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. 0382 _____ Санников В.А.

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 теоретические положения.

Стек – это структура данных, в которой элементы поддерживают принцип LIFO (“Last in – first out”): последним зашёл – первым вышел. Или первым зашёл – последним вышел.

Стек позволяет хранить элементы и поддерживает, обычно, две базовые операции:

- **PUSH** – кладёт элемент на вершину стека
- **POP** – снимает элемент с вершины стека, перемещая вершину к следующему элементу

Также часто встречается операция PEEK, которая получает элемент на вершине стека, но не снимает его оттуда.

Стек является одной из базовых структур данных и используется не только в программировании, но и в схемотехнике, и просто в производстве, для реализации технологических процессов и т.д.; стек используется в качестве вспомогательной структуры данных во многих алгоритмах и в других более сложных структурах.

Классы в C++ — это абстракция описывающая методы, свойства, ещё не существующих объектов. Объекты — конкретное представление абстракции, имеющее свои свойства и методы. Созданные объекты на основе одного класса называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса.

Выполнение работы.

Ход работы:

Для того, чтобы начать работать со стеком (реализованном на классе CustomStack) в main объявляем переменную choice типа char для определения функции, которую мы хотим выполнить, также не забываем объявить сам

класс Stack. Через цикл while вводим данные через cin и выбираем функцию с помощью условия if else. Если введено некорректное название функции, то программа предоставляет пользователю возможность ввести название снова. Если пользователь выполняет функцию top или pop на пустом стеке, то выводится сообщение об ошибке и программа завершается. (функции для работы со стеком реализованы в методах класса).

Конструктор и деструктор: CustomStack() и ~CustomStack.

Методы класса:

void push(int val) — добавляет элемент в стек (с помощью функции extend, которая добавляет ячейку памяти для нового элемента, и присвоения этой 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.

Исходный код см. в приложении А.

Тестирование.

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	ok 1 ok 2 2 1 1 0	Программа выполнила все методы класса без ошибок.

	cmd_exit	bye	
2.	cmd_push 123 cmd_push -100 cmd_size cmd_pop cmd_exit	ok ok 2 -100 bye	Программа работает верно.

Выводы.

Были выполнена работа с динамическими структурами данных.

Разработана программа, которая реализовывает стек на базе класса и выполняет методы по работе со стеком, которые написаны в соответствующем классе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

    ~CustomStack(){
        delete[] mData;
    }

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

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

    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(){
```

```
    int n;
    char choice[100];
    CustomStack Stack;

    while(true){
        choice[0] = '\0';
        cin >> choice;
        if(!strcmp(choice, "cmd_push")){
            cin >> n;
            Stack.push(n);
            cout << "ok" << endl;
        }else if(!strcmp(choice, "cmd_size")){
            cout << Stack.size() << endl;
        }else if(!strcmp(choice, "cmd_pop")){
            if(Stack.empty() == 1){
                cout << "error";
                exit(0);
            }
            Stack.pop();
        }else if(!strcmp(choice, "cmd_top")){
            if(Stack.empty() == 1){
                cout << "error";
                exit(0);
            }
            cout << Stack.top() << endl;
        }else if(!strcmp(choice, "cmd_exit")){
            cout << "bye";
            exit(0);
        }else{
            continue;
        }
    }
}
```