**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ**

**ПО НИЖЕГОРОДСКОЙ ОБЛАСТИ**

**УНИВЕРСИТЕТ ИМ.Н.И.ЛОБАЧЕВСКОГО**

**ИНСТУТ ИТММ.**

**Отчет по лабораторной работе №3.**

**«Вычисление арифметических выражений (стеки)»**

**Выполнил**:

студент группы 1703-2

Хайрутдинов Э.Ш.

**Проверил**:

доцент кафедры МОСТ

Сысоев А.В.

Нижний Новгород

2018

**Оглавление**

[**Введение.** 2](#_Toc515493166)

[**Постановка задачи 1.** 3](#_Toc515493167)

[**Руководство пользователя:** 4](#_Toc515493168)

[**Руководство программиста** 6](#_Toc515493169)

[**Эксперимент** 7](#_Toc515493170)

[**Список литературы** 8](#_Toc515493171)

[**Постановка задачи 2** 9](#_Toc515493172)

[**Руководство пользователя** 10](#_Toc515493173)

[**Руководство программиста.** 14](#_Toc515493174)

[**Эксперимент** 15](#_Toc515493175)

[**Список литературы** 16](#_Toc515493176)

[**Постановка к задаче 3.** 17](#_Toc515493177)

[**Руководство пользователя** 18](#_Toc515493178)

[**Руководство программиста** 21](#_Toc515493179)

[**Эксперимент** 22](#_Toc515493180)

[**Список литературы** 23](#_Toc515493181)

[**Постановка задачи 4.** 24](#_Toc515493182)

[**Руководство пользователя** 26](#_Toc515493183)

[**Руководство программиста** 28](#_Toc515493184)

[**Список литературы** 30](#_Toc515493185)

[**Постановка задачи 5.** 31](#_Toc515493186)

[**Руководство пользователя** 32](#_Toc515493187)

[**Руководство программиста** 35](#_Toc515493188)

[**Эксперимент** 37](#_Toc515493189)

[**Список литературы** 40](#_Toc515493190)

[**Постановка задачи 6** 41](#_Toc515493191)

[**Руководство пользователя** 42](#_Toc515493192)

[**Руководство программиста** 44](#_Toc515493193)

[**Эксперимент** 45](#_Toc515493194)

[**Список литературы** 47](#_Toc515493195)

[**Приложение** 48](#_Toc515493196)

# **Введение.**

Целью данной работы является совершенствование навыков разработки и использования классов и их систем в языке программирования С++ путем реализации динамической структуры данных Стек. Также целью работы является освоение алгоритма вычисления арифметического выражения за один просмотр. Для реализации данного алгоритма необходима чтобы запись была в постфиксной форме.

# **Постановка задачи 1.**

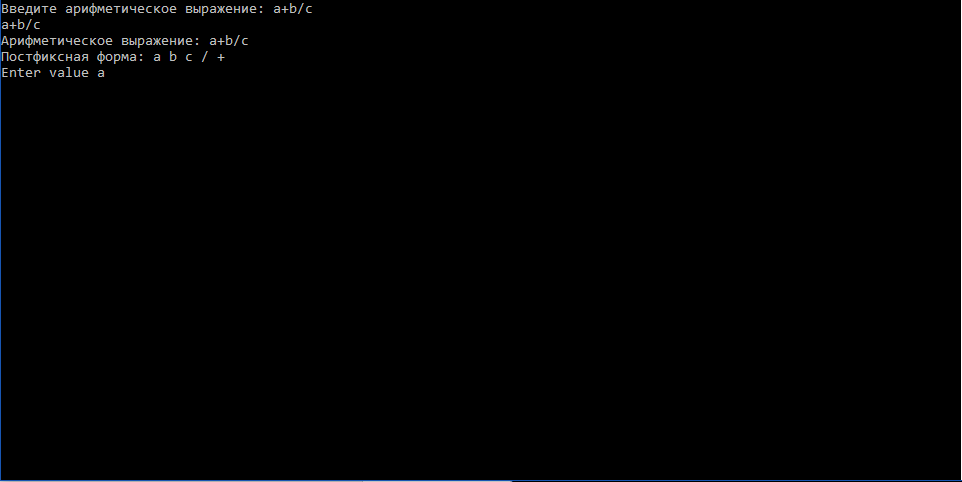
Передо мной была поставлена задача написать класс, которая реализует классы Стек и Постфикс и их функции:

* Методы добавления и удаления элемента из стека.
* Методы проверки стека на полноту и пустоту.
* Метод перевода арифметического выражения в постфикс и его проверки на корректность.
* Метод подсчета арифметического выражения в постфиксном виде.

# **Руководство пользователя:**

1. Перед Вами появилась консоль с меню (рис.1).

Рис.1

1. Введите выражение. Можете ввести как формулу, так и численное выражение.
   1. Если Вы ввели формулу, программа переведет формулу в постфиксную форму и попросит ввести значения операндов(рис.2).

­Рис.2

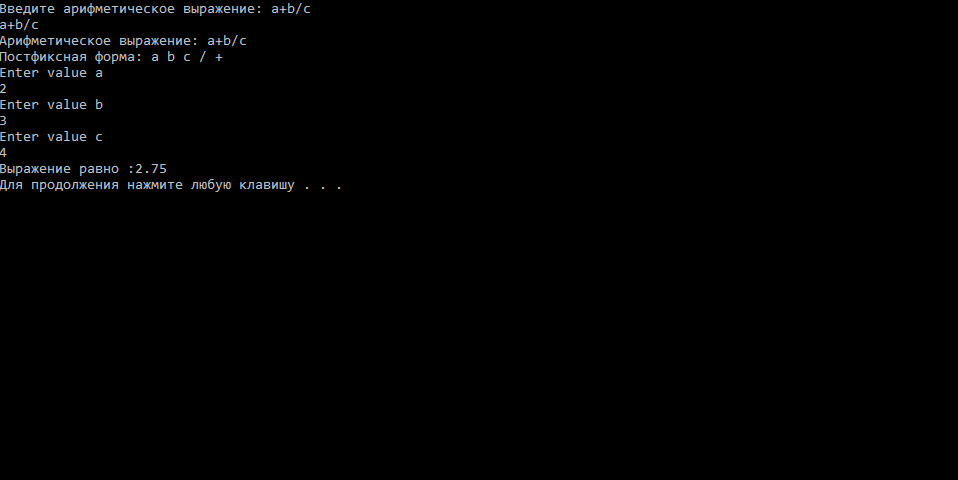
* 1. Когда Вы введете значение, то программа вычислит значение выражения(рис.3).

Рис.3

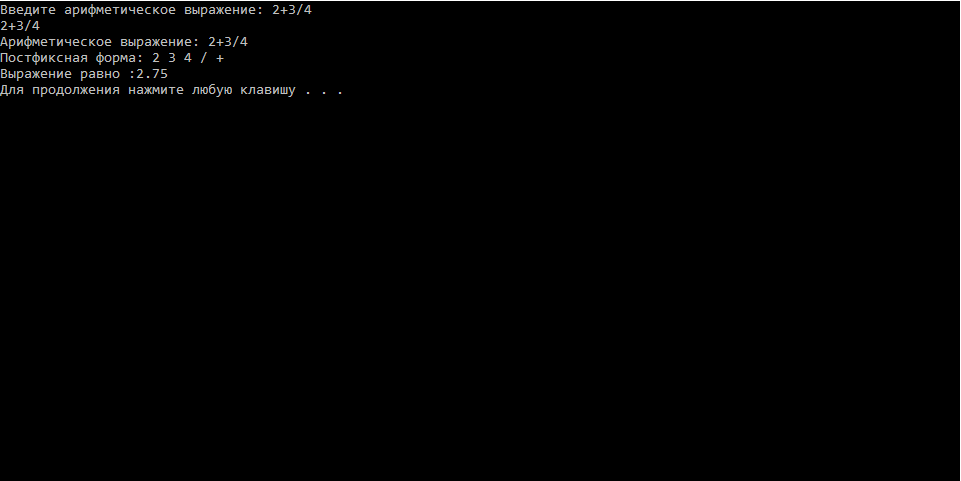
1. Если Вы ввели численное выражение, то программа сразу переведет в постфикс и вычислит значение выражение(рис.4).

Рис.4

1. Чтобы продолжить нажмите на любую кнопку (кроме кнопки выключения).

# **Руководство программиста**

Класс TStack имеет методы, которые осуществляют проверку стека на полноту и пустоту, методы добавления и удаление элемента из стека.

**bool IsEmpty()**

**bool IsFull()**

**void Push(const Telem &val)**

**Telem Pop()**

**Telem Back()**

Класс TPostfix имеет методы, которые проверяют корректность введенных данных, производят перевод обычной записи в постфиксную и вычисления выражения, а также вспомогательные методы.

**bool CorrectRecord();**

**void ToPostfix();**

**void Converter(string &str);**

**double Calculate();**

Вспомогательные методы:

**void SetInfix(string \_infix);**

**string GetInfix()**

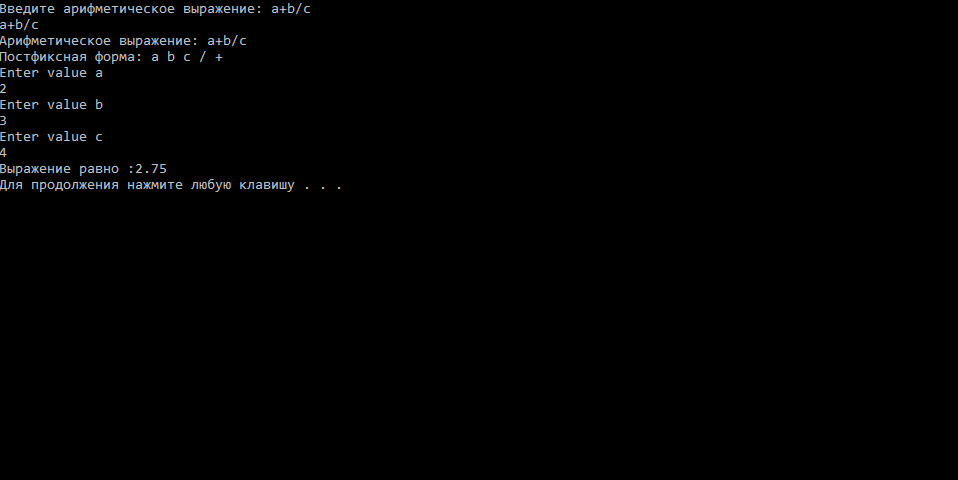
**bool IsOperations(char s);**

**bool IsOperand(char s);**

**string GetPostfix()**

**int Compare(char \_t1);**

# **Эксперимент**

В результате эксперимента, показанного на рис.3 и рис 4, мы видим, что программа считает правильно выражение, введенное как в численном, так и в буквенном виде. Программа функционирует и работает правильно.

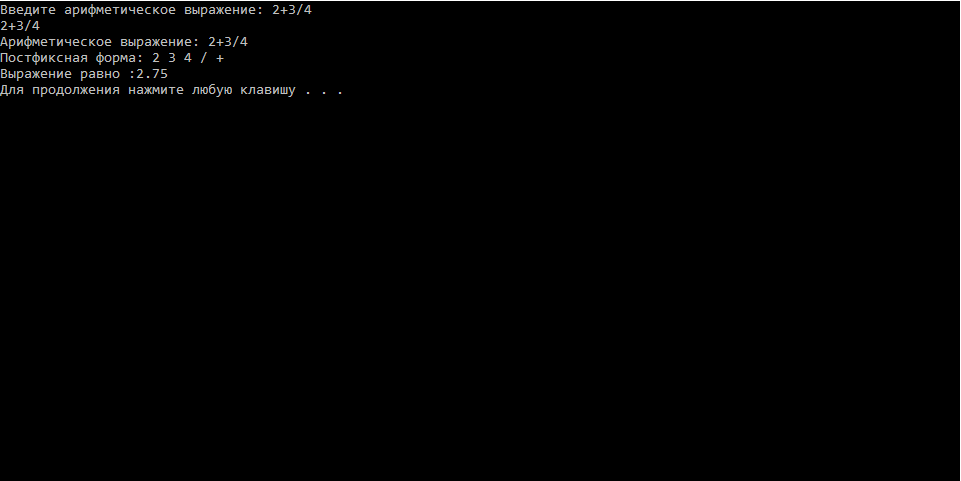
Рис.3

Рис.4

# **Список литературы**

* http://cpp.mazurok.com/tag/стек/
* http://www.lib.unn.ru/students/src/Pract\_ADS.pdf
* http://cpp.mazurok.com/tag/постфиксная-форма/

# **Приложение**

#include "postfix.h"

#include "stack.h"

#include "iostream"

#include "vector"

#include <cctype>

void TPostfix::ToPostfix()

{

if (CorrectRecord() == false)

throw "error";

TStack<char> op(infix.size());

postfix = "";

for (int i = 0; i < infix.size(); i++)

{

if (isalpha(infix[i]) || isdigit(infix[i]))

{

postfix += infix[i];

while (isalpha(infix[i + 1]) || isdigit(infix[i + 1]))

{

postfix += infix[i + 1];

i++;

}

postfix += ' ';

}

else

if (infix[i] == '(')

op.Push(infix[i]);

else

if (infix[i] == ')')

{

while (op.Back() != '(')

{

postfix += op.Pop();

postfix += ' ';

}

op.Pop();

}

else

if (op.IsEmpty() == true)

{

op.Push(infix[i]);

}

else

if (Compare(infix[i]) <= Compare(op.Back()))

{

postfix += op.Pop();

if (Compare(infix[i]) <= Compare(op.Back()))

{

postfix += op.Pop();

postfix += ' ';

}

postfix += ' ';

op.Push(infix[i]);

}

else

op.Push(infix[i]);

}

while (op.IsEmpty() == false)

{

postfix += op.Pop();

postfix += ' ';

}

}

void TPostfix::SetInfix(string \_infix)

{

infix = \_infix;

}

void TPostfix::Converter(string &str)

{

string tmp;

for (int i = 0; i < str.size(); i++)

{

if (IsOperand(str[i]))

{

cout << "Enter value " << str[i] << endl;

cin >> tmp;

char sym = str[i];

int j = i;

while (j < str.size())

{

if (sym == str[j])

{

str.erase(j, 1);

str.insert(j, tmp);

j += tmp.size();

}

else j++;

}

}

}

}

double TPostfix::Calculate()

{

Converter(postfix);

TStack<double> value(postfix.size());

string tmp;

double tmp1;

double tmp2;

for (int i = 0; i < postfix.size(); i++)

{

if (postfix[i] == ' ')

continue;

if (IsOperations(postfix[i]) == false)

{

while (postfix[i] != ' ')

{

tmp += postfix[i];

i++;

}

value.Push(atof(tmp.c\_str()));

tmp = "";

}

if (IsOperations(postfix[i]) == true)

{

tmp1 = value.Pop();

tmp2 = value.Pop();

switch (postfix[i])

{

case '+':

value.Push(tmp1 + tmp2);

break;

case '-':

value.Push(tmp2 - tmp1);

break;

case '\*':

value.Push(tmp1\*tmp2);

break;

case'/':

value.Push(tmp2 / tmp1);

break;

}

}

}

return value.Pop();

}

int TPostfix::Compare(char tmp1)

{

if (tmp1 == '\*' || tmp1 == '/')

return 2;

if (tmp1 == '-' || tmp1 == '+')

return 1;

if (tmp1 == '(')

return 0;

}

bool TPostfix::CorrectRecord()

{

int lb = 0, rb = 0;

int operations = 0, operands = 0;

for (int i = 0; i < infix.size(); i++)

{

if (infix[i] == '(')

{

lb++;

continue;

}

else if (infix[i] == ')')

{

rb++;

continue;

}

if (rb > lb)

return false;

if (infix[i] == '+' || infix[i] == '-' || infix[i] == '\*' || infix[i] == '/')

operations++;

else

{

while (isalpha(infix[i + 1]) || isdigit(infix[i + 1]))

{

i++;

}

operands++;

}

}

if ((IsOperations(infix[0]) == true || IsOperations(infix[infix.size() - 1]) == true))

return false;

if ((operands == operations + 1) && (lb == rb))

return true;

else return false;

}

bool TPostfix::IsOperations(char s)

{

if (s == '+' || s == '-' || s == '\*' || s == '/')

return true;

return false;

}

bool TPostfix::IsOperand(char s)

{

if ((int(s) >= 65 && int(s) <= 90) || (int(s) >= 97 && int(s) <= 122))

return true;

return false;

}

#ifndef \_\_STACK\_H\_\_

#define \_\_STACK\_H\_\_

const int MaxStackSize = 100;

template <class Telem>

class TStack

{

Telem \*pMem;

int size;

int top;

public:

TStack(int \_size)

{

size = \_size;

top = -1;

if ((size < 1) || (size > MaxStackSize))

throw size;

pMem = new Telem[size];

}

~TStack()

{

delete[] pMem;

}

bool IsFull()

{

if (top == (size - 1))

return true;

return false;

}

bool IsEmpty()

{

if (top == -1)

return true;

return false;

}

void Push(const Telem &val)

{

if (IsFull() == false)

{

top++;

pMem[top] = val;

}

else throw "error";

}

Telem Pop()

{

if (IsEmpty() == false)

return pMem[top--];

else

throw "error";

}

Telem Back()

{

return pMem[top];

}

};

#endif