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. 1304 _____ Заика Т.П

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Моделирование стека. 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:

Указатель на массив должен быть protected.

Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.

Предполагается, что пространство имен std уже доступно.

Использование ключевого слова using также не требуется.

Методы не должны выводить ничего в консоль.

Основные теоретические положения.

Стек.

Синтаксис языка C++.

Выполнение работы.

В ходе работы для решения поставленной задачи было принято создать программу, моделируюущую работу стека. Для этого, согласно пункту 1, был реализован класс CustomStack, работающий с типом данных int, и благодоря своим методам, реализующий работу со стеком на базе массива. Для решения пункта 2 было решено использовать ввод с клавиатуры при помощи cin.getline() и цикла while, чтобы работа программы прекращалась по введению определенной команды. Для обработки программой введенных команд была создана строковая переменная, в которую записываются введенные с клавиатуры значения. Был использован блок try except для того, чтобы обработать команду cmd_push, которая является единственной командой, состоящей из двух слов. Благодоря обработке строковой переменной, каждая введенная команда реализует заданный в пунтке 2 функционал. Таким образом, полноценное выполнение пункта 1 и пункта 2 задачи целиком реализует необходимую программу.

Переменные:

unsigned int capacity — максимальный размер стека

int n — текущий размер стека (или элемент массива)

int* mData — указатель на массив данных

int c — переменная, хранящая максимальный размер стека по умолчанию

int new_capacity — новый размер стека, используемый при расширении исходного массива

int *tmp — временный массив нового размера, используемый для записи значения старого массива и инициализации массива стека нового размера

CustomStack s — экземпляр класса CustomStack, который моделирует работу стека

char *str — строка, в которую записываются введенные с клавиатуры команды

char (*tmp_str)[10] — временный массив строк, используемый для записи слов в случае вызова команды cmd_push

char* pch — указатель на строку, используемый для функционала strtok

char tmp[strlen(pch)] — временная строка размера текущего слова, используемая для записи слова в временный массив строк

Функции (Методы класса CustomStack):

CustomStack() - конструктор класса CustomStack, инициализирует массив

~CustomStack() - деструктор класса CustomStack, удаляет из динамической памяти массив

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

void extend(int n) — расширяет исходный массив на n ячеек

Разработанный программный 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	Успешный тест

Выводы.

Была исследован, изучен стек на базе массива, а также синтаксис языка C++.

Разработана программа, выполняющая моделирование работы стека на базе массива, и обеспечивающая считывание из потока stdin последовательности команд, выполняющих заданную функциональность.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <string.h>

using namespace std;


class CustomStack {
        private:
            unsigned int capacity;
            int n;
        public:
            CustomStack(){
                int c = 100;
                capacity = c;
                mData = new int[capacity];
                n = -1;
            }

            ~CustomStack(){
                delete[] mData;
            }

            void push(int val){
                n++;
                mData[n] = val;
            }

            void pop(){
                n--;
            }

            int top(){
                return mData[n];
            }

            size_t size(){
                return n+1;
            }

            bool empty(){
                return n == -1;
            }

            void extend(int n){
                int new_capacity = capacity + n;
                int *tmp = new int[new_capacity];
                for(int i=0; i<n; ++i){
                    tmp[i] = mData[i];
                }
                delete[] mData;
```

```
        mData = tmp;
        capacity = new_capacity;
    }
protected:
    int* mData;
};

int main() {
    CustomStack s;

    char *str = new char[20];


    while(cin.getline(str, 20)){
        try{
            char (*tmp_str)[10] = new char[2][10];
            int n = 0;
            char* pch;
            pch = strtok(str, " \n");
            while(pch != NULL){
                char tmp[strlen(pch)];
                strcpy(tmp, pch);
                strcpy(tmp_str[n], tmp);
                n++;
                pch = strtok(NULL, " \n");
            }

            if(!(strcmp(tmp_str[0], "cmd_push"))){
                s.push(atoi(tmp_str[1]));
                cout<<"ok"<<endl;
            } else {
                throw 1;
            }
        }
        catch(int a){
            if(!(strcmp(str, "cmd_top"))){
                if(s.empty() == 1){
                    cout<<"error"<<endl;
                    return 0;
                }
                cout<<s.top()<<endl;
            }

            if(!(strcmp(str, "cmd_pop"))){
                if(s.empty() == 1){
                    cout<<"error"<<endl;
                    return 0;
                }
                cout<<s.top()<<endl;
                s.pop();
            }

            if(!(strcmp(str, "cmd_size"))){
                cout<<s.size()<<endl;
            }
```

```
            if(!(strcmp(str, "cmd_exit"))){
                cout<<"bye"<<endl;
                return 0;
            }
        }
    }


    return 0;
}
```