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

# ОТЧЕТ

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: Динамические структуры данных.**

Студент гр. 1382 _____ Коренев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

**Цель работы.**

Освоить основы языка программирования C++, научится моделировать работы стека на базе списка.

**Задание (Вариант 4).**

Требуется написать программу, моделирующую работу стека на базе списка. Для этого необходимо:

1) Реализовать класс CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных int.

Структура класса узла списка:

```
struct ListNode {
    ListNode* mNext;
    int mData;
};
```

Объявление класса стека:

```
class CustomStack {
public:
// методы push, pop, size, empty, top + конструкторы, деструктор
private:
// поля класса, к которым не должно быть доступа извне
protected:  // в этом блоке должен быть указатель на голову
    ListNode* mHead;
};
```

Перечень методов класса стека, которые должны быть реализованы:

- void push(int val) - добавляет новый элемент в стек
- void pop() - удаляет из стека последний элемент
- int top() - возвращает верхний элемент
- size_t size() - возвращает количество элементов в стеке
- bool empty() - проверяет отсутствие элементов в стеке

2) Обеспечить в программе считывание из потока stdin последовательности команд (каждая команда с

новой строки), в зависимости от которых программа выполняет ту или иную операцию и выводит результат ее выполнения с новой строки.

Перечень команд, которые подаются на вход программе в stdin:

- cmd_push n - добавляет целое число n в стек. Программа должна вывести "ok"
- cmd_pop - удаляет из стека последний элемент и выводит его значение на экран
- cmd_top - программа должна вывести верхний элемент стека на экран не удаляя его из стека
- cmd_size - программа должна вывести количество элементов в стеке
- cmd_exit - программа должна вывести "bye" и завершить работу

Если в процессе вычисления возникает ошибка (например вызов метода pop или top при пустом стеке), программа должна вывести "error" и завершиться.

**Выполнение работы.**

В функции main создается объект stack класса CustomStack и переменная input типа string. В цикле while каждую итерацию считывается значение в переменную input. С помощью функции find() определяется что было введено пользователем.

- cmd_push n, где n – число, вызовется метод push у объекта stack, в консоль выводится “ok
- cmd_top, в консоль выведется значение, которое вернет метод top()
- cmd_pop, вызовутся методы top() и pop(), pop() удаляет последний элемент в стеке
- cmd_size, в консоль выведется значение возвращаемое методом size() – количество элементов в стеке

- cmd_empty, выведет в консоль true если стек пустой и false в противном случае
- cmd_exit, программа завершит свою работу и выведет в консоль “bye”.

Класс CustomStack имеет конструктор, который присваивает nullptr в переменную mHead объявленную в модификаторе доступа protected, и деконструктор, который удаляет память выделенную для стека. Также он имеет методы:

- push() выделяет память для нового элемента стека, помещает в него значение поступившее в качестве аргумента, присваивает переменной mHead значение cur – указатель на текущий элемент
- pop() выводит “error” и заканчивает программу, если в стеке отсутствуют элементы или удаляет последний элемент и присваивает новое значение в переменную mHead
- size() возвращает количество элементов в стеке. Начиная с последнего элемента он прибавляет 1 в переменную size и переключается на предыдущий элемент до тех пор, пока это возможно
- empty() возвращает true если указатель на текущий элемент стека равен nullptr (т.е. он отствутвует) и false в противном случае
- top() выводит “error” и заканчивает программу, если в стеке отсутствуют элементы или возвращает значение последнего элемента стека.

**Тестирование.**

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1 | cmd_push 1<br>cmd_top<br>cmd_push 2<br>cmd_top<br>cmd_pop<br>cmd_size<br>cmd_pop<br>cmd_size<br>cmd_exit | ok<br>1<br>ok<br>2<br>2<br>1<br>1<br>0<br>bye | Программа работает верно. |
| 2 | cmd_size<br>cmd_top | 0<br>error | Программа работает верно. |
| 3 | cmd_empty<br>cmd_push 42<br>cmd_epmty<br>cmd_pop<br>cmd_pop | true<br>ok<br>false<br>42<br>error | Программа работает верно. |

**Выводы.**

Я изучил и освоил основы языка программирования C++ и работу с классами, методами, объектами. Основываясь на этих знаниях, была реализована работа стека на основе списка, который являлся объектом написанного класса. Для считывания пользовательских данных был использован оператор cin, а для их обработки метод find(). Далее основываясь на полученной информации производился вызов требуемых методов и ожидаемый результат выводился в консоль с помощью оператора cout.

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название программы: PR_Korenev_DA_lb4.cpp

```
#include <cstdlib>
#include <string>
#include <iostream>

using namespace std;

struct ListNode{
    ListNode* mNext;
    int mData;
};

class CustomStack {
public:
    CustomStack(){
        mHead = nullptr;
    }
    ~CustomStack(){
        ListNode* cur = mHead;
        while (cur != nullptr){
            ListNode* before = cur->mNext;
            delete cur;
            cur = before;
        }
    }

    void push(int val){
        ListNode* cur;
        cur = new ListNode;
        cur->mNext = mHead;
        cur->mData = val;
        mHead = cur;
    }

    void pop(){
        if (mHead == nullptr){
            cout << "error" << endl;
            exit(0);
        }
        ListNode* tmp = mHead;
        mHead = mHead->mNext;
        delete tmp;
    }

    size_t size(){
        ListNode* cur = mHead;
        size_t size = 0;
        while (cur != nullptr){
            size++;
            cur = cur->mNext;
        }
        return size;
    }
```

```
    bool empty(){
        return (mHead == nullptr);
    }

    int top(){
        if (mHead == nullptr) {
            cout << "error" << endl;
            exit(0);
        }
        return mHead->mData;
    }

protected:
    ListNode* mHead;
};

int main() {
    CustomStack stack;
    string input;

    while(cin >> input){
        if(input.find("cmd_push") == 0){
            int value;
            cin >> value;
            stack.push(value);
            cout << "ok" << endl;

        }else if (input.find("cmd_top") == 0){
            cout << stack.top() << endl;

        } else if (input.find("cmd_pop") == 0){
            cout << stack.top() << endl;
            stack.pop();

        } else if (input.find("cmd_size") == 0){
            cout << stack.size() << endl;

        } else if (input.find("cmd_exit") == 0){
            cout << "bye";
            break;

        } else if (input.find("cmd_empty") == 0){
            cout << stack.empty() << endl;
            break;
        }
    }

    return 0;
}
```