Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе №3**

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

**Выполнил**:студент группы 381703-2

Лугин М.Д.

**Проверил**:

Доцент кафедры МОСТ

к.т.н. Сысоев А.В.

Нижний Новгород

2018

**Содержание**

[Введение 3](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476590)

[Постановка задачи 4](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476592)

[Руководство пользователя 5](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476593)

[Руководство программиста. 7](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476594)

[Описание алгоритмов 7](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476595)

[Литература 10](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476596)

[Приложение 11](file:///C:\Users\Asus\Downloads\Gurov-mp2-lab3-stack-otchet.doc#_Toc529476597)

# Введение

# Выполнение данной лабораторной работы необходимо, чтобы понять как работает и как можно реализовать абстрактный тип данных стек.

**Стек** (англ. *stack* — стопка; читается *стек*) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу *LIFO* (англ. *last in — first out*, «последним пришёл — первым вышел»).

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

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

# Руководство пользователя

# Данная программа написана с помощью программной платформы .NET Framework в среде Microsoft Visual Studio 2017 на языке C++.

После запуска программа выводит список доступных операций и просит пользователя ввести алгебраическое выражение.



Рисунок 1 – Главное меню

После ввода, программа покажет постфиксный вид, но если некоторые значения выражения являются не численными, программа будет требовать от пользователя их ввода.

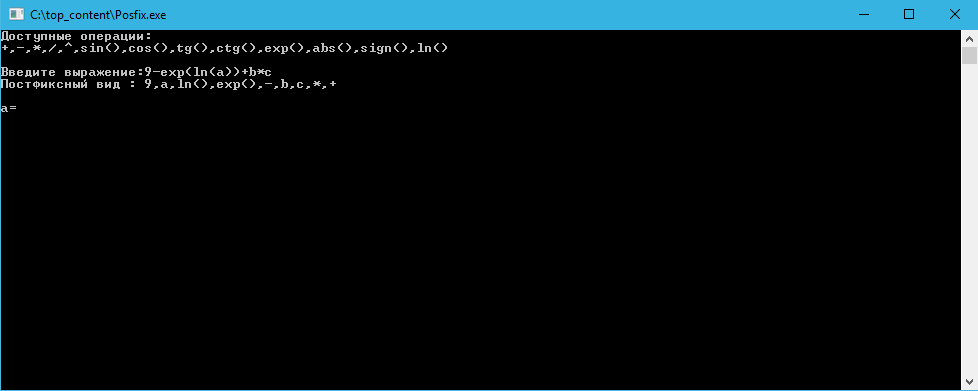


Рисунок 2.1 – Ввод выражения с переменными

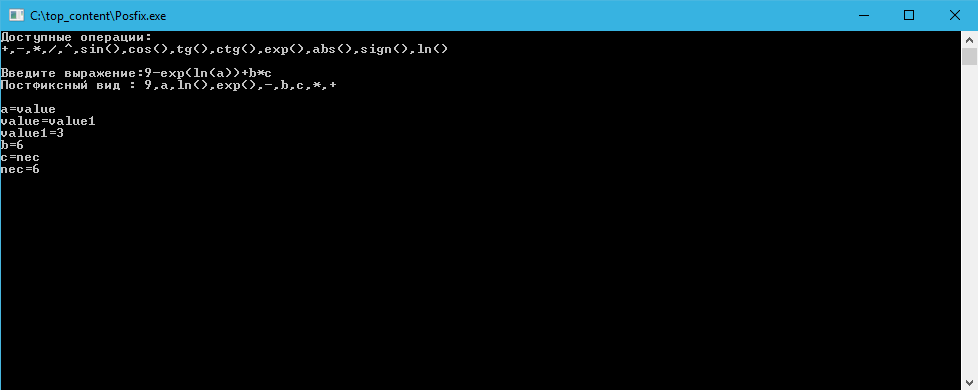


Рисунок 2.2 – доопределение переменных

После получения всех нужных значений программа выведет исходную строку, введённую пользователем, постфиксный вид получившегося выражения и результат вычисления.



Рисунок 3 – Результат

**Руководство программиста**

**Class TStack**

Template <class T> class TStack

**Глобальные переменные:**

const int STACK\_MAX\_SIZE = 100 – максимальный размер стека

**Поля:**

int Lii; - номер последнего элемента в стеке

ElemType Stack[STACK\_MAX\_SIZE]; - массив данных стека

**Методы:**

TStack() - конструктор

TStack( const TStack<ElemType> &TS) – конструктор копирования

~TStack() - деструктор

bool is\_full() – проверка на наличие свободного места в стеке

bool is\_empty() – проверка на пустоту

void push(ElemType Element) – добавить элемент в стек

ElemType pop() – удалить и получить элемент из стека

ElemType top() – получить верхний элемент стека

int size() – размер стека

TStack<ElemType>& operator=(const TStack<ElemType> &TS) – оператор присваивания

**Class Postfix**

**Поля:**

string PRF = {} - префиксный вид

vector<string> OP\_ex1 - доступные одно символьные операции

vector<string> OP\_ex2 - доступные много символьные операции

vector<string> OP\_prf = {} - упорядоченные операции префиксного вида

vector<string> operand = {} - упорядоченные операнды префиксного вида

vector<string> PSTF = {} - все символы в постфиксном виде

string PSTF\_str = {} - постфиксный вид , т.е. сумма всех символов в постфиксном виде

**Методы:**

void fill\_OP\_ex1() – заполнение поля OP\_ex1 операциями

void fill\_OP\_ex2() – заполнение поля OP\_ex2 операциями

bool disassemble(string PRF1) – разделение строчки на операции и опреанды

bool Make\_Postfix() – составление постфиксного вида из операций и операндов

void Make\_PSTF\_str() – составление строчки постфиксного вида

Postfix(string PRF1 = "") – конструктор

Postfix(const Postfix &P) – конструктор копирования

~Postfix() – деструктор

void Set\_prf(string PRF1) – изменяет префиксный вид

void Set\_unknow\_operand(vector<string> Ts) – изменяет первые N неизвестных операторов

bool this\_unknow\_operand(string opn) – это не число?

bool there\_is\_unknow\_value() – здесь есть не числа?

string erase\_gap(string STR1) – очистка префиксного вида от пробелов

double calc() – подсчёт и вывод результата

int check\_level\_OP(string OP) – определяет ступень опреации

vector<string> Get\_unknow\_value() – получить операторы, которые не числа

string Get\_PRF() – получить префиксный вид

vector<string> Get\_OP\_prf() – получить операторы в префиксном порядке

vector<string> Get\_operands() – получить операнды в префиксном порядке

string Get\_Postfix() – получить постфикс

vector<string> Get\_OP\_ex1() – получить одно символьные операции

vector<string> Get\_OP\_ex2() – получить много символьные операции

**Описание алгоритмов**

**­**1)Метод disassemble, разделения инфикса на составляющие: Метод принимает на вход строку инфиксного вида и начинает посимвольный перебор, если встречена операция 1 типа (кроме скобок), то она заносится сразу в вектор операций OP\_prf, если встречена открывающаяся скобка, то: 1.Если следующий символ минус, то если следующими элементами не станут операнд и закрывающаяся скобка, то это ошибка, иначе занести в вектор OP\_prf (-) , а операнд в вектор operand. 2.Запускается проверка символов стоящих перед этой скобкой, если они совпадаю с много символьной операцией, то эта операция заносится в вектор OP\_prf, иначе ошибка, т.к. перед открывающееся скобкой не должен стоять операнд. Далее если встречена закрывающаяся скобка, но при этом не было до этого встречено открывающейся скобки или последним символом была операция, то это ошибка, иначе закрывающая скобка заносится в вектор OP\_prf. В любом другом случае это будет какой-либо символ неизвестный нам, его мы записываем в буфер и идем к следующему символу в инфиксе, если будет встречена какая-либо операция, то из буфера операнд записывается в вектор operand.

2)Метод Make\_Postfix, устанавливает порядок элементов постфиксного вида: Метод работает с векторами OP\_prf и operand, заполняя вектор PSTF, первым делом в PSTF добавляется первый операнд, далее запускается цикл, который идет по операциям в векторе OP\_prf, если это одно символьная операция (кроме скобок) то в PSTF добавляется следующий операнд, а операция добавляется в стек, если ступень операции меньше или равна последней операции в стеке, то все операции выносятся в PSTF, а потом добавляется эта операция в стек, иначе это много символьная операция или скобка, в таком случае нового операнда не требуется, а операция заносится в стек и снижает уровень ступени до минимального, если встречена закрывающая скобка, то все операции до много символьной или открывающейся скобки выносятся из стека в PSTF, а уровень ступени ставится в соответствии с последней операцией в стеке, а если стек после этого пуст, то уровень ступени ставится минимальный.

3) Метод calc, подсчёт результата выражения: Запускается цикл, который идет по элементам вектора PSTF, если это операнд, то он заносится в стек, если операция, то если это много символьная операция, то данная операция выполняется над верхним элементом стека и результат заносится обратно в стек, если операция одно символьная, то она выполняется над двумя верхними элементами стека и результат заносится обратно в стек, по итогу в стеке должен остаться лишь один элемент, его функция и возвращает.

**Заключение**

В результате написания программы было выпито три чашки кофе с молоком, съедено 5 шоколадок Alpen Gold и пролита кока-кола на диван.

А так же написан класс TStack, эмулирующий работу стека, и класс Postfix, который переводит алгебраическое выражение в постфиксный вид и высчитывает его результат, с использование стека.

# Литература

1. Кнут Д. Э. Искусство программирования. Том 1. Основные алгоритмы = The Art of Computer Programming. Volume 1. Fundamental Algorithms / под ред. С. Г. Тригуб (гл. 1), Ю. Г. Гордиенко (гл. 2) и И. В. Красикова (разд. 2.5 и 2.6). — 3. — Москва: Вильямс, 2002. — Т. 1. — 720 с. — ISBN 5-8459-0080-8.
2. Кнут Д. Э. Искусство программирования. Том 2. Получисленные алгоритмы = The Art of Computer Programming. Volume 2. Seminumerical Algorithms / под ред. Л. Ф. Козаченко (гл. 3, разд. 4.6.4 и 4.7), В. Т. Тертышного (гл. 4) и И. В. Красикова (разд. 4.6). — 3. — Москва: Вильямс, 2001. — Т. 2. — 832 с. — ISBN 5-8459-0081-6.
3. 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. — ISBN 5-8459-0082-1.
4. Кнут Д. Э. Искусство программирования, том 4, A. Комбинаторные алгоритмы, часть 1 = The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1 / под ред. Ю. В. Козаченко. — 1. — Москва: Вильямс, 2013. — Т. 4. — 960 с. — ISBN 978-5-8459-1744-7.