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.0382 _____ Диденко Д.В.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2020

Цель работы.

Изучение динамических структур данных.

Задание.

Вариант 4.

Моделирование стека.

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.

Примечания:

Указатель на голову должен быть protected.

Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.

Предполагается, что пространство имен std уже доступно.

Использование ключевого слова using также не требуется.

Структуру ListNode реализовывать самому не надо, она уже реализована.

Основные теоретические положения.

Стек - это структура данных, в которой хранятся элементы в виде последовательности, организованной по принципу LIFO (Last In — First Out). Такую структуру данных можно сравнить со стопкой тарелок или магазином автомата. Стек не предполагает прямого доступа к элементам и список основных операций ограничивается операциями помещения элемента в стек и извлечения элемента из стека. Их принято называть PUSH и POP

соответственно. Также, обычно есть возможность посмотреть на верхний элемент стека не извлекая его (TOP) и несколько других функций, таких как проверка на пустоту стека и некоторые другие.

Очередь - это структура данных, в которой хранятся элементы в виде последовательности, организованной по принципу FIFO (First In — First Out). Эта структура данных более естественна - например, очередь в магазине. Также как и стек, очередь не предполагает прямого доступа к элементам, а основные операции: добавление ENQ (enqueue) и извлечение DEQ(dequeue). Также обычно есть функции получения первого элемента без его извлечения, определения размера очереди, проверки на пустоту и некоторые другие.

Выполнение работы.

Исходный код решения задачи см.в приложении А.

Объявлена структура ListNode с полями mNext и mData. Создан класс — CustomStack. Приватные члены класса — size_stack, хранящий текущий размер стека, защищенные — mHead, хранящий указатель на верхний элемент стека, общедоступные функции-члены — деструктор; push() - добаляет новый элемент в стек; pop() - удаляет верхний элемент; top() - показывает верхний элемент; size() - возвращает текущий размер стека; empty() - возвращает пустой ли стек в текущий момент.

Void push(int val) — принимает целочисленное значение и добавляет его в стек. В теле с каждым вызовом функции создается новая структура ListNode - B, в которую записывает val, следующим элементом для B становится mHead, а mHead присваивается адрес B. Размер стека инкрементируется. В консоль выводится «ok».

Void pop() - в теле функции происходит проверка на размер стека — если он пустой, то функции удалять нечего, программа завершает работу с выводом ошибки. Если стек не пуст, в консоль выводится удаляемое значение, создается указатель на структуру ListNode, ему присваивается адрес следующего элемента за mHead, память под mHead освобождается и

mHead присваивается адрес нового указателя. Т.е. mHead теперь имеет адрес следующего за ним элемента(нижнего).

Void top() - если стек не пуст — выводит в консоль верхний элемент без удаления, если пуст — программа завершается.

В функции main() пользователь через stdin вводит поочередно предоставленные команды. Выбор действия осуществляется с помощью оператора swich.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	cmd_push 5 ok cmd_top 5 cmd_pop 5 cmd_size 0 cmd_pop	error	Программа работает верно
2.	cmd_push 6 ok cmd_pop 6 cmd_exit	bye	Программа работает верно

Выводы.

Были изучены динамические структуры данных – стек и очередь.

Разработана программа, в которой реализован стек на базе списка. Программа выполняет команды, введенные пользователем, а в случае невозможности исполнения выводит «error» и завершает работу.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4_4.cpp

```
/*#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;            Уже подключены.

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

class CustomStack {

private:    // поля класса, к которым не должно быть доступа извне

    int size_stack = 0;

protected:  // в этом блоке должен быть указатель на голову
    ListNode* mHead = NULL;

public:
    ~CustomStack(){
        delete mHead;
    }
    void push(int val){
        ListNode* B = new ListNode;
        B->mData = val;
        B->mNext = mHead;
        mHead = B;
        size_stack++;
        cout << "ok\n";
    };                              //  добавляет новый элемент в стек
    void pop(){
        if(size_stack == 0){
            cout << "error" << endl;
            exit(0);
        }
        cout << mHead->mData << endl;
        ListNode* ptr;
        ptr = mHead->mNext;
        delete mHead;
        mHead = ptr;
        size_stack--;
    };                      //  удаляет из стека последний элемент
    int top(){
       if(mHead != NULL)
           return mHead->mData; //  доступ к верхнему элементу
        else{
           cout << "error" << endl;
           exit(0);
       }
    };

    size_t size(){
```

```
        return size_stack;
    };                  //  возвращает количество элементов в стеке
    bool empty(){
        if(size_stack == 0){
            return true;
        }else return false;
    };                  //  проверяет отсутствие элементов в стеке
};

int main()
{
    const int len_input = 10;
    CustomStack Stack;
    char comands[5][len_input] = {"cmd_push","cmd_pop","cmd_top",
                                 "cmd_size","cmd_exit"};
    int choice;
    char el_push[20];
    char input[len_input];
    while(true){
        cin >> input;
        if(!strcmp(input,comands[0])){
            cin >> el_push;
            choice = 1;
        }
        for(int j = 1;j<5;j++){
            if(!strcmp(input,comands[j])){
                choice = j+1;
            }
        }
        switch (choice) {
            case 1:
                Stack.push(atoi(el_push));
                break;
            case 2:
                Stack.pop();
                break;
            case 3:
                cout << Stack.top() << endl;
                break;
            case 4:
                cout << Stack.size() << endl;
                break;
            case 5:
                cout << "bye\n";
                exit(0);
            default:
                cout << "try again\n";
        }
        choice = 0;
    }
    return 0;
}
```