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

Отчёт по учебной практике

Вычисление арифметических выражений

Выполнил:

студент гр. 381806-01

Шерстнев И.И.

Проверил:

к. т. н., доцент кафедры МОСТ

Кустикова В.Д.

Нижний Новгород

2019 г.

Содержание

[Введение 3](#_Toc28240487)

[Постановка задачи 4](#_Toc28240488)

[Руководство пользователя 5](#_Toc28240489)

[Действия, приводящие к ошибке 5](#_Toc28240490)

[Руководство программиста 7](#_Toc28240491)

[Описание структуры программы 7](#_Toc28240492)

[Описание структур данных 7](#_Toc28240493)

[Класс Stack 7](#_Toc28240494)

[Класс Calculator 9](#_Toc28240495)

[Описание алгоритмов 10](#_Toc28240496)

[Организация работы стека 10](#_Toc28240497)

[Перевод в постфиксную форму 10](#_Toc28240498)

[Вычисление арифметического выражения 16](#_Toc28240499)

[Заключение 21](#_Toc28240500)

[Литература 22](#_Toc28240501)

[Приложения 23](#_Toc28240502)

[Приложение 1. Исходный код основной функции 23](#_Toc28240503)

[Приложение 2. Класс TStack 23](#_Toc28240504)

[Приложение 3. Класс Calculator 24](#_Toc28240505)

[Calculator.h 24](#_Toc28240506)

[Calculator.cpp 25](#_Toc28240507)

# Введение

Лабораторная работа направлена на практическое освоение динамической структуры данных стек. С этой целью в лабораторной работе изучается способ реализации стека на массиве и разрабатываются методы и программы решения ряда задач с использованием стеков. В качестве области приложений выбрана тема для вычисления арифметических выражений, возникающей при трансляции программ на языке программирования высокого уровня в исполняемые программы.

# Постановка задачи

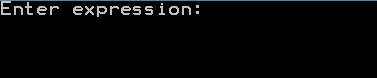
Задача: разработать алгоритм и составить программ для перевода арифметического выражения в постфиксную форму записи.

Входные данные: арифметическое выражение

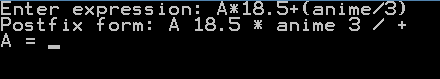
Выходные данные: арифметическое выражение в постфиксной форме записи и значение данного выражения.

# Руководство пользователя

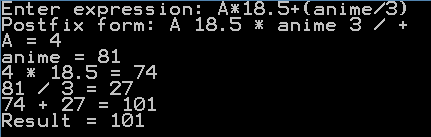
1. Запустить файл **TStack.exe**. Перед вами откроется консольное приложение с предложением ввести инфиксную форму арифметического выражения (рис. 1).



1. Запуск программы
2. После будет напечатана постфиксная форма записи арифметического выражения и вам предложат ввести значения переменных, присутствующих в арифметическом выражении (рис. 2).



1. Ввод значений переменных
2. После ввода значений переменных, будет напечатан результат выражения. (см. рис. 3)



1. Значение выражения

### Действия, приводящие к ошибке

1. Введена постфиксная форма, вместо инфиксной формы.
2. Не каждой открывающей скобке соответствует закрывающая скобка.
3. Введены два подряд операнда.
4. Введены две подряд операции.
5. Первый символ выражения - операция.
6. При подсчете выражение, выполняется деление на ноль.
7. Закрывающая скобка введена раньше, чем открывающая скобка.

Пример обработки ошибок. (см.рис. 4)



1. Пример ошибки

# Руководство программиста

## Описание структуры программы

* TStack.h – модуль с классом, содержащий описание и реализацию функций стека;
* Calculator.h, Calculator.cpp – модуль с классом, реализующий обработку арифметических выражений;
* main.cpp – модуль программы тестирования.

## Описание структур данных

### Класс Stack

#### Объявление класcа Stack

template<typename V>

class TStack {

size\_t size;

size\_t top;

V\* elems;

public:

TStack(size\_t);

TStack(const TStack&);

~TStack();

bool Empty()const;

bool Full()const;

void Push(V);

void Pop();

V Top()const;

TStack&operator=(const TStack&);

};

Тип “V” описывает тип значения элементов стека.

#### Поля

* size\_t size– реальный размер стека
* size\_t top–позиция вершины стека
* V\* elems– указатель на первый элемент массива

#### Методы

TStack(size\_t)

**Назначение:** Обеспечение выделение динамической памяти в куче.

**Входные параметры:** Размер стека

**Выходные данные:** Отсутствуют

TStack(constTStack&)

**Назначение:** Инициализирует объекты, копируя значения элементов из объекта того же типа.

**Входные параметры:** Константная ссылка на объект типа TStack.

**Выходные данные:** Отсутствуют

~TStack()

**Назначение:** Выполняет освобождение выделенной памяти.

**Входные параметры:** Отсутствуют

**Выходные данные:** Отсутствуют

size\_t GetSize()const

**Назначение:** Получение значения размера стека.

**Входные параметры:** Отсутствуют

**Выходные данные:** Значение переменной size

size\_t GetTop()const

**Назначение:** Проверка стека на полноту.

**Входные параметры:** Отсутствуют

**Выходные данные:** Значение переменной top

bool Empty()const

**Назначение:** Проверка стека на пустоту.

**Входные параметры:** Отсутствуют

**Выходные данные:** Значение типа boolean

bool Full()const

**Назначение:** Проверка стека на полноту.

**Входные параметры:** Отсутствуют

**Выходные данные:** Значение типа Boolean

void Push(V)

**Назначение:** Добавление элемента в стек.

**Входные параметры:** Элемент типа V

**Выходные данные:** Отсутствуют

voidPop()

**Назначение:** Изымает верхний элемент из стека

**Входные параметры:** Отсутствуют

**Выходные данные:** Отсутствуют

V Top()const

**Назначение:** Возвращает значение верхнего элемента стека, не изменяя стек

**Входные параметры:** Отсутствуют

**Выходные данные:** Значение верхнего элемента стека

### Класс Calculator

static class Calculator {

public:

struct Operand {

std::string name;

double value;

Operand\* next;

};

static std::string ToPostfixForm(const std::string&);

static double Calculate(const std::string&, Operand\*&);

static void Fill(Operand\*&, const std::string&);

private:

enum class TypeSymbol {

operation, operand, number, space, openingBracket, closingBracket, invalid

};

static void AddUnique(const std::string&, Operand\*&);

static bool IsVarExist(const std::string&, Operand\*);

static double GetValue(const std::string&, Operand\*);

static TypeSymbol GetType(const char);

static int PriorityOperations(const char);

static bool IsNumber(Operand\*);

};

Содержит:

* Struct Operand для хранения имени переменной, ее значения и указателя на следующий элемент
* Enum class TypeSymbol для определения типа символа;
* Статические методы для обработки арифметического выражения.

#### Описание методов класса:

static voidAddUnique(const std::string, Operand\*&)

**Назначение:** Добавление уникальной переменной в связный список

**Входные параметры:** Имя переменной, адрес указателя на начало списка

**Выходные данные:** Отсутствуют

static bool IsVarExist(const std::string, Operand\*)

**Назначение:** Проверка существования элемента в связном списке по имени

**Входные параметры:** Имя искомой переменной, указатель на начало списка

**Выходные данные:** Значение типа Boolean

Static void Fill(Operand\*&, const std::string)

**Назначение:** Заполнение связного списка именами переменных

**Входные параметры:** Адрес указателя на начало списка, постфиксная форма

**Выходные данные:** Отсутствуют

Static double GetValue(const std::string, Operand\*)

**Назначение:** Получает значение переменной по позиции в связном списке

**Входные параметры:** Имя искомой переменной, указатель на начало списка

**Выходные данные:** Значение переменной

Static bool IsNumber(Operand\*)

**Назначение:** Проверка: является ли операнд числом

**Входные параметры:** Операнд

**Выходные данные:** Значение типа Boolean

Static std::stringToPostfixForm(const std::string&)

**Назначение:** Перевод выражения в постфиксную форму

**Входные параметры:** Исходное выражение

**Выходные данные:** Постфиксная форма

static double Calculate(const std::string, Operand\*)

**Назначение:** Подсчет значения выражения

**Входные параметры:** Постфиксная форма, указатель на начало списка

**Выходные данные:** Значение выражения

Static TypeSymbol GetType(const char)

**Назначение:** Определяет тип символа

**Входные параметры:** Символ

**Выходныеданные:** Тип символа

Static int PriorityOperations(const char)

**Назначение:** Определяет приоритет операции

**Входные параметры:** Символ операции

**Выходные данные:** Значение приоритета операции

## Описание алгоритмов

### Организация работы стека

Стек – структура данных, состоящая, в нашем случае, из массива, переменной реального размера массива и переменной позиции верхнего элемента стека в этом массиве. Стек обладает функциями добавления, изъятия элемента, а также просмотра верхнего элемента. Элементы стека располагаются в массиве по принципу last in – first out (LIFO), т. е. последний добавленный элемент будет изъят первым. Чтобы избежать попыток доступа в области вне массива существуют функции проверки массива на пустоту и полноту.

### Перевод в постфиксную форму

Данный алгоритм основан на использовании стека.

На вход алгоритма поступает арифметическое выражение (строка символов), на выходе постфиксное арифметическое выражение (строка символов).

Каждой операции и скобками приписывается приоритет.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Знак операции | ( | ) | +- | \*/ |
| Приоритет | 0 | 1 | 2 | 3 |

1. Создать Стек №1 (для операций) и Стек №2 (для операндов)

2. Входная строка просматривается посимвольно слева направо до достижения конца строки, содержимое строки попадает либо в Стек №1 (для операций), либо в Стек №2 (для операндов и промежуточные постфиксные формы):

* Если пришел операнд, то кладем его в Стек №2;
* Если пришла левая скобка, то кладем ее в Стек №1.
* Если пришла правая скобка, то изымаем операции из Стека №1 и перекладываем их в Стек №2, пока не встретим левую скобку. После удаляем ее.
* Если пришла операция, то смотрим приоритет операции, которая хранится на верхушке Стека №1. Если приоритет пришедшей операции больше либо равен, чем приоритет операции, которая хранится на верху Стека №1, тогда записываем пришедшую операцию в Стек №1. Иначе перекладываем все операции, приоритет которых больше либо равен приоритету операции на верху Стека №2, из Стека №1 в Стек №2.

3. Если дошли до конца выражения, то необходимо переложить все операции из Стека №1 в Стек №2.

Пример: A+B\*(C-D)/(F+E)+K

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
| ↑ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Операция, кладем в Стек №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  | ↑ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  | ↑ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |

1. Операция кладем в Стек №1 (перед этим проверив приоритет).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  | ↑ |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |

1. Левая открывающая скобка, кладем ее в Стек №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  | ↑ |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **(** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  | ↑ |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **(** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **С** |  |  |  |  |  |  |  |  |  |  |  |

1. Операция кладем в Стек №1, проверяя приоритет.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  | ↑ |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **(** | **-** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **С** |  |  |  |  |  |  |  |  |  |  |  |

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  | ↑ |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **(** | **-** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **С** | **D** |  |  |  |  |  |  |  |  |  |  |

1. Закрывающаяся скобка, перекладываем операции в Стек №2, пока не встретим открывающую скобку, затем изымаем ее из Стека №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  | ↑ |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **С** | **D** | **-** |  |  |  |  |  |  |  |  |  |

1. Операция, проверяем приоритет операции, кладем в Стек №1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  | ↑ |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** |  |  |  |  |  |  |  |  |  |

1. Левая открывающая скобка, кладем ее в Стек №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  | ↑ |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** | **(** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** |  |  |  |  |  |  |  |  |  |

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  | ↑ |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** | **(** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** |  |  |  |  |  |  |  |  |

1. Операция, кладем в Стек №1, проверяя приоритет.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  | ↑ |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** | **(** | **+** |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** |  |  |  |  |  |  |  |

1. Операнд кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  |  | ↑ |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** | **(** | **+** |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** |  |  |  |  |  |  |  |

1. Закрывающаяся скобка, перекладываем операции в Стек №2, пока не встретим открывающую скобку, затем изымаем ее из Стека №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | ↑ |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** | **\*** | **/** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** |  |  |  |  |  |  |

1. Операция, приоритет операции меньше, поэтому перекладываем операции из Стека №1 в Стек №2, затем кладем операцию в Стек №1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ↑ |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** |  |  |  |

1. Операнд, кладем его в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ↑ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 | **+** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** |  |  |

1. Закончили просмотр, извлекаем все операции из Стека №1 и кладем их в Стек №2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **+** | **B** | **\*** | **(** | **C** | **-** | **D** | **)** | **/** | **(** | **F** | **+** | **E** | **)** | **+** | **K** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек № 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Стек №2 | **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |  |

### Вычисление арифметического выражения

1. Создать стек для подсчета выражения
2. Выражение просматривается посимвольно слева направо:

* При обнаружении операнда производится перевод в числовую форму и помещение в стек.
* При обнаружении знака операции происходит извлечение из стека двух значений, которые рассматриваются как операнд1 и операнд2. Выполнение операции происходит в порядке «Операнд2 операция Операнд1». Результат операции помещается в стек.

1. По окончании просмотра всего выражения из стека извлекается окончательный результат.

Пример: ABCD-FE+/\*+K+ (В качестве значений переменных, будем считать имя переменных)

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
| ↑ |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  | ↑ |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  | ↑ |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  | ↑ |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C** | **D** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  | ↑ |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C-D** |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  | ↑ |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C-D** | **F** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  | ↑ |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C-D** | **F** | **E** |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем вСтек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  | ↑ |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **C-D** | **F+E** |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | | |
| Выражение |  | | | | | | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  |  | ↑ |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B** | **(C-D)/(F+E)** |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | |
| Выражение | **(C-D)/(F+E)** | | | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  |  |  | ↑ |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B\*(C-D)/(F+E)** |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
| Выражение | **B\*(C-D)/(F+E)** | | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  |  |  |  | ↑ |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A+B\*(C-D)/(F+E)** |  |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| Выражение | **A+B\*(C-D)/(F+E)** | | | | | | | | |

1. Операнд кладем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  |  |  |  |  | ↑ |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A+B\*(C-D)/(F+E)** | **K** |  |  |  |  |  |  |  |
|  | | | | | | | | | |
| Выражение | **A+B\*(C-D)/(F+E)** | | | | | | | | |

1. Операция, изымаем 2 операнда из Стека и записываем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **-** | **F** | **E** | **+** | **/** | **\*** | **+** | **K** | **+** |
|  |  |  |  |  |  |  |  |  |  |  |  | ↑ |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A+B\*(C-D)/(F+E)+K** |  |  |  |  |  |  |  |
|  | | | | | | | | |
| Выражение | **A+B\*(C-D)/(F+E)+K** | | | | | | | |

# Заключение

Был рассмотрен способ перевода арифметического выражения в постфиксную форму записи, а также вычисления этого выражения при вводе значений переменных. Изучены структуры данных стек, его реализация на массиве, принципы работы, а также операции для работы с ним.

# Литература

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гегреля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.
2. Рабочие материалы к учебному курсу «Методы программирования» (часть 1). Профессор кафедры Программная инженерия, д.т.н., Гегрель В.П. Нижний Новгород, 2015г. – 100с.
3. Справочник по языку C++ [<https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=vs-2019>].

# Приложения

## Приложение 1. Исходный код основной функции

#include "Calculator.h"

#include <iostream>

#include <string>

void main() {

std::string Exp, postExp;

Calculator::Operand\* head = nullptr;

std::cout << "Enter expression: ";

std::getline(std::cin, Exp);

try {

postExp = Calculator::ToPostfixForm(Exp);

std::cout << "Postfix form: " << postExp << std::endl;

}

catch (std::string k) {

std::cout << k << std::endl;

return;

}

try {

Calculator::Fill(head, postExp);

}

catch (std::string k) {

std::cout << k << std::endl;

return;

}

try {

double res = Calculator::Calculate(postExp, head);

std::cout << "Result: " << res << std::endl;

}

catch (std::string k) {

std::cout << k << std::endl;

}

}

## Приложение 2. Класс TStack

#pragma once

template <typename V>

class TStack {

size\_t size;

size\_t top;

V\* elems;

public:

TStack(size\_t);

TStack(const TStack&);

~TStack();

bool Empty() const;

bool Full() const;

void Push(V);

void Pop();

V Top() const;

TStack& operator=(const TStack&);

};

template <typename V>

TStack<V>::TStack(size\_t maxsize) {

if (maxsize == 0) throw "Wrong size";

size = maxsize;

elems = new V[maxsize];

top = 0;

}

template <typename V>

TStack<V>::TStack(const TStack<V>& s) {

this->size = s.size;

this->top = s.top;

elems = new V[size];

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

elems[i] = s.elems[i];

}

}

template <typename V>

TStack<V>::~TStack() {

delete[] elems;

}

template <typename V>

bool TStack<V>::Empty() const {

return (top == 0);

}

template <typename V>

bool TStack<V>::Full() const {

return (top == size);

}

template <typename V>

void TStack<V>::Push(V value) {

if (Full()) throw "Stack overflow";

elems[top++] = value;

}

template <typename V>

void TStack<V>::Pop() {

if (Empty()) throw "Stack is empty";

top--;

}

template <typename V>

V TStack<V>::Top() const {

if (Empty()) throw "Stack is empty";

return elems[top - 1];

}

template<typename V>

TStack<V>& TStack<V>::operator=(const TStack<V>& tmp)

{

if (size != tmp.size)

{

size = tmp.size;

delete[] elems;

elems = new V[size];

}

for (size\_t i = 0; i < size; i++)

elems[i] = tmp.elems[i];

return \*this;

}

## Приложение 3. Класс Calculator

### Calculator.h

#pragma once

#include "TStack.h"

#include <iostream>

#include <string>

#include <cstdlib>

static class Calculator {

public:

struct Operand {

std::string name;

double value;

Operand\* next;

};

static std::string ToPostfixForm(const std::string&);

static double Calculate(const std::string&, Operand\*&);

static void Fill(Operand\*&, const std::string&);

private:

enum class TypeSymbol {

operation, operand, number, space, openingBracket, closingBracket, invalid

};

static void AddUnique(const std::string&, Operand\*&);

static bool IsVarExist(const std::string&, Operand\*);

static double GetValue(const std::string&, Operand\*);

static TypeSymbol GetType(const char);

static int PriorityOperations(const char);

static bool IsNumber(Operand\*);

};

### Calculator.cpp

#include "Calculator.h"

void Calculator::AddUnique(const std::string& op, Calculator::Operand\*& head) {

if (!head) {

head = new Operand;

head->name = op;

head->value = 0.0;

head->next = nullptr;

}

else if(!IsVarExist(op, head)){

Operand\* tmp = head;

while (tmp->next != nullptr)

tmp = tmp->next;

tmp->next = new Operand;

tmp->next->name = op;

tmp->next->value = 0.0;

tmp->next->next = nullptr;

}

}

bool Calculator::IsVarExist(const std::string& k, Calculator::Operand\* tmp) {

while (tmp != nullptr) {

if (tmp->name == k)

return true;

tmp = tmp->next;

}

return false;

}

void Calculator::Fill(Calculator::Operand\*& head, const std::string& postfixForm) {

std::string name = "";

char lastSymbol = '(';

TypeSymbol type, ltype;

for (size\_t i = 0; i < postfixForm.length(); i++) {

type = GetType(postfixForm[i]);

ltype = GetType(lastSymbol);

if (type == TypeSymbol::operand || type == TypeSymbol::number) {

name.append(1, postfixForm[i]);

}

else if (type == TypeSymbol::space && (ltype == TypeSymbol::operand || ltype == TypeSymbol::number)) {

AddUnique(name, head);

name.clear();

}

lastSymbol = postfixForm[i];

}

Calculator::Operand\* tmp = head;

while (tmp != nullptr) {

if (!Calculator::IsNumber(tmp)) {

std::cout << tmp->name << " = ";

std::cin >> tmp->value;

}

tmp = tmp->next;

}

}

double Calculator::GetValue(const std::string& wtf, Calculator::Operand\* head) {

Operand\* tmp = head;

while (tmp != nullptr) {

if (tmp->name == wtf)

return tmp->value;

tmp = tmp->next;

}

throw (std::string)"Operand doesn`t exist";

}

bool Calculator::IsNumber(Calculator::Operand\* op) {

if ((!op->name.empty()) && (op->name.find\_first\_not\_of("0123456789.") == op->name.npos)) {

op->value = atof((op->name).c\_str());

return true;

}

else return false;

}

Calculator::TypeSymbol Calculator::GetType(char temp)

{

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

return TypeSymbol::operation;

if (temp == ')')

return TypeSymbol::closingBracket;

if (temp == '(')

return TypeSymbol::openingBracket;

if (isalpha(temp))

return TypeSymbol::operand;

if ((temp == '0') || (temp == '1') || (temp == '2') || (temp == '3') || (temp == '4') || (temp == '5') || (temp == '6') || (temp == '7') || (temp == '8') || (temp == '9') || (temp == '.'))

return TypeSymbol::number;

if (temp == ' ')

return TypeSymbol::space;

else

return TypeSymbol::invalid;

}

int Calculator::PriorityOperations(const char op)

{

if ((op == '\*') || (op == '/')) return 3;

if ((op == '+') || (op == '-')) return 2;

if (op == ')') return 1;

return 0;

}

std::string Calculator::ToPostfixForm(const std::string& expression)

{

std::string buffer;

int lengthOfExpression = expression.length();

int numberOfOperators = 0, numberOfOperands = 0, numberOfLeftBrackets = 0, numberOfRightBrackets = 0;

TypeSymbol type, ltype;

char lastSymbol = 0;

std::string postfixForm = "";

TStack<char> operators(lengthOfExpression);

TStack<std::string> operands(lengthOfExpression);

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

{

type = GetType(expression[i]);

ltype = GetType(lastSymbol);

if (buffer.length() != 0)

{

if (type == TypeSymbol::operation || type == TypeSymbol::openingBracket || type == TypeSymbol::closingBracket || type == TypeSymbol::space)

{

numberOfOperands++;

operands.Push(buffer);

buffer.clear();

}

}

if (type == TypeSymbol::space);

else if (type == TypeSymbol::operand || type == TypeSymbol::number)

{

if (numberOfOperands == (numberOfOperators + 1))

{

throw (std::string)"Extra operand";

}

buffer.push\_back(expression[i]);

if (i == (lengthOfExpression - 1))

{

numberOfOperands++;

operands.Push(buffer);

buffer.clear();

}

}

else if (type == TypeSymbol::openingBracket)

{

if (ltype == TypeSymbol::operand || ltype == TypeSymbol::number)

{

throw (std::string)"No operation between operand and opening bracket";

}

numberOfLeftBrackets++;

operators.Push(expression[i]);

}

else if (type == TypeSymbol::closingBracket)

{

if (ltype == TypeSymbol::operation || ltype == TypeSymbol::openingBracket) {

throw (std::string)"Right bracket after operation or opening bracket";

}

if (numberOfRightBrackets >= numberOfLeftBrackets) {

throw (std::string)"Extra right bracket";

}

numberOfRightBrackets++;

while (operators.Top() != '(')

{

std::string tmp;

try

{

tmp.push\_back(operators.Top());

operators.Pop();

operands.Push(tmp);

}

catch (std::string k)

{

throw k;

}

}

operators.Pop();

}

else if (operators.Empty())

{

if (lastSymbol == 0)

{

throw (std::string)"Extra operator";

}

operators.Push(expression[i]);

numberOfOperators++;

}

else if (PriorityOperations(expression[i]) > PriorityOperations(operators.Top()))

{

operators.Push(expression[i]);

numberOfOperators++;

}

else

{

while ((!operators.Empty()) && (operators.Top() != '(')

&& (PriorityOperations(expression[i]) <= PriorityOperations(operators.Top())))

{

std::string tmp;

tmp.push\_back(operators.Top());

operators.Pop();

operands.Push(tmp);

}

operators.Push(expression[i]);

numberOfOperators++;

}

if (type != TypeSymbol::space)

lastSymbol = expression[i];

}

if (numberOfOperands != numberOfOperators + 1)

{

throw (std::string)"Wrong amount of operands/operators";

}

if (numberOfLeftBrackets != numberOfRightBrackets)

{

throw (std::string)"Wrong amount of brackets";

}

while (!operators.Empty())

{

std::string tmp;

tmp.push\_back(operators.Top());

operators.Pop();

operands.Push(tmp);

}

TStack<std::string> reverseStackOfOperands(lengthOfExpression);

for (; !operands.Empty(); operands.Pop())

reverseStackOfOperands.Push(operands.Top());

int startPosition = 0;

while (!reverseStackOfOperands.Empty())

{

startPosition = postfixForm.length();

postfixForm.insert(startPosition, reverseStackOfOperands.Top());

reverseStackOfOperands.Pop();

buffer = ' ';

postfixForm.insert(postfixForm.length(), buffer);

}

return postfixForm;

}

double Calculator::Calculate(const std::string& postfixForm, Calculator::Operand\*& head)

{

size\_t sizeForm = postfixForm.size();

TStack<double> calc(sizeForm);

double res = 0.0;

std::string Name;

char lastSymbol = '(';

for (size\_t i = 0; i < sizeForm; i++) {

TypeSymbol type = GetType(postfixForm[i]);

TypeSymbol ltype = GetType(lastSymbol);

if (type == TypeSymbol::operand || type == TypeSymbol::number)

Name.append(1, postfixForm[i]);

else if (type == TypeSymbol::operation) {

double a, b;

try {

b = calc.Top();

calc.Pop();

a = calc.Top();

calc.Pop();

}

catch (std::string k) {

throw (std::string)"Wrong postfix form: " + k;

}

switch (postfixForm[i]) {

case '+':

calc.Push(a + b);

break;

case '-':

calc.Push(a - b);

break;

case '\*':

calc.Push(a \* b);

break;

case '/':

if (b == 0)

throw (std::string)"Division by zero";

calc.Push(a / b);

break;

default:

throw (std::string)"Wrong operation";

break;

}

}

else if ((type == TypeSymbol::space && ltype == TypeSymbol::operand) || (type == TypeSymbol::space && ltype == TypeSymbol::number)) {

try {

calc.Push(GetValue(Name, head));

}

catch (std::string k) {

std::cout << k << std::endl;

}

Name.clear();

}

if(type != TypeSymbol::space)

lastSymbol = postfixForm[i];

}

try {

res = calc.Top();

calc.Pop();

}

catch (std::string k) {

throw k;

}

while (head->next != nullptr) {

Operand\* tmp = head;

while (tmp->next->next != nullptr) {

tmp = tmp->next;

}

delete tmp->next;

tmp->next = nullptr;

}

delete head;

if (calc.Empty())

return res;

else

throw (std::string)"Stack isn`t empty after calculating";

}