***Кафедра комп’ютерної та програмної інженерії***

**ЛАБОРАТОРНА РОБОТА № 2**

**з курсу**

**«Універсальна кроссплатформна**

**розробка на платформі .Net Core»**

**Студентка: Щеголь.А.А.**

**Група: 42ІПЗ**

**Факультет: Математики, інформатики**

**та фізики**

**Перевірив: Січкаренко В.О.**

**Тема**: .NET Standard в розробці міжплатформного програмного забезпечення.

**Мета**: Ознайомитись та перевірити механізм .NET Standard для побудови між платформного програмного забезпечення.

Завдання

1. Ознайомитися з теоретичною частиною.
2. Створити новий репозиторій на GitHub під поточну лабораторну роботу;
3. Перейти в локальний репозиторій з лабораторними роботами;
4. Створити окрему гілку (git checkout -b) в системі контролю версій Git для поточної лабораторної роботи;
5. В локальному репозиторії створити новий проект з цільовою платформою .NET Standard 2.1;
6. Перенести код з попередньої лабораторної роботи в новий проект;
7. Зкомпілювати код;
8. Використовуючи іншу ОС (за допомогою віртуальної машини, чи іншому фізичному пристрої), створити консольний проект .NET Core;
9. Додати та підключити створену .NET Standard-бібліотеку в проект .NET Core;
10. Продемонструвати використання .NET Standard-бібліотеки в проекті .NET Core на іншій ОС;
11. Проект .NET Core створений під іншою ОС, перенести в локальний репозиторій;
12. Продемонструвати роботу програми;
13. Результат роботи програми зберегти у вигляді скріншоту (png або jpeg) в репозиторії з проектом;
14. Індексувати обидва проекти (.NET Standard та .NET Core) (git add);
15. Зафіксувати зміни (git commit);
16. Надіслати зміни у віддалений репозиторій (git push);
17. Створити та виконати запит на зміни (pull request);
18. Надіслати посилання на поточну лабораторну роботу у віддаленому репозиторії в GitHub.
19. Створення нового репозиторію на GitHub:

https://github.com/Anastasiie/UniversalCrossPlatformDevelopment.git

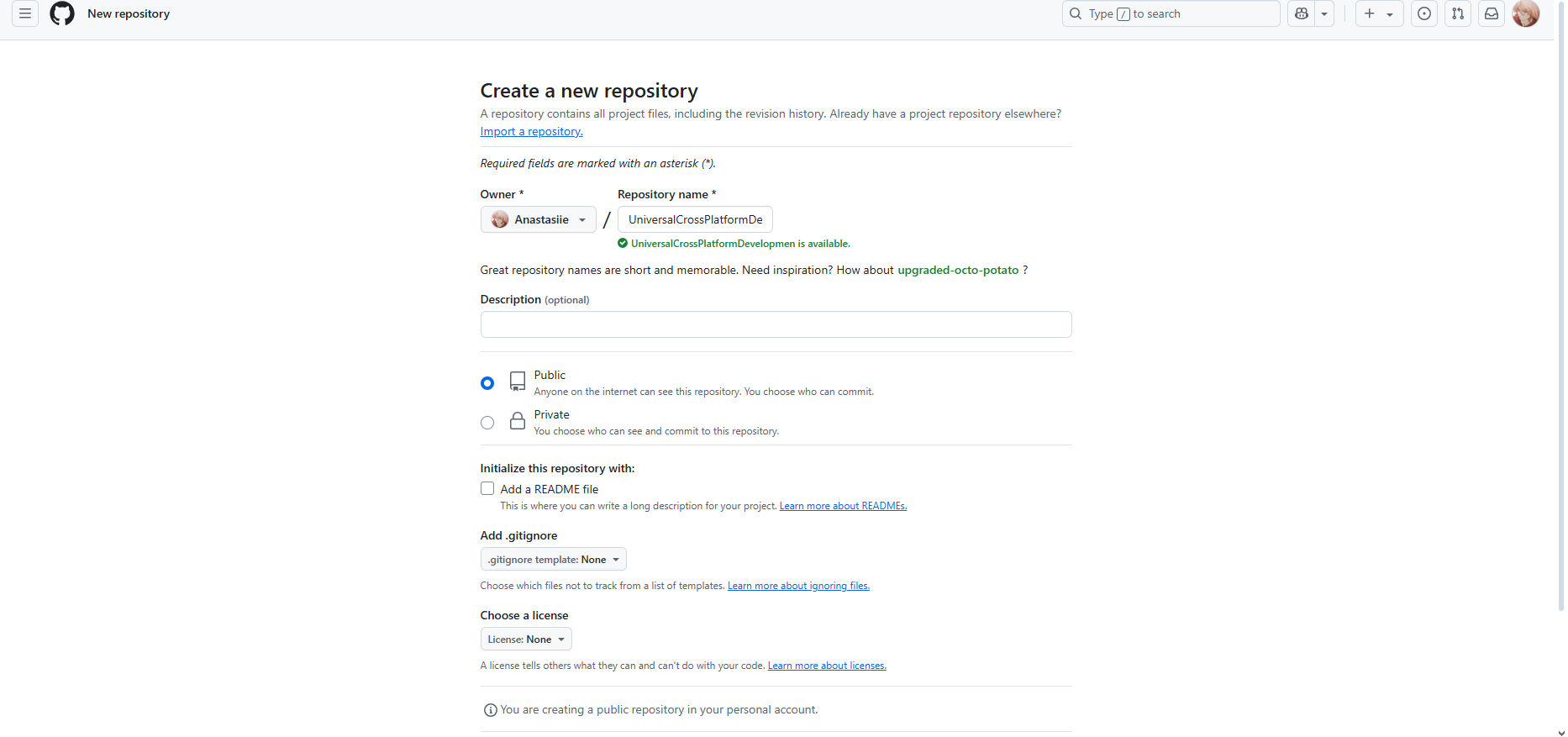


Рис.1 Створюємо новий репозиторій на GitHub

1. Перейти в локальний репозиторій з лабораторними роботами

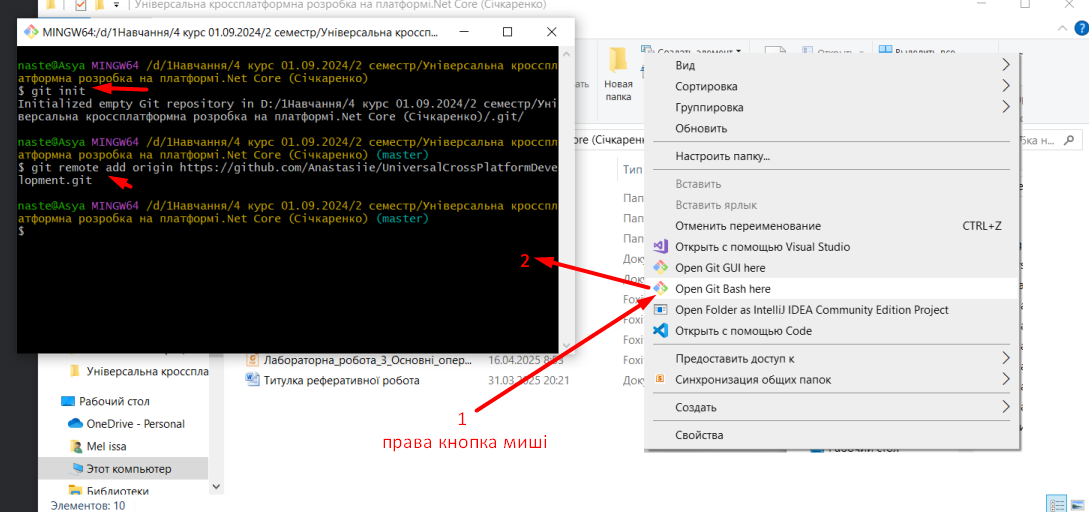


Рис.2 Локальний репозиторій

$ git init

git remote add origin https://github.com/Anastasiie /Universal CrossPlatformDevelopment.git

git checkout -b lab2\_geometry\_calculator

1. Створюємо окрему гілку (git checkout -b) в системі контролю версій Git для поточної лабораторної роботи

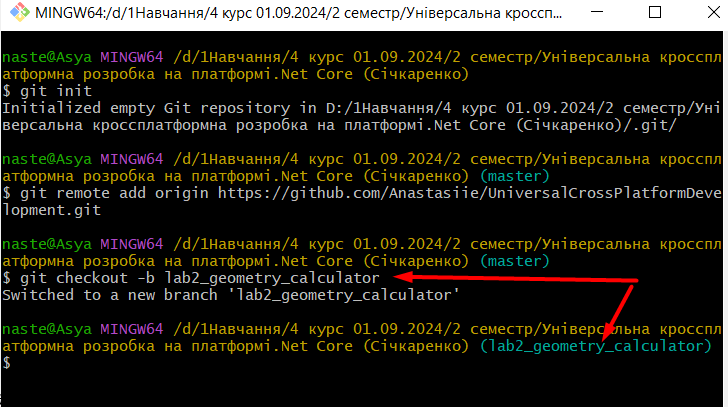


Рис.3 Створення окремої гілки

1. В локальному репозиторії створити новий проект з цільовою платформою .NET Standard 2.1;

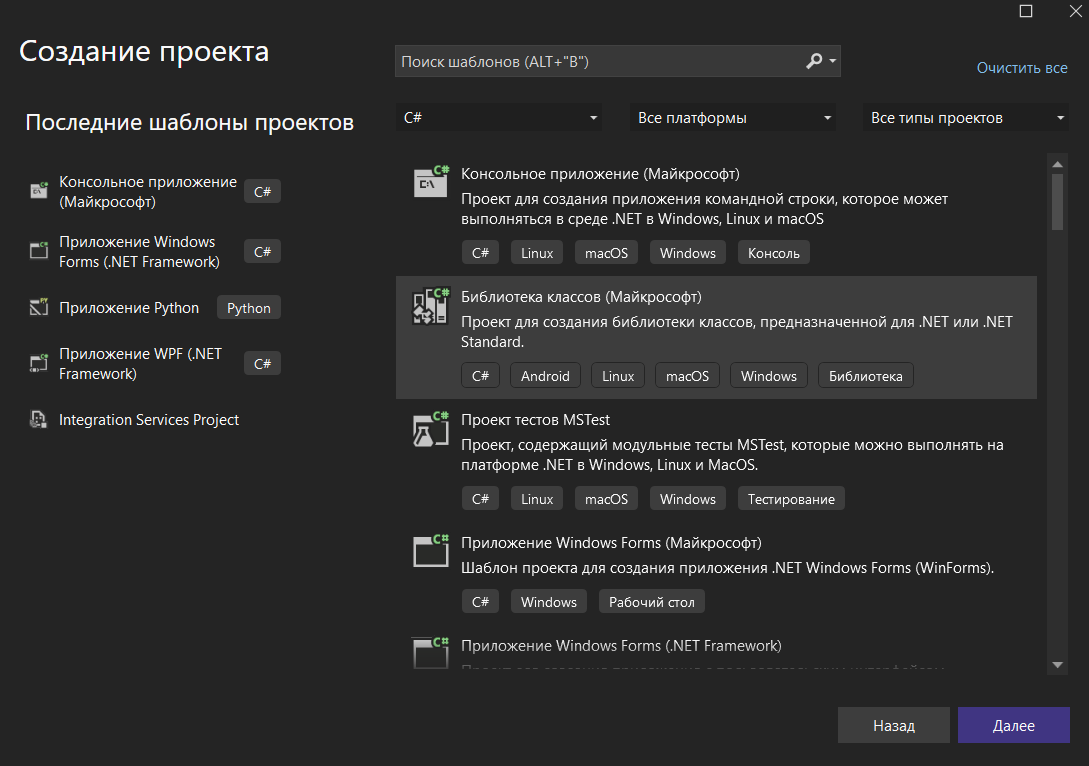


Рис.4 Створення бібліотеки

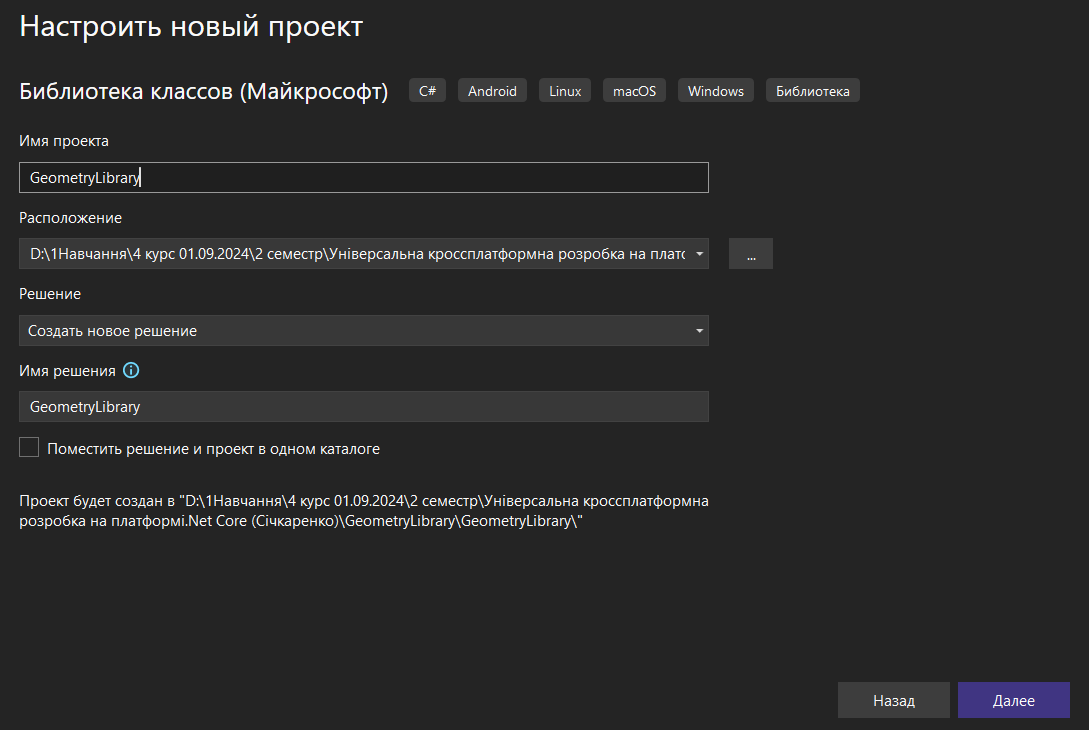
. 

Рис.5 Створення бібліотеки

1. Перенести код з попередньої лабораторної роботи в новий проект

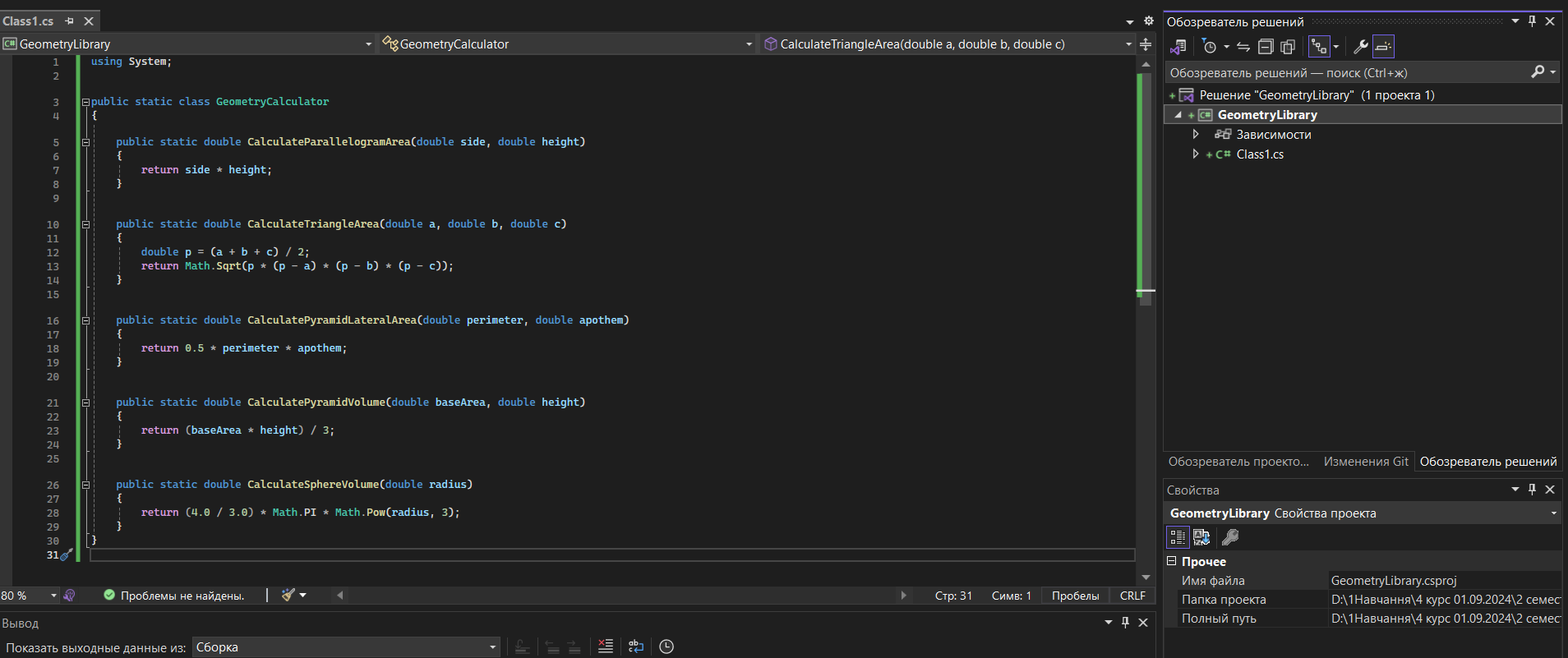


Рис.6 Копіюємо код

1. Зкомпілювати код

**Створюємо другий проєкт — консольний**

* У цьому ж рішенні (solution) створюємо новий проєкт:
  + **Console App (.NET Core)** або **Console App (.NET Framework)** (не важливо, головне — щоб міг запускатись).

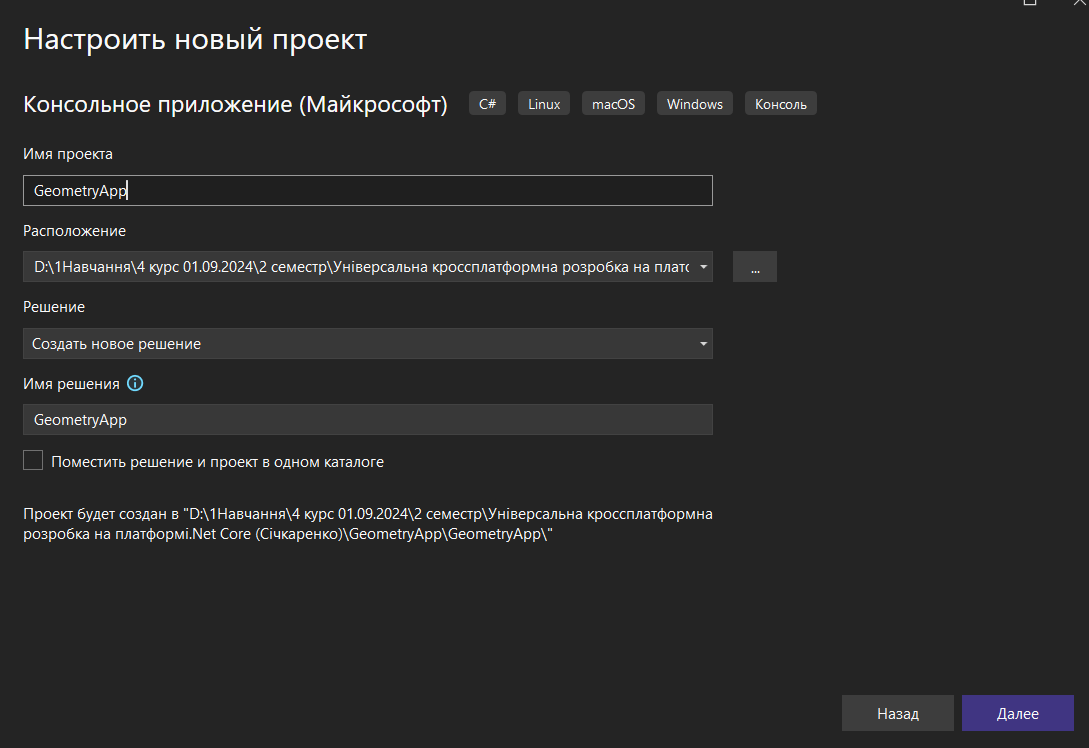


Рис 7.Створюємо новий проект у тому ж рішенні

* Переносимо код

public static class GeometryCalculator

{

public static double CalculateParallelogramArea(double side, double height)

{

return side \* height;

}

public static double CalculateTriangleArea(double a, double b, double c)

{

double p = (a + b + c) / 2;

return Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

public static double CalculatePyramidLateralArea(double perimeter, double apothem)

{

return 0.5 \* perimeter \* apothem;

}

public static double CalculatePyramidVolume(double baseArea, double height)

{

return (baseArea \* height) / 3;

}

public static double CalculateSphereVolume(double radius)

{

return (4.0 / 3.0) \* Math.PI \* Math.Pow(radius, 3);

}

}

**Додаємо посилання на бібліотеку**

* У проєкті GeometryApp:
  + Клік правою кнопкою по **Dependencies** → **Add Reference...**
  + Обери проєкт GeometryLibrary.
  + Тепер консольна програма бачить і може використовувати методи з бібліотеки.

**Побудуй і запусти**

* Клікни правою по GeometryApp → Set as Startup Project.
  + Побудуй рішення (Ctrl + Shift + B).
  + Запусти (Ctrl + F5).

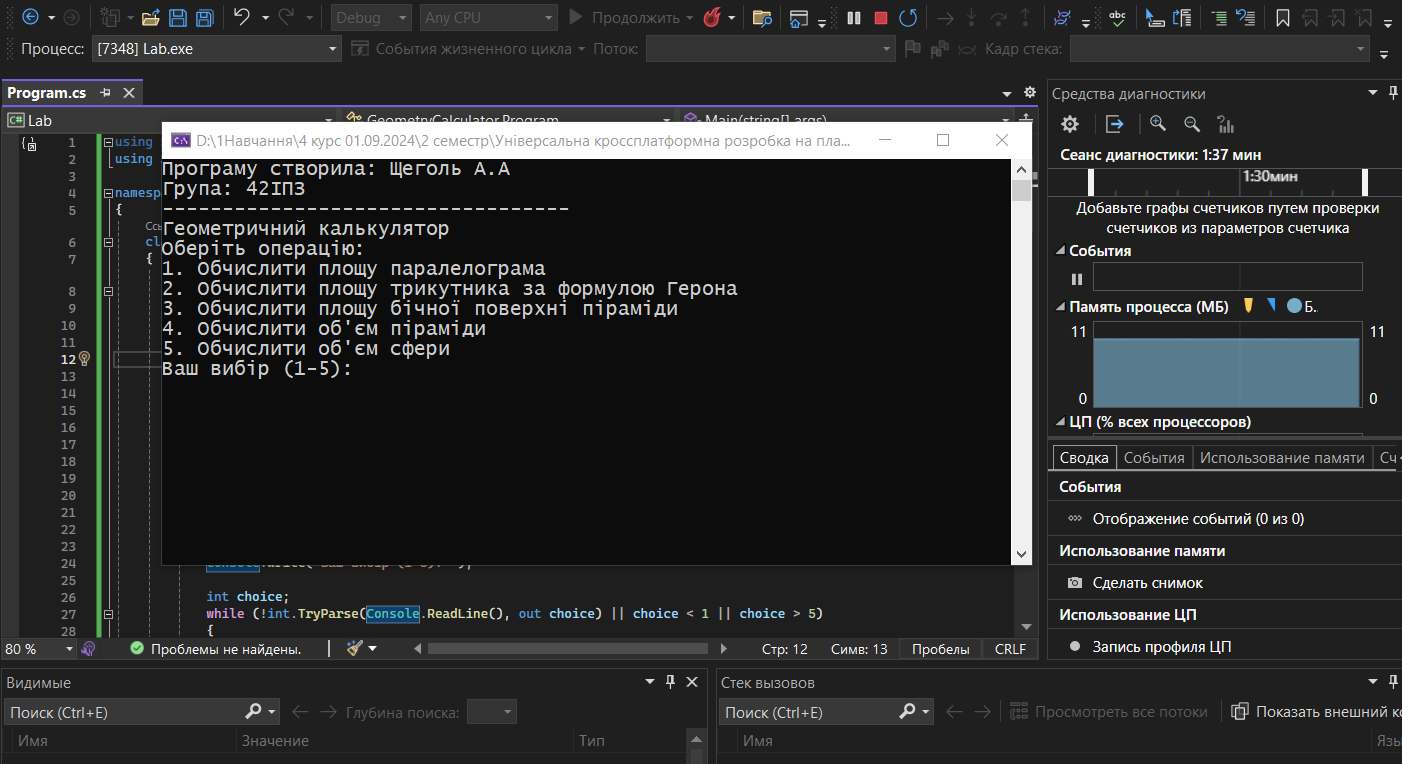


Рис.8 Вигляд проекту

# source\_code.cs

GeometryApp

using System;

using System.Text;

using GeometryLibrary;

namespace GeometryApp

{

class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

Console.InputEncoding = Encoding.UTF8;

Console.WriteLine("Програму створила: Щеголь A.A");

Console.WriteLine("Група: 42ІПЗ");

Console.WriteLine("----------------------------------");

Console.WriteLine("Геометричний калькулятор");

Console.WriteLine("Оберіть операцію:");

Console.WriteLine("1. Обчислити площу паралелограма");

Console.WriteLine("2. Обчислити площу трикутника за формулою Герона");

Console.WriteLine("3. Обчислити площу бічної поверхні піраміди");

Console.WriteLine("4. Обчислити об'єм піраміди");

Console.WriteLine("5. Обчислити об'єм сфери");

Console.Write("Ваш вибір (1-5): ");

int choice;

while (!int.TryParse(Console.ReadLine(), out choice) || choice < 1 || choice > 5)

{

Console.Write("Будь ласка, введіть число від 1 до 5: ");

}

try

{

switch (choice)

{

case 1:

Console.Write("Введіть довжину сторони: ");

double side = GetPositiveNumber();

Console.Write("Введіть висоту: ");

double height = GetPositiveNumber();

Console.WriteLine($"Площа паралелограма: {GeometryCalculator.CalculateParallelogramArea(side, height):F2}");

break;

case 2:

Console.Write("Введіть довжину першої сторони: ");

double a = GetPositiveNumber();

Console.Write("Введіть довжину другої сторони: ");

double b = GetPositiveNumber();

Console.Write("Введіть довжину третьої сторони: ");

double c = GetPositiveNumber();

Console.WriteLine($"Площа трикутника: {GeometryCalculator.CalculateTriangleArea(a, b, c):F2}");

break;

case 3:

Console.Write("Введіть периметр основи: ");

double perimeter = GetPositiveNumber();

Console.Write("Введіть апофему: ");

double apothem = GetPositiveNumber();

Console.WriteLine($"Площа бічної поверхні піраміди: {GeometryCalculator.CalculatePyramidLateralArea(perimeter, apothem):F2}");

break;

case 4:

Console.Write("Введіть площу основи: ");

double baseArea = GetPositiveNumber();

Console.Write("Введіть висоту: ");

double h = GetPositiveNumber();

Console.WriteLine($"Об'єм піраміди: {GeometryCalculator.CalculatePyramidVolume(baseArea, h):F2}");

break;

case 5:

Console.Write("Введіть радіус: ");

double radius = GetPositiveNumber();

Console.WriteLine($"Об'єм сфери: {GeometryCalculator.CalculateSphereVolume(radius):F2}");

break;

}

}

catch (Exception ex)

{

Console.WriteLine($"Помилка: {ex.Message}");

}

Console.WriteLine("

Натисніть будь-яку клавішу для виходу...");

Console.ReadKey();

}

static double GetPositiveNumber()

{

double number;

while (!double.TryParse(Console.ReadLine(), out number) || number <= 0)

{

Console.Write("Будь ласка, введіть додатне число: ");

}

return number;

}

}

}

GeometryLibrary

using System;

namespace GeometryLibrary

{

public static class Geometry

{

public static double CalculateParallelogramArea(double side, double height)

{

return side \* height;

}

public static double CalculateTriangleArea(double a, double b, double c)

{

if (a + b <= c || a + c <= b || b + c <= a)

{

throw new ArgumentException("Трикутник з такими сторонами не існує!");

}

double p = (a + b + c) / 2;

return Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

public static double CalculatePyramidLateralArea(double perimeter, double apothem)

{

return 0.5 \* perimeter \* apothem;

}

public static double CalculatePyramidVolume(double baseArea, double height)

{

return (baseArea \* height) / 3;

}

public static double CalculateSphereVolume(double radius)

{

return (4.0 / 3.0) \* Math.PI \* Math.Pow(radius, 3);

}

}

}