МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Программная инженерия»

ОТЧЕТ

по лабораторной работе №3

***Вычисление арифметических выражений (стеки)***

**Выполнил:** студент группы

3822Б1ПР2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.А. Соловьева

Подпись

Нижний Новгород  
2023

**Содержание**

[1.Введение: 3](#_Toc151063293)

[2. Постановка задачи: 4](#_Toc151063294)

[3. Руководство пользователя 5](#_Toc151063295)

[3.1. Консольное приложение 5](#_Toc151063296)

[3.2. Визуальное приложение: 6](#_Toc151063297)

[4. Руководство программиста:](#_Toc151063298) 7

[4.1 Описание алгоритмов: 7](#_Toc151063299)

[4.2.Описание программы: 8](#_Toc151063300)

[5.Заключение:](#_Toc151063301) 10

[6.Литература:](#_Toc151063303) 11

[7.Приложение:](#_Toc151063304) 12

1.Введение:

Лабораторная работа направлена на практическое освоение динамической структуры данных Стек. С этой целью в лабораторной работе изучаются различные варианты структуры хранения стеков и разрабатываются методы и программы решения ряда задач с использованием стеков. В качестве области приложений выбрана тема вычисления арифметических выражений, возникающей при трансляции программ на языке программирования высокого уровня в исполняемые программы.

При вычислении произвольных арифметических выражений возникают две основные задачи: проверка корректности введённого выражения и выполнение операций в порядке, определяемом их приоритетами и расстановкой скобок. Существует алгоритм, позволяющий реализовать вычисление произвольного арифметического выражения за один просмотр без хранения промежуточных результатов. Для реализации данного алгоритма выражение должно быть представлено в постфиксной форме. Рассматриваемые в данной лабораторной работе алгоритмы являются начальным введением в область машинных вычислений.

Арифметическое выражение - выражение, в котором операндами являются объекты, над которыми выполняются арифметические операции. Например, (1+2)/(3+4\*6.7)-5.3\*4.4

При такой форме записи (называемой инфиксной, где знаки операций стоят между операндами) порядок действий определяется расстановкой скобок и приоритетом операций. Постфиксная (или обратная польская) форма записи не содержит скобок, а знаки операций следуют после соответствующих операндов. Тогда для приведённого примера постфиксная форма будет иметь вид: 1 2+ 3 4 6.7\*+/ 5.3 4.4\* -

Обратная польская нотация была разработана австралийским ученым Чарльзом Хэмблином в середине 50-х годов прошлого столетия на основе польской нотации, которая была предложена в 1920 году польским математиком Яном Лукасевичем. Эта нотация лежит в основе организации вычислений для арифметических выражений. Известный ученый Эдсгер Дейкстра предложил алгоритм для перевода выражений из инфиксной в постфиксную форму. Данный алгоритм основан на использовании стека.

Стек (англ. stack), магазин – схема запоминания информации, при которой каждый вновь поступающий ее элемент как бы «проталкивает» вглубь отведенного участка памяти находящиеся там элементы (подобно патрону, помещаемому в магазин винтовки) и занимает крайнее положение (так называемую вершину стека). При выдаче информации из стека выдается элемент, расположенный в вершине стека, а оставшиеся элементы продвигаются к вершине; следовательно, элемент, поступивший последним, выдается первым [1]. Более строгое определение структуры дано в разделе 2 описания данной лабораторной работы.

2. Постановка задачи:

В рамках лабораторной работы ставится задача реализации программ, обеспечивающих поддержку стеков, и разработки программных средств, производящих обработку арифметических выражений, включая проверку правильности записи выражения, перевод в постфиксную форму и вычисление результата. В начальной – самой простой постановке – можно предполагать, что проверка записи выражения состоит в контроле правильности расстановки скобок, перевод в постфиксную форму производится только для корректных выражений, а вычисление – для корректных выражений, содержащих только числовые операнды и допустимые знаки операций.

Класс Арифметическое Выражение должен предоставлять, как минимум, следующие операции

* Получение, хранение и возврат исходной (инфиксной) формы
* Преобразование в постфиксную форму
* Возврат постфиксной формы
* Вычисление арифметического выражения при заданных значениях операндов

3. Руководство пользователя

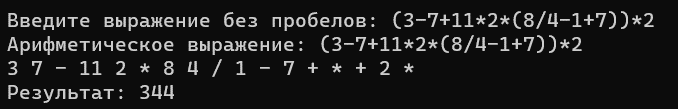
3.1. Консольное приложение

В начале работы программы появляется окно (рис.1) и предлагается ввести любое математическое выражение для подсчета и представления его в постфиксном виде:



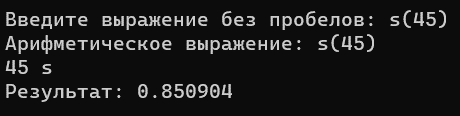
(рис.1)

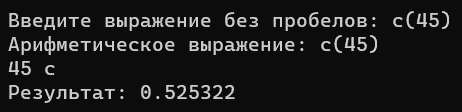
Далее, после ввода выражения выводится окно (рис.2), в котором считается и выводится как само выражение, так и его постфиксная запись, так и сам результат вычисления:

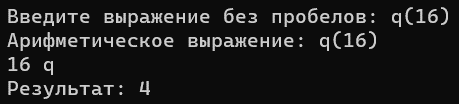


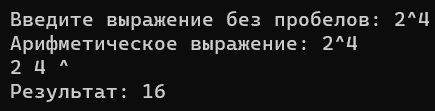
(рис.2)

В программе есть реализация не только базовых операций, но и синуса (рис.3), косинуса (рис.4), квадратного корня (рис.5) и возведения в степень одного числа в другое (рис.6). Демонстрация работы ниже:

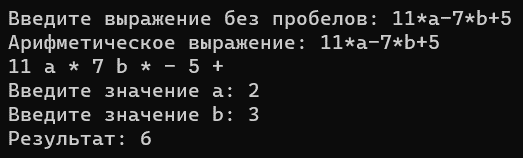
 (Рис. 3)

 (Рис. 4)

 (Рис. 5)

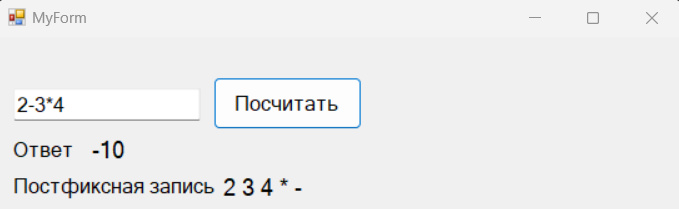
 (Рис. 6)

Также в программе есть возможность работы с переменными (рис.7):

 (Рис. 7)

3.2. Визуальное приложение:

Перейдем к визуальной части программы, она выглядит следующим образом (рис.8) В окно пользователь вводит математическое выражение, нажимает на клавишу «Посчитать» и ниже появляется ответ выражения и запись математического выражения в постфиксном виде.

 (Рис. 8)

4. Руководство программиста:

4.1 Описание алгоритмов:

Алгоритма преобразования инфиксной строки в обратную польскую запись:

Рассматриваем поочередно каждый символ:

Если этот символ - число (или переменная), то просто помещаем его в выходную строку.

Если символ - знак операции (+, -, \*, /), то проверяем приоритет данной операции:

Умножение и деление имеют наивысший приоритет (например, он равен 3)

Сложение и вычитание имеют меньший приоритет (например, 2)

Наименьший приоритет у открывающейся скобки (1)

Получив один из этих символов мы должны проверить стек:

Если стек все еще пуст или находящиеся в нем символы (а находиться в нем могут только знаки операций и открывающая скобка) имеют меньший приоритет, чем приоритет текущего символа, то символ помещаем в стек

Если символ, находящийся на вершине стека имеет приоритет больший или равный приоритету текущего символа, то извлекаем символы из стека в выходную строку до тех пор, пока выполняется это условие, затем переходим к пункту А

Если текущий символ - открывающая скобка, то помещаем ее в стек

Если текущий символ - закрывающая скобка, то извлекаем символы из стека в выходную строку, пока не встретим открывающую скобку (т.е. символ с приоритетом, равным 1), которую следует просто уничтожить. Закрывающая скобка также уничтожается

Если вся входная строка разобрана, а в стеке еще остаются знаки операций, то извлекаем их из стека в выходную строку.

Алгоритм вычисления по постфиксной форме:

Для каждой лексемы в постфиксной форме

– Если лексема –операнд, поместить ее значение в стек

– Если лексема –операция

• Извлечь из стека значения двух операндов

• Выполнить операцию (верхний элемент из стека является правым операндом, следующий за ним – левым

• Положить результат операции в стек

По исчерпании лексем из постфиксной формы на вершине стека будет

результат вычисления выражения

4.2.Описание программы:

Программа состоит из нескольких основных классов:

Класс стек:

Класс стек – шаблонный класс с изменяемым типом хранения содержащий:

protected:

T\* pMem;

int size;

int first;

public:

MyStack(int \_size = 100); //конструктор

MyStack(const MyStack<T>& st); //копирование

MyStack(MyStack<T>&& st); //перемещение

~MyStack(); //деструктор

MyStack<T>& operator=(const MyStack<T>& st);

MyStack<T>& operator=(MyStack<T>&& st);

bool operator==(const MyStack<T>& st) const;

bool operator!=(const MyStack<T>& st) const;

bool isEmpty(); //проверяет, пуст ли стек

bool isFull(); //проверяет, заполнен ли стек

void Push(T elem); //позволяет «протолкнуть» элемент в начало стека

T Pop(); //позволяет удалить элемент из начала стека

Класс мой постфикс:

string infix; // объявление строки infix

vector<string> postfix; // объявление вектора строк postfix

static map<char, int> priority; // объявление статического ассоциативного контейнера priority с ключами типа char и значениями типа int

vector <string> lexems; // объявление вектора строк lexems

map<string, double> operands; // объявление ассоциативного контейнера operands с ключами типа string и значениями типа double

void ToPostfix(); // объявление метода ToPostfix() без возвращаемого значения

void Parse(); // объявление метода Parse() без возвращаемого значения //реализован метод для разбиения инфиксного выражения на лексемы

//реализованы методы для разбиения инфиксного выражения на лексемы

public:

MyPostfix();

MyPostfix(std::string infix); // объявление конструктора класса MyPostfix с параметром типа string

string GetInfix(); // объявление метода GetInfix() с возвращаемым значением типа string

vector<string> GetPostfix(); // объявление метода GetPostfix() с возвращаемым значением типа vector<string>

vector <string> GetOperands(); // объявление метода GetOperands() с возвращаемым значением типа vector<string>

double Calculate(const map<string, double>& values); // Ввод переменных, вычисление по постфиксной форме

Класс

5.Заключение

В ходе выполнения этой лабораторной работы, я успешно справилась со всеми поставленными задачами. После изучения темы постфиксной записи был реализован класс для обработки арифметических выражений, после в него были добавлены различные улучшения такие как определения синуса, косинуса, квадратного корня, возведение в степень одного числа в другое, а также использование переменных. Реализовали тесты для проверки основных функций класса. Для данного класса был реализован дополнительно класс стек для обработки арифметического выражения, который был построен на упрошенном одностороннем списке. Так же мы реализовали main для проверки работоспособности программы, после мы реализовали визуальный интерфейс для приложения который имел вид стандартного калькулятора, визуальный интерфейс создавался с помощью WindowsForms.

**6.Литература:**

1. Википедия/Стек - <https://ru.wikipedia.org/wiki/Стек>
2. Википедия/Список - <https://ru.wikipedia.org/wiki/Связный_список>
3. Учебно-методическое пособие - [https://drive.google.com/drive](https://drive.google.com/drive/folders/1HP8ljKsupkmpcOhCmBTk4kzr6_h1nLyw)

7.Приложение:

MyStack.h

#pragma once

#include <iostream>

template<typename T>

class MyStack

{

protected:

T\* pMem;

int size;

int first;

public:

MyStack(int \_size = 100);

MyStack(const MyStack<T>& st);

MyStack(MyStack<T>&& st);

~MyStack();

MyStack<T>& operator=(const MyStack<T>& st);

MyStack<T>& operator=(MyStack<T>&& st);

bool operator==(const MyStack<T>& st) const;

bool operator!=(const MyStack<T>& st) const;

bool isEmpty(); //проверяет, пуст ли стек

bool isFull(); //проверяет, заполнен ли стек

void Push(T elem); //позволяет «протолкнуть» элемент в начало стека

T Pop(); //позволяет удалить элемент из начала стека

};

template<typename T>

inline MyStack<T>::MyStack(int \_size)

{

if (\_size > 0)

{

size = \_size;

first = 0;

pMem = new T[size];

}

else

{

throw "Сбой в стеке инициализации";

}

}

template<typename T>

inline MyStack<T>::MyStack(const MyStack<T>& st)

{

if (st.pMem == nullptr)

{

pMem = nullptr;

size = 0;

first = 0;

}

else

{

size = st.size;

first = st.first;

pMem = new T[size];

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

{

pMem[i] = st.pMem[i];

}

}

}

template<typename T>

inline MyStack<T>::MyStack(MyStack<T>&& st)

{

size = st.size;

first = st.first;

pMem = st.pMem;

st.size = 0;

st.first = 0;

st.pMem = nullptr;

}

template<typename T>

inline MyStack<T>::~MyStack()

{

if (pMem != nullptr)

{

delete[] pMem;

pMem = nullptr;

}

size = 0;

first = 0;

}

template<typename T>

inline MyStack<T>& MyStack<T>::operator=(const MyStack<T>& st)

{

if (this == &st)

{

return \*this;

}

if (pMem != nullptr)

{

delete[] pMem;

}

if (st.pMem == nullptr)

{

pMem = nullptr;

size = 0;

first = 0;

}

else

{

size = st.size;

first = st.first;

pMem = new T[size];

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

{

pMem[i] = st.pMem[i];

}

}

return \*this;

}

template<typename T>

inline MyStack<T>& MyStack<T>::operator=(MyStack<T>&& st)

{

if (this == &st)

{

return \*this;

}

if (pMem != nullptr)

{

delete[] pMem;

}

pMem = st.pMem;

size = st.size;

first = st.first;

st.pMem = nullptr;

st.size = 0;

st.first = 0;

return \*this;

}

template<typename T>

inline bool MyStack<T>::operator==(const MyStack<T>& st) const

{

if (size != st.size || first != st.first)

{

return false;

}

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

{

if (pMem[i] != st.pMem[i])

{

return false;

}

}

return true;

}

template<typename T>

inline bool MyStack<T>::operator!=(const MyStack<T>& st) const

{

if (size != st.size || first != st.first)

{

return true;

}

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

{

if (pMem[i] != st.pMem[i])

{

return true;

}

}

return false;

}

template<typename T>

inline bool MyStack<T>::isEmpty()

{

if (first == 0)

{

return true;

}

else

{

return false;

}

}

template<typename T>

inline bool MyStack<T>::isFull()

{

if (size == first)

{

return true;

}

else

{

return false;

}

}

template<typename T>

inline void MyStack<T>::Push(T elem)

{

if (isFull())

{

throw "Полный стек сбоев";

}

pMem[first] = elem;

first += 1;

}

template<typename T>

inline T MyStack<T>::Pop()

{

if (isEmpty())

{

throw "Стек сбоев пуст";

}

first--;

return pMem[first];

}

Main:

#include <iostream>

#include "MyStack.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

MyStack<int> a(10);

a.Push(1);

a.Push(2);

a.Push(3);

while (!a.isEmpty()) {

cout << a.Pop() << endl;

}

}

+ тесты (для стека и для постфикса)

MyPostf.h:

#pragma once

#define \_\_MyPostf\_h\_\_

#include <iostream>

#include <string>

#include <map>

#include <vector>

#include <iostream>

#include "MyStack.h"

#include <stdexcept>

#include <math.h>

#include <string>

#include <cstdlib>

using namespace std;

class MyPostfix

{

string infix; // объявление строки infix

vector<string> postfix; // объявление вектора строк postfix

static map<char, int> priority; // объявление статического ассоциативного контейнера priority с ключами типа char и значениями типа int

vector <string> lexems; // объявление вектора строк lexems

map<string, double> operands; // объявление ассоциативного контейнера operands с ключами типа string и значениями типа double

void ToPostfix(); // объявление метода ToPostfix() без возвращаемого значения

void Parse(); // объявление метода Parse() без возвращаемого значения //реализован метод для разбиения инфиксного выражения на лексемы

//реализованы методы для разбиения инфиксного выражения на лексемы

public:

MyPostfix();

MyPostfix(std::string infix); // объявление конструктора класса MyPostfix с параметром типа string

string GetInfix(); // объявление метода GetInfix() с возвращаемым значением типа string

vector<string> GetPostfix(); // объявление метода GetPostfix() с возвращаемым значением типа vector<string>

vector <string> GetOperands(); // объявление метода GetOperands() с возвращаемым значением типа vector<string>

double Calculate(const map<string, double>& values); // Ввод переменных, вычисление по постфиксной форме

};

MyPostfix::MyPostfix()

{

}

MyPostfix::MyPostfix(std::string infx) {

if (infx.empty())

throw "Stack is empty"; // стек пуст

infix = infx;

ToPostfix();

}

string MyPostfix::GetInfix() {

return infix;

}

vector<string> MyPostfix::GetPostfix() {

return postfix;

}

vector<string> MyPostfix::GetOperands() {

vector<string> op(operands.size());

size\_t i = 0;

for (const auto& item : operands) {

op[i++] = item.first;

}

return op;

}

//sin = s

//cos = c

//sqrt = q

void MyPostfix::Parse() { //реализован метод для разбиения инфиксного выражения на лексемы

string tmp = "";

for (char c : infix) {

if (c == 's' || c == 'c' || c == 'q' || c == '^' || c == '+' || c == '-' || c == '\*' || c == '/' || c == '(' || c == ')') {

if (!tmp.empty()) {

lexems.push\_back(tmp);

}

lexems.push\_back(string{ c });

tmp = "";

continue;

}

tmp += c;

}

if (!tmp.empty()) {

lexems.push\_back(tmp);

}

}

map<char, int>MyPostfix::priority = {

{'s', -1 },

{'c', -1 },

{'q', -1 },

{'(', 0 },

{')', 1 },

{'+', 2},

{ '-', 2 },

{ '\*', 3 },

{ '/', 3 },

{'^', 4 }

};

void MyPostfix::ToPostfix() {

Parse();

MyStack<string> st(100);

string stackItem;

for (auto& item : lexems) {

switch (item[0]) {

case '(':

st.Push(item);

break;

case ')':

stackItem = st.Pop();

while (stackItem != "(") {

postfix.push\_back(stackItem);

stackItem = st.Pop();

}

break;

case '+': case '-': case '\*': case '/':case 's':case 'c':case 'q':case '^':

while (!st.isEmpty()) {

stackItem = st.Pop();

if (priority[item[0]] <= priority[stackItem[0]]) {

postfix.push\_back(stackItem);

}

else {

st.Push(stackItem);

break;

}

}

st.Push(item);

break;

default:

operands.insert({ item, 0.0 });

postfix.push\_back(item);

break;

}

}

while (!st.isEmpty()) {

stackItem = st.Pop();

postfix.push\_back(stackItem);

}

}

double MyPostfix::Calculate(const map<string, double>& values) {

for (auto& val : values)

{

try

{

operands.at(val.first) = val.second;

}

catch (out\_of\_range& e) {}

}

MyStack<double> st(100);

double leftOperand, rightOperand;

for (const string& lexem : postfix)

{

const char c = lexem[0];

switch (c)

{

case 's':

rightOperand = st.Pop();

st.Push(sin(rightOperand));

break;

case 'c':

rightOperand = st.Pop();

st.Push(cos(rightOperand));

break;

case 'q':

rightOperand = st.Pop();

st.Push(sqrt(rightOperand));

break;

case '+':

rightOperand = st.Pop();

leftOperand = st.Pop();

st.Push(leftOperand + rightOperand);

break;

case '-':

rightOperand = st.Pop();

leftOperand = st.Pop();

st.Push(leftOperand - rightOperand);

break;

case '\*':

rightOperand = st.Pop();

leftOperand = st.Pop();

st.Push(leftOperand \* rightOperand);

break;

case '/':

rightOperand = st.Pop();

leftOperand = st.Pop();

st.Push(leftOperand / rightOperand);

break;

case '^':

rightOperand = st.Pop();

leftOperand = st.Pop();

st.Push(pow(rightOperand, leftOperand));

break;

default:

st.Push(operands[lexem]);

break;

}

}

return st.Pop();

}

Main:

#include <iostream>

#include <string>

#include "MyPostf.h"

#include <map>

#include <vector>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

string expression;

cout << "Введите выражение без пробелов: ";

cin >> expression;

MyPostfix expr(expression);

cout << "Арифметическое выражение: " << expr.GetInfix() << endl;

const auto postfix = expr.GetPostfix();

for (const auto& lexem : postfix)

{

cout << lexem << ' ';

}

cout << endl;

//cout << "Постфиксная форма: " << expr.GetPostfix() << endl;

vector<string> operands = expr.GetOperands();

map<string, double> values;

double val;

for (const auto& op : operands)

{

try {

double val = stod(op);

values[op] = val;

}

catch (...) { // ловит любой тип

cout << "Введите значение " << op << ": ";

cin >> val;

values[op] = val;

}

}

cout << "Результат: " << expr.Calculate(values) << endl;

return 0;

}

MyForm.cpp:

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void Main(cli::array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Graph::MyForm form;

Application::Run(% form);

}

MyForm.h:

#pragma once

#include"../postflib/MyPostf.h"

#include<msclr\marshal\_cppstd.h>

#include<string>

namespace Graph {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(231, 40);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(159, 52);

this->button1->TabIndex = 0;

this->button1->Text = L"Посчитать";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(17, 51);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(198, 30);

this->textBox1->TabIndex = 1;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 13.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(96, 96);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(19, 29);

this->label1->TabIndex = 2;

this->label1->Text = L":";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 13.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(236, 133);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(19, 29);

this->label2->TabIndex = 3;

this->label2->Text = L":";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(12, 99);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(72, 25);

this->label3->TabIndex = 4;

this->label3->Text = L"Ответ";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(12, 134);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(209, 25);

this->label4->TabIndex = 5;

this->label4->Text = L"Постфиксная запись";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(730, 172);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button1);

this->Name = L"MyForm";

this->Text = L"MyForm";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

//стандартная строка

std::string infix;

//конвертация строки String^ в std::string

infix = msclr::interop::marshal\_as<std::string>(textBox1->Text);

MyPostfix temp(infix);

auto a = temp.GetPostfix();

string t;

for (auto b : a) {

t += b + " ";

}

label2->Text= msclr::interop::marshal\_as<String^>(t);

vector<string> operands = temp.GetOperands();

map<string, double> values;

double val;

for (const auto& op : operands)

{

try {

double val = stod(op);

values[op] = val;

}

catch (...) { // ловит любой тип

cout << "Введите значение " << op << ": ";

cin >> val;

values[op] = val;

}

}

//вычисление и вывод на форму

label1->Text = Convert::ToString(temp.Calculate(values));

}

};

}