Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

**Лабораторная работа №4**

Выполнил:

Зенин Д.Д.

Проверил Иванов С. Е.

Санкт-Петербург,

2024

Оглавление

[ВВЕДЕНИЕ 3](#_Toc177765913)

[1. Упражнение 1 4](#_Toc177765914)

[2. Упражнение 2 5](#_Toc177765917)

[3. Упражнение 3 6](#_Toc177765921)

[4. Упражнение 4 9](#_Toc177765922)

[5. Упражнение 5 12](#_Toc177765923)

[ЗАКЛЮЧЕНИЕ 14](#_Toc177765927)

# ВВЕДЕНИЕ

Цель лабораторной работы №4 состояла в изучении и приобретении навыков работы с методами класса

# Упражнение 1

# В этом упражнении мы создадим класс Utils, в котором определим метод Greater. Этот метод будет принимать два целочисленных параметра и возвращать больший из них.

# 

Рисунок 1 – Код класса Utils и метод Greater

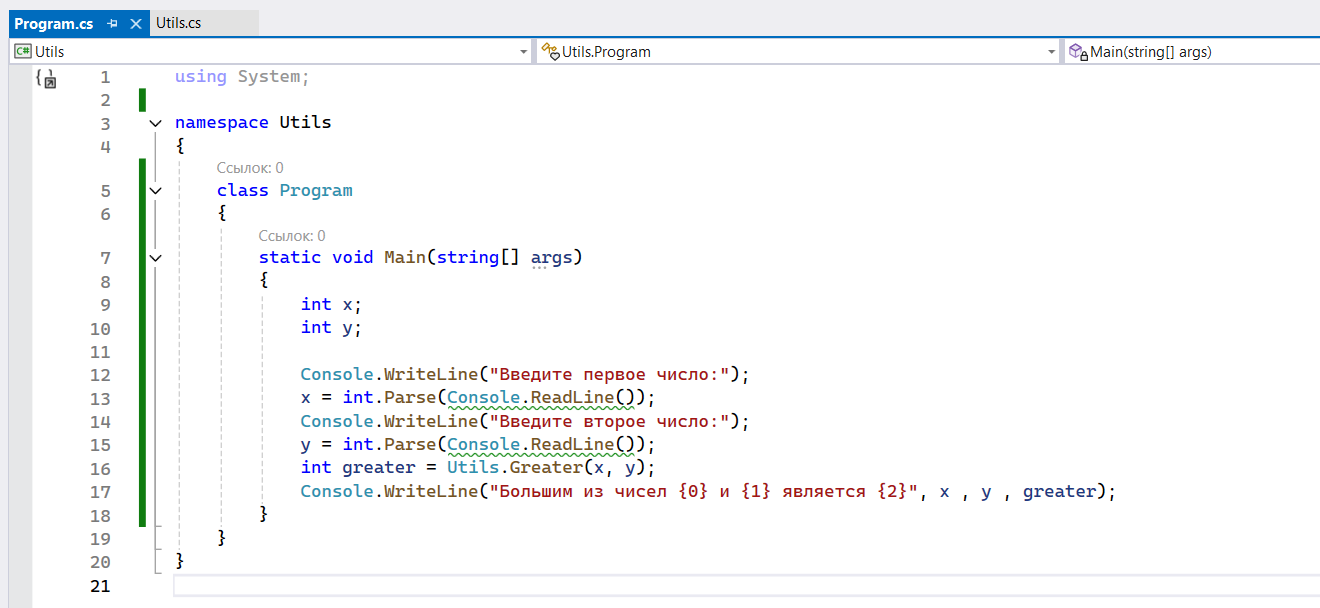


Рисунок 2 – Код класса Program

Вывод программы:

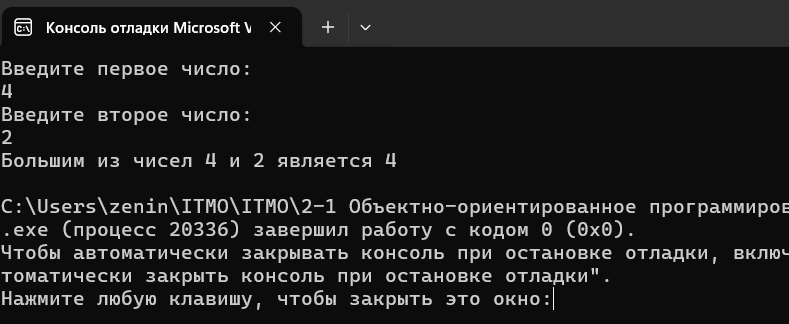


Рисунок 3 – Компиляция программы и ее запуск

Программа работает корректно и без ошибок.

# Упражнение 2

# В этом упражнении мы создадим метод Swap, который поменяет местами значения параметров. При этом мы будем использовать параметры, передаваемые по ссылке.

# Добавим в класс Utils код метода Swap

# 

Рисунок 3 – Класс Utils

Добавим в класс Program код, меняющий местами введенные значения:

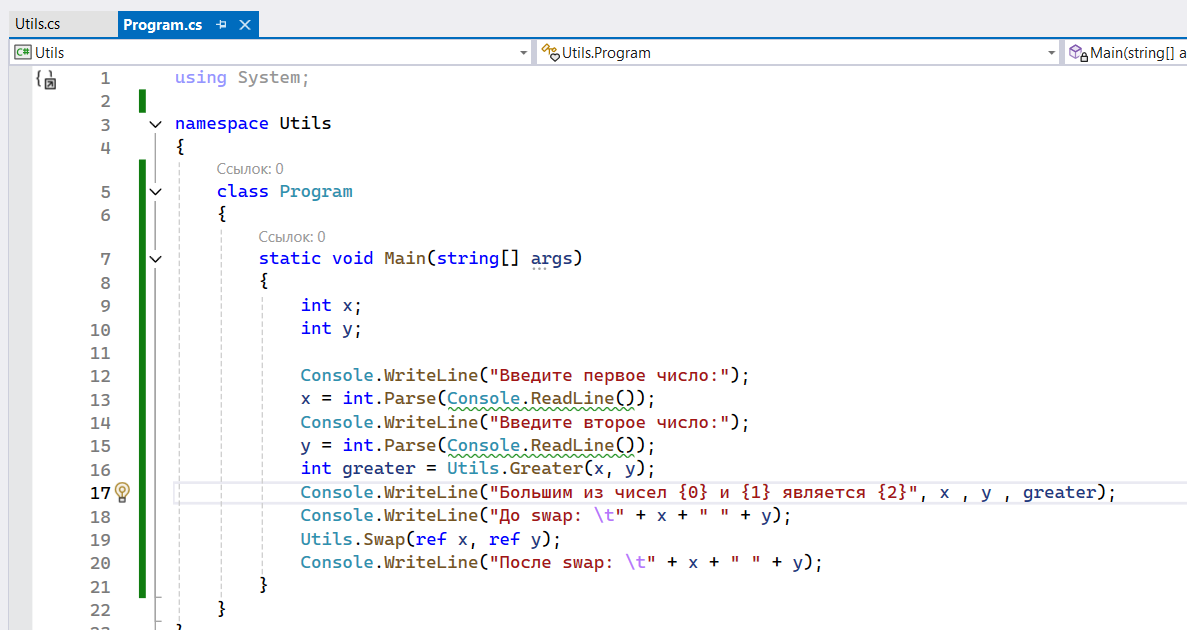


Рисунок 4 – Код класса Program

Результат работы программы:

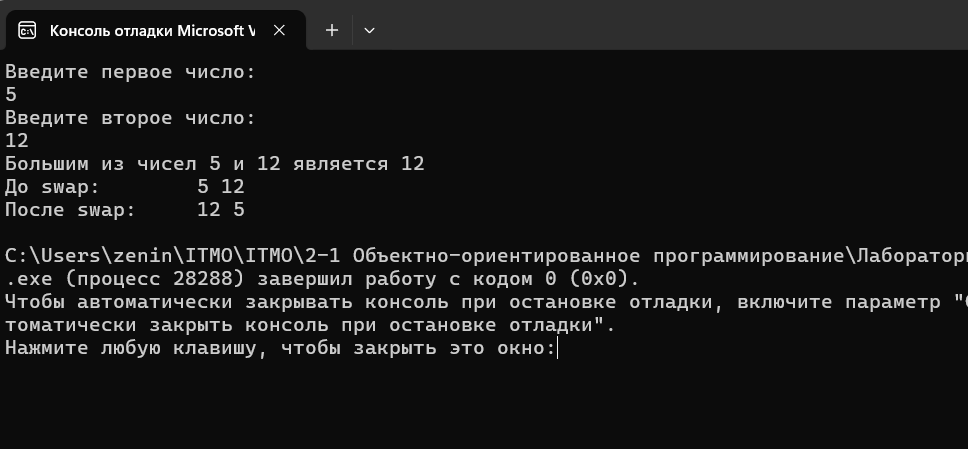


Рисунок 5 – Результат работы приложения

Приложение работает корректно и выводит нужный результат.

# Упражнение 3

В этом упражнении мы создадим метод Factorial, принимающий целочисленную переменную и рассчитывающий ее факториал по итерационному алгоритму.

В классе Utils напишем метод Factorial, используя try-catch для отлова ошибок.

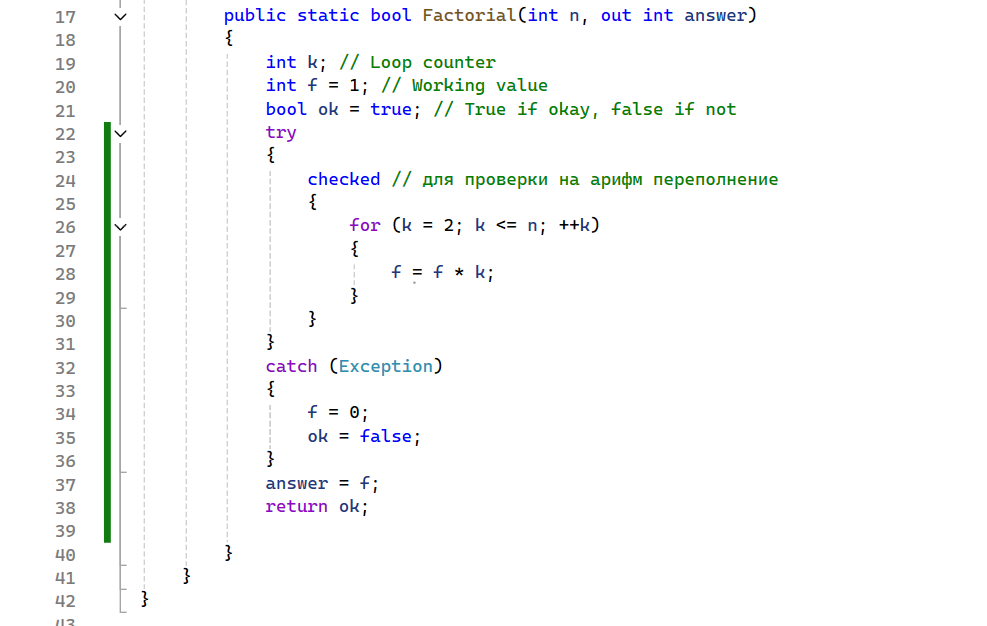


Рисунок 6 – Код метода Factorial

Для работы измененного класса отредактируем класс Program:

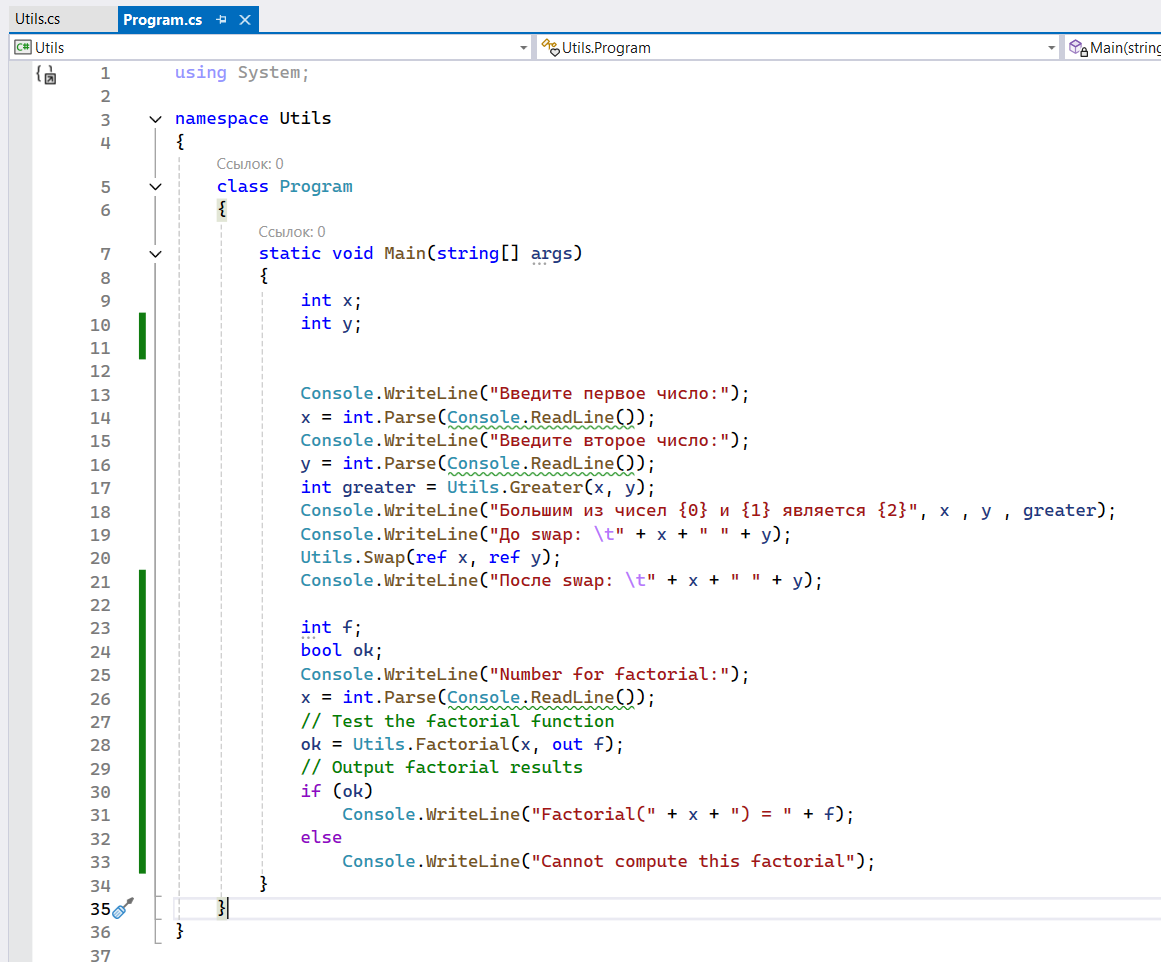


Рисунок 7 – Код класса Program

Протестируем приложение:

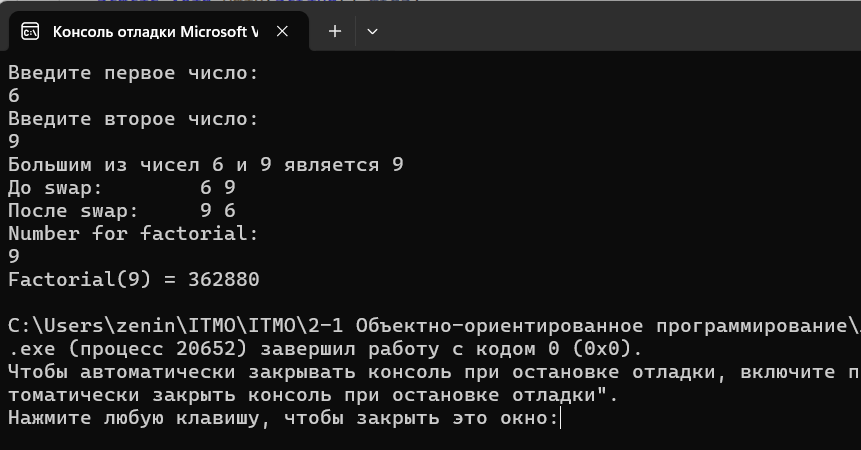


Рисунок 8 – Результат вывода программы

Программа работает верно.

# Упражнение 4

В этом упражнении требуется создать класс Operation, который будет вычислять площадь как равностороннего треугольника, так и обычного(используя перегрузку).

Чтобы использовать перегрузку без копирования кода, создадим отдельный класс:

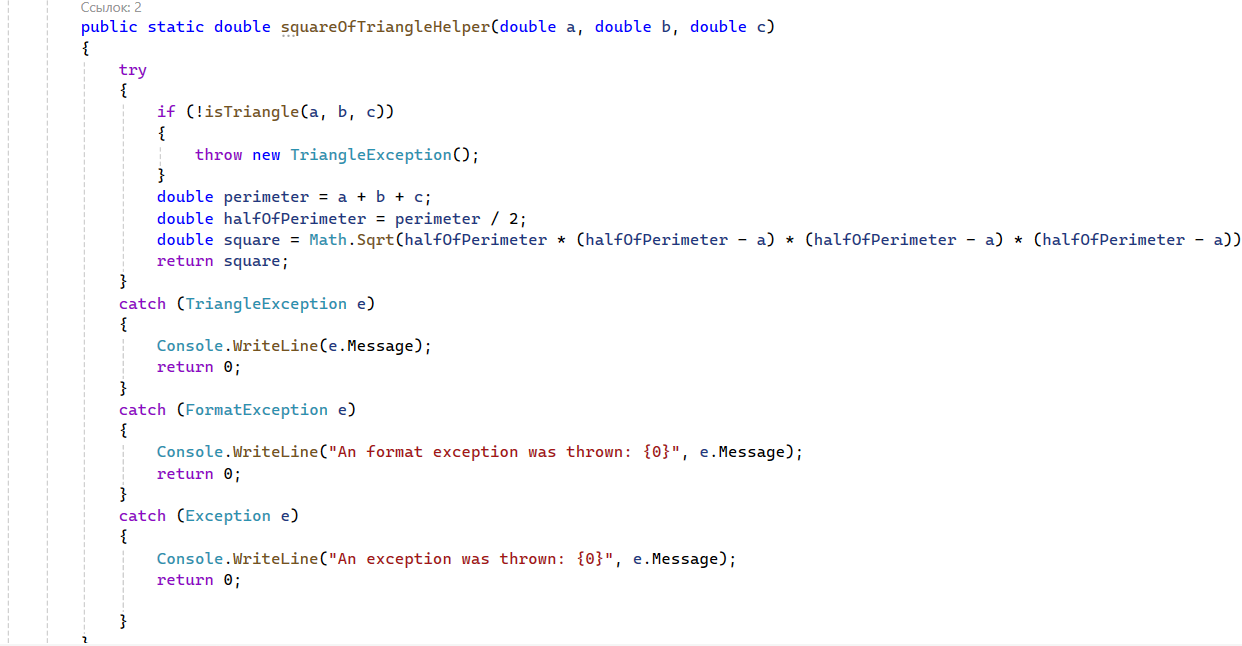


Рисунок 9 – Код класса squareOfTriangleHelper

Данный класс обрабатывает всевозможные ошибки, в том числе и то, что треугольник с заданными сторонами может не существовать.

Далее представлен код метода проверки треугольника на существование и код класса ошибки обработки существования.

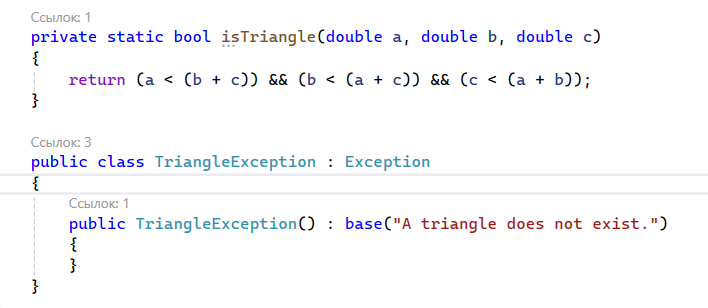


Рисунок 10 – Код метода isTriangle() и класса TriangleException

Реализуем перегрузку метода:

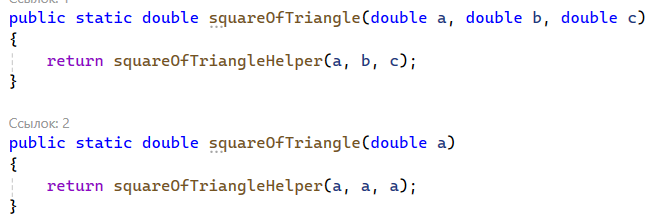


Рисунок 11 – Перегрузка метода squareOfTriangleHelper()

Теперь представим код класса Program:

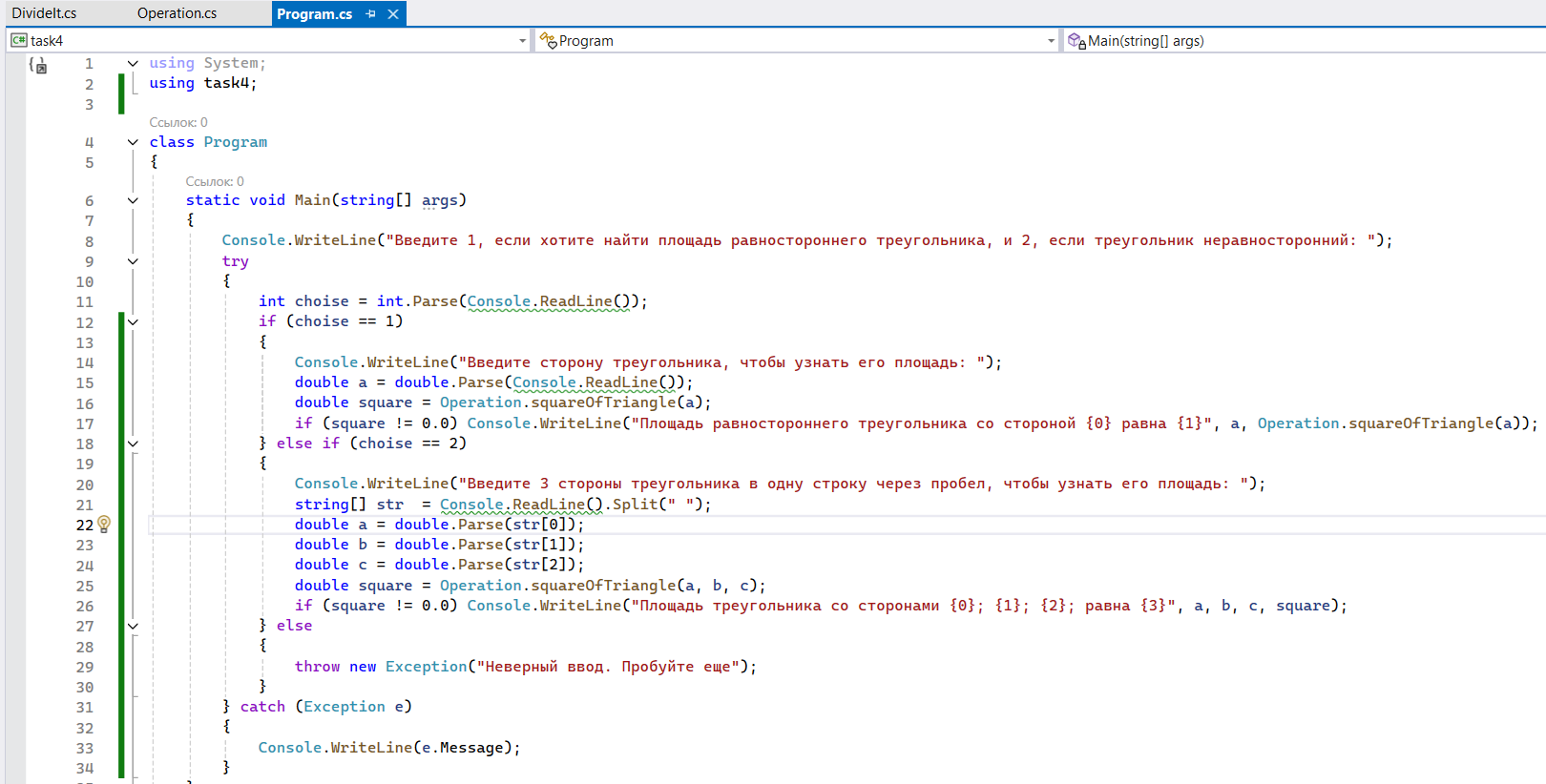


Рисунок 12 – Код класса Program

Программа обрабатывает всевозможные ошибки и правильно выдает ответы.

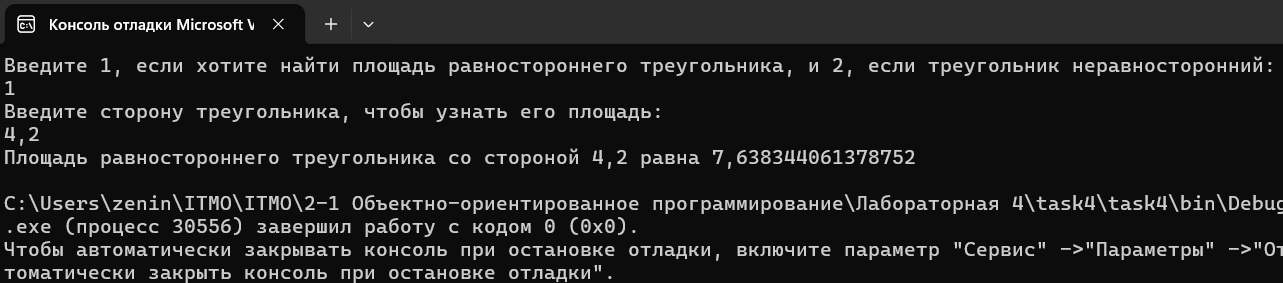


Рисунок 13 – Вывод программы для нахождения площади равностороннего треугольника

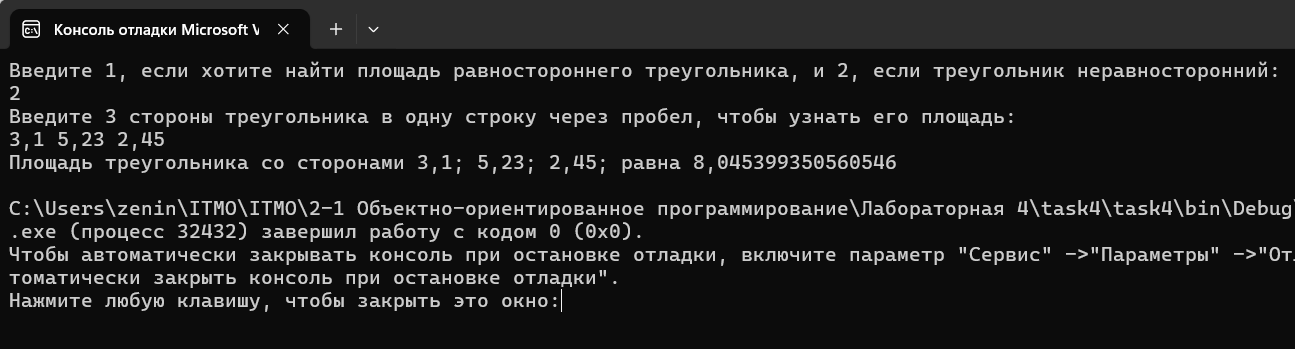


Рисунок 14 – Вывод программы для нахождения площади неравностороннего треугольника

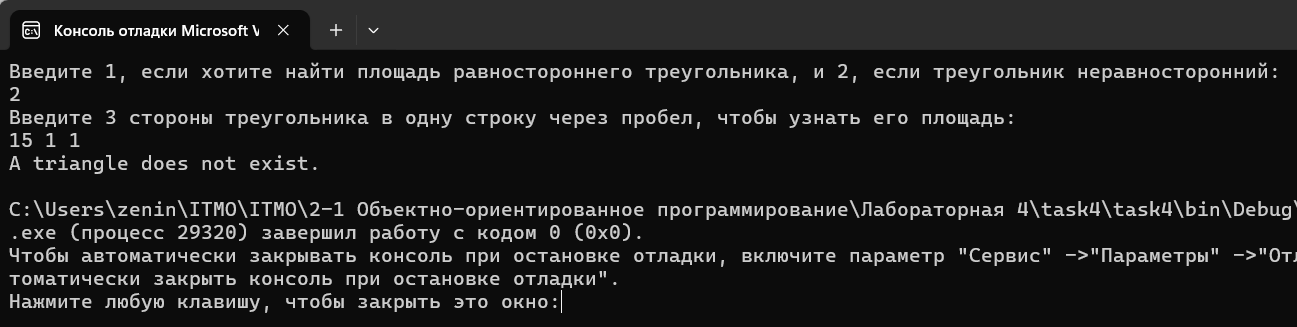


Рисунок 15 – Обработка несуществующего треугольника

Программа работает исправно и обрабатывает все ошибки.

# Упражнение 5

# В этом упражнении требуется реализовать метод вычисления корней квадратного уравнения.

# Напишем код класса QuadraticEquation

# 

Рисунок 17 – Код класса Program

Запустим программу:

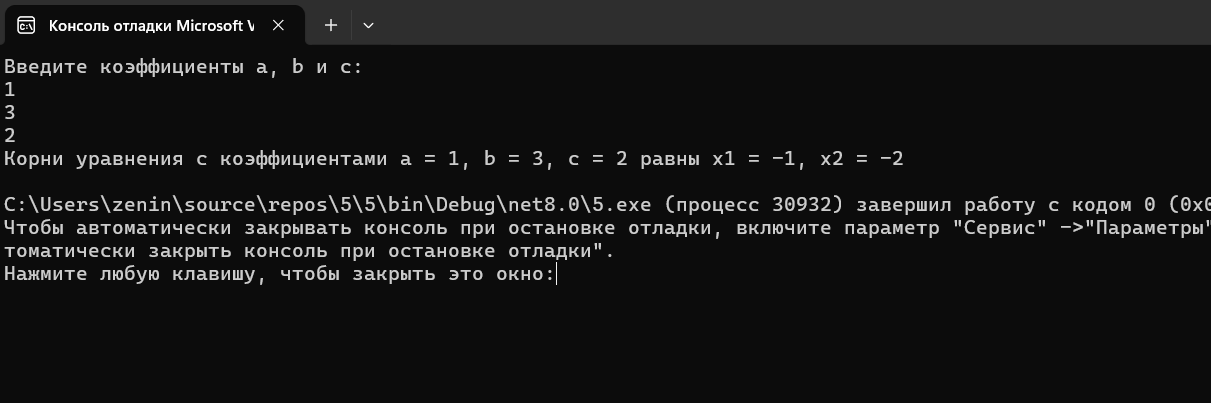


Рисунок 18 – Результат работы программы

Программа работает верно и ошибок не выдает.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы №4 были получены навыки работы с языком программирования С#