Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Реализация алгоритмов-сортировок»**

**Выполнила**:

Студентка группы 381903-2

Кабанова В.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_

одпись

Нижний Новгород  
2020

Содержание

[1 Постановка задачи 3](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470213)

[2 Методы Решения 4](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470214)

[3 Руководство пользователя 8](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470215)

[4 Описание программной реализации 10](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470216)

[5 Подтверждение корректности 11](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470217)

[6 Заключение 12](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470218)

[7 Приложение 13](file:///C:\Users\Сергей\Downloads\_._._.docx#_Toc59470219)

# Постановка задачи

Необходимо разработать программу, выполняющую вычисление арифметического выражения с вещественными числами.

Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны: +, -, /, \*.

Допускается наличие знака "-" в начале выражения или после открывающей скобки. Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки.

3

1. **Методы Решения**

Для решения данной задачи реализован шаблонный класс TStack, класс Split ,класс arithmetic.

Класс TStack используется для реализации методов в классе arithmetic.

Класс Split написан для разбиения заданного выражения на операции и

операнды (числа). Для операции разрешены +, -, \*, /, (, ). Если для операции или для числа написаны некорректные символы – выбрасывается исключение “ not correct operation position..” или “incorrect expression”

Класс arithmetic содержит методы Process,Prioritet,oper, calculate.

Метод Prioritet определяет приоритет операций.

Метод oper выполняет операции сложения, вычитания, деления и умножения.

Метод Process реализует перевод исходного выражения в польскую запись.

1. Создать пустой стек для хранения операторов. Создать пустой список для вывода.
2. Преобразовать инфиксную строку в список, используя строковый метод split.
3. Сканировать список слева направо.
   * Если символ является операндом, то добавить его в конец выходного списка.
   * Если символ является левой скобкой, положить его в стек.
   * Если символ является правой скобкой, то выталкивать элементы из стека пока не будет найдена соответствующая левая скобка. Каждый оператор добавлять в конец выходного списка.
   * Если символ является оператором \*, /, + или -, поместить его в стек. Однако, перед этим вытолкнуть любой из операторов, уже находящихся в стеке, если он имеет больший или равный приоритет, и добавить его в результирующий список.

Когда входное выражение будет полностью обработано, проверить стек. Любые операторы, всё ещё находящиеся в нём, следует вытолкнуть и добавить в конец итогового списка. Использует стек для целых чисел(int), в котором хранятся знаки операций.

Метод calculat вычисление выражений по обратной польской записи. Стек из double, для хранения промежуточных значений

1. Обработка входного символа

Если на вход подан операнд, он помещается на вершину стека.

Если на вход подан знак операции, то соответствующая операция выполняется над требуемым количеством значений, извлечённых из стека, взятых в порядке добавления. Результат выполненной операции кладётся на вершину стека.

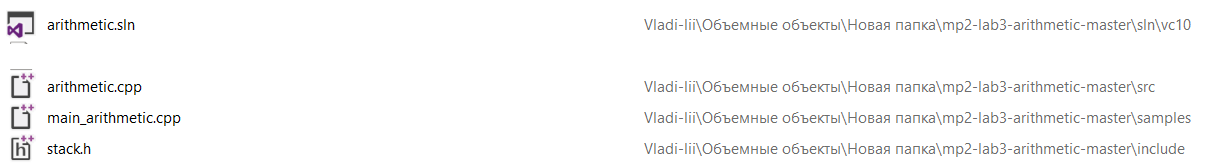
2. Если входной набор символов обработан не полностью, перейти к шагу 1.

3. После полной обработки входного набора символов результат вычисления выражения лежит на вершине стека

Для получения результата необходимо воспользоваться методом GetResult.

# Руководство пользователя

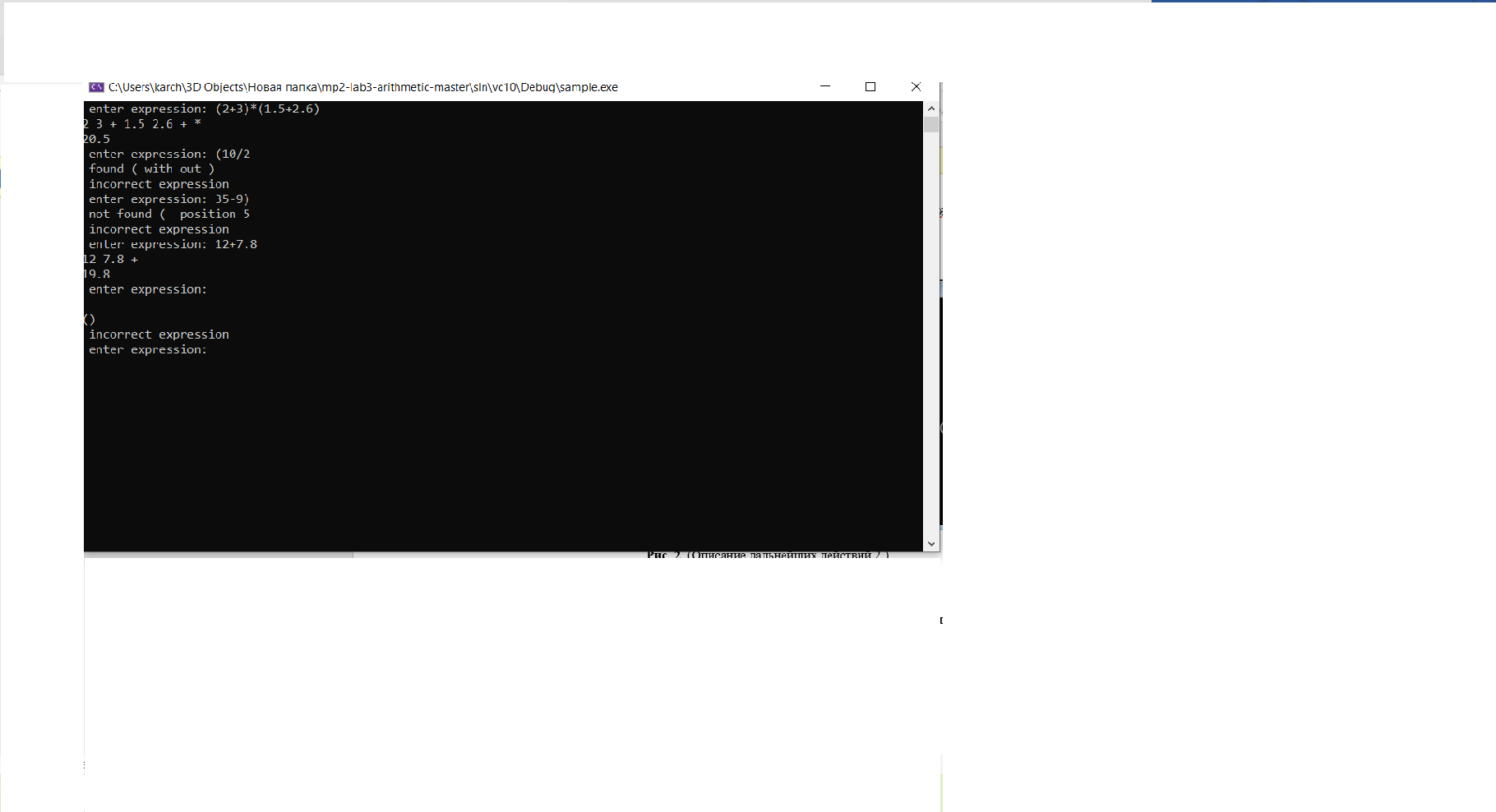
Для использования программы необходимо открыть файл arithmetic.sln (рис.1).

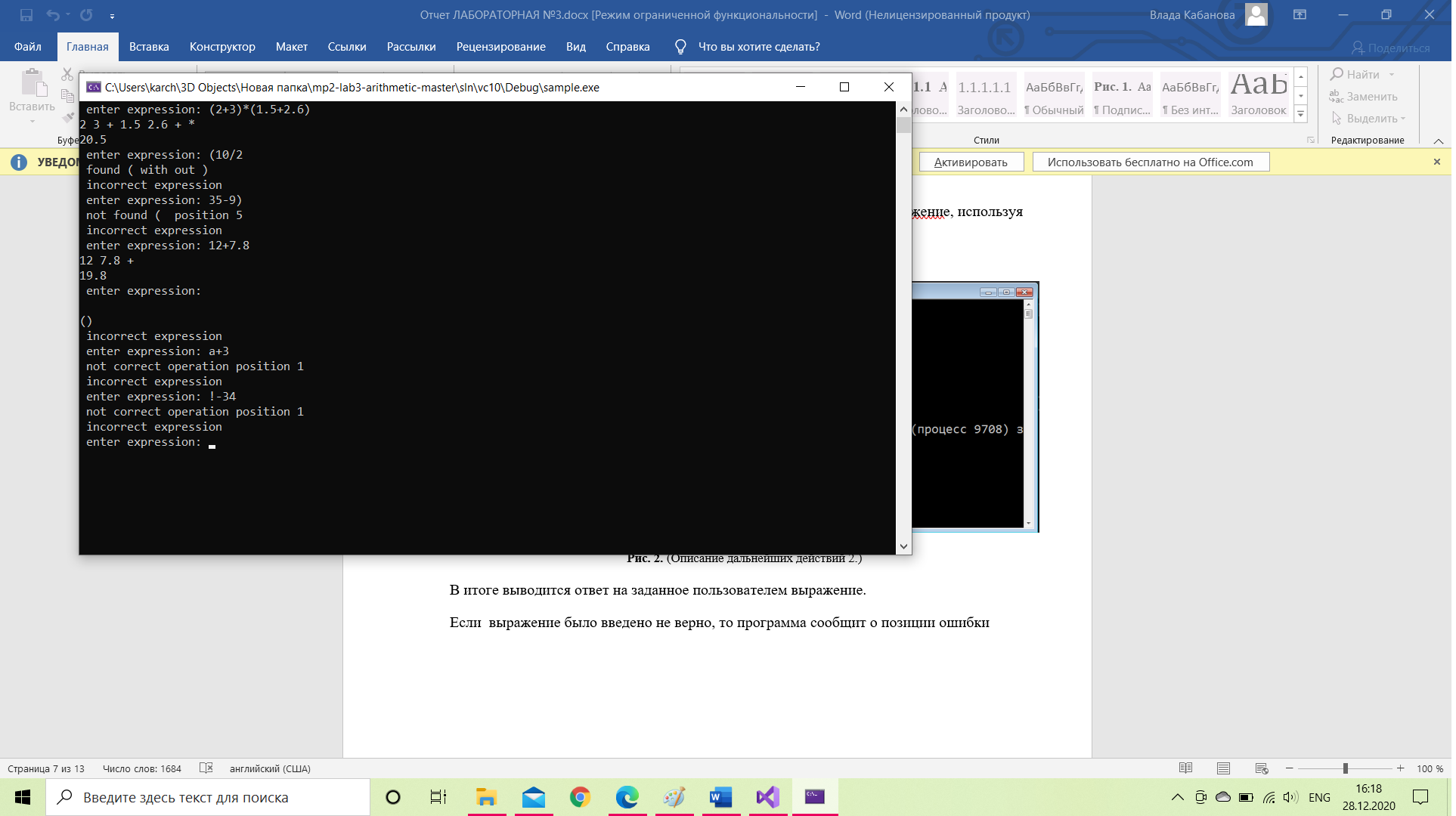


1. (Скриншот нахождения программы “arithmetic.sln”)

При открытии программы откроется консоль. Необходимо задать выражениеe, используя цифры 0,…,1,десятичную точку, операции +, -, \*, /, (, ).

Пусть задано выражение: (2 + 3)\* (1.5 + 2.6) или 12+7.8 (рис.2):

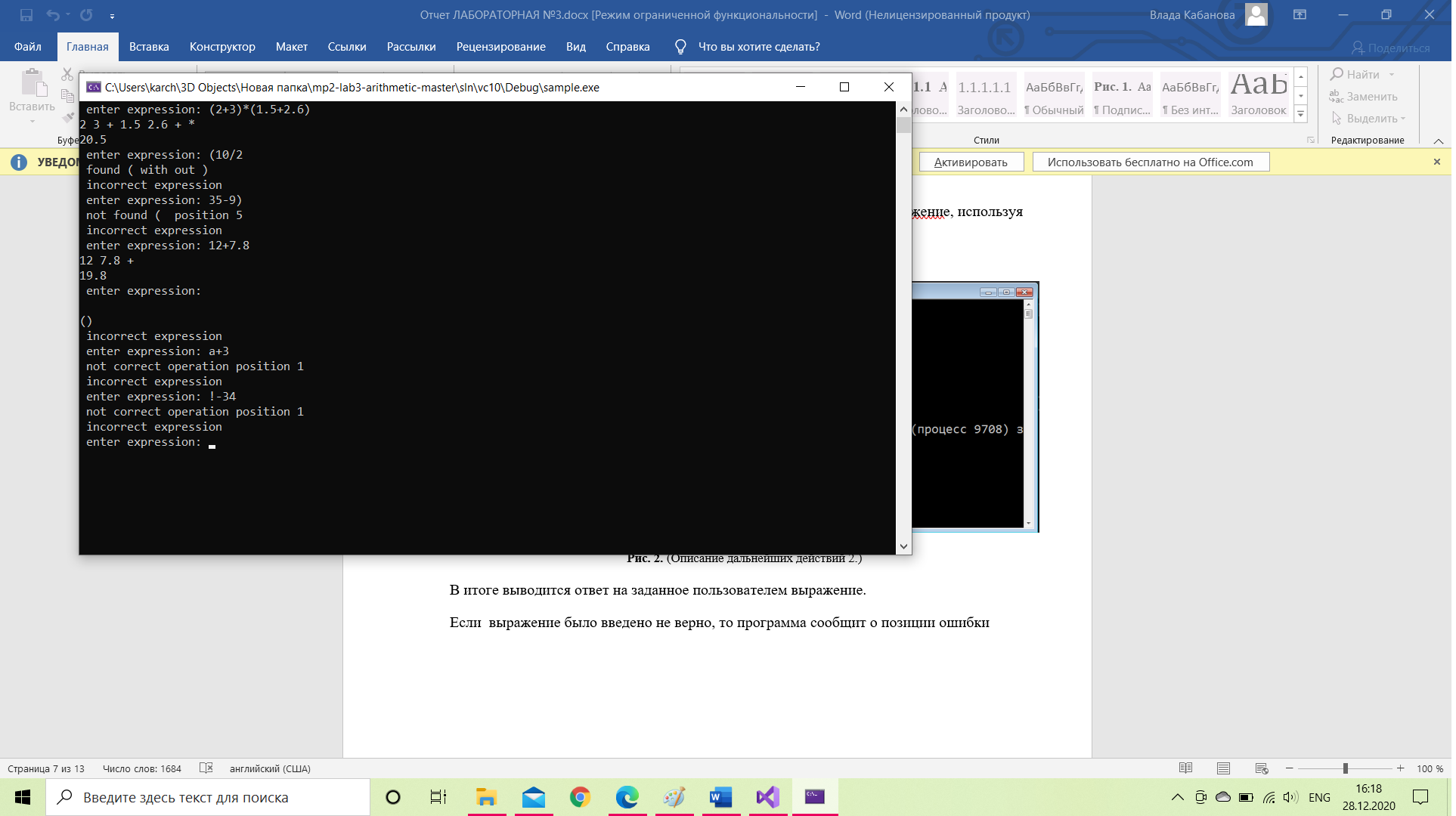




1. (Описание дальнейших действий .)

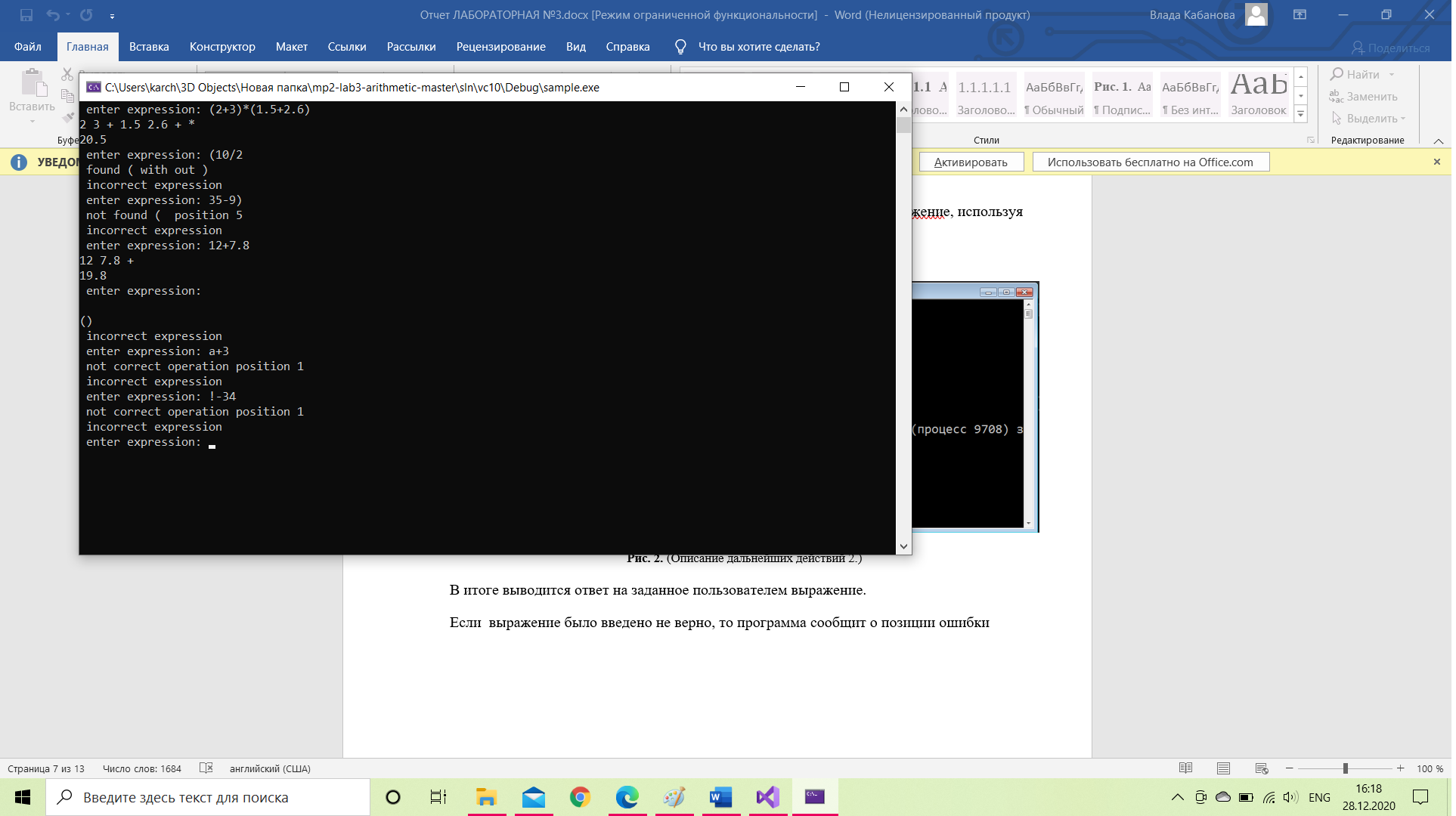
В итоге выводится ответ на заданное пользователем выражение.

Если выражение было введено не верно, то программа сообщит о позиции ошибки(рис.3):



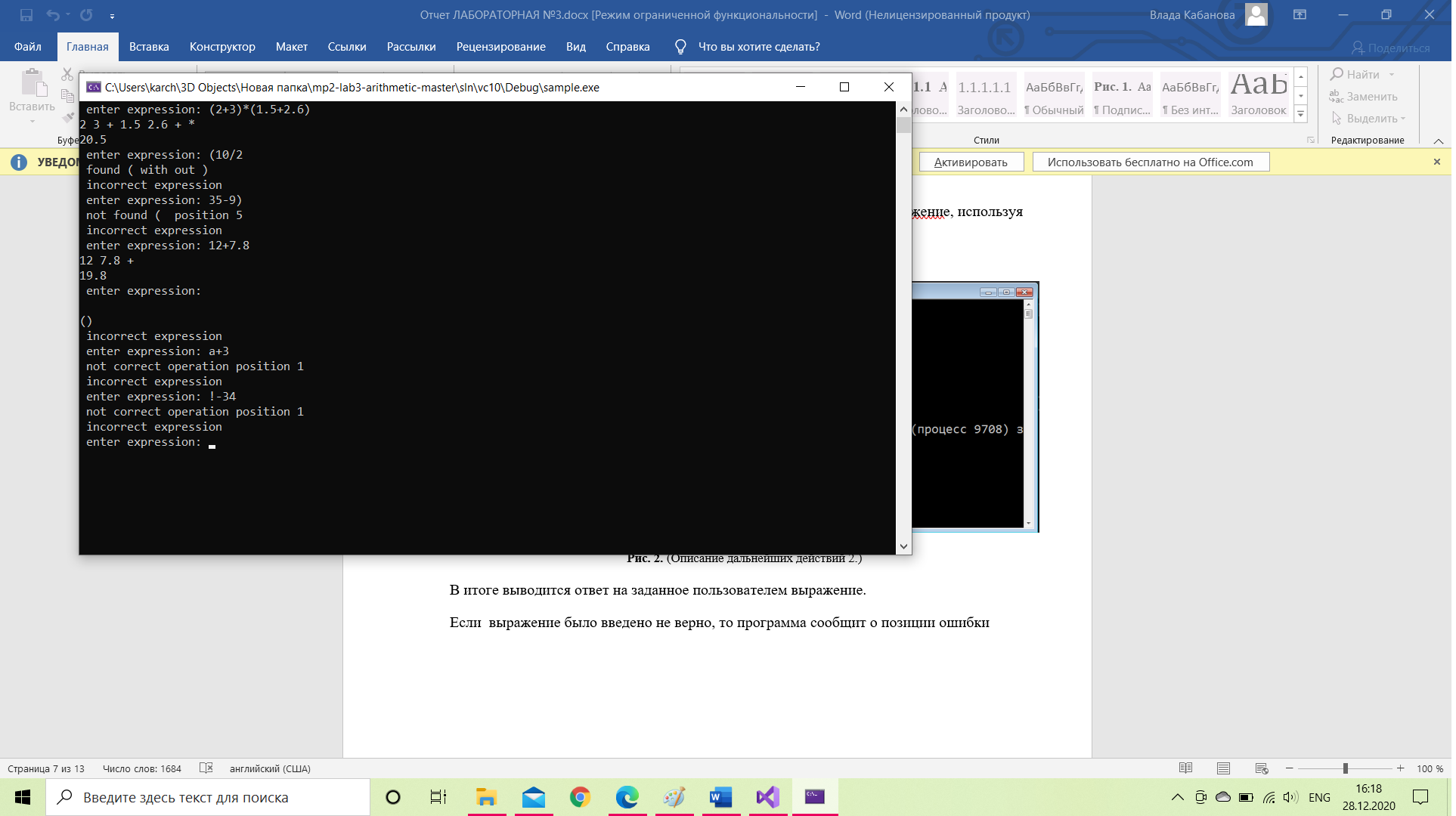
1. (Описание дальнейших действий .)

Если в выражении не хватает открывающейся или закрывающейся скобки, то программа также сообщит об ошибке (рис.4):



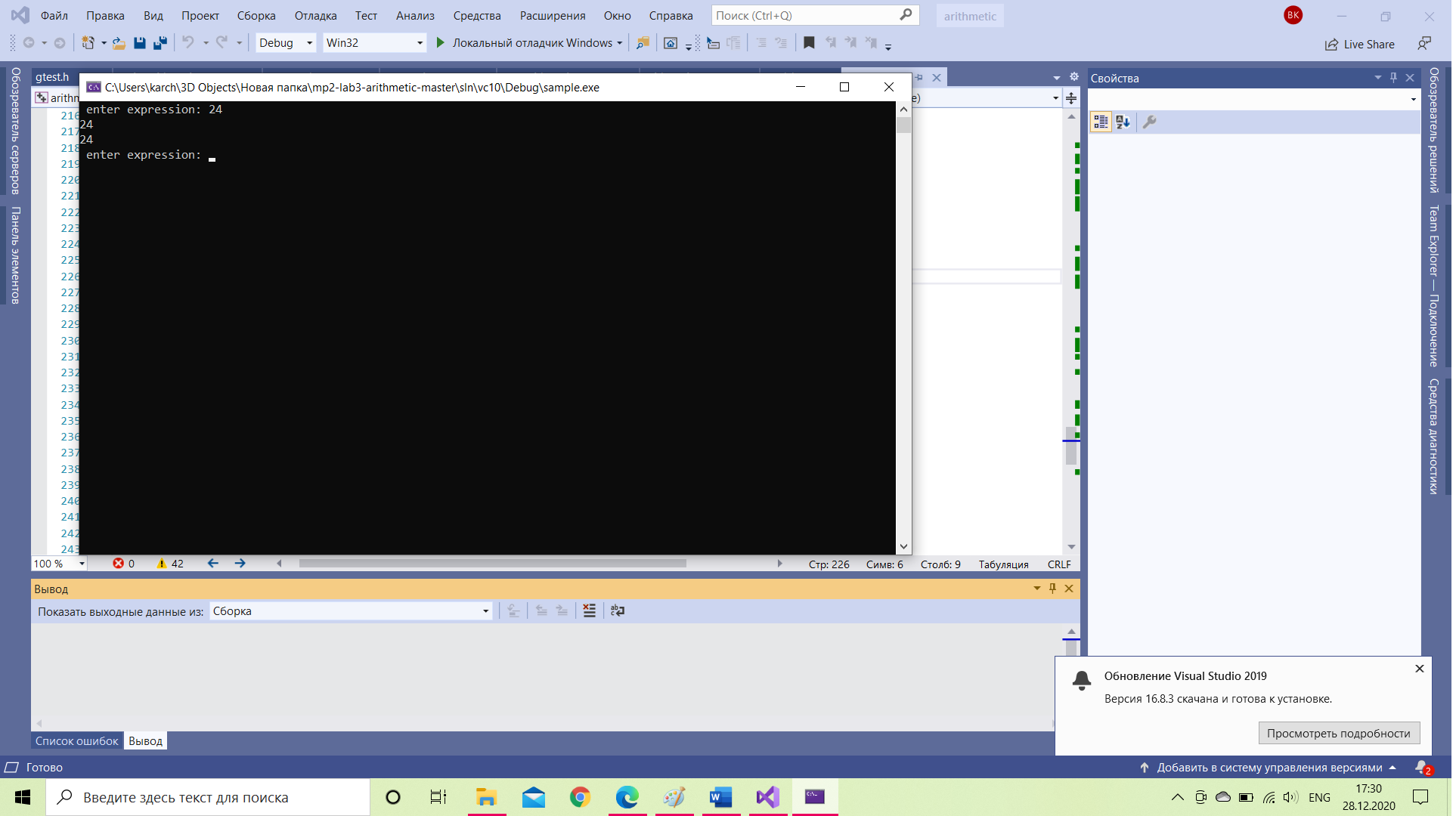
1. (Описание дальнейших действий .)

Если в выражении нет никаких операций или чисел, то программа сообщит о неправильном вводе (рис.5):



1. (Описание дальнейших действий .)

Если в выражении только число без дальнейших операций и иных чисел, то программа выдаст исходное выражение (рис.6):



# Описание программной реализации

Программная реализация для работы с арифметическими выражениями была выполнена на языке С++ в среде Microsoft Visual Studio 2019.

Для данной реализации используются 4 проекта arithmetic, gtest, sample, tests.

В проекте test написаны google-тесты для классов TStack, для класса Split и для класса arithmetic.

В проекте Sample один .cpp файл – main\_arithmetic.cpp, в котором находится main

В проекте arithmetic 1 .cpp файл и 2 .h файла:

1. В Файле stack.h реализован шаблонный класс TStack.

В классе TStack protected поля: T\*Mem – указатель на область памяти, в котором хранятся элементы(Т – шаблонный тип данных), size – размер стека, Top - последний не пустой элемент.  
Реализованы следующие public методы: IsEmpty, IsFull, See, Count, Put, Get, Clear.  
Деструктор очищает память, на которую указывает Mem.

2. В Файле arithmetic.h реализован класс Split и класс arithmetic

2.1. В классе Split написаны методы GetLex, Getposition для получения соответствующей лексемы, типа лексемы.

2.2. В классе arithmetic написаны следующие методы:Process, oper, calculat, GetResult, Prioritet.

# Подтверждение корректности

Для подтверждении корректности правильности работы программы с арифметическими выражениями написаны google-тесты на разные ситуации. Google-тесты написаны для класса TStack,Split,arithmetic

# Заключение

В ходе лабораторной работы была разработана программа для работы с арифметическими выражениями. Программа также выполняет проверку на корректность ввода выражения пользователем. Были написаны google-тесты для проверки реализации на различные ситуации.

Таким образом, данная программа для работы с арифметическими выражениями работает правильно.

# Приложение

class arithmetic

{

    string input;

    string output;

    double result;

public:

    arithmetic(string s)

    {

        input = s;

        output = "";

        result = 0;

    }

    string Process()

    {

        string nomber = "";

         output = "";

        TStack<int> st;

        Split str(input);

        string lexsema = str.GetLex();

        while (lexsema != "")

        {

            if (lexsema[0] >= '0' && lexsema[0] <= '9')// лежит число

            {

                output += lexsema;

                output += " ";

            }

            else // + - \*  / ()

            {

                //output += " ";

                if (lexsema[0] == '(')

                {

                    st.Put(lexsema[0]);

                }

                else if (st.IsEmpty())

                {

                    int d = Prioritet(lexsema[0]);

                    if (d < 0)

                    {

                        std::cout << " not correct operation";

                        std::cout << " position " << str.Getposition() << endl;

                        return "";

                    }

                    st.Put(lexsema[0]);

                }

                else

                {

                    if (lexsema[0] == ')')// извлекаем символы из стека в выходную строку до тех пор, пока не встретим в стеке открывающую скобку

                    {

                        int c;

                        if (!st.IsEmpty())

                            c = st.Get();

                        else

                        {

                            std::cout << " not found ( ";

                            std::cout << " position " << str.Getposition() << endl;

                            return "";

                        }

                        while (c != '(')

                        {

                            output += c;

                            output += " ";//

                            if (!st.IsEmpty())

                                c = st.Get();

                            else

                            {

                                std::cout << " not found ( ";

                                std::cout << " position " << str.Getposition() << endl;

                                return "";

                            }

                        }

                    }

                    //if (Prioritet(input[i] > Prioritet(st.See())))

                    else

                    {

                        int d = Prioritet(lexsema[0]);

                        if (d < 0)

                        {

                            std::cout << " not correct operation";

                            std::cout << " position " << str.Getposition() << endl;

                            return"";

                        }

                        if (Prioritet(lexsema[0]) > Prioritet(st.See()))

                        {

                            st.Put(lexsema[0]);// помещение в стек( на вершину)

                        }

                        else

                        {

                            while (Prioritet(lexsema[0]) <= Prioritet(st.See()))

                            {

                                output += st.Get();

                                output += " ";

                            }

                            st.Put(lexsema[0]);

                        }

                    }

                }

            }

            lexsema = str.GetLex();

        }

        while (!st.IsEmpty())

        {

            int c = st.Get();

            if (c == '(')

            {

                std::cout << " found ( with out )" << endl;

                return "";

            }

            output += c;

            output += " ";

        }

        return output;

    }

Приведен код основного класса arithmetic