Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №1

на тему:

**«ПРОСТОЕ КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ»**

БГУИР 6-05-0612-02 34

|  |
| --- |
| Выполнил студент группы 353505  ХОРОШКО Кирилл Николаевич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание:**

1 Создать консольное приложение. Название проекта: \_НомерГруппы\_Фамилия.

2 Ознакомиться со структурой проекта.

3 Найти файл \*.csproj. Найдите в нем указание на целевую платформу и тип приложения.

4 Добавьте в проект NuGet пакет AutoMapper. Найдите ссылку на установленный пакет в файле \*.csproj.

5 Написать программу, которая выводит в консоль частное от деления чисел, введенных с клавиатуры. Введенные данные и результат сохранять в отдельных переменных.

6 Запустить проект в двух режимах: с отладкой и без отладки. Ввести сначала корректные, а затем заведомо неправильные значения чисел (строка вместо числа, ноль в качестве делителя). Сравнить результат в режиме отладки и выпуска.

7 Найти скомпилированные файлы (для режима отладки и режима выпуска)

8 Поставить точки останова. Запустить проект в двух режимах: с отладкой и без отладки. Сравнить результат. Прочитать в отладчике значения переменных. Провести пошаговое выполнение программы.

# 2 Выполнение работы

В visual studio был создан проект для написания кода на языке программирования C# (см. рисунок 1-2). Далее прошло ознакомление со структурой проекта (см. рисунок 3-4). Этот проект включал в себя файл с разрешением **.csproj** (см. рисунок 5)**,** в котором были указаны целевая платформа и типприложения (см. рисунок 6). После подключения внешних зависимостей в этом файле они будут отображаться. Нужно открыть папку Lb\_C#, в ней находится проект. Сразу же можно заметить файл с расширением \*.csproj. Нужно открыть файл и там сразу же можем заметить указание на целевую платформу и тип приложения.

Через вкладку проект обратились к функции “Управление пакетами NuGEt” (см. рисунок 7). Добавили через NuGet пакет AutoMapper (см. рисунок 8). Открыв после файл \*.csproj можно заметить, что добавилась строка кода с указанием подключенного пакета и его версии (см. рисунок 9).

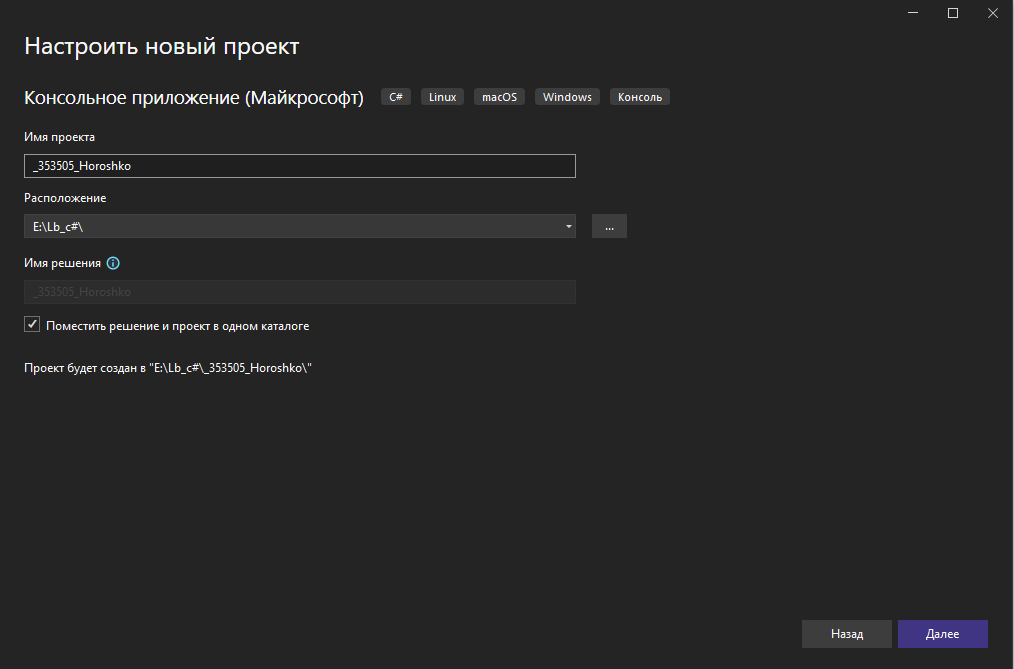


Рисунок 1 – Создание консольного приложения

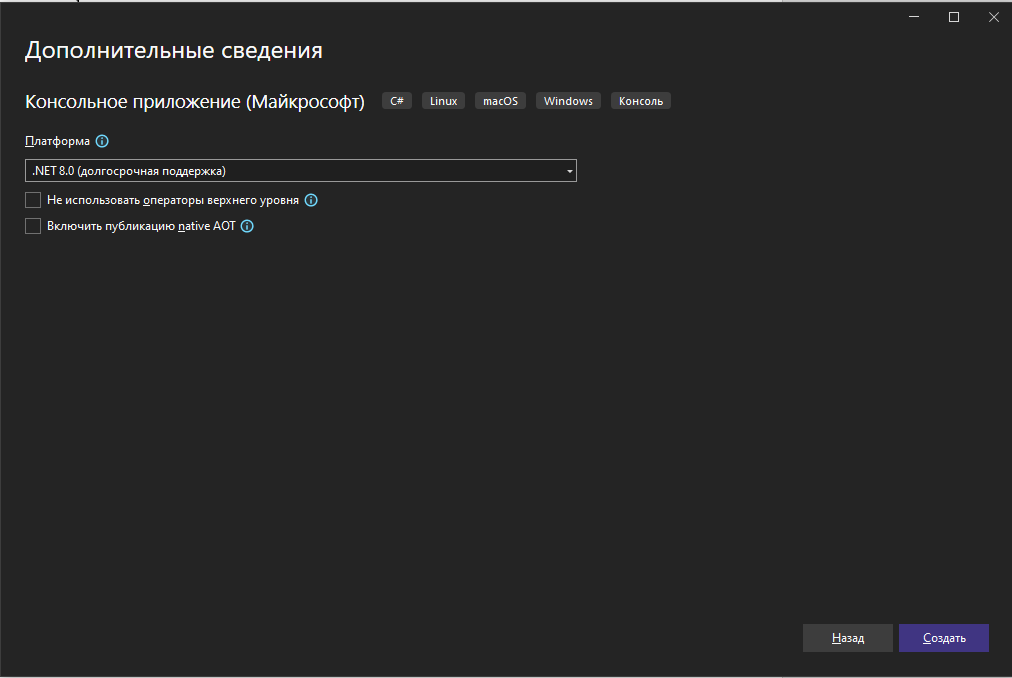


Рисунок 2 – Создание консольного приложения

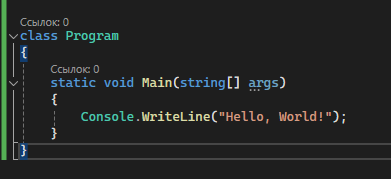


Рисунок 3 – Знакомство со структурой проекта

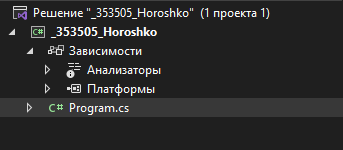


Рисунок 4 – Знакомство со структурой проекта

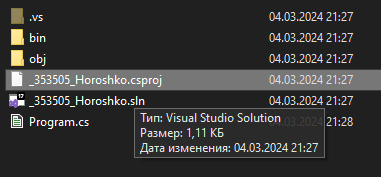


Рисунок 5 – Файл \*csproj

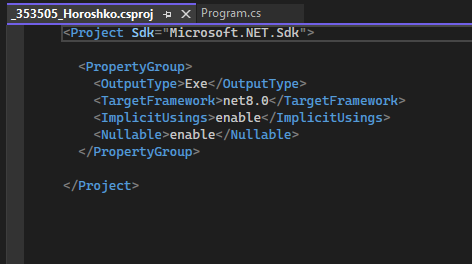


Рисунок 6 – Указания на целевую платформу и тип приложения

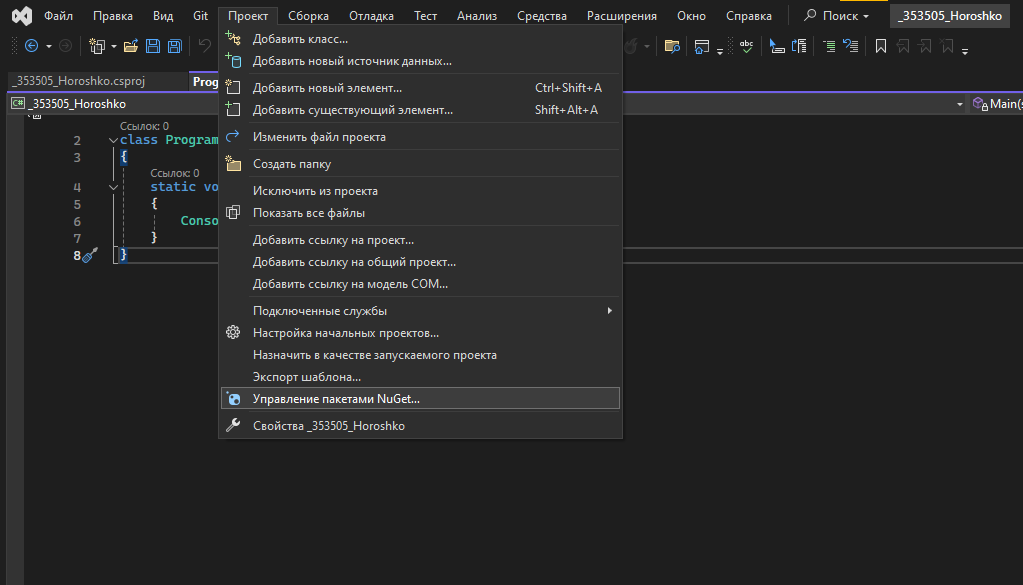


Рисунок 7 – Добавление пакета AutoMapper

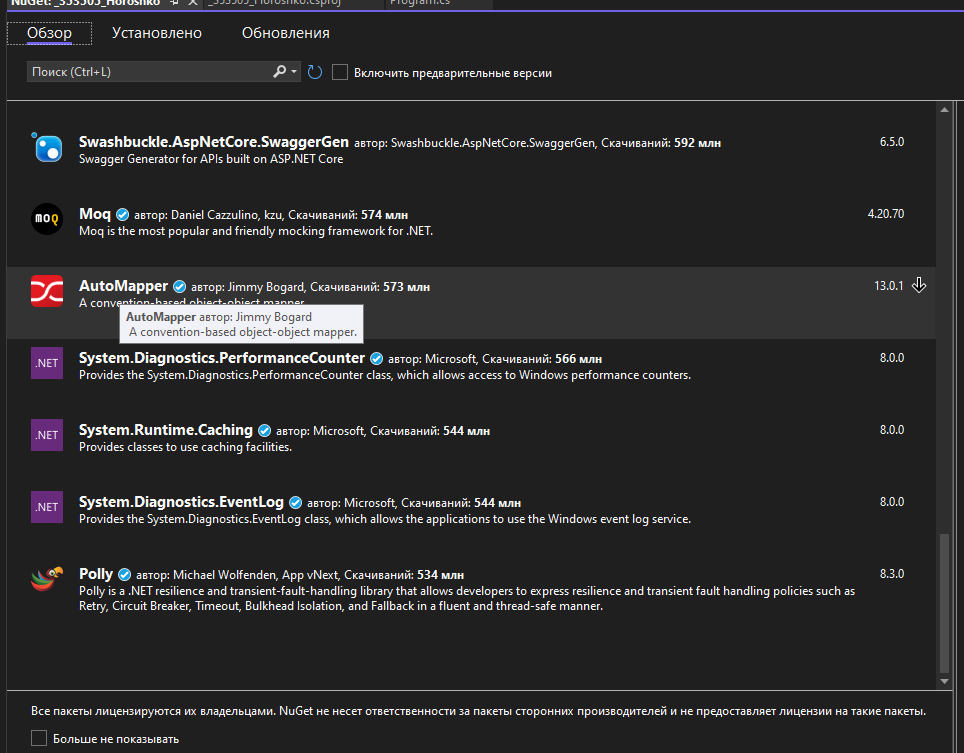


Рисунок 8 - Добавление пакета AutoMapper

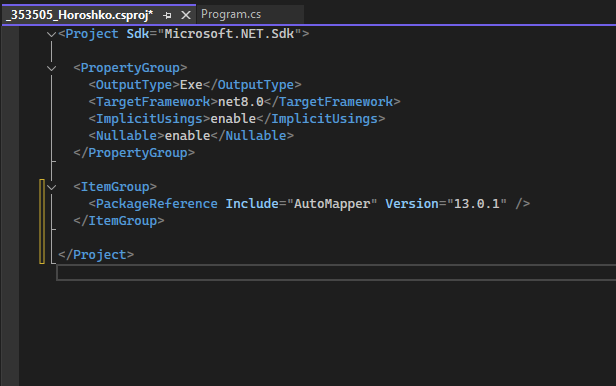


Рисунок 9 – Файл \*csproj после добавления AutoMapper

Далее была написана программа, которая выводит в консоль частное от

деления чисел, введенных с клавиатуры (см рисунок 10). Введенные данные и результат были сохранены в отдельных переменных. Имя переменной, которая являлась делимым, было “a”. Имя переменной, которая являлась делителем, было “b”. Результат сохранялся в переменной “result”. Все переменные имели тип данных double для нецелочисленных преобразований. Что бы считать строку введенную пользователем с клавиатуры, использовался метод ReadLine() класса Console. Далее, используя метод ToDouble() класса Convert, строка была преобразована в тип данных double. Вывод подсказок пользователю, а так же значения переменной result = a/b, использовался метод Write() или WriteLine() класса Console. Весь этот фрагмент кода являлся телом метода static void Main(string[] args), который является точкой входа в программу, а сам метод принадлежит классу Program.

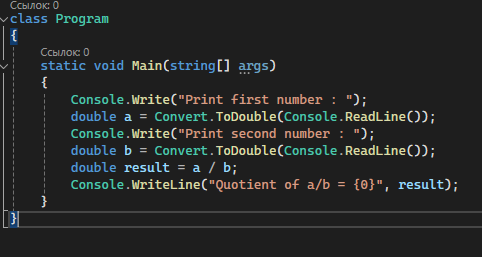


Рисунок 10 – Код программы

Программа была запущена в режиме с откладкой (см. рисунок 11) и в режиме выпуска (см. рисунок 12). Сразу были введены корректные, а позже некорректные значения, такие как 0 или символ(строка). Результат отличался в режиме откладки (см. рисунок 13) от режима выпуска (см. рисунок 14). Это связано с тем, что режим откладки предназначен для поиска ошибок в программе, поэтому, как мы можем видеть на рисунках ниже, после ввода некорректного значения режим откладки указал на ошибку и вернул ее код(см. рисунок 15), в то время как в режиме выпуска программа аварийно завершилась и указала на метод, который не выполнился (см. рисунок 16).

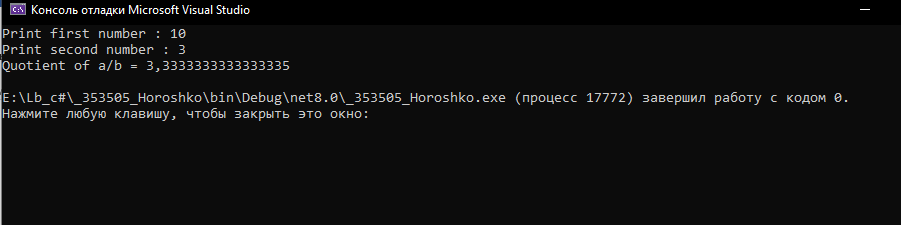


Рисунок 11 – Ввод корректных значений в режиме с откладкой

# 

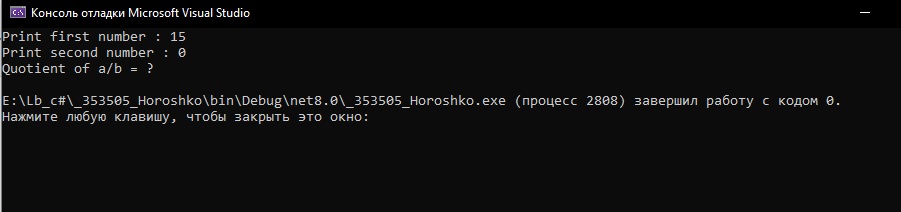


Рисунок 12 – Ввод корректных значений в режиме выпуска

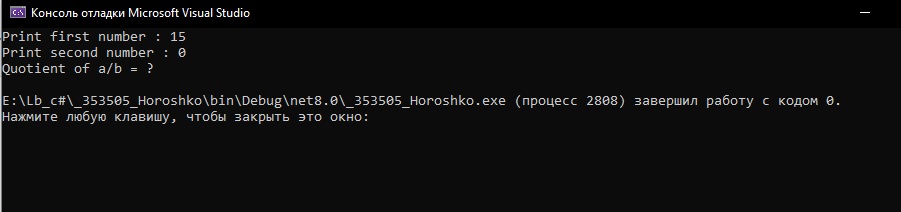


Рисунок 13 – Ввод некорректных значений в режиме с откладкой

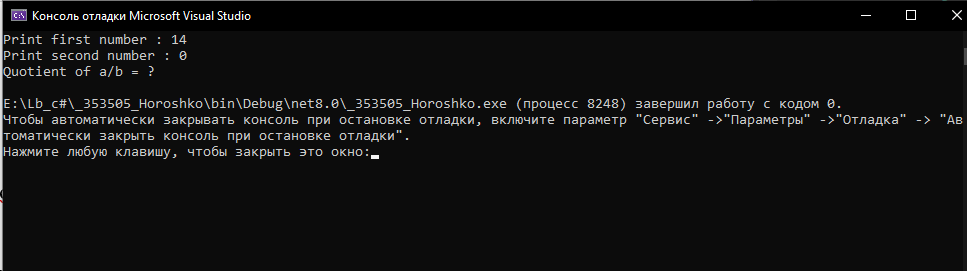


Рисунок 14 – Ввод некорректных значений в режиме выпуска

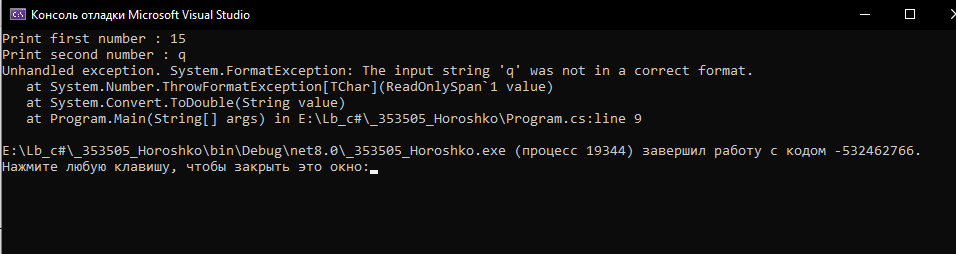


Рисунок 15 – Ввод некорректных значений в режиме с откладкой

# 

Рисунок 16 – Ввод некорректных значений в режиме выпуска

Далее открываем папку с проектом и находим в ней папку bin в этой папке обычно хранятся еще две папки Debug и Release, так и было в моем случае. В папке Debug находятся скомпилированные файлы для режима откладки (см. рисунок 17). В папке Release находятся скомпилированные файлы для режима выпуска (см. рисунок 18). Открываем папки для просмотра содержимого, можем заметить, что у нас есть 2 файла с расширением .exe.

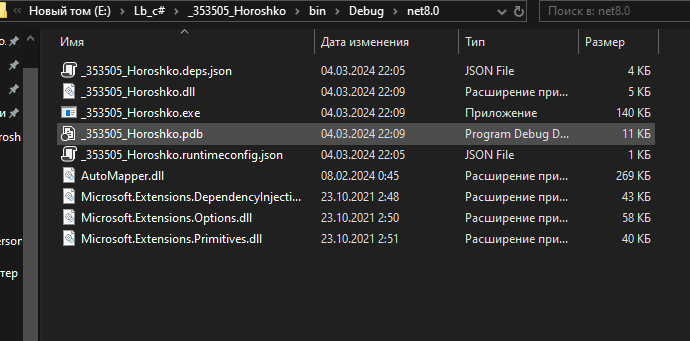


Рисунок 17 – Скомпилированные файлы для режима откладки

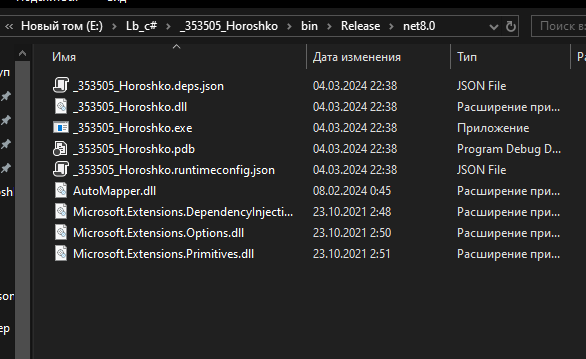


Рисунок 18 – Скомпилированные файлы для режима выпуска

Далее в программе расставили точки останова для пошагового выполнения программы (см. рисунок 19). Запустили в режиме с откладкой (см. рисунок 21) и без (см. рисунок 20). Заметили что в режиме без откладки точки останова были проигнорированы.

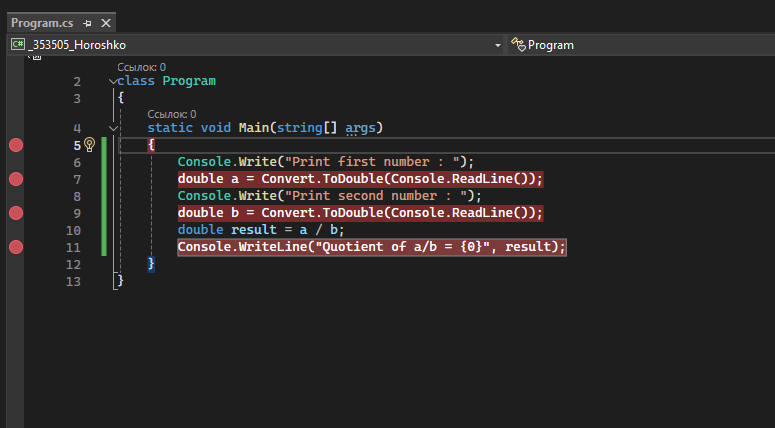


Рисунок 19 – Точки останова

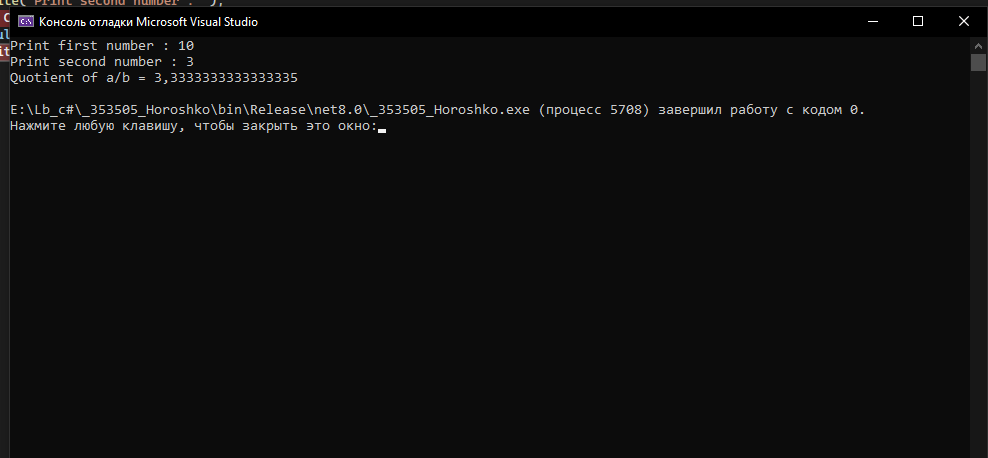


Рисунок 20 – Программа запущена в режиме без откладки

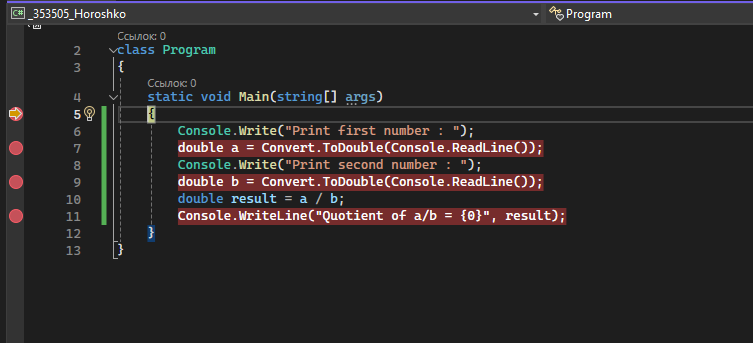


Рисунок 21 – Программа запущена в режиме с откладкой

# Вывод

В ходе лабораторной работы были изучены способы выполнения программы, а также способы ее компиляции. Были изучены некоторые методы для работы с классом Console, такие как Write, WriteLine, Read, ReadLine. Так же познакомились с типами данных в языке C#. Было изучено приведение типов с использованием методов класса Convert, таких как ToDouble.