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 (стеки)

Выполнил:

студент гр. 381806-01

Тактаев А.А.

Проверил:

к. т. н., доцент кафедры МОСТ

Кустикова В.Д.

Нижний Новгород

2019 г.

Содержание

[Введение 3](#_Toc27957825)

[Постановка задачи 4](#_Toc27957826)

[Руководство пользователя 5](#_Toc27957827)

[Действия, приводящие к ошибке 5](#_Toc27957828)

[Руководство программиста 7](#_Toc27957829)

[Описание структуры программы 7](#_Toc27957830)

[Описание структур данных 7](#_Toc27957831)

[Класс TStack 7](#_Toc27957832)

[Структура Variable 9](#_Toc27957833)

[Структура Variables 9](#_Toc27957834)

[Класс Postfix 10](#_Toc27957835)

[Описание алгоритмов 12](#_Toc27957836)

[Работа Стека 12](#_Toc27957837)

[Перевод в постфиксную форму 13](#_Toc27957838)

[Вычисление Арифметического выражения 18](#_Toc27957839)

[Заключение 21](#_Toc27957840)

[Литература 22](#_Toc27957841)

[Приложения 23](#_Toc27957842)

[Приложение 1. Исходный код основной функции(main) 23](#_Toc27957843)

[Приложение 2. Класс TStack 23](#_Toc27957844)

[Приложение 3. Класс PostfixForm 27](#_Toc27957845)

# 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 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.

# Постановка задачи

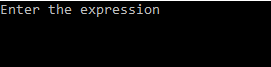
Задача: реализовать программу, обеспечивающую поддержку стеков, производящую обработку арифметических выражений, включая проверку правильности записи выражения, перевод в постфиксную форму и вычисление результата.

Входные данные: арифметическое выражение инфиксной формы записи (без констант).

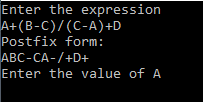
Выходные данные: арифметическое выражение в постфиксной форме записи и значение данного выражения. В случае некорректного ввода – вывод ошибки.

# Руководство пользователя

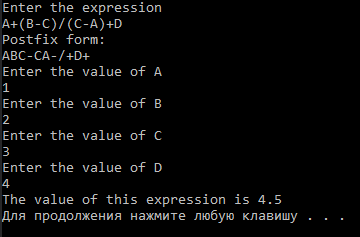
1. Запустить файл **02\_Stack.exe**. Ввести выражение инфиксной формы записи (см. Рис. 1).



1. Запуск программы
2. После вывода постфиксной формы записи вашего выражения ввести значения соответствующих переменных (см. Рис. 2).



1. Ввод значений переменных
2. После вывода результата введенного выражения нажать любую клавишу для прекращения работы программы (см. Рис. 3).

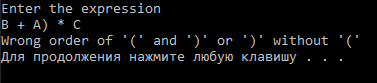


1. Значение выражения

### Действия, приводящие к ошибке

1. Отсутствие операций во введенном выражении.
2. Выражение начинается с операции.
3. Выражение заканчивается на операцию.
4. Незакрытая скобка ‘(’.
5. Открывающаяся скобка ‘(’ в выражении идет позже, чем соответствующая ей закрывающаяся скобка ‘)’.

Пример обработки ошибок (см Рис. 4).



1. Пример ошибки

# Руководство программиста

## Описание структуры программы

* 02\_Stack.h – модуль с классом, содержащим описание и реализацию функций стека;
* PostfixForm.h, PostfixForm.cpp – модуль с классом, реализующим обработку арифметических выражений;
* main.cpp – модуль программы тестирования.

## Описание структур данных

### Класс TStack

template <typename ValType>

class TStack

{

private:

ValType\* elems;

size\_t size;

size\_t top;

public:

TStack(size\_t);

TStack(const TStack&);

~TStack();

void Push(ValType);

void Pop();

ValType Top() { return elems[top - 1]; }

size\_t GetTop() { return top; }

bool IsEmpty() const;

bool IsFull() const;

TStack& operator=(const TStack&);

};

Тип “ValType” описывает тип значения элементов стека.

#### Поля класса:

* **elems** – элементы стека.
* **size** – размер стека.
* **top** – индекс верхнего элемента стека.

#### Описание методов класса:

TStack(size\_t);

**Назначение:** выделение памяти в куче под массив элементов стека.

**Входные параметры:** размер стека.

**Выходные данные:** отсутствуют.

TStack(const TStack&);

**Назначение:** копирование полей элемента класса из элемента того же класса.

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

**Выходные данные:** отсутствуют.

~TStack();

**Назначение:** освобождение выделенной памяти.

**Входные параметры:** отсутствуют.

**Выходные данные:** отсутствуют.

void Push(ValType);

**Назначение:** добавление элемента на вершину стека.

**Входные параметры:** элемент типа ValType.

**Выходные данные:** отсутствуют.

void Pop();

**Назначение:** изъятие верхнего элемента стека.

**Входные параметры:** отсутствуют.

**Выходные данные:** отсутствуют.

ValType Top();

**Назначение:** возвращение значения верхнего элемента стека, не изменяя стек.

**Входные параметры:** отсутствуют.

**Выходные данные:** значение верхнего элемента стека.

size\_t GetTop();

**Назначение:** возвращение индекса верхнего элемента стека.

**Входные параметры:** отсутствуют.

**Выходные данные:** индекс вершины стека.

bool IsEmpty() const;

**Назначение:** проверка стека на пустоту.

**Входные параметры:** отсутствуют.

**Выходные данные:** значение типа boolean.

bool IsFull() const;

**Назначение:** проверка стека на полноту.

**Входные параметры:** отсутствуют.

**Выходные данные:** значение типа boolean.

TStack& operator=(const TStack&);

**Назначение:** присваивание одного стека другому.

**Входные параметры:** константная ссылка на объект класса TStack.

**Выходные данные:** ссылка на объект класса TStack.

### Структура Variable

struct Variable

{

char Name;

double Value;

};

**Поля класса:**

* **Name** – имя переменной из выражения.
* **Value** – значение переменной из выражения.

### Структура Variables

struct Variables

{

Variable\* varmas;

int size;

Variables(int);

};

**Поля класса:**

* **varmas** – массив переменных из выражения.
* **size** – размер массива.

#### Описание методов класса:

Variables(int);

**Назначение:** выделение памяти в куче под массив переменных.

**Входные параметры:** размер массива.

**Выходные данные:** отсутствуют.

### Класс Postfix

class Postfix

{

private:

static int Priority(const char);

static int TypeCheck(const char);

static int OperationsCount(string);

public:

static Variables FillVariables(string);

static string CreatePostfixForm(string);

static double Calculate(string, Variables);

};

#### Описание методов класса:

static int Priority(const char);

**Назначение:** определение приоритета символа.

**Входные параметры:** символ.

**Выходные данные:** приоритет введенного символа(число).

static int TypeCheck(const char);

**Назначение:** определение типа символа(буква, операция).

**Входные параметры:** символ.

**Выходные данные:** тип введенного символа(число).

static int OperationsCount(string);

**Назначение:** подсчет количества операций в строке.

**Входные параметры:** строка, содержащая выражение.

**Выходные данные:** количество операции в строке.

static Variables FillVariables(string);

**Назначение:** заполнение значений переменных из строки.

**Входные параметры:** строка.

**Выходные данные:** заполненный объект типа Variables.

static string CreatePostfixForm(string);

**Назначение:** создание постфиксной формы выражения.

**Входные параметры:** строка, содержащая выражение.

**Выходные данные:** строка, содержащая постфиксную форму выражения.

static double Calculate(string, Variables);

**Назначение:** вычисление значение выражения.

**Входные параметры:** постфиксная форма выражения, массив значений переменных.

**Выходные данные:** значение выражения.

## Описание алгоритмов

### Работа 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.

В качестве *структуры хранения* стека предлагается использовать одномерный  
(одноиндексный) массив, размещаемый в динамической области памяти. Для описания  
структуры хранения следует использовать следующие переменные:

* ***elems*** – указатель на память, выделенную для хранения стека,
* ***size***– размер выделенной памяти,
* ***top*** – индекс элемента массива, в котором хранится последнее добавленное значение стека.

Для работы со стеком предлагается реализовать следующие операции:

* Метод **Push –** добавить элемент;

При добавлении элемента в стек необходимо переместить указатель вершина стека, увеличить на 1, записать элемент в соответствующую позицию динамического массива.

* Метод **Pop** – удаление элемента;

При удалении элемента из стека необходимо переместить указатель вершины стека, уменьшить на 1.

* Метод **Top** – просматривает верхний элемент стека;

Возвращает значение из динамического массива по индексу вершины стека.

* Метод **IsEmpty** – проверить стек на пустоту;

Стек пуст, если в нем нет ни одного элемента, т.е. когда **size** = **top**.

* Метод **IsFull** – проверить стек на полноту;

Стек полон при исчерпании всей отведенной под хранение элементов памяти, т.е.когда значение *size* совпадает со значением *top*.

### Перевод в постфиксную форму

Данный алгоритм основан на использовании стека.

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

Каждой операции и скобками приписывается приоритет.

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

Предполагается, что входная строка содержит правильное выражение.

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 промежуточных постфиксных форм). Операндами будем считать односимвольные переменные.

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

Пример: A+B\*(C-A)/(B+E)+D

1. Операнд кладем его в Стек №2.

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

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

1. Операция, кладем в Стек №1.

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

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

1. Операнд кладем его в Стек №2.

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

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

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

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

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

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

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

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

1. Операнд кладем его в Стек №2.

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

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

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

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

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

1. Операнд кладем его в Стек №2.

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

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

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

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

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

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

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

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

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

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

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

1. Операнд кладем его в Стек №2.

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

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

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

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

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

1. Операнд кладем его в Стек №2.

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

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

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

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

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

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

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

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

1. Операнд, кладем его в Стек №2.

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

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

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

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

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

### Вычисление Арифметического выражения

Выражение просматривается посимвольно слева направо. При обнаружении операнда производится перевод в числовую форму и помещение в стек. При обнаружении операции происходит извлечение из стека двух значений(1 и 2 в порядке извлечения). Выполнение операции происходит в порядке «2 операция 1». Результат операции помещается в стек. В конце на вершине стека будет лежать результат всего выражения.

Пример: ABCA-\*BE+/+D+ (без значений)

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операция, изымаем 2 операнда из стека и кладем в стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операция, изымаем 2 операнда из стека и кладем в стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B\*(C-A)** |  |  |  |  |  |  |  |  |  |  |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B\*(C-A)** | **B** |  |  |  |  |  |  |  |  |  |

1. Операнд, кладем значение в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B\*(C-A)** | **B** | **E** |  |  |  |  |  |  |  |  |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стек | **A** | **B\*(C-A)** | **B+E** |  |  |  |  |  |  |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

1. Операнд кладем в Стек.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A** | **-** | **\*** | **B** | **E** | **+** | **/** | **+** | **D** | **+** |

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

# Заключение

В лабораторной работе была изучена структура данных «Стек» посредством написания алгоритма, переводящего выражение из инфиксной формы записи в постфиксную и подсчитывающего значение этого выражения по его постфиксной форме.

# Литература

1. 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.
2. Рабочие материалы к учебному курсу «Методы программирования» (часть 1). Профессор кафедры Программная инженерия, д.т.н., Гергель В.П. Нижний Новгород, 2015г. – 100с.
3. <https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=vs-2019>

# Приложения

## Приложение 1. Исходный код основной функции(main)

#include "PostfixForm.h"

int main()

{

string line, line1;

double res;

cout << "Enter the expression" << endl;

getline(cin, line);

try

{

line1 = Postfix::CreatePostfixForm(line);

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

cout << line1 << endl;

res = Postfix::Calculate(line1, Postfix::FillVariables(line1));

cout << "The value of this expression is " << res << endl;

}

catch (const char\* k)

{

cout << k << endl;

}

system("pause");

}

## Приложение 2. Класс TStack

#pragma once

#include <iostream>

#include <string>

using namespace std;

template <typename ValType>

class TStack

{

private:

ValType\* elems;

size\_t size;

size\_t top;

public:

TStack(size\_t);

TStack(const TStack&);

~TStack();

void Push(ValType);

void Pop();

ValType Top() { return elems[top - 1]; }

size\_t GetTop() { return top; }

bool IsEmpty() const;

bool IsFull() const;

TStack& operator=(const TStack&);

};

////////////////////////////////////////////////////////////////////////////////////

template<typename ValType>

TStack<ValType>::TStack(size\_t \_size)

{

size = \_size;

top = 0;

elems = new ValType[size];

}

template<typename ValType>

TStack<ValType>::TStack(const TStack& tmp)

{

if (tmp.top >= tmp.size) throw "Stack overflow";

size = tmp.size;

top = tmp.top;

elems = new ValType[size];

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

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

}

template<typename ValType>

TStack<ValType>::~TStack()

{

delete[] elems;

}

template<typename ValType>

void TStack<ValType>::Push(ValType tmp)

{

if (IsFull()) throw "Stack is full";

elems[top++] = tmp;

}

template<typename ValType>

void TStack<ValType>::Pop()

{

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

top--;

}

template<typename ValType>

bool TStack<ValType>::IsEmpty() const

{

return (top == 0);

}

template<typename ValType>

bool TStack<ValType>::IsFull() const

{

return (top == size);

}

template<typename ValType>

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

{

if (size != tmp.size)

{

size = tmp.size;

delete[] elems;

elems = new ValType[size];

}

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

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

return \*this;

}

## Приложение 3. Класс PostfixForm

#### PostfixForm.h

### #pragma once

### #include "02\_Stack.h"

### using namespace std;

### struct Variable

### {

### char Name;

### double Value;

### };

### struct Variables

### {

### Variable\* varmas;

### int size;

### Variables(int);

### };

### class Postfix

### {

### private:

### static int Priority(const char);

### static int TypeCheck(const char);

### static int OperationsCount(string);

### public:

### static Variables FillVariables(string);

### static string CreatePostfixForm(string);

### static double Calculate(string, Variables);

### };

### PostfixForm.cpp

#include "PostfixForm.h"

Variables::Variables(int \_size)

{

size = \_size;

varmas = new Variable[size];

}

int Postfix::Priority(const char s)

{

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

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

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

return 0;

}

int Postfix::TypeCheck(const char s)

{

if ( ( (s >= 'a') && (s <= 'z') ) || ( (s >= 'A') && (s <= 'Z') ) ) return 1;

if (Priority(s) > 0) return 2;

if (s == ')') return 3;

return 0;

}

int Postfix::OperationsCount(string line)

{

int res = 0;

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

if (TypeCheck(line[i]) == 2) res++;

return res;

}

string Postfix::CreatePostfixForm(string line)

{

size\_t CountLeft = 0, CountRight = 0;

if (OperationsCount(line) == 0) throw "There is no operations";

TStack<char> Stack1(OperationsCount(line)), Stack2(line.length());

size\_t i = 0;

size\_t count = 0;

if ((TypeCheck(line[0]) == 2) && (line[0] != '(')) throw "Expression can't start with this symbol";

if (TypeCheck(line[line.length() - 1]) == 2) throw "Expression can't end with this symbol";

while (line[i] != '\0')

{

if (line[i] == ' ')

{

i++;

continue;

}

if (TypeCheck(line[i]) == 1)

{

Stack2.Push(line[i]);

}

else if (TypeCheck(line[i]) == 2)

{

if ((line[i] != '(') && (!Stack1.IsEmpty()))

{

while (Priority(Stack1.Top()) >= Priority(line[i]))

{

Stack2.Push(Stack1.Top());

Stack1.Pop();

}

Stack1.Push(line[i]);

}

else Stack1.Push(line[i]);

if (line[i] == '(') CountLeft++;

}

else if (TypeCheck(line[i]) == 3)

{

CountRight++;

if (CountRight > CountLeft) throw "Wrong order of '(' and ')' or ')' without '(' ";

while (Stack1.Top() != '(')

{

Stack2.Push(Stack1.Top());

Stack1.Pop();

}

Stack1.Pop();

}

else throw "Unacceptable symbol";

i++;

}

if (CountRight < CountLeft) throw "There is a non-closed '(' ";

while (!(Stack1.IsEmpty()))

{

Stack2.Push(Stack1.Top());

Stack1.Pop();

}

string res;

res.resize(Stack2.GetTop(), 0);

for (string::reverse\_iterator j = res.rbegin(); !Stack2.IsEmpty(); j++)

{

\*j = Stack2.Top();

Stack2.Pop();

}

return res;

}

Variables Postfix::FillVariables(string line)

{

Variables mas(line.length() - OperationsCount(line));

size\_t i = 0, j = 0;

while (line[i] != '\0')

{

if (TypeCheck(line[i]) == 1)

{

if (line.find\_first\_of(line[i]) == i)

mas.varmas[j++].Name = line[i];

}

i++;

}

for (size\_t l = 0; l < j; l++)

{

cout << "Enter the value of " << mas.varmas[l].Name << endl;

cin >> mas.varmas[l].Value;

}

return mas;

}

double Postfix::Calculate(string line, Variables mas)

{

size\_t Variables = line.length() - OperationsCount(line);

TStack<double> Stack(Variables);

size\_t i = 0, j = 0;

double a = 0, b = 0;

while (line[i] != '\0')

{

if (TypeCheck(line[i]) == 1)

{

for (size\_t m = 0; m < mas.size; m++)

if (mas.varmas[m].Name == line[i]) Stack.Push(mas.varmas[m].Value);

}

else if (TypeCheck(line[i]) == 2)

{

b = Stack.Top();

Stack.Pop();

a = Stack.Top();

Stack.Pop();

if (line[i] == '+') Stack.Push(a + b);

else if (line[i] == '-') Stack.Push(a - b);

else if (line[i] == '\*') Stack.Push(a \* b);

else if (line[i] == '/') Stack.Push(a / b);

}

i++;

}

return Stack.Top();

}