Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М.Машерова

**Отчет**

по учебной практике по программированию

«Модуль 2. Объектно-ориентированное программирование (ООП)»

Выполнил учащийся Липинский М.А.

группы 3ПОИС23 01.10.2025г.

Проверил Алейников М.А.

01.10.2025г.

Орша, 2025

**Цель:** разработать Модуль 2.

**Общее задание для всех**

**Задание 1**

Создайте класс Person, представляющий человека. У этого класса должны быть поля для хранения имени, возраста и адреса. Добавьте методы для установки и получения значений этих полей. Затем создайте объекты этого класса и выведите информацию о них.

**Задание 2**

Создайте базовый класс Shape, представляющий геометрическую фигуру, и производные классы Circle и Rectangle. В базовом классе определите метод Area(), который возвращает площадь фигуры, и метод Perimeter(), который возвращает периметр фигуры. В производных классах переопределите эти методы для соответствующих фигур (круг и прямоугольник). Создайте объекты всех классов и выведите их площади и периметры.

**Задание 3**

Создайте классы Author и Book. Класс Author должен содержать информацию об авторе (имя и год рождения). Класс Book должен содержать информацию о книге (название, год выпуска и автора). Используйте композицию, чтобы связать объекты Author и Book. Создайте несколько объектов Author и Book и выведите информацию о них.

**Задание 4**

Создайте интерфейс IDrawable с методом Draw(), который выводит информацию о рисуемом объекте. Создайте классы Circle, Rectangle и Triangle, реализующие этот интерфейс. Создайте массив объектов, реализующих интерфейс IDrawable, и вызовите метод Draw() для каждого из них.

**Задание 5**

Создайте класс TemperatureSensor, который генерирует событие TemperatureChanged, когда измеренная температура меняется. Создайте класс Thermostat, который подписывается на событие TemperatureChanged и реагирует на изменение температуры, включая или выключая отопление.

**Вариант 1.**

**Задание 6.**

1. Создайте класс "Студент" со свойствами: имя, фамилия, возраст и средний балл. Создайте несколько экземпляров этого класса и выведите информацию о них.

2. Создайте структуру с именем train, содержащую поля: название пункта назначения, номер поезда, время отправления. Ввести данные в массив из пяти элементов типа train, упорядочить элементы по номерам поездов. Добавить возможность вывода информации о поезде, номер которого введен пользователем. Добавить возможность сортировки массив по пункту назначения, причем поезда с одинаковыми пунктами назначения должны быть упорядочены по времени отправления.

3. Разработайте иерархию классов для геометрических фигур (круг, прямоугольник, треугольник). Реализуйте методы вычисления площади и периметра для каждой фигуры.

**Ход выполнения работы.**

**Выполнение практических заданий.**

**Задание 1.**

Для выполнения первого задания нужно создать класс Person, представляющий человека. У этого класса должны быть поля для хранения имени, возраста и адреса. Добавьте методы для установки и получения значений этих полей. Затем создайте объекты этого класса и выведите информацию о них.

Код на языке C#, представленный ниже в листинге 1.

Листинг 1. Код программы на языке программирования C#.

using System;

class Person

{

private string name;

private int age;

private string address;

// Методы для установки значений

public void SetName(string name)

{

this.name = name;

}

public void SetAge(int age)

{

this.age = age;

}

public void SetAddress(string address)

{

this.address = address;

}

// Методы для получения значений

public string GetName()

{

return name;

}

public int GetAge()

{

return age;

}

public string GetAddress()

{

return address;

}

// Метод для вывода информации

public void DisplayInfo()

{

Console.WriteLine($"Имя: {name}, Возраст: {age}, Адрес: {address}");

}

}

class Program

{

static void Main()

{

Person person1 = new Person();

person1.SetName("Михаил");

person1.SetAge(17);

person1.SetAddress("Орша");

person1.DisplayInfo();

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 1.

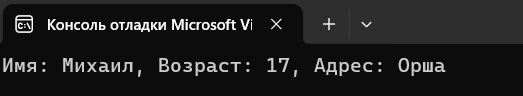


Рисунок 1 – Вывод класса на консоль

**Задание 2.**

Для выполнения второго задания нужно создать базовый класс Shape, представляющий геометрическую фигуру, и производные классы Circle и Rectangle. В базовом классе определите метод Area(), который возвращает площадь фигуры, и метод Perimeter(), который возвращает периметр фигуры. В производных классах переопределите эти методы для соответствующих фигур (круг и прямоугольник). Создайте объекты всех классов и выведите их площади и периметры

Код на языке C#, представленный ниже в листинге 2.

Листинг 2. Код программы на языке программирования C#.

using System;

class Shape

{

public virtual double Area() { return 0; }

public virtual double Perimeter() { return 0; }

}

class Circle : Shape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public override double Area()

{

return Math.PI \* radius \* radius;

}

public override double Perimeter()

{

return 2 \* Math.PI \* radius;

}

}

class Rectangle : Shape

{

private double width;

private double height;

public Rectangle(double width, double height)

{

this.width = width;

this.height = height;

}

public override double Area()

{

return width \* height;

}

public override double Perimeter()

{

return 2 \* (width + height);

}

}

class Program

{

static void Main()

{

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

Console.WriteLine($"Площадь круга: {circle.Area()}, Периметр: {circle.Perimeter()}");

Console.WriteLine($"Площадь прямоугольника: {rectangle.Area()}, Периметр: {rectangle.Perimeter()}");

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 2.

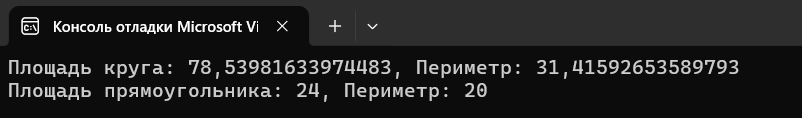


Рисунок 2 – Площадь и периметр фигур

**Задание 3.**

Для выполнения третьего задания нужно создать классы Author и Book. Класс Author должен содержать информацию об авторе (имя и год рождения). Класс Book должен содержать информацию о книге (название, год выпуска и автора). Используйте композицию, чтобы связать объекты Author и Book. Создайте несколько объектов Author и Book и выведите информацию о них.

Код на языке C#, представленный ниже в листинге 3.

Листинг 3. Код программы на языке программирования C#.

using System;

// Класс для автора

class Author

{

// Имя автора

public string Name { get; set; }

// Год рождения автора

public int BirthYear { get; set; }

// Конструктор: задаёт имя и год рождения

public Author(string name, int birthYear)

{

Name = name;

BirthYear = birthYear;

}

}

// Класс для книги

class Book

{

// Название книги

public string Title { get; set; }

// Год выпуска книги

public int ReleaseYear { get; set; }

// Автор книги

public Author Author { get; set; }

// Конструктор: задаёт название, год выпуска и автора

public Book(string title, int releaseYear, Author author)

{

Title = title;

ReleaseYear = releaseYear;

Author = author;

}

// Метод для вывода информации о книге

public void DisplayInfo()

{

Console.WriteLine($"Книга: {Title}, Год выпуска: {ReleaseYear}, Автор: {Author.Name}, Год рождения автора: {Author.BirthYear}");

}

}

class Program

{

static void Main()

{

// Создаём автора.

Author author = new Author("Владимир Маяковский", 1893);

// Создаём книгу

Book book = new Book("Облако в штанах", 1915, author);

// Выводим информацию о книге

book.DisplayInfo();

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 3.

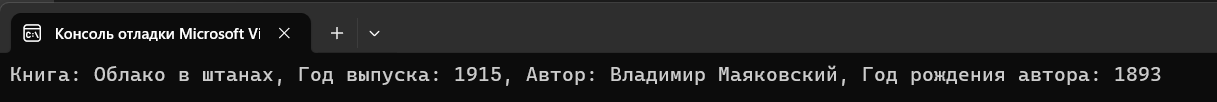


Рисунок 3 – Информация об книге

**Задание 4.**

Для выполнения четвертого задания нужно создать интерфейс IDrawable с методом Draw(), который выводит информацию о рисуемом объекте. Создайте классы Circle, Rectangle и Triangle, реализующие этот интерфейс. Создайте массив объектов, реализующих интерфейс IDrawable, и вызовите метод Draw() для каждого из них.

Код на языке C#, представленный ниже в листинге 4.

Листинг 4. Код программы на языке программирования C#.

using System;

// Интерфейс для рисования фигур

interface IDrawable

{

// Метод для рисования фигуры

void Draw();

}

// Класс для круга, реализует IDrawable

class Circle : IDrawable

{

// Рисует круг

public void Draw()

{

Console.WriteLine("Рисую круг");

}

}

// Класс для прямоугольника, реализует IDrawable

class Rectangle : IDrawable

{

// Рисует прямоугольник

public void Draw()

{

Console.WriteLine("Рисую прямоугольник");

}

}

// Класс для треугольника, реализует IDrawable

class Triangle : IDrawable

{

// Рисует треугольник

public void Draw()

{

Console.WriteLine("Рисую треугольник");

}

}

class Program

{

static void Main()

{

// Массив фигур типа IDrawable

IDrawable[] shapes = new IDrawable[]

{

new Circle(), // Круг

new Rectangle(), // Прямоугольник

new Triangle() // Треугольник

};

// Перебор массива и рисование каждой фигуры

foreach (IDrawable shape in shapes)

{

shape.Draw();

}

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 4.

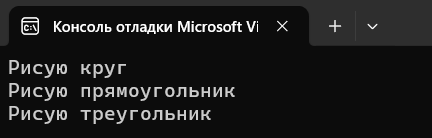


Рисунок 4 – Рисование фигур

**Задание 5.**

Для выполнения пятого задания нужно создать класс TemperatureSensor, который генерирует событие TemperatureChanged, когда измеренная температура меняется. Создайте класс Thermostat, который подписывается на событие TemperatureChanged и реагирует на изменение температуры, включая или выключая отопление

Код на языке C#, представленный ниже в листинге 5.

Листинг 5. Код программы на языке программирования C#.

using System;

// Класс для сенсора температуры

class TemperatureSensor

{

// Событие изменения температуры (тип Action<int>)

public event Action<int> TemperatureChanged;

// Приватное поле для текущей температуры

private int temperature;

// Свойство Temperature с геттером и сеттером

public int Temperature

{

get { return temperature; }

set

{

// Если температура изменилась, обновляем и вызываем событие

if (temperature != value)

{

temperature = value;

OnTemperatureChanged(value);

}

}

}

// Защищённый виртуальный метод для вызова события

protected virtual void OnTemperatureChanged(int newTemperature)

{

TemperatureChanged?.Invoke(newTemperature);

}

}

// Класс для термостата

class Thermostat

{

// Метод, который реагирует на изменение температуры

public void OnTemperatureChanged(int newTemperature)

{

// Проверка: если выше 25, включаем кондиционер

if (newTemperature > 25)

{

Console.WriteLine("Температура слишком высокая, включаю кондиционер.");

}

// Если ниже 18, включаем отопление

else if (newTemperature < 11)

{

Console.WriteLine("Температура слишком низкая, включаю отопление.");

}

// Иначе - норма

else

{

Console.WriteLine("Температура в норме.");

}

}

}

class Program

{

static void Main()

{

// Создаём сенсор температуры

TemperatureSensor sensor = new TemperatureSensor();

// Создаём термостат

Thermostat thermostat = new Thermostat();

// Подписываем термостат на событие изменения температуры

sensor.TemperatureChanged += thermostat.OnTemperatureChanged;

// Изменяем температуру

sensor.Temperature = 30; // Температура слишком высокая

sensor.Temperature = 10; // Температура слишком низкая

sensor.Temperature = 22; // Температура в норме

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 5.

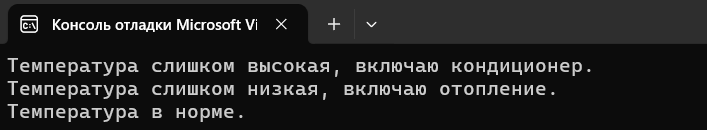


Рисунок 5 – Изменение температуры

**Вариант 1.**

**Задание 6.**

Для выполнения шестого задания нужно создать класс "Студент" со свойствами: имя, фамилия, возраст и средний балл. Создайте несколько экземпляров этого класса и выведите информацию о них.

Код на языке C#, представленный ниже в листинге 6.

Листинг 6. Код программы на языке программирования C#.

using System;

using System.Diagnostics.Metrics;

using System.Xml.Linq;

namespace zd\_6

{

// Струкртура Студент и ее данные (имя; фамилия; возраст; средний балл)

struct Student

{

public string Name;

public string SurName;

public int age;

public double AverageScore;

// Модификатор для передачи переменных

public Student (string Name, string SurName, int age , double AverageScore)

{

this.Name = Name;

this.SurName = SurName;

this.age = age;

this.AverageScore = AverageScore;

}

// Метод для вывода

public void WriteUserInfo()

{

Console.WriteLine($"Имя студента: {Name}");

Console.WriteLine($"Фамилия студента: {SurName}");

Console.WriteLine($"Возраст студента: {age}");

Console.WriteLine($"Средний балл студента: {AverageScore}");

}

}

internal class Program

{

static void Main(string[] args)

{

Student student;

Console.Write("Введите имя студента: ");

student.Name = Console.ReadLine();

Console.Write("Введите Фамилию студента:");

student.SurName = Console.ReadLine();

Console.Write("Введите возраст студента:");

student.age = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите средний балл студента:");

student.AverageScore = Convert.ToDouble(Console.ReadLine());

student.WriteUserInfo(); // Вызов метода

Console.ReadKey();

}

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 6.

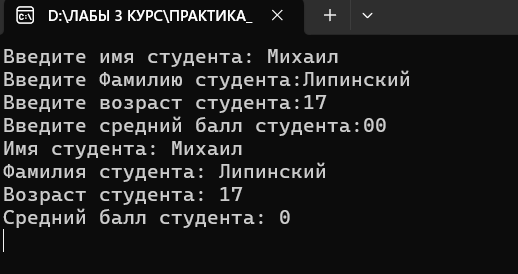


Рисунок 6 – Наглядное использование структуры

**Задание 7.**

Для выполнения седьмого задания нужно создать структуру с именем train, содержащую поля: название пункта назначения, номер поезда, время отправления. Ввести данные в массив из пяти элементов типа train, упорядочить элементы по номерам поездов. Добавить возможность вывода информации о поезде, номер которого введен пользователем. Добавить возможность сортировки массив по пункту назначения, причем поезда с одинаковыми пунктами назначения должны быть упорядочены по времени отправления.

Код на языке C#, представленный ниже в листинге 7.

Листинг 7. Код программы на языке программирования C#.

using System;

struct Train

{

public string Destination; // Пункт назначения

public int TrainNumber; // Номер поезда

public TimeSpan DepartureTime; // Время отправления

// Конструктор структуры

public Train(string destination, int trainNumber, TimeSpan departureTime)

{

Destination = destination;

TrainNumber = trainNumber;

DepartureTime = departureTime;

}

// Метод для вывода информации о поезде

public void DisplayInfo()

{

Console.WriteLine($"Пункт назначения: {Destination}, Номер поезда: {TrainNumber}, Время отправления: {DepartureTime}");

}

}

class Program

{

static void Main()

{

// Создаем массив поездов

Train[] trains = new Train[5];

// Ввод данных о поездах

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

{

Console.WriteLine($"Введите данные для поезда {i + 1}:");

// Ввод пункта назначения

Console.Write("Пункт назначения: ");

string destination = Console.ReadLine();

// Ввод номера поезда

Console.Write("Номер поезда: ");

int trainNumber = int.Parse(Console.ReadLine());

// Ввод времени отправления

Console.Write("Время отправления (чч:мм): ");

TimeSpan departureTime = TimeSpan.Parse(Console.ReadLine());

// Заполняем массив поездами

trains[i] = new Train(destination, trainNumber, departureTime);

}

// Сортировка поездов по номерам

SortTrainsByNumber(trains);

// Запрос у пользователя номера поезда для вывода информации

FindTrainByNumber(trains);

// Сортировка поездов по пункту назначения и времени отправления

SortTrainsByDestinationAndTime(trains);

// Вывод всех отсортированных поездов

Console.WriteLine("\nПоезда, отсортированные по пункту назначения и времени отправления:");

foreach (Train train in trains)

{

train.DisplayInfo();

}

}

// Метод для сортировки поездов по номерам

static void SortTrainsByNumber(Train[] trains)

{

// Стандартная сортировка массива поездов с помощью метода CompareTrainsByNumber

Array.Sort(trains, CompareTrainsByNumber);

}

// Метод сравнения поездов по номеру

static int CompareTrainsByNumber(Train firstTrain, Train secondTrain)

{

// Сравниваем номера поездов

if (firstTrain.TrainNumber > secondTrain.TrainNumber)

return 1; // Если номер первого поезда больше, то возвращаем 1

else if (firstTrain.TrainNumber < secondTrain.TrainNumber)

return -1; // Если номер первого поезда меньше, возвращаем -1

else

return 0; // Если номера равны, возвращаем 0

}

// Метод для поиска поезда по номеру, введенному пользователем

static void FindTrainByNumber(Train[] trains)

{

Console.Write("Введите номер поезда для вывода информации: ");

int inputNumber = int.Parse(Console.ReadLine());

// Ищем поезд по номеру

Train? foundTrain = null;

// Проходим по каждому поезду в массиве

foreach (Train train in trains)

{

if (train.TrainNumber == inputNumber)

{

foundTrain = train; // Если поезд найден, сохраняем его

break;

}

}

// Если поезд найден, выводим информацию

if (foundTrain != null)

{

foundTrain.Value.DisplayInfo();

}

else

{

Console.WriteLine("Поезд с таким номером не найден.");

}

}

// Метод для сортировки поездов по пункту назначения и времени отправления

static void SortTrainsByDestinationAndTime(Train[] trains)

{

// Сортируем массив поездов с помощью метода CompareTrainsByDestinationAndTime

Array.Sort(trains, CompareTrainsByDestinationAndTime);

}

// Метод сравнения поездов по пункту назначения и времени отправления

static int CompareTrainsByDestinationAndTime(Train firstTrain, Train secondTrain)

{

// Сначала сравниваем пункты назначения

int destinationComparison = firstTrain.Destination.CompareTo(secondTrain.Destination);

// Если пункты назначения одинаковые, сравниваем по времени отправления

if (destinationComparison == 0)

{

if (firstTrain.DepartureTime > secondTrain.DepartureTime)

return 1; // Если время отправления первого больше

else if (firstTrain.DepartureTime < secondTrain.DepartureTime)

return -1; // Если время отправления первого меньше

else

return 0; // Время отправления одинаковое

}

return destinationComparison; // Возвращаем результат сравнения по пунктам назначения

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 7.

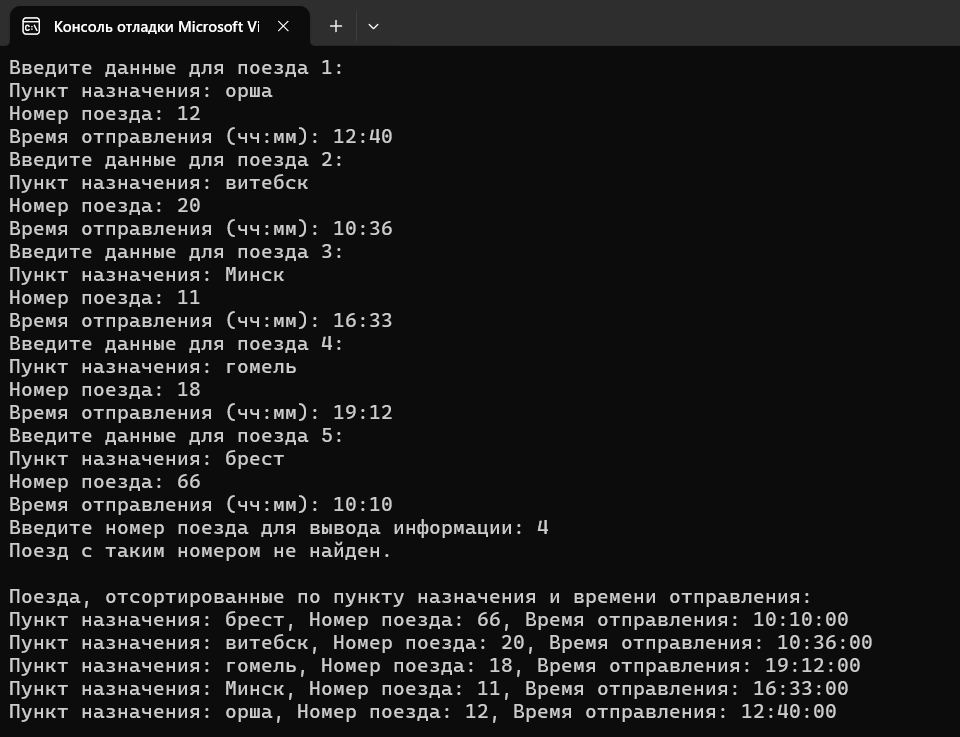


Рисунок 7 – Справка о поездах

**Задание 8.**

Для выполнения восьмого задания нужно разработать иерархию классов для геометрических фигур (круг, прямоугольник, треугольник). Реализуйте методы вычисления площади и периметра для каждой фигуры.

Код на языке C#, представленный ниже в листинге 8.

Листинг 8. Код программы на языке программирования C#.

using System;

// Абстрактный базовый класс для фигур

abstract class Shape

{

// Абстрактный метод для площади (должен быть реализован в подклассах)

public abstract double Area();

// Абстрактный метод для периметра (должен быть реализован в подклассах)

public abstract double Perimeter();

}

// Класс для круга, наследует от Shape

class Circle : Shape

{

private double radius; // Радиус круга

// Конструктор: задаёт радиус

public Circle(double radius)

{

this.radius = radius;

}

// Площадь круга: π \* r²

public override double Area()

{

return Math.PI \* radius \* radius;

}

// Периметр круга: 2 \* π \* r

public override double Perimeter()

{

return 2 \* Math.PI \* radius;

}

}

// Класс для прямоугольника, наследует от Shape

class Rectangle : Shape

{

private double width; // Ширина

private double height; // Высота

// Конструктор: задаёт ширину и высоту

public Rectangle(double width, double height)

{

this.width = width;

this.height = height;

}

// Площадь: ширина \* высота

public override double Area()

{

return width \* height;

}

// Периметр: 2 \* (ширина + высота)

public override double Perimeter()

{

return 2 \* (width + height);

}

}

// Класс для треугольника, наследует от Shape

class Triangle : Shape

{

private double sideA; // Сторона A

private double sideB; // Сторона B

private double sideC; // Сторона C

// Конструктор: задаёт все стороны

public Triangle(double sideA, double sideB, double sideC)

{

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

}

// Площадь по формуле Герона: sqrt(s \* (s-a) \* (s-b) \* (s-c)), где s - полупериметр

public override double Area()

{

double s = Perimeter() / 2; // Полупериметр

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

}

// Периметр: сумма всех сторон

public override double Perimeter()

{

return sideA + sideB + sideC;

}

}

class Program

{

static void Main()

{

// Создаём круг с радиусом 6

Shape circle = new Circle(6);

// Создаём прямоугольник 5x7

Shape rectangle = new Rectangle(5, 7);

// Создаём треугольник со сторонами 5, 3, 6

Shape triangle = new Triangle(5, 3, 6);

// Выводим площадь и периметр круга

Console.WriteLine($"Площадь круга: {circle.Area()}, Периметр: {circle.Perimeter()}");

// Выводим площадь и периметр прямоугольника

Console.WriteLine($"Площадь прямоугольника: {rectangle.Area()}, Периметр: {rectangle.Perimeter()}");

// Выводим площадь и периметр треугольника

Console.WriteLine($"Площадь треугольника: {triangle.Area()}, Периметр: {triangle.Perimeter()}");

}

}

Далее нужно выполнить запуск приложения для демонстрирования его работоспособности, представлено на рисунке 8.

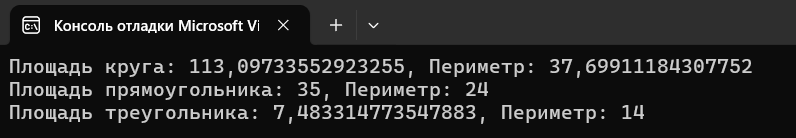


Рисунок 8 – Сумма элементов массива

Все коды хранятся на GitHub в репозитории ().

**Вывод:** в ходе выполнения практического задания был разработан модуль 2.