Министерство науки и высшего образования РФ

ФГБОУ ВПО «Пермский государственный национальный исследовательский университет»

Кафедра прикладной математики и информатики

**Индивидуальное задание №1**

**«Работа с массивами»**

***по дисциплине «Алгоритмизация и программирование»***

|  |  |
| --- | --- |
|  | **Выполнил:** |
|  | Студент 1-го курса механико-математического ф-та  \_\_\_\_\_\_\_\_\_\_\_ Белоусов А.А.,    группа ПМИ-3,4. |
|  | **Проверил:** |
|  | Доцент кафедры прикладной математики и информатики ПГУ, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Перескокова  О.И. |
|  | «16» декабря 2020 г. |

Пермь 2020

**Постановка задачи**

Даны номер автобусного билета – шестизначное натуральное число и целое неотрицательное число N. Между некоторыми парами соседних цифр разрешается ставить знаки арифметических операций (+, -, \*) так, чтобы результат полученного арифметического выражения стал равен N.

Требуется:

1) определить количество различных решений для заданного номера билета и N;

2) вывести все решения на экран.

Замечания

● Разрешается предполагать, что число различных решений не превосходит число

строк на экране

● Разрешается пренебречь грамматической согласованностью ответа.

● Значение выражения вычисляется с учетом приоритета заданных

арифметических операций.

Техническое требование:

● Программа должна проверять корректность ввода шестизначного числа

**Определение идеи алгоритма,**

**выбор методов решения и структур данных.**

Для решения задачи была выбрана система программирования Microsoft Visual Studio.

Идея алгоритма:

Для решения данной задачи была использована польская запись.

1. Проверим корректность всех введеных данных и если, есть проблемы программа завершится и сообщит и проблеме.
2. Разобьем число на три части – левую, среднюю и правую. Создавая новый вариант постановки знаков, в функции perebor будет вызываться функция new\_part, которая собирает новую левую и среднюю части, а запустит рекурсию функции perebor, с новой левой частью, а новая правая часть = новая средняя + «?» + правая.
3. В функции perebor запускается функция zamena, которая последовательно будет заменять все знаки вопроса в строке, на знаки операции. ТО есть сначала заменим первый знак вопроса на первый знак, затем второй заменим на первый знак. А последний будем менять и проверять подходит ли нам эта комбинация. Далее заменится предпоследний знак на следующий и опять последний знак будет меняться и проверятся возможность такой комбинации.
4. В функции zamena запустим функцию Calculate, которая с помощью стэка будем высчитывать результат получившейся комбинации и сравнивать с результатом, который нам нужен. Мы проходим по строке, когда видим число, кладем его в стэк, если мы нашли знак, то достаем два числа из стэка, выполняем операцию и кладем в стэк получившийся результат. Выполняем, пока не пройдем строку и в стэке не останется одно число, которое является результатом выполнения данного случая операций.
5. Если Данный случай расстановки знаков нам подходит, нужно проверить, а действительно ли этот случай нам подходит, потому что польская запись – это запись без постановки скобок. То есть, если перевести обратно в инфиксную запись, то результат без скобок может оказаться другим. Перевод из постфиксной записи в инфиксную следующий: Используем стэк типа string. Собираем из строки число. Кладем его в стэк, если нашли знак, то достаем из стэка два числа и кладем в стэк «первое число + знак + второе число». Делаем так, пока строка не закончится, тогда нужная нам инфиксная запись находится в топе стэка. Достаем строку.
6. Проверяем возможность такого случая в инфиксной записи. Посчитаем количество операций, которые нужно выполнить. Сначала сделаем умножение, потому что у этой операции приоритет выполнения выше. Проходя по строке, если найдем знак, действия следующие : Через два вайла будем увеличивать переменные, чтобы узнать скольки - циферные числа стоят слева и справа от знака. Переведем собранные числа из string в int. Произведем операцию умножения над ними. Вырежем из строки первое число, знак и второе число и вставим на это место получившееся значение. Уменьшим количество оставшихся операций на один. Далее выполним те же самые действия, что и с умножением, с сложением и вычитаем и как только количество операций станет равно 0, в строке останется только результат. Переведем получившийся результат в int, сверим его с нужным нам результатом, то теперь осталось последнее действие.
7. Нужно проверить, не было ли уже такой же комбинации знаков. Для этого был создан массив строк, в котором будут храниться новые, уникальные случаи. Проходим по массиву и если такого случая не было. Кладем его в новую ячейку массива. А также сам случай выводим на экран в формате – случай = результат.

**Тестирование программы**

Проведя анализ возможных данных, было выделено 4 класса входных данных, для каждого из них ниже дано описание, представлен тест и результат прохождения данного теста программой.

1. Исходные данные корректны. Даны такие данные, что в результате должно быть 1023 случая. Это все возможные вариации расстановки знаков.

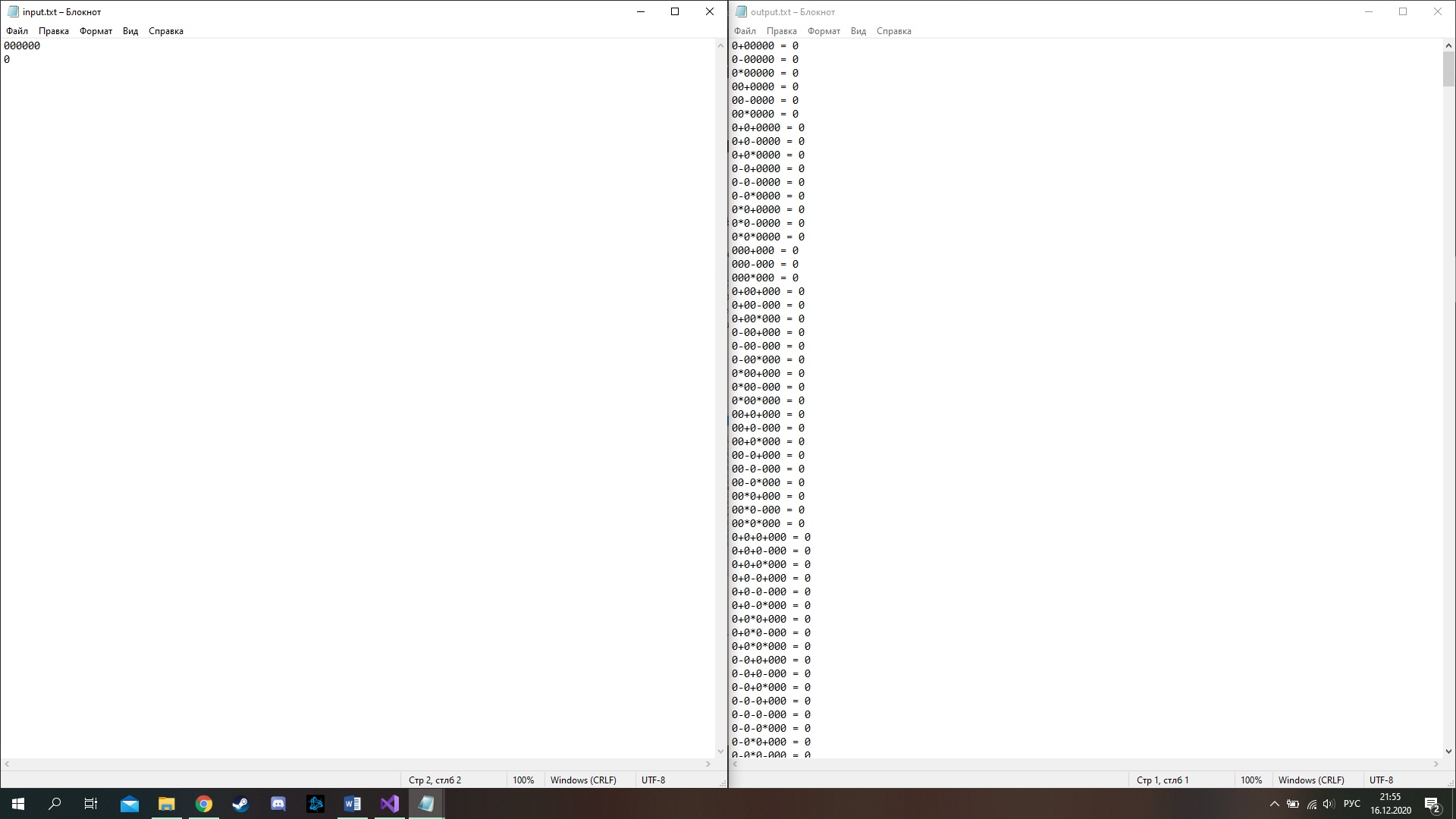
*входной файл (прилагается к отчету под именем input.txt)*

*000000*

*0*

*Ожидаемый результат*

*1023 варианта (все случаи постановки знаков)*



1. *Исходные данные корректны. Варианты возможны*

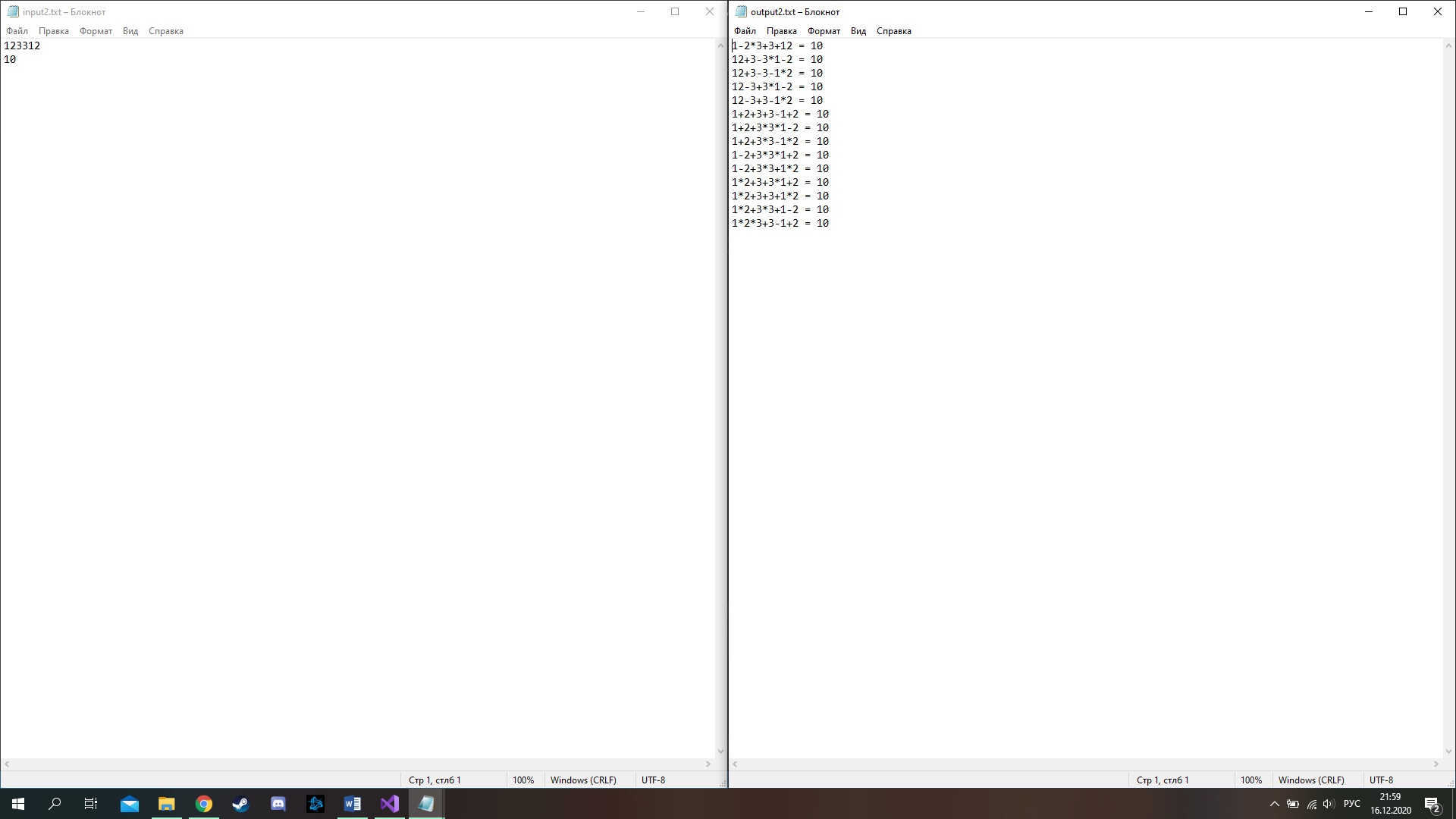
*входной файл (прилагается к отчету под именем input2.txt)*

*123312*

*10*

*Ожидаемый результат*

*Варианты есть*



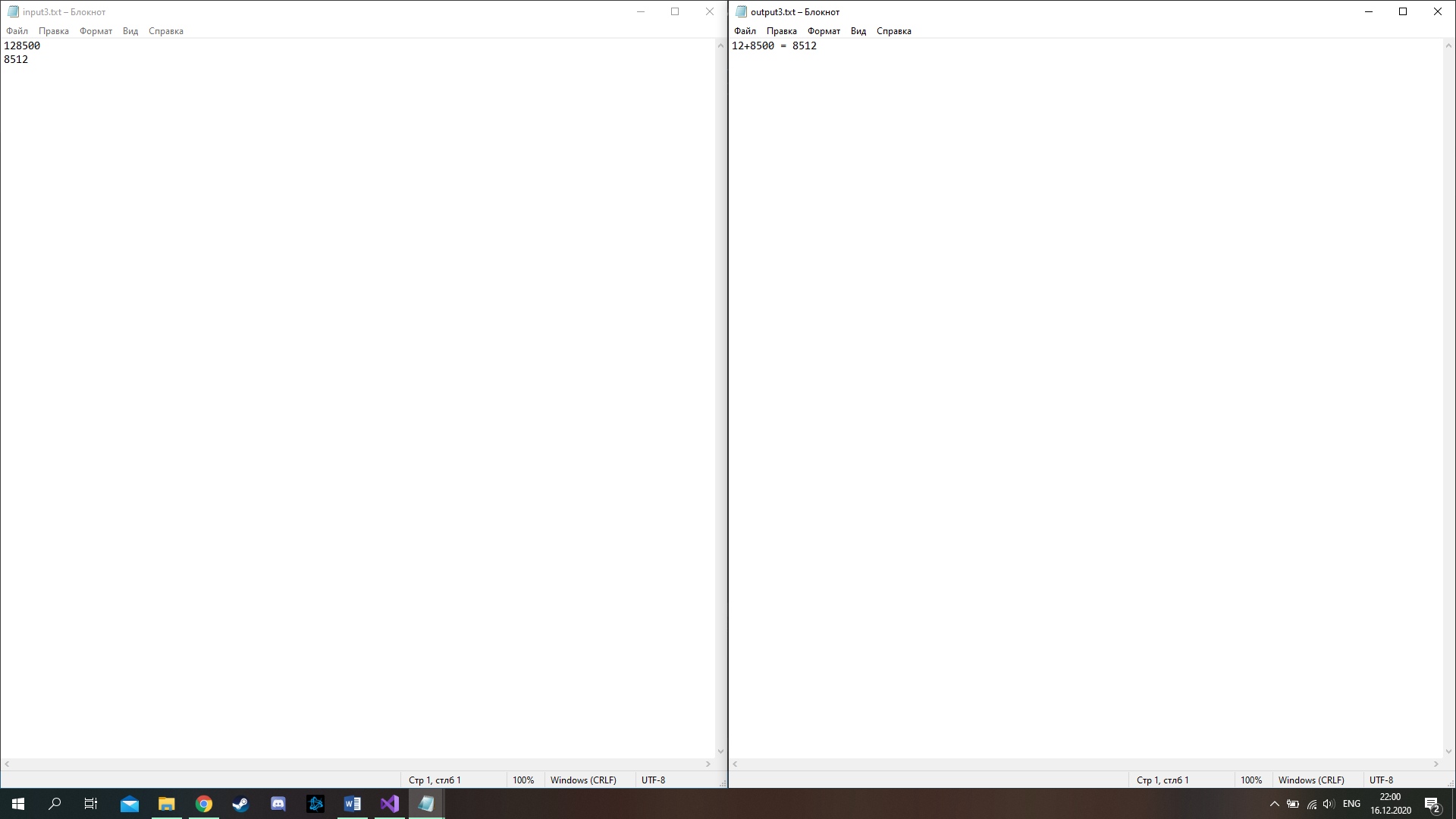
*входной файл (прилагается к отчету под именем input3.txt)*

*128500*

*8512*

*Ожидаемый результат*

*Варианты есть*



1. *Исходные данные корректны. Варианты невозможны*

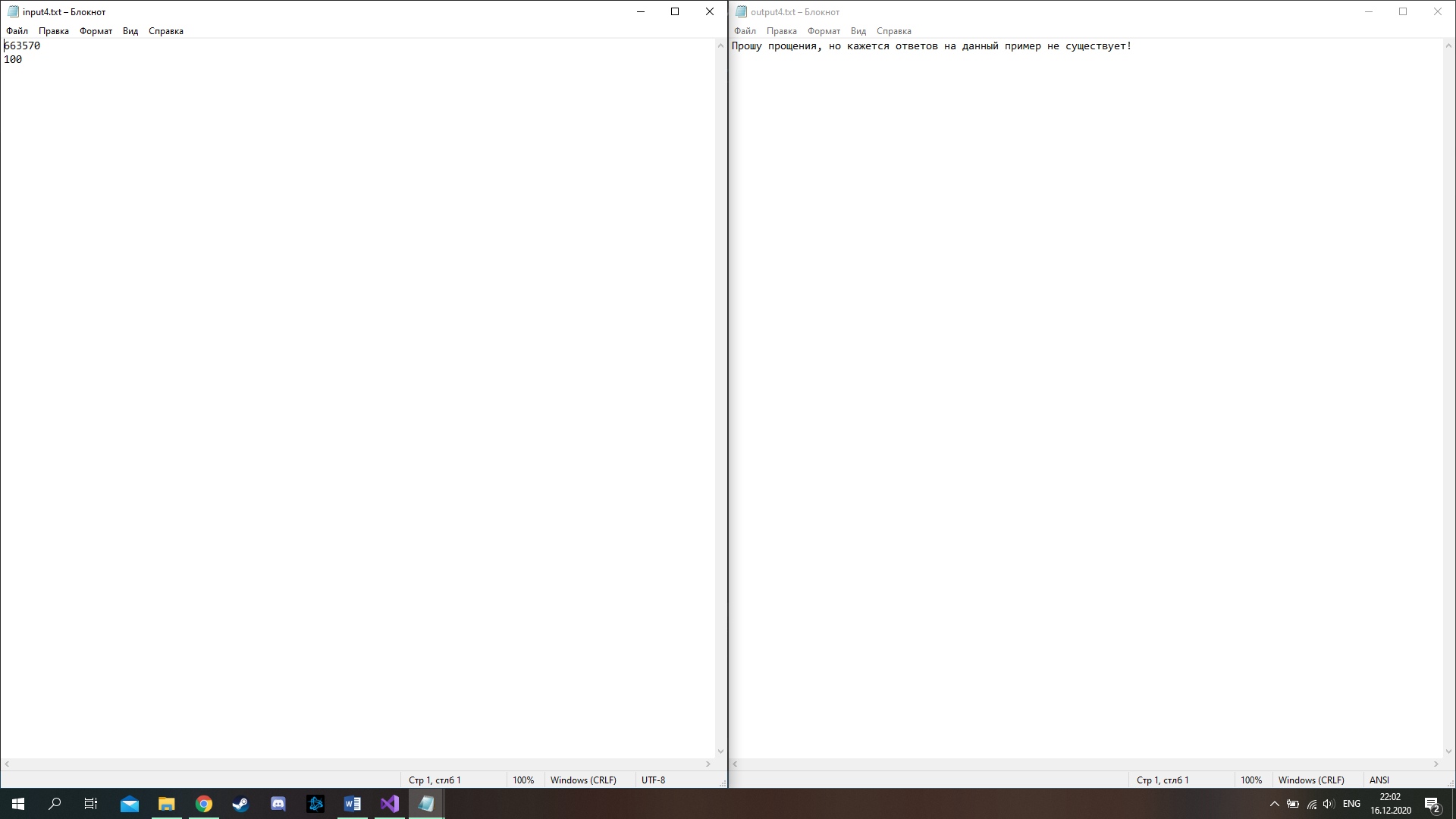
*входной файл (прилагается к отчету под именем input4.txt)*

*663570*

*100*

*Ожидаемый результат*

*Вариантов нет*



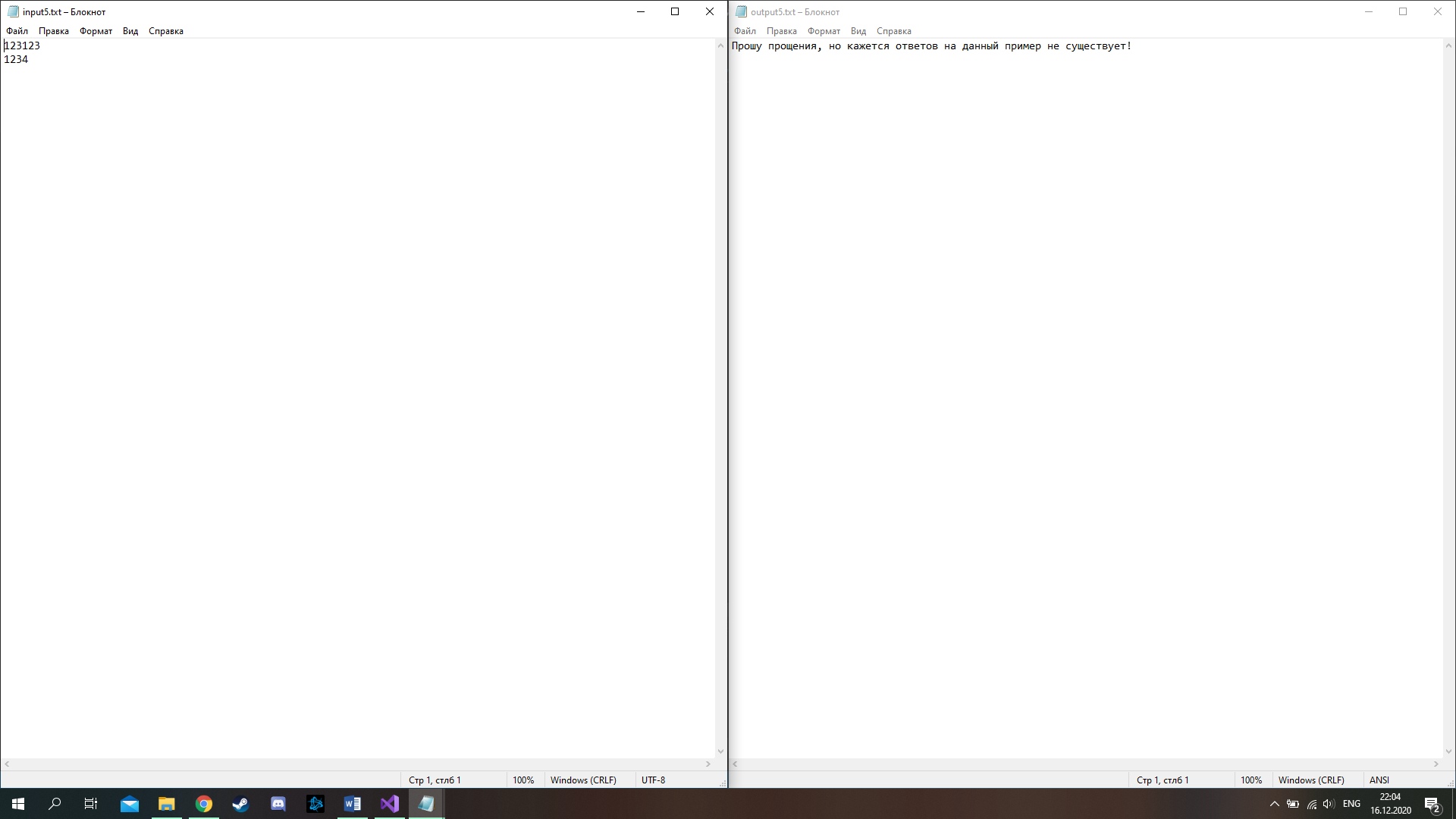
*входной файл (прилагается к отчету под именем input5.txt)*

*123123*

*1234*

*Ожидаемый результат*

*Вариантов нет*



1. *Исходные данные некорректны.*

