**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

Федеральное государственное автономное образовательное учреждение высшего образования

“Нижегородский государственный университет им. Н.И. Лобачевского (ННГУ)”

Кафедра Институт информационных технологий, математики и механики

**ОТЧЁТ**

О прохождении учебной практики

**Тема**:” Вычислениеарифметических выражений (стеки)”

Выполнил студент

2 курса, группы 381906-3:

Баринов Сергей Владимирович

Руководитель практики:

Гетманская Александра Александровна

**г. Нижний Новгород, 2021 год**

Содержание:

[История возникновения 3](#_Toc26889234)

[Описание задач 3](#_Toc26889235)

[Математическая постановка задачи 3](#_Toc26889236)

[Описание методов 4](#_Toc26889237)

[Программная реализация 5](#_Toc26889238)

[Выбор инструментов 5](#_Toc26889239)

[Руководство программиста 5](#_Toc26889240)

[Заключение 6](#_Toc26889243)

**История возникновения**

Перед программистом нередко возникает задача вычисления арифметических или иных выражений, не известных на этапе компиляции программы. В интернете можно найти компоненты и законченные примеры вычисления выражений, но нередко требуется создать что-то свое. Изучив основы теории синтаксического анализа, можно понять, что можно легко написать код для синтаксического разбора любых выражений, которые подчиняются некоторому формальному описанию синтаксиса.

Итак, **арифметическое выражение** - выражение, в котором операндами являются объекты, над которыми выполняются арифметические операции.

Структура данных в программировании — это механизм организации данных для их эффективного использования. Cтек – хороший тому пример.

**Стек** (англ. **stack** — стопка) — абстрактный тип данных (структура данных), представляющий собой список элементов, организованных по принципу LIFO (англ. Last In — First Out, «Последним прибыл – первым обслужен»).

**Стек,** а точнее шаблон стека,содержится в стандартной библиотеке stack (подключить: #include <stack>). Он содержит **методы**, то есть, функции, используемые обычно для контейнеров типа очереди и стека.

В ходе работы был написан стек (на массиве и на односвязном списке) и односвязный список, содержащие необходимый набор методов и объектов.

**Описание задач**

**Математическая постановка задачи.**

Необходимо реализовать программу, которая обеспечит поддержку стека, и разработку программных средств, производящих обработку арифметических выражений, перевод в постфиксную форму и вычисление результата.

**Описание методов.**

Структура данных stack удовлетворяет следующим требованиям:

* Структура данных **stack** и все ее методы реализованы в виде шаблонов.
* В качестве структуры хранения стека используется одномерный массив, размещаемый в динамической области памяти.
* Для работы со **стеком** реализовываются следующие операции:
* Метод Push - добавить элемент;
* Метод Pop – удалить верхний элемент;
* Метод Top – получить значение верхнего элемента;
* Метод isEmpty - проверить стек на пустоту;

**Программная реализация.**

Примечание:

Source.cpp– исполнительный файл, содержит функцию main;

Steck.h– стек на основе массива;

Calc.h – операции по переведению в постфиксную форму и вычисление

Calculator.cpp – определения функций из Calc.h

**Выбор инструментов**

Для реализации указанных выше методов была использована интегрированная среда разработки (IDE) от Microsoft – **Visual** **Studio 2019.** Программы написаны на языке программирования **C++**.

**Руководство программиста**

1. #include <iostream> - заголовочный файл с классами, функциями и переменными для организации ввода вывода.
2. #include <string> - стандартная библиотека из STL, содержащая методы со строками и переменные - строки, использованные при выполнении лабораторной работы.
3. int main() - глобальная функция, отвечающая за ввод уравнений и вызов функции вычеслений

#include "Stack.h" – подключение заголовочного файла, содержащего стек

#include "Calc.h" – подключение заголовочного файла, содержащего функции вычислений

Методы, реализованные в Stack.h:

void Push(T);//добавить элемент

T Pop();// удаляет последний элемент

T Top();// выводит последний элемент

bool isEmpty();//проверяет стек на пустоту

bool IsFull();// проверяет стек на заполненость

void clear();// полностью отчищает стек

Методы, реализованные в Calc,h:

void push\_back(T Data); //добавить элемент

int GetSize() { return Size; };

T pop();//считать и удалить верхний элемент

T top();//вернуть верхний элемент

bool isEmpty();//проверка на пустоту

void print();//вывести список на экран

Методы, реализованные в Calculator.cpp:

int GetPriority(char m) //Определяет приоритет операций

void ToPostFix();// переводит выражение в постфиксную форму

void SetInfix(string inf) //получает инфиксную форму

void GetPostfix() //возвращает постфиксную форму

double CalcS() // вычисляет выражение

**Заключение**

В результате выполнения лабораторной работы я успешно выполнил отмеченные мной задачи, в том числе провел тест программы по следующем критериям: Выполняется перевод записи выражения в постфиксную форму, правильно вычисляются корректно записанные арифметические выражения.

Также были проведены следующие тесты:

1) 1+2

Постфиксная форма: 1 2 +

Результат: 3

2) 1+2\*(3-2)-4

Постфиксная форма: 1 2 3 2 -\*+4 –

Результат: -1

3) 1+2/(3-4)

Постфиксная форма: 1 2 3 4-/+

Результат: -1