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 _____ Крючков А. М.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучение динамических структур данных.

Задание.

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.

Выполнение работы.

В качестве стека используется *class CustomStack*, в нём реализованы следующие функции:

- Конструктор — инициализирует приватные переменные.
- Деструктор — освобождает память из-под списка.
- *void push(int val)* — добавляет элемент в стэк.
- *void pop()* — удаляет последний элемент из стэка, если стэк пуст, то выводит «error» и завершает программу с кодом 0.
- *int top()* — возвращает последний элемент, добавленный в стэк. Если стэк пуст, то выводит «error» и завершает программу с кодом 0.
- *size_t size()* — выводит размер стэка.
- *Bool empy()* — возвращает значение true, если стэк имеет хотя бы один элемент, и false в противном случае.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	cmd_push 1 cmd_top cmd_push 2 cmd_top cmd_pop cmd_size cmd_pop cmd_size cmd_exit	ok 1 ok 2 2 1 1 0 bye	Ок

Выводы.

Были изучены динамические структуры данных – стек и очередь. Разработана программа, в которой реализован стек на базе списка.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src/main.cpp

```
#include <iostream>
#include <sstream>
#include <cstring>

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

class CustomStack {

public:
       // методы push, pop, size, empty, top + конструкторы, деструктор
    CustomStack() {
                                    mHead = static_cast<ListNode *>(malloc(sizeof(ListNode)));
        lastNode = NULL;
        mHead = NULL;
    }

    ~CustomStack() {
        ListNode *current = mHead;
        while (current != NULL) {
            mHead = mHead->mNext;
            free(current);
            current = mHead->mNext;


        }
    }

    void push(int val) {
        if (mHead == NULL) {
                                        mHead = static_cast<ListNode *>(malloc(sizeof(ListNode)));
            mHead->mData = val;
            mHead->mNext = NULL;
            lastNode = mHead;
            return;
        }
                            lastNode->mNext = static_cast<ListNode *>(malloc(sizeof(ListNode)));
        lastNode = lastNode->mNext;
        lastNode->mNext = NULL;
        lastNode->mData = val;
    }
```

```
    void pop() {
        if (mHead == NULL) {
            std::cout << "error" << std::endl;
            exit(0);
        }
        if (mHead->mNext == NULL) {
            std::cout << mHead->mData << std::endl;
            free(mHead);
            mHead = NULL;

            return;
        }
        ListNode *current = mHead;
        while (current->mNext->mNext != NULL) {
            current = current->mNext;
        }
        std::cout << current->mNext->mData << std::endl;
        free(current->mNext);
        lastNode = current;
        current->mNext = NULL;
    }

    int top() {
        if (mHead == NULL) {
            std::cout << "error" << std::endl;
            exit(0);
        }

        return lastNode->mData;
    }

    size_t size() {
        ListNode *current = mHead;
        size_t size = 0;
        while (current != NULL) {
            size++;
            current = current->mNext;
        }
        return size;
    }


    bool empty() {
        return mHead == NULL;
    }


private:
    ListNode *lastNode;




// поля класса, к которым не должно быть доступа извне
```

```
protected:
    // в этом блоке должен быть указатель на массив данных
    ListNode *mHead;
};


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) {
        std::cin >> input;
        if (!strcmp(input, comands[0])) {
            std::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));
                std::cout << "ok\n";
                break;
            case 2:
                Stack.pop();
                break;
            case 3:
                std::cout << Stack.top() << std::endl;
                break;
            case 4:
                std::cout << Stack.size() << std::endl;
                break;
            case 5:
                std::cout << "bye\n";
                exit(0);
            default:
                std::cout << "try again\n";
        }
        choice = 0;
    }
}
```