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 №3**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 3822Б1ПР1

Ахметов Д.Д.

**Проверил**:

доцент кафедры МОСТ, к.т.н.,

Сысоев А.В.

Нижний Новгород

2023

# Содержание

[Введение 3](#_Toc118232948)

[1. Постановка задачи 4](#_Toc118232949)

[2. Руководство пользователя 5](#_Toc118232950)

[3. Руководство программиста 6](#_Toc118232951)

[3.1. Описание структуры программы 6](#_Toc118232952)

[3.2. Описание алгоритмов 6](#_Toc118232953)

[4. Результаты экспериментов 7](#_Toc118232954)

[Заключение 8](#_Toc118232955)

[Литература 9](#_Toc118232956)

[Приложение 10](#_Toc118232957)

# Введение

В программировании, вычисление арифметических операций является одной из фундаментальных задач. При работе с числами и математическими выражениями, правильная и эффективная обработка операций играет ключевую роль. В языке программирования C++, существуют различные способы записи и вычисления арифметических выражений, одним из которых является постфиксная форма записи выражений.

Постфиксная форма записи выражений (также известная как обратная польская запись) представляет собой способ записи математических выражений, где операторы следуют после операндов. Например, вместо выражения "2+3" в постфиксной форме оно будет записано как "23+".

Одной из основных причин использования постфиксной формы записи выражений является её простота и удобство в вычислении. В отличие от инфиксной формы записи (стандартного математического представления), где приоритет операций определяется скобками и правилами приоритета операторов, постфиксная форма не требует скобок и в ней операторы вычисляются в порядке их появления. Постфиксная форма записи выражений также обладает преимуществами в плане удобства хранения и передачи выражений в компьютерных системах. Она может быть легко представлена в виде строки символов или последовательности элементов без необходимости использования дополнительных данных, таких как скобки или операторы приоритета.

Кроме того, постфиксная форма записи выражений также позволяет избежать проблемы, связанной с приоритетом операторов, которая может возникнуть при использовании инфиксной формы. Это облегчает процесс вычисления выражений и делает код более читабельным и понятным.

В результате, использование постфиксной формы записи выражений предоставляет нам простой и эффективный способ вычисления арифметических операций, минимизируя сложность и позволяя работать с математическими выражениями более естественным и интуитивным способом. В данной лабораторной работе мы будем изучать основные алгоритмы и методы вычисления арифметических выражений в постфиксной форме, а также реализовывать соответствующий код на языке программирования C++.

# Постановка задачи

Цель данной работы — разработка структуры данных Стек и ее использование для расчета арифметических выражений с использованием обратной польской записи (постфиксной формы).

Выполнение работы предполагает решение следующих задач:

* Разработка интерфейса шаблонного класса TStack.
* Реализация методов шаблонного класса TStack.
* Разработка интерфейса класса TPostfix для работы с постфиксной формой.
* Реализация методов класса TPostfix.
* Разработка и реализация тестов для классов TStack и TPostfix на базе Google Test.
* Публикация исходных кодов в личном репозитории на GitHub.

# Руководство пользователя

Заголовочный файл "TStack" в C++ определяет класс TStack, который реализует структуру данных "Стек" по принципу "последний вошел - первый вышел" (LIFO - last in, first out). TStack включает в себя различные методы и свойства, позволяющие выполнять основные операции со стеком, такие как добавление элемента в стек (push), удаление элемента из стека (pop), получение последнего элемента без удаления (top), проверка, является ли стек пустым (isEmpty) и другие.

Класс “Postfix” представляет собой интерфейс для проведения вычислений с использованием польской нотации. Пользователем задаётся выражение в инфиксной форме с использованием переменных, а затем значения переменных. При компиляции программы пользователь получает результат постфиксную форму выражения и его результат.

# Руководство программиста

## Описание структуры программы

Структура проекта:

* gtest — библиотека Google Test.
* base — каталог с основным проектом ЛР.
* base\_test — каталог с проектом с модульными тестами.

Служебные файлы:

* .gitignore — перечень расширений файлов, игнорируемых Git при добавлении файлов в репозиторий.

## Описание алгоритмов

Алгоритм состоит из следующих действий:

1. Создание пустого стека операторов.

2. Создание пустой строки для хранения выходного постфиксного выражения.

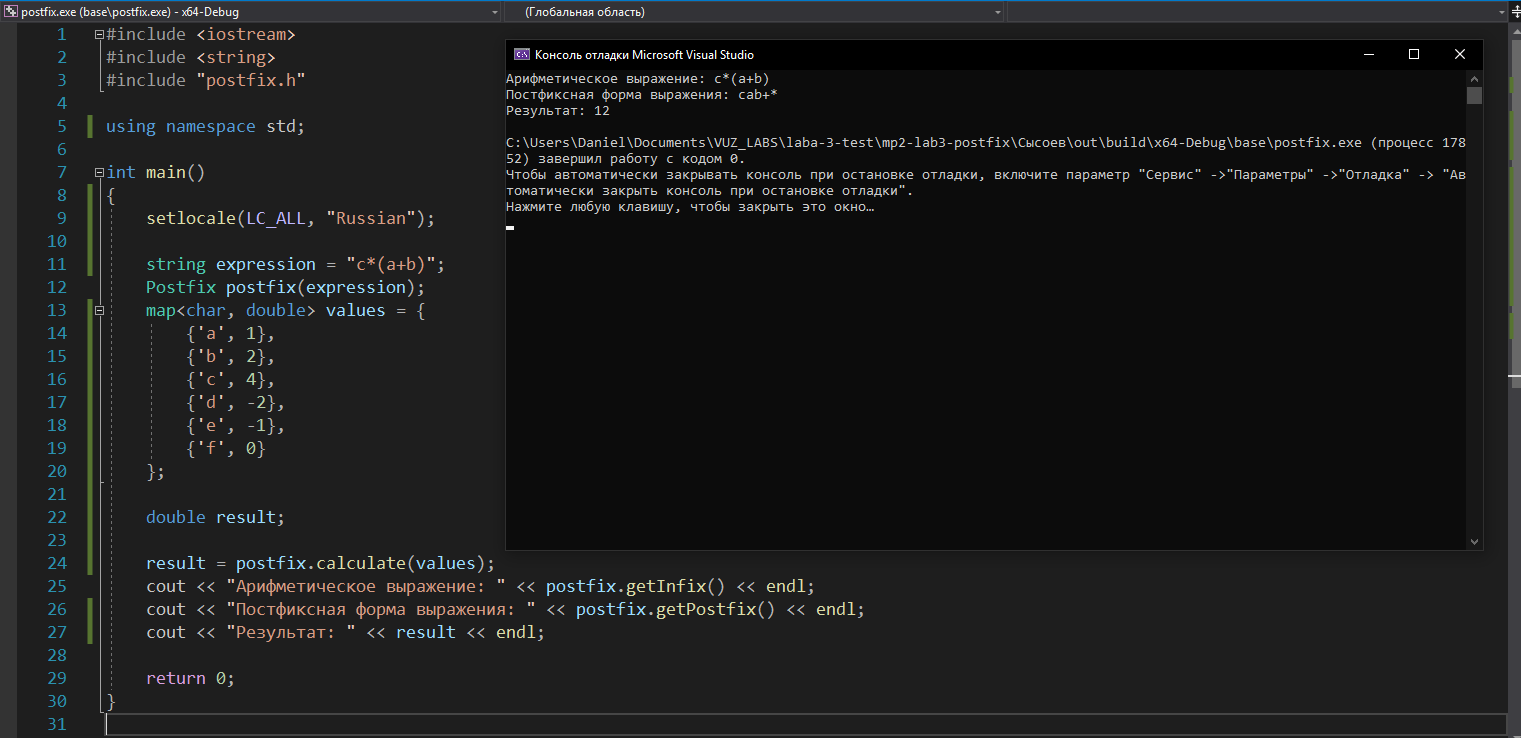
3. Разбор исходного инфиксного выражения слева направо:

* Если лексема - операнд (число или переменная), добавить его к выходному выражению.
* Если лексема - открывающаяся скобка "(", поместить ее в стек.
* Если лексема - закрывающаяся скобка ")", операторы извлекаются из стека и добавляются к выходному выражению до тех пор, пока в стеке не встретится открывающаяся скобка. Затем открывающаяся скобка удаляется из стека.
* Если лексема - оператор, проверяется его приоритет. Если приоритет оператора меньше или равен приоритету оператора на вершине стека, операторы извлекаются из стека и добавляются к выходному выражению, пока это возможно. Затем текущий оператор помещается в стек.

1. После обработки всех символов входного выражения, все оставшиеся операторы извлекаются из стека и добавляются к выходному выражению.
2. Выходное выражение будет являться постфиксной формой исходного выражения.

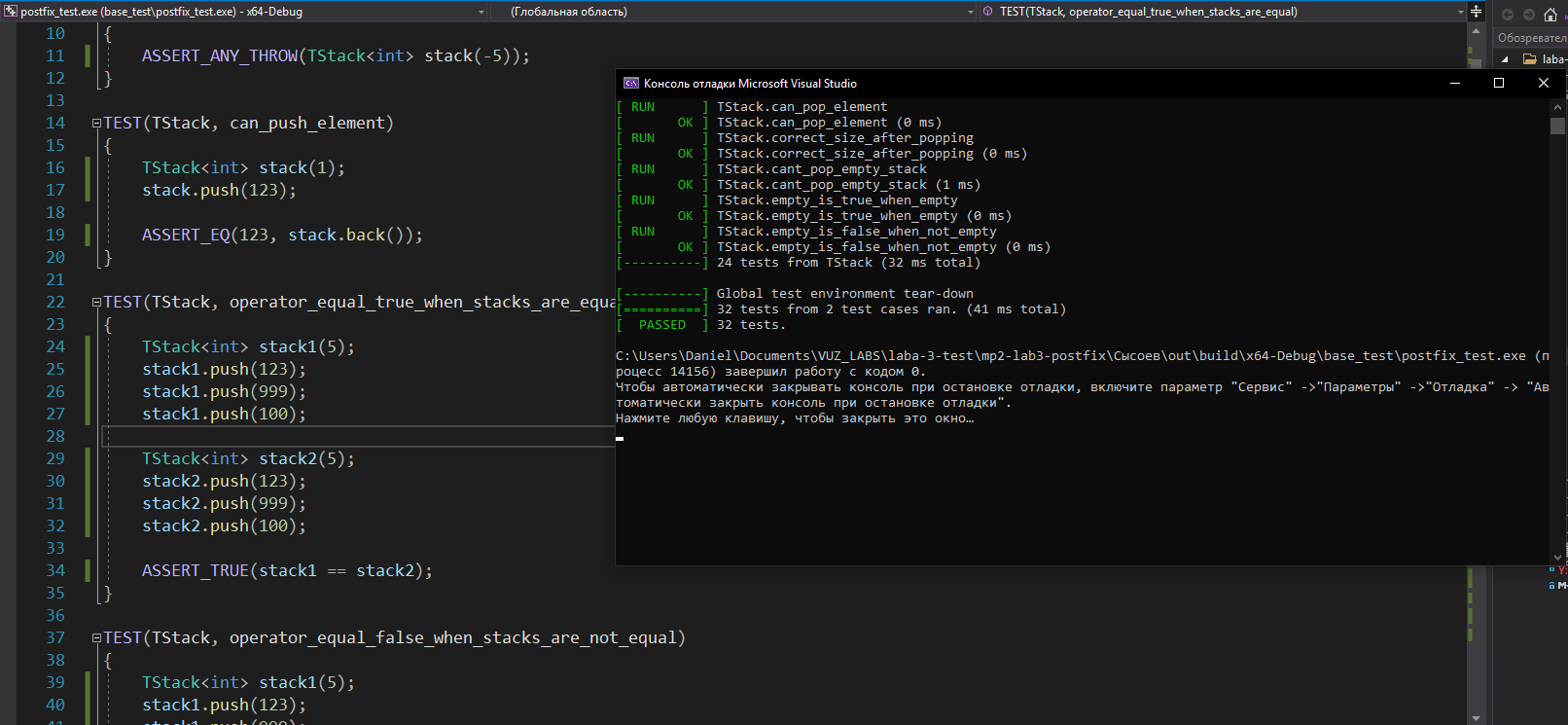
# Результаты экспериментов

Как пример арифметической операции, введём операцию “c\*(a+b)”, что равно “4\*(1+2)”. Программа верно выводит постфиксную форму введённого выражения и результат:

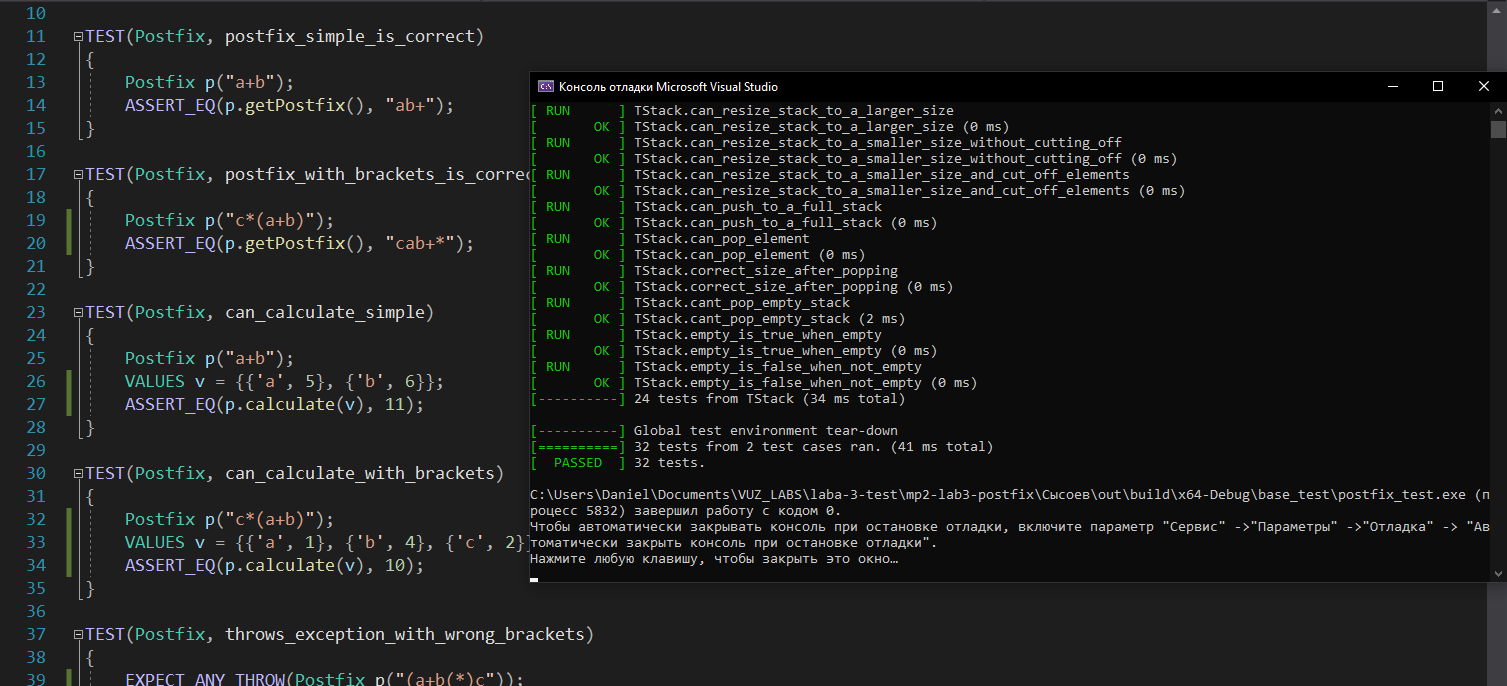


*Рис. 1 – Использование класса Postfix*

Также программа успешно справляется со всеми реализованными на базе GoogleTest тестами:



*Рис. 2 – Успешное прохождение теста test\_tpostfix*



*Рис. 2 – Успешное прохождение теста test\_tstack*

# Заключение

В ходе работы мы успешно реализовали структуры данных Стек для расчета арифметических выражений с использованием обратной польской записи, а также реализовали тесты на базе GoogleTest для проверки нашей программы.

Изучение основных алгоритмов и методов вычисления в постфиксной форме поможет расширить наши знания и навыки в области математических вычислений. В результате, мы сможем более естественным и интуитивным способом работать с математическими выражениями, быстро получая результаты и улучшая эффективность нашего кода.

# Литература

1. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). – 2-е изд. – Москва: Вильямс, 2007. – Т. 3. – 832 с.
2. Страуструп Бьерн Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2017 г. – 1136 с.
3. Сайт Algolist. Сортировка выбором – <http://algolist.manual.ru/sort/select_sort.php>.
4. Сайт Algolist. Сортировка пузырьком – <http://algolist.manual.ru/sort/bubble_sort.php>.

…

# Приложение

**Postfix.h**

#ifndef \_\_POSTFIX\_H\_\_

#define \_\_POSTFIX\_H\_\_

#include <string>

#include <iostream>

#include "stack.h"

using namespace std;

class TPostfix

{

string infix;

string postfix;

string operands = "+-\*/";

//int arity[4] = { 2,2,2,2 };

int priority[4] = { 1,1,2,2 };

bool isCorrect(string str);

public:

TPostfix(string inf = "a+b")

{

if (!isCorrect(inf))

throw "Error";

for (int i = 0; i < inf.length(); i++)

if (inf[i] != ' ')

infix += inf[i];

if (infix.length() < 1)

throw "Error";

}

string GetInfix() { return infix; }

string GetPostfix() { return postfix; }

string ToPostfix();

double Calculate(); // Ввод переменных, вычисление по постфиксной форме

};

#endif

**Postfix.cpp**

#include "postfix.h"

#include "stack.h"

bool TPostfix::isCorrect(string str)

{

string unacceptable = "!@#^&.,<>{}[]|=\_?";

int count = 0, k = 0, countOperation = 0, length = str.length();

if ((operands.find(str[0]) != string::npos) || (operands.find(str[length - 1]) != string::npos))

return false;

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