МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**ЛАБОРАТОРНАЯ РАБОТА**

на тему:

**«Стек. Постфиксная форма.»**

**Выполнил(а):** студент группы 3822Б1ФИ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Савченко М.П./

Подпись

**Проверил:** к.т.н, доцент каф. ВВиСП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кустикова В.Д./

Подпись

Нижний Новгород  
2023

**Содержание**

[Введение 3](#_Toc147915966)

[1 Постановка задачи 4](#_Toc147915967)

[2 Руководство пользователя 5](#_Toc147915968)

[2.1 Приложение для демонстрации работы битовых полей 5](#_Toc147915969)

[2.2 Приложение для демонстрации работы множеств 5](#_Toc147915970)

[2.3 «Решето Эратосфено» 5](#_Toc147915971)

[3 Руководство программиста 6](#_Toc147915972)

[3.1 Описание алгоритмов 6](#_Toc147915973)

[3.1.1 Битовые поля 6](#_Toc147915974)

[3.1.2 Множества 6](#_Toc147915975)

[3.1.3 «Решето Эратосфена» 6](#_Toc147915976)

[3.2 Описание программной реализации 6](#_Toc147915977)

[3.2.1 Описание класса TBitField 6](#_Toc147915978)

[3.2.2 Описание класса TSet 7](#_Toc147915979)

[Заключение 8](#_Toc147915980)

[Литература 9](#_Toc147915981)

[Приложения 10](#_Toc147915982)

[Приложение А. Реализация класса TBitField 10](#_Toc147915983)

[Приложение Б. Реализация класса TSet 10](#_Toc147915984)

# Введение

Стек и постфиксная (или обратная польская) форма арифметического выражения - это концепции, которые могут быть актуальными и полезными в различных областях программирования и вычислительной математики.

1. **Стек (Stack):**

Стек - это структура данных, работающая по принципу "последний вошел, первый вышел" (Last In, First Out - LIFO). Это означает, что элементы добавляются и удаляются только с одного конца стека (вершины). Стек может быть реализован как массив или связанный список.

Применение стека:

1. **Управление вызовами функций:** Стек используется для хранения информации о вызовах функций, чтобы знать, куда возвращаться после завершения каждой функции.
2. **Обратная трассировка (debugging):** Стек помогает отслеживать порядок вызовов функций и точки, в которых произошла ошибка.
3. **Вычисления с использованием рекурсии:** Рекурсивные алгоритмы часто используют стек для хранения промежуточных результатов.
4. **Постфиксная форма:**

Постфиксная форма (или обратная польская запись) - это способ записи арифметических выражений, при котором операторы расположены после своих операндов. Это исключает необходимость в скобках и упрощает вычисление выражений.

Применение постфиксной формы:

1. **Калькуляторы:** Некоторые карманные калькуляторы используют постфиксную форму для упрощения вычислений.
2. **Оптимизация вычислений:** Постфиксная форма позволяет избежать проблем с приоритетом операторов и порядком операций, делая выражения более однозначными.
3. **Автоматическая генерация кода:** Некоторые компиляторы используют постфиксную форму внутри своих промежуточных представлений.

Вместе стек и постфиксная форма могут использоваться, например, для вычисления постфиксных выражений без использования рекурсии. Выражение обрабатывается слева направо, операнды помещаются в стек, и когда встречается оператор, извлекаются нужное количество операндов из стека, выполняется операция, и результат помещается обратно в стек.

Хотя стек и постфиксная форма могут казаться несколько абстрактными, они оказываются полезными инструментами в различных областях программирования и алгоритмов, особенно в тех случаях, когда нужно эффективно управлять порядком операций и сохранять контекст выполнения.

# Постановка задачи

Цель – Реализовать шаблонный класс для представления стека TStack, и на его основе реализовать класс для представления арифметического выражения с его постфиксной формой TArithmeticExpression.

Задачи при реализации класса TStack:

1. Описать и реализовать конструктор, конструктор копирования, деструктор.
2. Описать и реализовать методы проверки заполнен ли стек и пуст ли стек.
3. Описать и реализовать методы добавления элемента на верхушку стека, удаление элемента с верхушки стека, получить элемент с верхушки стека.

Задачи при реализации класса TArithmeticExpression:

1. Описать и реализовать конструктор.
2. Описать и реализовать метод парсинга арифметического выражения для поиска операторов, операндов и констант.
3. Реализовать алгоритм перевода арифметического выражения в его постфиксную форму.
4. Реализовать алгоритм вычисления значения арифметического выражения по его постфиксной форме.
5. Описать и реализовать разные способы ввода значений арифметического выражения.
6. Описать и реализовать вспомогательные методы: проверка на оператор, проверка на константу, поиск первого встречного оператора от какой-нибудь позиции.
7. Описать и реализовать методы получения инфиксной и постфиксной форм арифметического выражения.
8. Описать и реализовать метод конвертации строки арифметического выражения в форму без сокращений (унарный минус, умножение скобок без символа «\*» и т.д.).
9. Описать и реализовать метод на проверку корректности арифметического выражения.

# Руководство пользователя

## Приложение для демонстрации работы стека

1. Запустите приложение с названием \*.exe. В результате появится окно, показанное ниже (рис. 1).



1. Основное окно программы

## Приложение для демонстрации работы постфиксной формы арифметического выражения

# Руководство программиста

## Описание алгоритмов

### Стек

Стек (или стопка) представляет собой абстрактную структуру данных, организованную по принципу "последний вошел - первый вышел" (Last In, First Out, LIFO). Это означает, что элементы добавляются и удаляются только с одного конца стека, который называется вершиной.

Операции, которые можно выполнять со стеком, включают добавление элемента (push) на вершину стека и удаление элемента (pop) с вершины стека. При этом доступ к остальным элементам стека осуществлять нельзя, кроме верхнего элемента.

### Постфиксная форма

Постфиксная форма арифметического выражения, также известная как обратная польская запись (ОПЗ) или постфиксная нотация, представляет собой способ записи математических выражений, при котором операторы расположены после своих операндов. В отличие от традиционной инфиксной формы, где операторы находятся между операндами, в постфиксной форме порядок операндов и операторов определяется последовательностью их появления.

Например, в инфиксной форме выражение записывается с оператором между операндами: .

В постфиксной форме это же выражение будет выглядеть следующим образом: .

**Алгоритм преобразования инфиксной формы арифметического выражения в постфиксную с использованием стека:**

1. Создайте пустой стек для операторов и операндов.
2. Инициализируйте пустой список для хранения выходного постфиксного выражения.
3. Пройдите по каждому символу в инфиксной форме слева направо.
4. Если символ - операнд (число), добавьте его в выходной список.
5. Если символ - открывающая скобка, поместите ее в стек.
6. Если символ - оператор, то:

* Пока стек не пуст и верхний элемент стека не является открывающей скобкой, и приоритет оператора на вершине стека больше или равен приоритету текущего оператора, извлеките оператор из стека и добавьте его в выходной список.
* Поместите текущий оператор в стек.

1. Если символ - закрывающая скобка, извлекайте операторы из стека и добавляйте их в выходной список до тех пор, пока не встретите открывающую скобку. Извлекните открывающую скобку, но не добавляйте ее в выходной список.
2. После обхода всего выражения извлеките оставшиеся операторы из стека и добавьте их в выходной список.

**Алгоритм вычисления значения арифметического выражения в постфиксной форме с использованием стека:**

1. Создайте пустой стек.
2. Пройдите по каждому символу в постфиксной форме слева направо.
3. Если символ - операнд (число), поместите его в стек.
4. Если символ - оператор, извлеките из стека необходимое количество операндов (в соответствии с арностью оператора), выполните операцию, и поместите результат обратно в стек.
5. Продолжайте этот процесс до тех пор, пока не обработаете все символы в постфиксной форме.
6. После обработки всех символов в постфиксной форме, результат вычисления будет находиться на вершине стека.

Пример преобразования инфиксной формы в постфиксную:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | / |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + | + |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | H | H | H |
|  |  |  |  |  |  |  |  |  |  |  |  | G | G | G | G | G |
|  |  |  |  |  |  |  |  |  |  | F | F | F | F | F | F | F |
|  |  |  |  |  |  |  |  |  | + | + | + | + | + | + | + | + |
|  |  |  |  |  |  |  |  |  | \* | \* | \* | \* | \* | \* | \* | \* |
|  |  |  |  |  |  |  |  | D | D | D | D | D | D | D | D | D |
|  |  |  |  |  |  | - | - | - | - | - | - | - | - | - | - | - |
|  |  |  |  |  | C | C | C | C | C | C | C | C | C | C | C | C |
|  |  |  |  | B | B | B | B | B | B | B | B | B | B | B | B | B |
| ст1 | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
|  |  |  |  |  |  |  |  |  |  |  |  |  | + | + |  |  |
|  |  |  |  | - | - |  |  |  |  |  | ( | ( | ( | ( |  |  |
|  |  |  | ( | ( | ( |  | \* | \* |  | / | / | / | / | / | / |  |
| ст2 |  | + | + | + | + | + | + | + | - | - | - | - | - | - | - |  |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

Полученная постфиксная форма:

Пример вычисления значения по префиксной форме:

A=0, B=1, C=2, D=-1, F=2, G=0.5, H=0.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  | 0.5 |  |  |  |
|  |  |  | 2 |  | -1 |  |  |  | 0.5 | 0.5 | 1 |  |  |
|  |  | 1 | 1 | -1 | -1 | 1 |  | 2 | 2 | 2 | 2 | 2 |  |
| Ст. | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| Преф. | A | B | C | - | D | \* | + | F | G | H | + | / | - |

Полученное значение: -1

## Описание программной реализации

### Описание класса TStack

template <typename T>

class TStack {

private:

int maxSize;

int top;

T\* elems;

void Realloc(int extraSize = 10);

public:

TStack(int \_maxSize = 50);

TStack(const TStack<T>& s, int extraSize = 0);

virtual ~TStack();

bool IsEmpty(void) const noexcept;

bool IsFull(void) const noexcept;

T Top();

void Push(const T& e);

void Pop();

};

Назначение: представление стека.

Поля:

BitLen – длина битового поля – максимальное количество битов.

pMem – память для представления битового поля.

MemLen – количество элементов для представления битового поля.

Методы:

int GetMemIndex(const int n) const;

Назначение: получение индекса элемента в памяти…

Входные параметры:

n – номер бита.

Выходные параметры:

Номер элемента в памяти.

### Описание класса TArithmeticExpressio

class TArithmeticExpression {

private:

string infix;

vector<string> postfix;

vector<string> lexems;

static map<string, int> priority;

map<string, double> operands;

void Parse();

void ToPostfix();

public:

TArithmeticExpression(const string& \_infix);

string GetInfix() const { return infix; }

vector<string> GetPostfix() const { return postfix; }

string GetStringPostfix() const;

vector<string> GetOperands() const;

void SetValues();

void SetValues(map<string, double>& values);

void ClearValues();

double Calculate();

double Calculate(const map<string, double>& values);

private:

bool IsOperator(const string& isopr) const;

bool IsConst(const string& isopd) const;

int FindOperator(int pos = 0) const;

void ConvertInfix();

void CorrectnessCheck();

};

Назначение: представление арифметического выражения.

# Заключение

Что сделано?

# Литература

1. Лекция «Динамическая структура данных Стек» Сысоева А.В. <https://cloud.unn.ru/s/jXmxFzAQoTDGfNe>
2. Лекция «Разбор и вычисление арифметических выражений с помощью постфиксной формы» Сысоева А.В. <https://cloud.unn.ru/s/4Pyf24EBmowGsQ2>

# Приложения

## Приложение А. Реализация класса TBitField

## Приложение Б. Реализация класса TSet