МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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 УЧЕБНОЙ ПРАКТИКЕ**

**Вычисление арифметических выражений**

**Выполнил:** студент группы 381806-1

Белик Юлия Андреевна

**Проверил:**кандидат технических наук доцент кафедры МОСТ ИИТММ

Кустикова Валентина Дмитриевна

Нижний Новгород

2019

[Содержание 2](#_Toc26538457)

[1. Введение 3](#_Toc26538459)

[2. Постановка задачи 4](#_Toc26538460)

[3. Руководство пользователя 5](#_Toc26538461)

[4. Руководство программиста 6](#_Toc26538462)

[4.1 Структура программы 6](#_Toc26538463)

[4.2 Описание алгоритма 6](#_Toc26538464)

[4.2.1 Класс Stack 6](#_Toc26538465)

[4.2.2 Преобразование выражения в постфиксную форму 6](#_Toc26538466)

[4.2.3 Значения операндов 7](#_Toc26538467)

[4.2.4 Подсчет итогового значения 7](#_Toc26538468)

[4.3 Описание функций 8](#_Toc26538469)

[4.3.1 Функции класса Stack 8](#_Toc26538470)

[4.3.2 Преобразование выражения в постфиксную форму 9](#_Toc26538471)

[4.3.3 Функция записи значений операндов 9](#_Toc26538472)

[4.3.4 Функция подсчета итогового значения 9](#_Toc26538473)

[4.3.5 Дополнительные функции 9](#_Toc26538474)

[Заключение 12](#_Toc26538475)

1. Введение

В настоящее время наука и технологии шагают вперед семимильными шагами. Чтобы идти в ногу с прогрессом, приходится постоянно совершенствовать свои знания, умения и навыки, а на это требуется достаточно много времени. В целях освобождения времени для таких фундаментальных задач нужно минимализировать его затраты на небольшие задачи, которые можно решить один раз вместо того, чтобы каждый раз делать все заново.

Именно с этой целью пишутся программы, совершающие элементарные задачи. В данной работе будет рассматриваться способ представления и вычисления арифметических операций на компьютере. Особую пользу приносит данная программа при вычислении больших выражений, так как на такую задачу у человека может уйти несколько минут, а у компьютера пара мгновений.

1. Постановка задачи

Разработать программу для вычисления значения арифметического выражения при введенном выражении и значениях переменных.

Входные данные:

* Арифметическое выражение
* Значения переменных

Выходные данные:

* Постфиксная форма выражения
* Значение выражение

1. Руководство пользователя

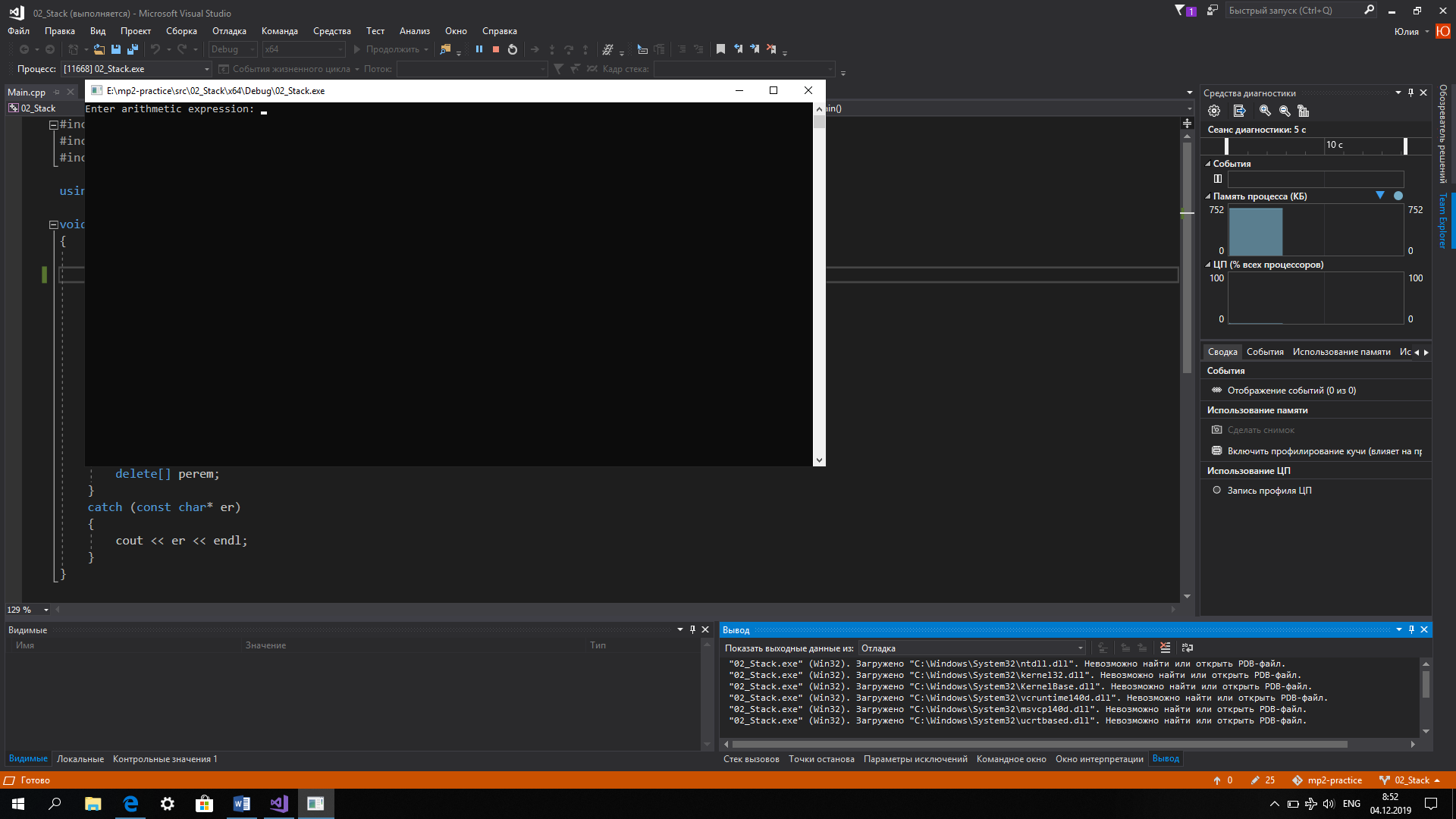
После запуска программа на экран выводится запрос на введение арифметического выражения[1] 

Рисунок 1. Запрос на ввод арифметического выражения

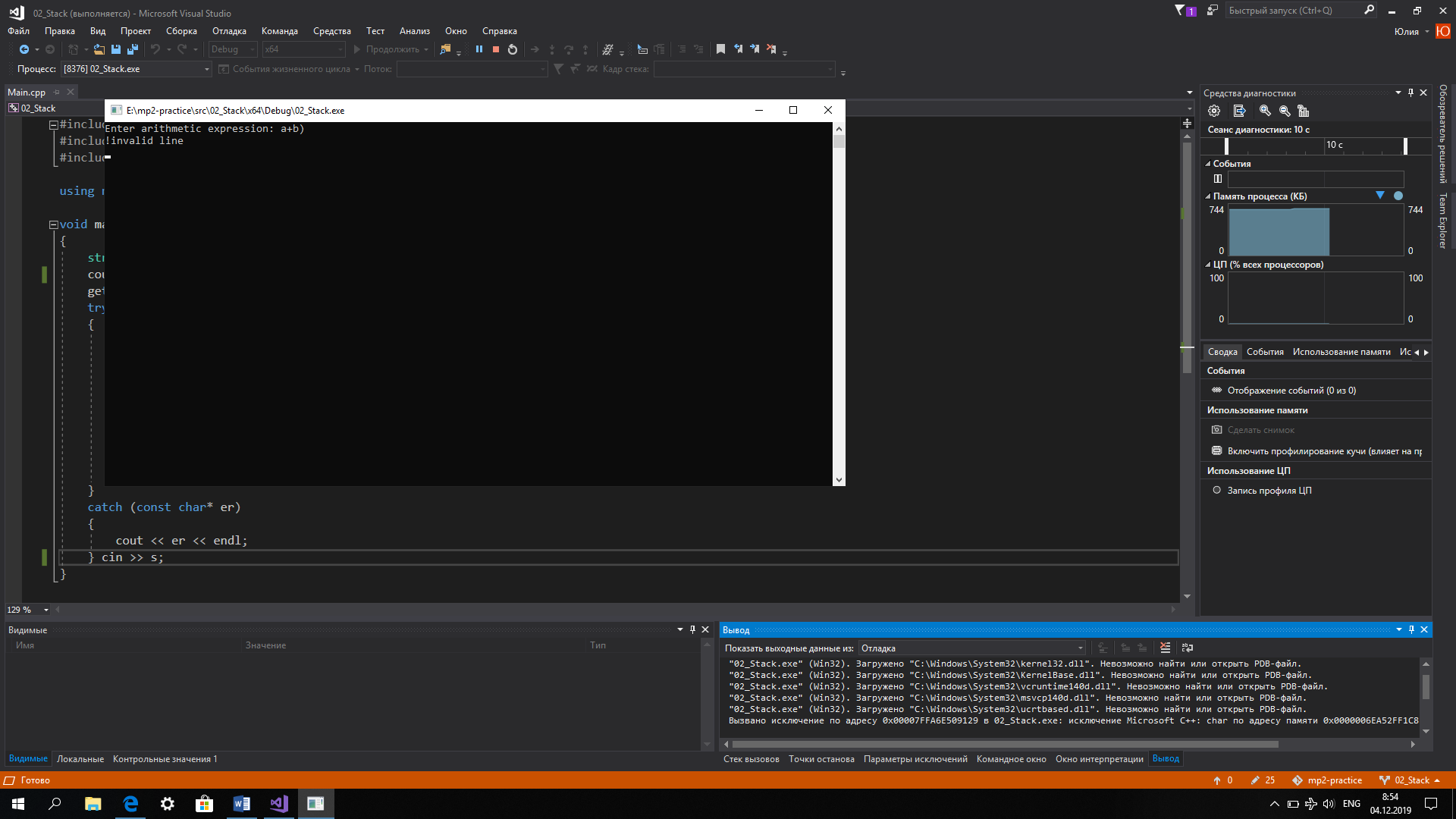
В случае некорректного ввода будет выведено сообщение об ошибке[2]. 

Рисунок 2. Пример некорректного ввода выражение

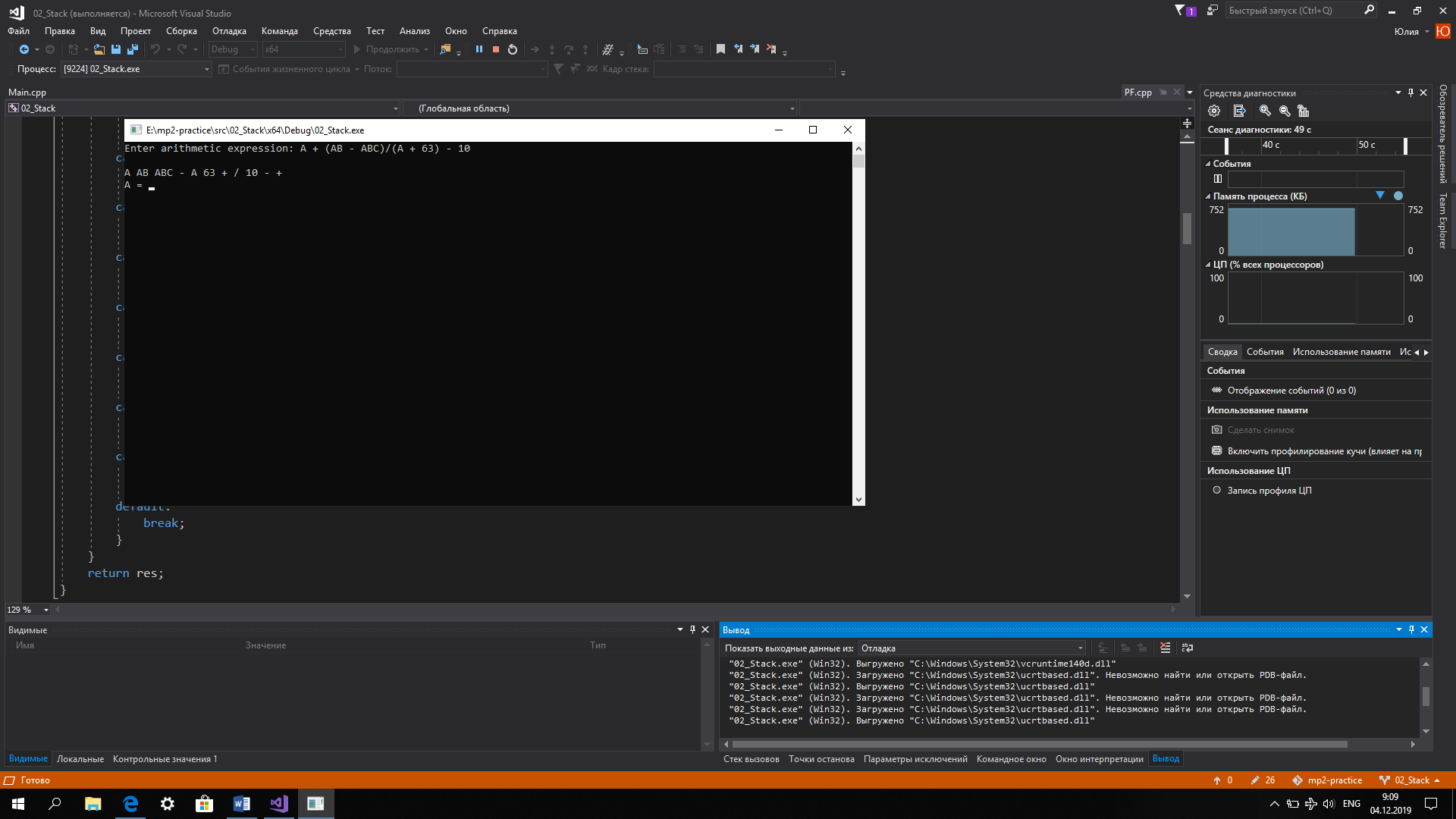
В случае корректно введенных данных на экране появится постфиксная форма выражения и запрос на ввод значений переменных, входящих в выражение. Переменные могут быть односимвольными, многосимвольными и константными числовыми значениями[3]. 

Рисунок 3. Вывод польской формы и запрос переменных

После ввода значений переменных на экране появится итоговое значение выражения[4].

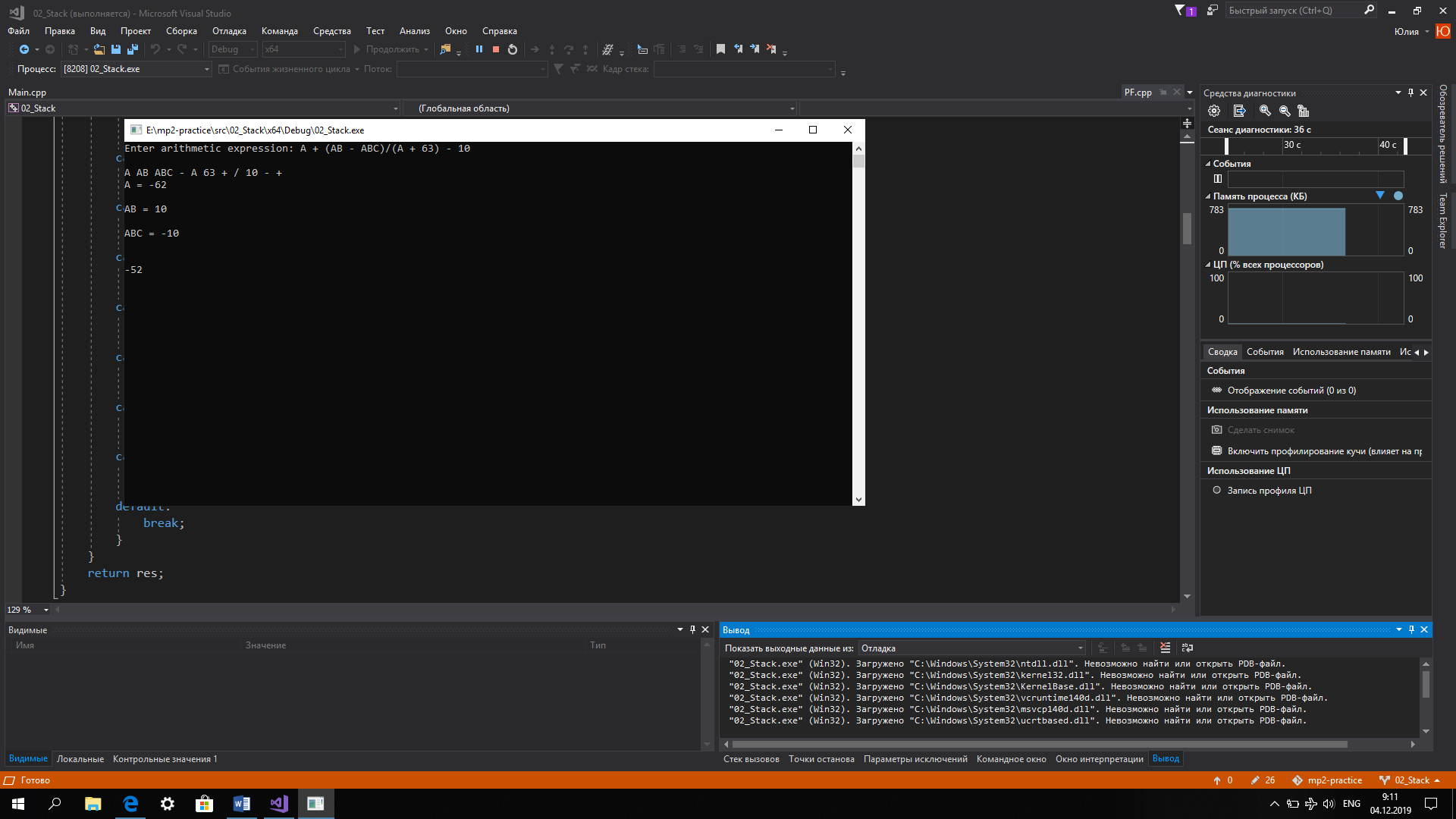


Рисунок 4. Пример выводимых данных

1. Руководство программиста
   1. Структура программы

Программа состоит из 4 файлов:

* *Main.cpp,* который содержит функцию *main*
* *Stack.h*, который содержит класс Stack
* *PF.h*,который содержит класс PolF
* *PF.cpp*, который содержит реализацию функций класса PolF
  1. Описание алгоритма
     1. Класс Stack
* Стек – структура данных, представляющая собой список элементов, организованных по принципу LIFO (last in - first out, т.е. последним пришёл-первым ушёл). Рассмотрим его реализацию с помощью массива, создав параметризованный класс для удобства работы с данными любого типа.
* Полями класса являются массив элементов стека, максимальный размер стека и текущее число элементов стека
* Методами класса являются: конструктор с параметром, конструктор копирования, деструктор, вставка в конец, удаление из конца, оператор приравнивания, проверка на пустоту и проверка на полноту
  + 1. Преобразование выражения в постфиксную форму
* Создаются 2 стека: постфиксная форма и стек операторов
* Поочередно просматривается каждый символ арифметического выражения, игнорируя пробелы
* Если символ является арифметическим знаком («+», «-», «\*», «/») или знак открывающейся скобки «(», он отправляется в стек операторов
* Если символ является закрывающейся скобкой «)», достаются все элементы из стека операторов до «(» и добавляются в стек постфиксной формы
* Если символ операнд, он добавляется в стек постфиксной формы
* После завершения цикла все элементы стека операторов перекладываются в стек постфиксной формы
* Все элементы стека постфиксной формы добавляются в итоговую строку, разделяясь пробелами [[1]](#Таблица1)

Таблица 1. Пример преобразования в постфиксную форму

Исходная строка: A + (B - 5) \* A

|  |  |  |
| --- | --- | --- |
| Элемент строки | Постфиксная форма | Стек операторов |
| A | A |  |
| + | A | + |
| ( | A | +( |
| B | AB | +( |
| - | AB | +(- |
| 5 | AB5 | +(- |
| ) | AB5- | + |
| \* | AB5- | +\* |
| A | AB5-A | +\* |
|  | AB5-A\*+ |  |

Итоговая строка: A B 5 – A \* +

* + 1. Значения операндов
* Поочередно просматриваются все элементы постфиксной формы, считается количество операндов
* Выделяется память для 2 массивов: массив названий переменных и массив их значений
* На втором просмотре выполняется проверка на константное значение и запись названий в нужный массив
* Если операнд является константой, его значение отправляется в массив значений
* В обратном случае будет выведен запрос на значение операнда, считывание и запись в массив[[2]](#Таблица2)

Таблица 2. Пример запоминания значения операндов

|  |  |  |  |
| --- | --- | --- | --- |
| Массив переменных | А | В | 5 |
| Массив значений | 10 | 15 | 5 |

* + 1. Подсчет итогового значения
* Создается стек итогового значения
* Поочередно просматриваются все элементы постфиксной формы
* Если элемент является операндом, его значение отправляется в итоговый стек
* Если элемент является оператором, из итогового массива достаются 2 последних элемента, производится необходимая арифметическая операция и значение отправляется обратно в итоговый стек
* Последнее значение в итоговом стеке является итоговым для всего выражения [[3]](#Таблица3)

Таблица 3. Пример подсчета итогового значения

|  |  |
| --- | --- |
| Элемент постфиксной формы | Итоговый стек |
| A | 10 |
| B | 10 15 |
| 5 | 10 15 5 |
| - | 10 10 |
| A | 10 10 10 |
| \* | 100 |
| + | 110 |

Итоговое значение выражения 110

* 1. Описание функций
     1. Функции класса Stack

TStack(int \_size = 10)

* Назначение

Создание объекта класса Stack с параметром количества элементов

* Входные параметры

\_size – количество элементов в стеке

TStack(const TStack<VT>& tmp)

* Назначение

Конструктор копирования класса Stack

* Входные параметры

tmp – объект класса, копия которого будет создаваться

~TStack()

* Назначение

Удаление объекта класса Stack

void Push(VT a)

* Назначение

Вставка нового элемента в конец стека

* Входные параметры

а – вставляемый элемент

* Выходные параметры

Отсутствуют

VT Pop()

* Назначение

Удаление последнего элемента из стека

* Входные параметры

Отсутствуют

* Выходные параметры

Элемент стека

TStack& operator=(const TStack& tmp);

* Назначение

Оператор присваивания

* Входные параметры

tmp - объект класса, значение которого присваивают

* Выходные параметры

Ссылка на новый объект класса

bool IsFull()const

* Назначение

Проверка на полноту

* Входные параметры

Отсутствуют

* Выходные параметры

True – если стек полностью заполнен, в противном случае – False

bool IsEmpty()const

* Назначение

Проверка на пустоту

* Входные параметры

Отсутствуют

* Выходные параметры

True – если стек пуст, в противном случае – False

* + 1. Преобразование выражения в постфиксную форму

static string PF(string st)

* Назначение

Преобразование строки, содержащей арифметическое выражение в постфиксную форму

* Входные параметры

st – строка,содержащая арифметическое выражение

* Выходные параметры

Строка, содержащая постфиксную форму

* + 1. Функция записи значений операндов

static void ZN(string pf, string\*& perem, double\*& zn, int\* count)

* Назначение

Выделение памяти для массивов названий операндов и их значений, считывание и запись названий переменных и их значений

* Входные параметры

pf – строка, содержащая постфиксную форму

\*&perem – ссылка на массив названий операндов

\*&zn – ссылка на массив значений операндов

\*count – указатель на количество операндов

* Выходные параметры

Отсутствуют (все преобразования массивов и количества переменных передаются через указатели)

* + 1. Функция подсчета итогового значения

static double Count(string PF, string\* perem, double\* zn)

* Назначение

Подсчет итогового значения арифметического выражения

* Входные параметры

PF – строка, содержащая постфиксную форму

\*perem – массив названий переменных

\*zn – массив значений переменных

* Выходные параметры

Итоговое значений выражения

* + 1. Дополнительные функции

static bool IsS(char zn)

* Назначение

Проверка элемента строки является ли он арифметическим знаком

* Входные параметры

zn – элемент строки

* Выходные параметры

True – если элемент является знаком арифметической операции, в противном случае – False

static double DO(string tmp, double a, double b)

* Назначение

Проведение арифметической операции

* Входные параметры

tmp – знак арифметической операции

a – первый операнд

b – второй операнд

* Выходные параметры

Значение арифметической операции над операндами

static int Prior(string s)

* Назначение

Определение приоритета операции

* Входные параметры

s – знак арифметической операции

* Выходные параметры

Значение приоритета арифметической операции

static double znach(string a, string\* perem, double\* zn)

* Назначение

Определение значения переменной

* Входные параметры

a – название операнда

\*perem – массив названий операндов

\*zn – массив значений операндов

* Выходные параметры

Значение операнда

static void Sign(string tmp, TStack<string>& s, TStack<string>& gl)

* Назначение

Добавление оператора в стек в зависимости от приоритета операции

* Входные параметры

tmp – знак арифметической операции

s – стек операторов

gl – стек постфиксной формы

* Выходные параметры

Отсутствуют (все изменения происходят над объектами классов)

static bool IsD(string a)

* Назначение

Проверка элемента строки не является ли он константным числовым значением

* Входные параметры

a – проверяемый элемент строки

* Выходные параметры

True – если элемент является константным числовым значением, в противном случае – False

static double StToD(string a)

* Назначение

Преобразование строки, содержащей константное числовое значение, в числовое значение

* Входные параметры

a – строка, содержащая константное числовое значение

* Выходные параметры

Числовое значение константы

static bool IsCorrect(string st)

* Назначение

Проверка строки на корректное значение

* Входные параметры

st – строка, содержащая арифметическое выражение

* Выходные параметры

True – если арифметическое выражение корректно, в противном случае – False

Заключение

Разработана программа для вычисления значения арифметического выражения при введенном выражении и значениях переменных с выводом постфиксной формы выражения.