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

ОТЧЁТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Динамические структуры данных.

Студент гр. 1304 _____ Мусаев А.И.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы:

Целью работы является написать программу, моделирующую стек на базе списка.

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.

Выполнение работы:

В работе было реализован стэк. В нём были реализованы функции push, pop, top, size и empty. Все функции выполняют поставленные им условия в задании. Программа считывает поданную ей команду и в зависимости от команды выполняет условие.

- **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

Программа распознает команду с помощью условного ветвления и в зависимости от этого вызывает метод в стеке.

- **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

Тестирование:
Input:
cmd_push 1
cmd_top
cmd_push 2
cmd_top
cmd_pop
cmd_size
cmd_pop
cmd_size
cmd_exit

Результат
Output:
ok
1
ok
2
2
1
1
0
bye

Комментарий
Верно.

Вывод:
Было реализован стек на базе списка.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: 1.cpp

```
#include <iostream>
#include <string>
using namespace std;

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

class CustomStack
{

public:
void push(int a)
{
    counter++;
    ListNode *new_el = (ListNode *)malloc(sizeof(ListNode));
    new_el->mNext = mHead;
    new_el->mData = a;
    mHead = new_el;
}

void pop()
{
    if (counter == 0)
    {
        cout << "error" << endl;
        exit(0);
    }
    ListNode *new_el = mHead->mNext;
    free(mHead);
    mHead = new_el;
    counter--;
}

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

size_t size()
{
    return counter;
}

bool empty()
{
    return counter == 0;
}
// методы push, pop, size, empty, top + конструкторы, деструктор
```

```
private:
    int counter = 0, flag, i;

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

int main()
{
    CustomStack myStack;
    string s;
    int a, flag = 1;
    while (flag)
    {
        cin >> s;
        if (s.compare("cmd_push") == 0)
        {
            cin >> a;
            myStack.push(a);
            cout << "ok" << endl;
            continue;
        }
        else if (s.compare("cmd_top") == 0)
        {
            cout << myStack.top() << endl;
            continue;
        }
        else if (s.compare("cmd_pop") == 0)
        {
            cout << myStack.top() << endl;
            myStack.pop();
            continue;
        }
        else if (s.compare("cmd_size") == 0)
        {
            cout << myStack.size() << endl;
            continue;
        }
        else if (s.compare("cmd_exit") == 0)
        {
            cout << "bye" << endl;
            break;
        }
        else
        {
            cout << "error" << endl;
            exit(0);
        }
    }
}
```