

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Стек

Студент гр. 7383

Лосев М. Л.

Преподаватель

Размочаева Н. В.

Санкт-Петербург

2018

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ.....	3
2. РЕАЛИЗАЦИЯ ЗАДАЧИ.....	4
3. ТЕСТИРОВАНИЕ.....	5
4 . ВЫВОД.....	5
ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ Код программы.....	6
ПРИЛОЖЕНИЕ Б. Тестовые случаи.....	10

1. ЦЕЛЬ РАБОТЫ

Цель работы: Познакомиться с часто используемыми на практике линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур, освоить на практике использование стека, очереди и дека для решения задач.

Формулировка задачи: вычислить как целое число значение выражения (без переменных), записанного в префиксной форме.

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

2.1 ИСПОЛЬЗУЕМЫЕ СТРУКТУРЫ ДАННЫХ

Stack – шаблонный класс с шаблонным параметром *base* (тип хранимых элементов). Представляет из себя стек в реализации на базе массива. Поля класса *Stack*:

- *arr* – массив элементов типа *base*, в котором хранятся все элементы стека.
- *size* – длина массива *arr*.
- *len* – количество элементов в стеке.

Методы класса *Stack*:

- Конструктор *Stack* – выделяет место под пять элементов.
- *top* – если стек не пуст, возвращает элемент, который хранит вершина стека.
- *pop* – если стек не пуст, удаляет его вершину, возвращает значение элемента, который хранила удаляемая вершина.
- *push* – добавляет в стек элемент.
- *isEmpty* – возвращает *true*, если стек пуст, или *false* во всех остальных случаях.
- *destroy* – освобождает место, занятое стеком.

2.2 ИСПОЛЬЗУЕМЫЕ ФУНКЦИИ

IsDigit – принимает символ, возвращает *true*, если этот символ является цифрой, или *false*, если не является.

IsOperator – принимает символ, возвращает *true*, если этот символ является оператором, или *false*, если не является.

CalcPrefixExp – принимает строку (выражение в префиксной форме), вычисляет значение выражения как целое число и возвращает его.

3. ТЕСТИРОВАНИЕ

3.1 ПРОЦЕСС ТЕСТИРОВАНИЯ

Программа собрана в операционной системе Ubuntu 18.04 компилятором gcc. В других ОС тестирование не проводилось.

3.1 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестовые случаи представлены в приложении Б. Ошибки выявить не удалось. По результатам тестирования было показано, что задача выполнена.

4 . ВЫВОД

В ходе работы была написана программа на языке C++, вычисляющая значение выражения, записанного в префиксной форме. Был получен опыт в использовании стека в ссылочной реализации.

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл MakeFile:

```
all: main2.o
    g++ main2.o -o main2
main2.o: main2.cpp st_interf2.h
    g++ -c main2.cpp
clean:
    rm *.o main2
```

Файл stack_vec.h:

```
#ifndef STACK_VEC_H
#define STACK_VEC_H

template <class base>
class Stack
{
    private:
        base *arr;
        int size;
        int len;
    public:
        Stack()
        {
            len = 0;
            size = 5;
            arr = new base[size];
        }

        ~Stack()
        {
            delete [] arr;
        }

        int lenth()
        {
            return len;
        }

        void push(base x)
        {
            if (len == size)
                new_size(size<<1);
            arr[len++] = x;
            //cout << "Push -> " << x << endl;
        }

        base pop()
        {
            if (len == 0)
```

```

        return arr[len];
        return arr[--len];
    }

    base top()
    {
        return arr[len];
    }

    void destroy()
    {
        delete [] arr;
    }

    bool isEmpty()
    {
        return len == 0;
    }

    void new_size(int s)
    {
        if (s <= 0)
            return;
        base *new_arr = new base[s];
        for (int i = 0; i < s; i++)
            new_arr[i] = arr[i];
        delete[] arr;
        size = s;
        arr = new_arr;
        //cout<<"NEW SIZE "<<size<<endl;
    }
};

#endif

```

Файл main2.cpp:

```

// Программа клиент вычисляет арифметическое выражение, заданное в постфиксной
// форме
// Ссылочная реализация в динамической (связанной) памяти
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <cstring>

#include "stack_vec.h"

using namespace std;

bool IsDigit( char c )
{
    return ( (c <= '9') && (c >= '0')));
}

```

```

bool IsOperator( char c )
{
    return (c == '+' || c == '-' || c == '*' || c == '/' || c == '^');
}

int CalcPrefixExp(string expr)
{
    Stack<int> numbers;
    int n = expr.length();

    for (int i = n - 1; i >= 0; i--)
    {
        // бежим от конца выражения к началу
        if (IsOperator(expr[i]))
        {
            // если попался оператор
            int x = numbers.pop();
            int y = numbers.pop();
            if (expr[i] == '+')
                numbers.push(x + y);
            if (expr[i] == '*')
                numbers.push(x * y);
            if (expr[i] == '-')
                numbers.push(x - y);
            if (expr[i] == '/')
                numbers.push(x / y);
            if (expr[i] == '^')
                numbers.push(pow(x, y));
        }
        if (IsDigit(expr[i])) //
        {
            // если попало число
            int dec_exp = 10;
            numbers.push(expr[i--] - '0'); // пушим младшую цифру
            while (IsDigit(expr[i]))
            {
                // пока идут цифры числа, продолжаем его ввод
                numbers.push(numbers.pop() + dec_exp * (expr[i--] - '0'));
                dec_exp *= 10;
            }
            if (expr[i] == '-') // если число отрицательное
                numbers.push(-1 * numbers.pop());
        }
    }

    int answ = numbers.pop(); // сохранение ответа

    numbers.destroy();

    return answ;
}

void UserInterface ()
{
    int key;
    string expr;

    while (true) {
        cout << "Выбери действие:" << endl;
        cout << "0. Завершить выполнение;" << endl;
        cout << "1. Загрузить префиксное выражение из пользовательского файла;" <<
endl;
    }
}

```



```

        cout << "2. Загрузить префиксное выражение из файла по умолчанию
(prefix.txt);" << endl;
        cout << "3. Ввести префиксное выражение с клавиатуры;" << endl;
        cin >> key;
        switch (key) {
            case 0:
                return;
                break;
            case 1:
                {
                    string filename;
                    cout << "Введите имя файла: ";
                    cin.ignore(256, '\n');// игнорируем оставшиеся в cin после
ввода '1' символы
                    getline(cin, filename);
                    ifstream fin(filename);
                    fin >> noskipws; // не пропускать пробелы
                    if (fin){
                        getline(fin, expr);
                        cout << expr << " = " << CalcPrefixExp(expr) << endl;
                    }
                    else cout << "Файл не открыт\n";
                }
                break;
            case 2:
                {
                    ifstream fin("prefix.txt");
                    fin >> noskipws; // не пропускать пробелы
                    if (fin){
                        getline(fin, expr);
                        cout << expr << " = " << CalcPrefixExp(expr) << endl;
                    }
                    else cout << "Файл не открыт\n";
                }
                break;

            case 3:
                {
                    cout << "Введите выражение: ";
                    cin.ignore(256, '\n');// игнорируем оставшиеся в cin после ввода '3'
символы
                    getline(cin, expr);
                    cout << expr << " = " << CalcPrefixExp(expr) << endl;
                }
                break;

            default : cout << "! - ...";
                    break;
        };
    }
}

int main ()
{
    UserInterface();
    return (0);
}

```

ПРИЛОЖЕНИЕ Б ТЕСТОВЫЕ СЛУЧАИ

Входное выражение	Вывод программы	Корректность выполнения
$+ 100\ 500$	$+ 100\ 500 = 600$	да
$+ 100 / 100\ 4$	$+ 100 / 100\ 4 = 125$	да
$- 169 \wedge 13\ 2 = 0$	$- 169 \wedge 13\ 2 = 0$	да
$- 100\ 500$	$- 100\ 500 = -400$	да
$* -3\ 14$	$* -3\ 14 = -42$	да
$+ + \wedge 10\ 2\ 10 * 2 * 3\ 10$	$+ + \wedge 10\ 2\ 10 * 2 * 3\ 10 = 170$	да