

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

Гусева Е.А.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучить указатели на функции, динамические структуры данных, юнит-тестирования, введение в C++, написание кода для лабораторной работы.

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, который имеет возможность хранить и работать с типом данных int, реализован метод push, который добавляет новый элемент в стек, метод pop, который удаляет из стека последний элемент, метод top, который возвращает верхний элемент, метод size, который возвращает количество элементов в стеке, метод empty, который проверяет отсутствие элементов в стеке, метод extend, который расширяет исходный массив на n ячеек. Было обеспечено 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. Был реализован перечень команд: 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 "Some error" и завершиться.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

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

Выводы

Были изучены указатели на функции, динамические структуры данных, юнит-тестирования, получены навыки работы на языке C++, написан код для лабораторной работы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла main_for_lb4.cpp

```
class CustomStack {
public:
    CustomStack(){
        mData = new int[10];
        len = 0;
        capacity = 10;
    }

    ~CustomStack() {
        delete[] mData;
    }

    void push(int val){
        if(len == capacity){
            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){
        capacity += n;
        int* temp = new int[capacity];
        memcpy(temp, mData, (capacity * sizeof(int)));
        delete[] mData;
        mData = temp;
    }
protected:
    int* mData;

private:
    size_t len;
    size_t capacity;
};

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\n";
            break;
        }

        cout << stack.top() << "\n";
        stack.pop();

    } else if (inp_cmd == "cmd_top") {

        if (stack.empty()) {
            cout << "error\n";
            break;
        }

        cout << stack.top() << "\n";

    } else if (inp_cmd == "cmd_size") {

        cout << stack.size() << "\n";

    } else if (inp_cmd == "cmd_exit") {

        cout << "bye\n";
        break;

    } else{

        cout<<"Some error";

    }
}
return 0;
}

```