**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. 3344** |  | **Сьомак Д.А.** |
| **Преподаватель** |  | **Глазунов** **С.А.** |

**Санкт-Петербург**

**2024**

**Цель работы**

Получение практических навыков работы с ООП на языке C++. Реализация программы, моделирующей работу стека на базе массива.

**Задание**

## Вариант 3

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

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

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 массив данных  
  
 int\* mData;  
};

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

extend(int n) - расширяет исходный массив на n ячеек

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**

Был реализован класс CustomStack. Весь функционал данного класса (его поля, методы) был описан в задании и реализован в соответствии. Помимо класса было реализовано считывание строки-команды из консоли. Внутри функции main создаётся объект класс CustomStack и программа считывает строку inp\_cmd из консоли, выполняет действие, которое соответствует данной команде по перечню, прописанному в задании. Если команда не является cmd\_exit, то программа считывает следующую строку-команду из консоли посредством цикла while, иначе заканчивает свою работу. Также предусмотрен вывод ошибки и завершение программы в случае вызова методов pop и top при пустом стеке.

Исходный код см. в приложении А.

**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 | - |

## **Выводы**

Был получен практический опыт работы с ООП на языке C++. Был освоен новый вид динамической структуры - стек. Была написана программа, внутри которой было реализовано моделирование работы стека.

# **ПРИЛОЖЕНИЕ А** **ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: Somak\_Demid\_lb4.c

class CustomStack {

public:

CustomStack(){

mData = new int[10];

len = 0;

memory = 10;

}

~CustomStack() {

delete[] mData;

}

void push(int val){

if(len == memory){

extend(1);

}

mData[len++] = val;

}

void pop(){

len--;

}

int top(){

return mData[len-1];

}

size\_t size(){

return len;

}

bool empty(){

return len == 0;

}

void extend(int n){

memory += n;

int \*tmp\_arr = new int[memory];

for (int i = 0; i < len; ++i) {

tmp\_arr[i] = mData[i];

}

delete[] mData;

mData = tmp\_arr;

}

protected:

int\* mData;

private:

size\_t len;

size\_t memory;

};

int main(){

CustomStack stack;

string inp\_cmd;

while (true) {

cin >> inp\_cmd;

if (inp\_cmd == "cmd\_push") {

int n;

cin >> n;

stack.push(n);

cout << "ok" << endl;

} else if (inp\_cmd == "cmd\_pop") {

if (stack.empty()) {

cout << "error" << endl;

break;

}

cout << stack.top() << endl;

stack.pop();

} else if (inp\_cmd == "cmd\_top") {

if (stack.empty()) {

cout << "error" << endl;

break;

}

cout << stack.top() << endl;

} else if (inp\_cmd == "cmd\_size") {

cout << stack.size() << endl;

} else if (inp\_cmd == "cmd\_exit") {

cout << "bye" << endl;

break;

} else{

cout << "Unknown command" << endl;

}

}

return 0;

}