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

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 *Elem* (тип хранимых элементов). Представляет из себя стек в ссылочной реализации.

Поля класса *Stack:*

- *node* – структура, содержащая два поля: указатель *hd* на элемент типа *Elem,* указатель *tl* на следующий элемент типа *node* – «хвост»; и конструктор *node*, обнуляющий оба поля.
- *topOfStack* – указатель на вершину стека.

Методы класса *Stack:*

- Конструктор *Stack,* обнуляющий указатель *topOfStack* на вершину стека.
- *top* – если стек не пуст, возвращает элемент, который хранит вершина сткеа.
- *pop* – если стек не пуст, удаляет его вершину (при этом вершиной становится следующий за ней элемент).
- *pop2* – если стек не пуст, удаляет его вершину (при этом вершиной становится следующий за ней элемент), возвращает значения элемента, который хранила удаляемая вершина.
- *push* – добавляет в стек элемент. Бывшая вершина после добавления нового элемента идет за ним, а сам новый элемент становится вершиной.
- *two_numbers* – возвращает *true*, если верхние два элемента в стеке отличны от нуля, или *false* во всех остальных случаях.
- *isNull* – возвращает *true*, если стек пуст, или *false* во всех остальных случаях.
- d*estroy* – освобождает место, занятое стеком.

2.2 ИСПОЛЬЗУЕМЫЕ ФУНКЦИИ

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

CalcPrefixExp – приимает указатель на начало строки (выражения в префиксной форме), вычисляет значение выражения как целое число и возвращает его.

Read_str – принимает указатель на поток, считывает из этого потока строку, возвращает указатель на ее начало.

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 st_interf2.cpp:

```
#ifndef ST_INTERF2_H_INCLUDED
#define ST_INTERF2_H_INCLUDED

// Интерфейс АТД "Стек" (ссылочная реализация в динамической памяти)с шаблоном класса
// И интерфейс, и реализация в одном заголовочном файле

#include <iostream>
#include <cstdlib>

using namespace std;

namespace st_modul2
{

//-------------------------------------
	template <class Elem>
	class Stack
	{
		private:
		struct node
		{ //
			Elem *hd;
			node *tl;
			// constructor
			node ()
				{
					hd = NULL;
```

```
                        tl = NULL;
                    };
        };// end node

        node *topOfStack;

        public:
        Stack ()
              { topOfStack = NULL;
              }//;

//-------------------------------------
      Elem top (void) //
      {// PreCondition: not null
            if (topOfStack == NULL) { cerr << "Error: top(null) \n"; return
NULL;  }
            else return *topOfStack->hd;
      }
//-------------------------------------
      void pop (void)//
      {// PreCondition: not null
            if (topOfStack == NULL) { cerr << "Error: pop(null) \n"; return; }
            else
            {     node *oldTop = topOfStack;
                  topOfStack = topOfStack->tl;
                  delete oldTop->hd;
                  delete oldTop;
            }
      }
//-------------------------------------
      Elem pop2(void)//
      {// PreCondition: not null
            if (topOfStack == NULL) { cerr << "Error: pop(null) \n"; return NULL; }
            else
            {     node *oldTop = topOfStack;
                  Elem r = *topOfStack->hd;
                  topOfStack = topOfStack->tl;
                  delete oldTop->hd;
                  delete oldTop;
                  return r;
            }
```

```
	}
//-------------------------------------
	void push (const Elem &x)//
	{	node *p;
		p = topOfStack;
		topOfStack = new node;
		if ( topOfStack != NULL) {
			topOfStack->hd = new Elem;
			*topOfStack->hd = x;

			topOfStack->tl = p;
		}
		else {cerr << "Memory not enough\n"; return; }
	}

	bool  two_numbers( void )
	{ // если последние два тега соответствуют числам, то возвращается true
		if ((*topOfStack).tl == NULL)
			return 0;
		return (*(*topOfStack).hd && *(*topOfStack).tl->hd);

	}
//-------------------------------------
	bool isNull(void)//
	{	return (topOfStack == NULL) ;
	}
//-------------------------------------
	void destroy (void)//
	{	while ( topOfStack != NULL) {
			pop();
		}
	}
	};
}

#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 "st_interf2.h"

using namespace std;
using namespace st_modul2;

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

int CalcPrefixExp(char *expr)
{
	Stack <int> numbers;
	Stack <char> operators;
	Stack <bool> tags; // 0 - operator; 1 - number
	int n = strlen(expr);

	for (int i = 0; i < n; i++)
	{
		if (expr[i] == '+' || expr[i] == '*' || expr[i] == '/' || expr[i] == '^') {
			operators.push(expr[i]);
			tags.push(false);
		}

		if (expr[i] == '-')
			if (IsDigit(expr[i + 1])){
				i += 1;
				numbers.push(0);
				while ((expr[i] >= '0') && (expr[i] <= '9'))
				{
```

```
                        numbers.push(10*numbers.pop2() + (expr[i++]-'0'));
                }
                numbers.push(numbers.pop2() * -1);
                tags.push(true);
        }
        else{
                operators.push(expr[i]);
                tags.push(false);
        }

    if (IsDigit(expr[i]))
    {
        numbers.push(0);
        while ((expr[i] >= '0') && (expr[i] <= '9'))
        {
                numbers.push(10*numbers.pop2() + (expr[i++]-'0'));
        }
        tags.push(true);

        while (tags.two_numbers()){
                char op = operators.pop2();
                tags.pop();
                int x = numbers.pop2();
                int y = numbers.pop2();

                if (op == '+')
                        numbers.push(y + x);
                if (op == '*')
                        numbers.push(y * x);
                if (op == '-')
                        numbers.push(y - x);
                if (op == '/')
                        numbers.push(y / x);
                if (op == '^')
                        numbers.push(pow(y, x));
                tags.pop();
                tags.pop();
                tags.push(true);
        }
    }
}
```

```
      int answ = numbers.pop2(); // сохраение ответа

      numbers.destroy();
      operators.destroy();
      tags.destroy();

      return answ;
}

template <typename t>
char *Read_str (t *stream)
{

    char *str, c;
    int size = 20; // размер стоки str
    str = new char[size];
    for (int i = 0;  (*stream).get(c); i++){
            if (c == '\n') // конец ввода строки
            break;
      if (i == size){ // когда i-ый символ уже не влезет в str
            char *new_str = new char[2 * size];
                  for (size_t l = 0; l < size; ++l)
                  new_str[l] = str[l];
            size *= 2;
            delete [] str;
            str = new_str;
      }
            str[i] = c;
      }
      return str;
}

void UserInterface ()
{
    int key;
      char *a;

    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;" << endl;
        cout << "3. Ввести префиксное выражение с клавиатуры;" << endl;
        cin >> key;
        switch (key) {
            case 0:
                return;
                break;
            case 1:
                  {
                        char *filename;

                        cout << "Введите имя файла: ";
                        cin.ignore(256, '\n');// игнорируем оставшиеся в cin после
ввода '1' символы
                        filename = Read_str(&cin);
                        ifstream fin(filename);
                        fin >> noskipws;   // не пропускать пробелы
                        if (fin){
                              a = Read_str(&fin);
                              cout << a << " = " << CalcPrefixExp(a) << endl;
                        }
                        else cout << "Файл не открыт\n";
                  }
            break;
            case 2:
            {
                        ifstream fin("prefix.txt");
                        fin >> noskipws;   // не пропускать пробелы
                        if (fin){
                              a = Read_str(&fin);
                              cout << a << " = " << CalcPrefixExp(a) << 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
                        a = Read_str(&cin);
                        cout << a << " = " << CalcPrefixExp(a) << 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 ^ 13 2 = 0	- 169 ^ 13 2 = 0	да
- 100 500	- 100 500 = -400	да
* -3 14	* -3 14 = -42	да
+ + ^ 10 2 10 * 2 * 3 10	+ + ^ 10 2 10 * 2 * 3 10 = 170	да