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. 1304 ____ Маркуш А.Е.

Преподаватель ____ Чайка К.В

Санкт-Петербург

2022

Цель работы.

Изучение работы динамических структур данных в языке C++.

Задание

- 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.

Экспериментальные результаты

Входные данные	Вывод	Комментарий
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	Корректная работа программы

Выводы.

Были изучены основы языка C++, рассмотрена работа динамических структур данных. В качестве практического задания был написан стек на основе массива и продемонстрирована его работа.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#define N 20
#define M 200

using std::cout;

class CustomStack
{
public:
    CustomStack()
    {
        buffer_size = 0;
        elements_count = 0;
        mData = NULL;
    }

    ~CustomStack()
    {
        delete[] mData;
    }

    void push(int val)
    {
        if (elements_count == buffer_size)
            extend(100);
        mData[elements_count] = val;
        elements_count++;
    }

    void pop()
    {
        elements_count--;
    }

    int top()
    {
        return mData[elements_count-1];
    }

    size_t size()
    {
        return elements_count;
    }

    bool empty()
    {
```

```
        if (!elements_count) return true;
        return false;
    }

private:
    void extend(int n)
    {
        int *tmp = new int [buffer_size + n];
        memcpy(tmp, mData, sizeof(int)*elements_count);
        delete[] mData;
        mData = tmp;
        buffer_size += n;
    }

    int buffer_size;

    int elements_count;

protected:
    int *mData;

};
void FreeAll(char** Arr,int n)
{
    for(int i=0;i<n;i++)
        free(Arr[i]);
    free(Arr);
}
int main()
{
    CustomStack();
    CustomStack stack;
    char** Cmd_Arr= (char**)malloc(N*sizeof(char*));
    char* Inp_s = (char*)malloc(M*sizeof(char));
    fgets(Inp_s,M,stdin);
    int n=0;
    while(strcmp(Inp_s,"cmd_exit\n\0"))
    {
        Cmd_Arr[n]=(char*)malloc(M*sizeof(char));
        strcpy(Cmd_Arr[n],Inp_s);
        n++;
        fgets(Inp_s,M,stdin);
    }
    Cmd_Arr[n]=(char*)malloc(M*sizeof(char));
    strcpy(Cmd_Arr[n],Inp_s);
    n++;
    free(Inp_s);
    for(int i=0;i<n;i++)
    {

        char* p= strtok(Cmd_Arr[i]," \n\0");
        if(!strcmp(p,"cmd_push"))
        {
```

```
            char* p=strtok(NULL," \n\0");
            int numb = atoi(p);
            stack.push(numb);
            cout<<"ok\n";
            continue;
        }
        if(!strcmp(Cmd_Arr[i],"cmd_pop"))
        {
            if(stack.empty())
            {
                cout<<"error\n";
                return 0;
            }

            cout<<stack.top()<<"\n";
            stack.pop();
            continue;
        }
        if(!strcmp(Cmd_Arr[i],"cmd_top"))
        {
            if(stack.empty())
            {
                cout<<"error\n";
                return 0;
            }
            cout<<stack.top()<<"\n";
            continue;
        }
        if(!strcmp(Cmd_Arr[i],"cmd_size"))
        {
            cout<<stack.size()<<"\n";
            continue;
        }
        if(!strcmp(Cmd_Arr[i],"cmd_exit"))
        {
            cout<<"bye\n";
            break;
        }
    }
    FreeAll(Cmd_Arr,n);
    return 0;
}
```