**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. 3342 |  | Колесниченко М.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью данной лабораторной работы является изучение работы с динамическими структурами данных и их создание. Также одна из целей – изучение основ работы с языком C++. Требуется написать программу, моделирующую 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.

**Примечания:**

1. Указатель на массив должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.
3. Предполагается, что пространство имен std уже доступно.
4. Использование ключевого слова using также не требуется.
5. Методы не должны выводить ничего в консоль.

## Выполнение работы

Реализован класс CustomStack, который имеет следующие методы: push, pop, size, empty, top. Он имеет приватные поля, содержащие размер и вместительность стека. В защищенном поле mData находятся данные стека. Реализован main() в котором считываются и выполняются пользовательские команды. Есть проверка на пустоту массива, при вызове pop и top.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 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

Была разработана программа на языке C++, которая создаёт динамическую структуру данных – стек на базе массива. Реализованы методы для работы с созданной структурой и считывание пользовательских команд и их выполнение.

# Приложение А Исходный код программы

Название файла: main.cpp

class CustomStack {

public:

CustomStack() : mData(nullptr), mCapacity(0), mSize(0) {}

CustomStack(const CustomStack& other) : mCapacity(other.mCapacity), mSize(other.mSize) {

mData = new int[mCapacity];

for (size\_t i = 0; i < mSize; ++i) {

mData[i] = other.mData[i];

}

}

~CustomStack() {

delete[] mData;

}

void push(int val) {

if (mSize == mCapacity) {

extend(1);

}

mData[mSize++] = val;

}

void pop() {

--mSize;

}

int top() {

return mData[mSize - 1];

}

size\_t size() {

return mSize;

}

bool empty() {

return mSize == 0;

}

void extend(int n) {

int\* newData = new int[mCapacity + n];

for (size\_t i = 0; i < mSize; ++i) {

newData[i] = mData[i];

}

delete[] mData;

mData = newData;

mCapacity += n;

}

protected:

int\* mData;

private:

size\_t mCapacity;

size\_t mSize;

};

int main() {

CustomStack stack;

std::string command;

while (std::cin >> command) {

if (command == "cmd\_push") {

int n;

std::cin >> n;

stack.push(n);

std::cout << "ok\n";

} else if (command == "cmd\_pop") {

if (stack.empty()) {

std::cout << "error";

return 0;

}

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

stack.pop();

} else if (command == "cmd\_top") {

if (stack.empty()) {

std::cout << "error";

return 0;

}

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

} else if (command == "cmd\_size") {

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

} else if (command == "cmd\_exit") {

std::cout << "bye\n";

break;

}

}

return 0;

}