МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Динамические структуры данных.

Студент гр. 0382 _____ Павлов С.Р

Преподаватель _____ Берленко Т.А

Санкт-Петербург

2021

Цель работы.

Работа с динамическими структурами данных.

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

Основные теоритические 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.

Классы в **C++** — это абстракция описывающая методы, свойства, ещёне существующих объектов. Объекты — конкретное представлениеабстракции, имеющее свои свойства и методы. Созданные объекты на основеодного класса называются экземплярами этого класса. Эти объекты могутиметь различное поведение, свойства, но все равно будут являться объектамиодного класса.

Выполнение работы.

Исходный код программы, начинается с описания класса ***CustomStack*** ,

Во-первых описывается конструктор класса ***CustomStack()***,

а затем *методы класса*:

- **void push(int val)** — добавляет элемент в стек (с помощью функции *extend()*, которая добавляет ячейку памяти для нового элемента, и присвоенияэтой ячейке элемента *val*).
- **void pop()** — удаляет элемент стека и выводит его значение на экран.
- **int top()** — выводит значение верхнего элемента стека.
- **size_t size()** — выводит размер стека.
- **bool empty()** — проверяет пустой стек или нет.
- **void extend()** — увеличивает стек на n ячеек.

После указания класса, описывается функция *main,* инициализируется переменные-объекты класса типа *string,* и *stack.* Для приема комманд пользователя (потока stdin) и создание стека, соответственно. Далее с помощью цикла *while*, и операторов if и else if, реализуется функционал программы и вызываются соотвествующие методы класса.

Разработанный программный код смотрите в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 — Результаты тестирования.

	Входные данные	Выходные данные	Коментарии
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	Программа работает верно
3.	cmd_pop	error	Программа работает верно

Выводы.

Была выполнена работа с динамическими структурами данных.
Разработана программа, которая создает стек на базе класса и выполняет методы по работе со стеком, которые написаны в соответствующем классе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: ***main.cpp***

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

void push(int val){
extend(1);
mData[stack_size] = val;
stack_size += 1;
}

void pop(){
mData[stack_size - 1] = 0;
stack_size -= 1;
}

int top(){
return mData[stack_size - 1];
}
size_t size(){
return stack_size;
}
bool empty(){
if(stack_size == 0)
return true;
else
return false;
}

private:
int stack_size;
void extend(int n){
mData = (int*) realloc(mData, n * sizeof(int));
}

protected:
int* mData;
};


int main() {

CustomStack stack;

int n;
```

```
string* commands;
string command = "a";

while (true){
cin >> command;

if (command == "cmd_exit"){
cout << "bye\n";
return 0;
}
else if (command == "cmd_push"){
cin >> n;
stack.push(n);
cout << "ok\n";
}
else if (command == "cmd_pop"){
if (stack.empty() == false){
cout << stack.top() << endl;
stack.pop();
}
else{
cout << "error\n";
return 0;
}
}
else if (command == "cmd_top"){
if (stack.empty() == false) {
cout << stack.top() << endl;
}
else{
cout << "error\n";
return 0;
}
}
else if (command == "cmd_size"){
cout << stack.size() << endl;
}
}
return 0;
}
```