МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

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»

Профиль подготовки: «Общий»

**Отчет по лабораторной работе**

на тему:

**«Вычисление арифметических выражений с**

**использованием стеков»**

**Выполнил:** студент группы 3822Б1ПР2

Чернова Наталья Алексеевна

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

Подпись

Нижний Новгород  
2023

**Содержание**

[1. ВВЕДЕНИЕ 3](#_Toc4)

[2. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc5)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 5](#_Toc6)

[4. РУКОВОДСТВО ПРОГРАММИСТА 6](#_Toc7)

[4.1 Описание структуры программы 6](#_Toc8)

[4.2 Описание структур данных 6](#_Toc9)

[5. ОПИСАНИЕ АЛГОРИТМОВ 8](#_Toc10)

[6. ЭКСПЕРИМЕНТЫ 10](#_Toc11)

[7. ЗАКЛЮЧЕНИЕ 11](#_Toc12)

[8.ЛИТЕРАТУРА 12](#_Toc13)

[9.ПРИЛОЖЕНИЕ 13](#_Toc14)

# 1. ВВЕДЕНИЕ

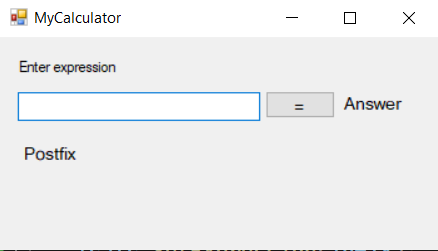
Выражение — это построенная по принятым в математике правилам комбинация чисел, букв, символов функций, символов операций, скобок и других математических обозначений. Обычно выражения записываются в инфиксной форме, где между операндами стоят знаки операторов. Например (1+2+(12/3)\*5+4)/9. Но в данном случае существует определённый приоритет операций, что упрощает вычисления человеку, но усложнят вычисления для машин. В 1920 годах польский логик Ян Лукасевич разработал форму записи арифметических и логических выражений, в которой операция располагается в выражении слева от ее операндов. Так, арифметическому выражению (a+b\*c)\*(c/d-e) будет соответствовать запись \*+a\*bc-/cde. Такой способ записи называется префиксной (польской или прямой польской) формой арифметического выражения. А в 1950 годах австралийский ученый Чарльз Хэмблин на основе польской нотации разработал форму записи арифметических и логических выражений, в которой операнды располагаются в выражении перед операциями. Так арифметическому выражению (a+b\*c)\*(c/d-e) будет соответствовать запись abc\*+cd/e-\*. Такой способ записи называется постфиксной (обратной польской) формой арифметического выражения. Плюсы постфиксной и префиксной формы заключаются в том, что порядок операций определяется самой записью и компьютер может вычислять пример последовательно. Хранить промежуточные значения вычислений удобно в стеках – абстрактном типе данных, где элементы хранятся по принципу LIFO (от англ. “last in – first out”, «последним вошёл, первым вышел»). Поэтому в данной работе мы напишем реализацию вычисления арифметического выражения с использованием стеков.

# 2. ПОСТАНОВКА ЗАДАЧИ

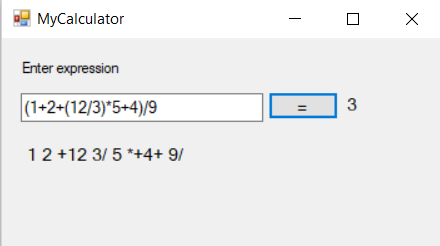
Задача данной работы – реализовать шаблонный класс стек, а также класс калькулятор, обладающий методами приведения выражения в постфиксную форму и вычисления значения данного выражения.

# 3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Изначально у нас есть окно, в которое нужно вводить математическое выражение.



После ввода данных и нажатия на кнопку «=» появится ответ на данное выражение и запись выражения в постфиксной форме.



# 4. РУКОВОДСТВО ПРОГРАММИСТА

## 4.1 Описание структуры программы

Программа состоит из нескольких проектов:

* Проект stack, содержащий файл main.cpp, где находится проверка работоспособности калькулятора
* Проект stacklib, содержащий файл MyStack.h, где находится реализация классов стека и калькулятора
* Проект stacktest, содержащий тесты для проверки корректности работы классов стека и калькулятора. Реализованы с помощью фреймворка GoogleTest.
* Графический проект MyCalculator, содержащий реализацию вывода на экран

## 4.2 Описание структур данных

TStack (шаблонный класс)

protected:

int size; //размер стека

int top; //индекс крайнего элемента

T\* mas; //массив размера size

public:

TStack(int n = 0); //конструктор

TStack(TStack<T>& stack); //конструктор копирования

~TStack(); //деструктор

void Push(T a); //добавить элемент в стек

T Pop(); //удалить элемент из стека

T TopView(); //получить крайний элемент

int GetSize(); //получить размер

int GetTop(); //получить индекс крайнего элемента

bool IsEmpty(); //проверка, пуст ли стек

bool IsFull(); //проверка, полон ли стек

TStack& operator=(const TStack<T>& stack); //оператор присваивания

friend ostream& operator<<(ostream& ostr, const TStack<T>& stack) noexcept //оператор вывода

friend istream& operator>>(istream& istr, const TStack<T>& stack) noexcept //оператор ввода

TCalculator

private:

string infix; //представление строки в обычном виде

string postfix; //представление строки в постфиксном виде

TStack<char> st; //стек с операндами

TStack<char> stop; //стек с операторами

public:

TCalculator(); //конструктор

TCalculator(string str); //конструктор

void SetExpression(string expression); //установление выражения

string GetExpression(); //возвращение выражения (в обычной форме)

string GetPostfix(); //возвращение выражения в постфиксной форме

int Priority(char elem); //установление приоритета для операций

void ToPostfix(); //перевод выражения в постфиксную форму

double CalculatePostfix(); //вычисление выражения в постфиксной форме

double CalculateAll(); //общее вычисление выражения

# 5. ОПИСАНИЕ АЛГОРИТМОВ

Удобный формат алгоритма преобразования инфиксной строки в обратную польскую запись:

Рассматриваем поочередно каждый символ:

1. Если этот символ - число (или переменная), то просто помещаем его в выходную строку.
2. Если символ - знак операции (+, -, \*, /), то проверяем приоритет данной операции:
   1. Умножение и деление имеют наивысший приоритет (например, он равен 3)
   2. Сложение и вычитание имеют меньший приоритет (например, 2)
   3. Наименьший приоритет у открывающейся скобки (1)

Получив один из этих символов мы должны проверить стек:

1. 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
2. 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
3. Если текущий символ - открывающая скобка, то помещаем ее в стек
4. 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

# 6. ЭКСПЕРИМЕНТЫ

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10

2. Процессор: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx, 2100 МГц

3. Версия Visual Studio: 2022

|  |  |  |
| --- | --- | --- |
| Кол-во элементов (n) | Время работы оператора сложения (ms) | Время работы оператора умножения (ms) |
| 10000 | 4 | 5 |
| 30000 | 14 | 16 |
| 60000 | 28 | 32 |

Здесь представлено время работы проекта, высчитывающего сумму и произведение n элементов. Сложность операторов примерно равно O(n).

# 7. ЗАКЛЮЧЕНИЕ

В ходе работы были реализованы классы стека (TStack) и калькулятора (TCalculator). Также был реализован визуальный интерфейс, для более удобного представления результатов работы.

# 8.ЛИТЕРАТУРА

[Стек — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA)

[Выражение (математика) — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))

[Обратная польская запись — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C)

[Pract\_ADS.pdf - Google Диск](https://drive.google.com/file/d/1aZEfnRSgA7IhMccdLTvbJ5noZlXPC1BK/view)

Лабораторный практикум. Составители:Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

# 9.ПРИЛОЖЕНИЕ

MyStack.h

//#ifndef \_MY\_STACK\_

#define \_MY\_STACK\_

#include <iostream>

#include <string>

using namespace std;

template <class T>

class TStack {

protected:

int size;

int top;

T\* mas;

public:

TStack(int n = 0);

TStack(TStack<T>& stack);

~TStack();

void Push(T a);

T Pop();

T TopView();

int GetSize();

int GetTop();

//операторы вводы и выводы

bool IsEmpty();

bool IsFull();

TStack& operator=(const TStack<T>& stack);

friend ostream& operator<<(ostream& ostr, const TStack<T>& stack) noexcept {

ostr << "My Stack: " << endl;

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

ostr << stack.mas[i] << endl;

}

return ostr;

}

friend istream& operator>>(istream& istr, const TStack<T>& stack) noexcept {

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

istr >> stack.mas[i];

}

return istr;

}

};

template <class T>

TStack<T>::TStack(int n) {

if (n > 0)

{

size = n;

top = 0;

mas = new T[size];

}

else

{

throw "Negative size";

}

}

template <class T>

TStack<T>::TStack(TStack<T>& stack) {

if (stack.mas == nullptr)

{

size = 0;

top = 0;

mas = nullptr;

}

else

{

size = stack.size;

top = stack.top;

mas = new T[size];

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

mas[i] = stack.mas[i];

}

}

}

template <class T>

TStack<T>::~TStack() {

delete[]mas;

size = 0;

top = 0;

}

template <class T>

bool TStack<T>::IsEmpty() {

if (top == 0) {

return true;

}

else {

return false;

}

}

template <class T>

bool TStack<T>::IsFull() {

if (top == size) {

return true;

}

else {

return false;

}

}

template <class T>

void TStack<T>::Push(T el) {

if (IsFull()) {

throw "Stack is full";

}

mas[top] = el;

top++;

}

template <class T>

T TStack<T>::Pop() {

if (IsEmpty()) {

throw "Stack is empty";

}

top--;

return mas[top];

}

template <class T>

int TStack<T>::GetSize() {

return size;

}

template <class T>

int TStack<T>::GetTop() {

return top;

}

template <class T>

T TStack<T>::TopView() {

if (IsEmpty()) {

throw "Stack is empty";

}

return mas[top - 1];

}

template <class T>

TStack<T>& TStack<T>::operator=(const TStack<T>& stack) {

if (this == &stack) {

return \*this;

}

delete[]mas;

size = stack.size;

top = stack.top;

mas = new T[size];

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

mas[i] = stack.mas[i];

}

}

class TCalculator {

private:

string infix;

string postfix;

TStack<char> st;

TStack<char> stop;

public:

TCalculator() :st(25), stop(25)

{

infix = "";

postfix = "";

}

TCalculator(string str) : st(str.length()), stop(str.length()) {

infix = str;

postfix = "";

}

void SetExpression(string expression) {

infix = expression;

}

string GetExpression() {

return infix;

}

string GetPostfix() {

return postfix;

}

int Priority(char elem);

void ToPostfix();

double CalculatePostfix();

double CalculateAll();

};

int TCalculator::Priority(char elem) {

switch (elem) {

case'(':

case')':return 1;

case'+':

case'-':return 2;

case'\*':

case'/':return 3;

default: throw elem;

}

}

void TCalculator::ToPostfix() {

postfix = "";

string src = "(" + infix + ")";

char elem = '!';

unsigned int i = 0;

while (i < src.size()) {

if (src[i] >= '0' && src[i] <= '9' || src[i] == '.') {

postfix += src[i];

}

else if (src[i] == '+' || src[i] == '-' || src[i] == '\*' || src[i] == '/') {

postfix += " ";

elem = st.Pop();

while (Priority(elem) >= Priority(src[i])) {

postfix += elem;

elem = st.Pop();

}

st.Push(elem);

st.Push(src[i]);

}

else if (src[i] == '(') {

st.Push(src[i]);

}

else if (src[i] == ')') {

elem = st.Pop();

while (elem != '(') {

postfix += elem;

elem = st.Pop();

}

}

i++;

}

}

double TCalculator::CalculatePostfix() {

unsigned int i = 0;

while (i < postfix.size()) {

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

double b = st.Pop();

double a = st.Pop();

switch (postfix[i]) {

case '+': {

st.Push(a + b);

break;

}

case '-': {

st.Push(a - b);

break;

}

case '\*': {

st.Push(a \* b);

break;

}

case '/': {

st.Push(a / b);

break;

}

}

}

else if (postfix[i] >= '0' && postfix[i] <= '9' || postfix[i] == '.') {

size\_t idx;

double tmp = stod(&postfix[i], &idx); //перевод строки в число (число double, количество символов) например stod("6.123, &idx) выведет 6.123 5

st.Push(tmp);

i += idx - 1;

}

i++;

}

return st.Pop();

}

double TCalculator::CalculateAll() {

ToPostfix();

return CalculatePostfix();

}

MyCalculator.h

#pragma once

#include <iostream>

#include <string>

#include <msclr\marshal\_cppstd.h>

#include <MyStack.h>

//#include "..\All\_labs\stacklib\MyStack.h"

namespace Calculator {

using namespace std;

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 MyCalculator

/// </summary>

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

{

public:

MyCalculator(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyCalculator()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

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

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

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

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

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

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

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

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(17, 21);

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

this->label1->Size = System::Drawing::Size(107, 16);

this->label1->TabIndex = 0;

this->label1->Text = L"Enter expression";

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10));

this->textBox1->Location = System::Drawing::Point(20, 54);

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

this->textBox1->Size = System::Drawing::Size(258, 26);

this->textBox1->TabIndex = 1;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10));

this->label2->Location = System::Drawing::Point(23, 103);

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

this->label2->Size = System::Drawing::Size(60, 20);

this->label2->TabIndex = 2;

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

//

// answer

//

this->answer->AutoSize = true;

this->answer->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10));

this->answer->Location = System::Drawing::Point(364, 54);

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

this->answer->Size = System::Drawing::Size(65, 20);

this->answer->TabIndex = 3;

this->answer->Text = L"Answer";

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10));

this->button1->Location = System::Drawing::Point(284, 53);

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

this->button1->Size = System::Drawing::Size(74, 27);

this->button1->TabIndex = 4;

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

this->button1->UseVisualStyleBackColor = true;

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

//

// MyCalculator

//

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

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

this->ClientSize = System::Drawing::Size(470, 209);

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

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

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

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

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

this->Name = L"MyCalculator";

this->Text = L"MyCalculator";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

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

string infix;

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

if (infix.size() == 0) return;

TCalculator Tcalc;

try {

Tcalc.SetExpression(infix);

Tcalc.ToPostfix();

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

answer->Text = Convert::ToString(Tcalc.CalculateAll());

}

catch (...) {

answer->Text = Convert::ToString("ERROR");

}

}

};

}

MyCalculator.cpp

//#include "..\All\_labs\stacklib\MyStack.h"

#include "MyCalculator.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

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

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Calculator::MyCalculator form;

Application::Run(% form);

}