**17.** Сборки, Библиотеки, Атрибуты, Директивы

Задание 1. Создать библиотеку с двумя классами: Треугольник (методы ввода сторон, проверки на существование, вычисления периметра, вычисления площади, определения вида треугольника (разносторонний, равнобедренный, равносторонний)); Прямоугольник (методы ввода сторон, вычисления периметра, вычисления площади).

using GeometryLibrary;

namespace task1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter triangle sides: ");

double a = double.Parse(Console.ReadLine());

double b = double.Parse(Console.ReadLine());

double c = double.Parse(Console.ReadLine());

try

{

Triangle t = new Triangle(a, b, c);

Console.WriteLine($"Perimeter: {t.Perimeter()}");

Console.WriteLine($"Area: {t.Area()}");

Console.WriteLine($"Type: {t.Type()}");

}

catch (ArgumentException e)

{

Console.WriteLine(e.Message);

}

Console.WriteLine("\nEnter rectangle sides: ");

double l = double.Parse(Console.ReadLine());

double w = double.Parse(Console.ReadLine());

try

{

Rectangle r = new Rectangle(l, w);

Console.WriteLine($"Perimeter: {r.Perimeter()}");

Console.WriteLine($"Area: {r.Area()}");

}

catch (ArgumentException e)

{

Console.WriteLine(e.Message);

}

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System;

namespace GeometryLibrary

{

public class Triangle

{

private double sideA;

private double sideB;

private double sideC;

public Triangle(double a, double b, double c)

{

if (a <= 0 || b <= 0 || c <= 0)

{

throw new ArgumentException("Sides must be positive numbers.");

}

if (!IsValidTriangle(a, b, c))

{

throw new ArgumentException("Invalid triangle.");

}

sideA = a;

sideB = b;

sideC = c;

}

private static bool IsValidTriangle(double a, double b, double c)

{

return a + b > c && b + c > a && a + c > b;

}

public double Perimeter()

{

return sideA + sideB + sideC;

}

public double Area()

{

double s = (sideA + sideB + sideC) / 2;

return Math.Sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

}

public string Type()

{

if (sideA == sideB && sideB == sideC)

{

return "Equilateral";

}

else if (sideA == sideB || sideA == sideC || sideB == sideC)

{

return "Isosceles";

}

else

{

return "Scalene";

}

}

}

public class Rectangle

{

private double length;

private double width;

public Rectangle(double l, double w)

{

if (l <= 0 || w <= 0)

{

throw new ArgumentException("Sides must be positive numbers.");

}

length = l;

width = w;

}

public double Perimeter()

{

return 2 \* (length + width);

}

public double Area()

{

return length \* width;

}

}

}

Анализ результатов:

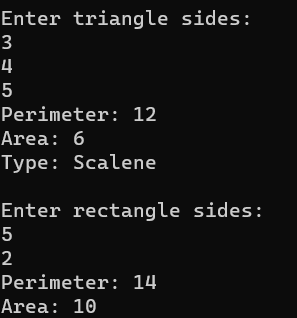


Рисунок 1.1 –работы программы

Задание 2. Создать свою библиотеку на индивидуальную тему и

продемонстрировать ее функциональность. (Расписание)

using System;

using ScheduleLibrary;

namespace task2

{

class Program

{

static void Main(string[] args)

{

Schedule schedule = new Schedule();

bool keepRunning = true;

while (keepRunning)

{

Console.WriteLine("Select an option:");

Console.WriteLine("1. Add event");

Console.WriteLine("2. Remove event");

Console.WriteLine("3. View schedule");

Console.WriteLine("4. Exit");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Enter date (mm/dd/yyyy): ");

DateTime date = DateTime.Parse(Console.ReadLine());

Console.Write("Enter event name: ");

string eventName = Console.ReadLine();

schedule.AddEvent(date, eventName);

break;

case "2":

Console.Write("Enter date of event to remove (mm/dd/yyyy): ");

DateTime dateToRemove = DateTime.Parse(Console.ReadLine());

schedule.RemoveEvent(dateToRemove);

break;

case "3":

schedule.DisplayEvents();

break;

case "4":

keepRunning = false;

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

}

}

}

using System;

using System.Collections.Generic;

namespace ScheduleLibrary

{

public class Schedule

{

private Dictionary<DateTime, string> \_schedule = new Dictionary<DateTime, string>();

public void AddEvent(DateTime date, string eventName)

{

if (\_schedule.ContainsKey(date))

{

Console.WriteLine($"There's already an event scheduled on {date.ToShortDateString()}. Would you like to replace it? (y/n)");

string response = Console.ReadLine().ToLower();

if (response == "y")

{

\_schedule[date] = eventName;

}

else

{

Console.WriteLine("Event not added.");

}

}

else

{

\_schedule.Add(date, eventName);

}

}

public void RemoveEvent(DateTime date)

{

if (\_schedule.ContainsKey(date))

{

\_schedule.Remove(date);

Console.WriteLine($"Event on {date.ToShortDateString()} has been removed.");

}

else

{

Console.WriteLine($"No event scheduled on {date.ToShortDateString()}.");

}

}

public void DisplayEvents()

{

if (\_schedule.Count == 0)

{

Console.WriteLine("No events scheduled.");

}

else

{

Console.WriteLine("Scheduled events:");

foreach (KeyValuePair<DateTime, string> item in \_schedule)

{

Console.WriteLine($"{item.Key.ToShortDateString()} - {item.Value}");

}

}

}

}

}

Анализ результатов:

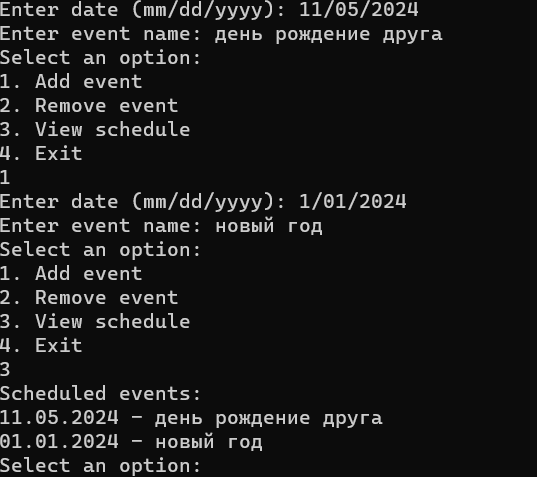
**

Рисунок 1.2 – Результат работы программы

работы программы