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		**Анахин Е.Д.**
Преподаватель		**Глазунов С.А.**

Санкт-Петербург

2024

Цель работы

Изучение работы с классами и структурами данных на языке программирования C++

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: Anakhin_Egor_lb4

```
class CustomStack {

public:

bool push(int number) {
if (!isFull()) {
mData[leng++] = number;
return true;
}
return false;
};

bool pop() {
if (size() == 0) {
std::cout « "error";
return false;
}
leng--;
return true;
};

int top() {
if (isEmpty()) {
std::cout « "error";
return 0;
}
return mData[leng-1];
};

bool isEmpty() {
return leng == 0;
}

bool isFull() {
return leng == capacity;
};

size_t size() {
return leng;
};

void extend(int newLength) {
int* newArray = new int[newLength];

for (int i = 0; i < leng && i < newLength; i++) {
newArray[i] = mData[i];
}

delete[] mData;
mData = newArray;
capacity = newLength;
```

```
};

CustomStack() {
capacity = 100;
leng = 0;
mData = new int[capacity];
}

~CustomStack() {
delete[] mData;
};

bool makeMath(char operation) {
if (size() < 2) {
return false;
}

int num2 = top();
pop();
int num1 = top();
pop();
int newNum;

if (num2 == 0 || num1 == 0) {
return false;
}

switch (operation)
{
case '/':
newNum = num1 / num2;
break;
case '*':
newNum = num1 * num2;
break;
case '+':
newNum = num1 + num2;
break;
case '-':
newNum = num1 - num2;
break;
default:
return false;
break;
}

push(newNum);
return true;
};

protected:
int* mData;

private:
int leng;
int capacity;
```

```
};

int main() {

CustomStack stack;

std::string line;
std::getline(std::cin, line);
line.push_back('\0');

char* str = &line[0];
char* token = std::strtok(str, " ");

while (token != nullptr) {
try {
int number = std::stoi(token);
stack.push(number);
} catch(std::invalid_argument) {
bool isCorrect = stack.makeMath(token[0]);
if (!isCorrect) {
std::cout « "error";
return 0;
}
}
token = std::strtok(nullptr, " ");
}

if (stack.size() != 1) {
std::cout « "error";
return 0;
}

std::cout « stack.top() « std::endl;

return 0;

}
```