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. 3343 _____ Поддубный В.А.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург

2024

Цель работы

Изучить особенности реализации классов на языке C++ и освоить работу с ними. Реализовать на основе списка динамическую структуру данных стек, с использованием ООП.

Задание

Требуется написать программу, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе **списка.**

1) Реализовать **класс** CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных ***int.***

Структура класса узла списка:

```
struct ListNode {
   ListNode* mNext;
   int mData;
};
```

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 голову
   ListNode* mHead;
};
```

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()** - доступ к верхнему элементу
- **size_t size()** - возвращает количество элементов в стеке
- **bool empty()** - проверяет отсутствие элементов в стеке

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 голову 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. Структуру **ListNode** реализовывать самому не надо, она уже реализована.

Выполнение работы

Описание класса *CustomStack*:

public методы:

- *CustomStack()* – конструктор класса, заполняющий поля нулевыми данными.
- *empty()* – проверка наличия элементов в стеке.
- *top()* – возвращает данные в верхнем элементе стека, если это возможно.
- *size()* – возвращает размер стека.
- *push(int value)* – добавляет новый элемент в стек.
- *pop()* – удаляет элемент из стека, если это возможно.
- *change(string value)* – удаляет два элемента из стека и в зависимости от полученного значения *value* добавляет сумму, разность, произведение или частное от деления удалённых элементов в стек.
- *~CustomStack()* – деконструктор класса, очищающий стек
- CustomStack() – конструктор класса, инициализирующий mHead

В области *protected* находится ссылка на голову стека *mHead*.

Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1.

№	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_exi	ok 1 ok 2 2 1 1 0 bye	Вывод соответствует ожиданиям.
3.			

Выводы

Во время выполнения лабораторной работы мы ознакомились с синтаксисом языка C++ по работе с классами, а также написали программу с использованием стека на основе списка.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
class CustomStack {
public:
    CustomStack() {
        mHead = nullptr;
    }

    ~CustomStack() {
        while (mHead) {
            pop();
        }
    }

    void push(int val) {
        auto current = mHead;
        auto *listNode = new ListNode;
        listNode->mNext = current;
        listNode->mData = val;
        mHead = listNode;
    }

    int pop() {
        auto current = mHead;
        int data = current->mData;
        mHead = mHead->mNext;
        delete current;
        return data;
    }

    int top() {
        return mHead->mData;
    }

    size_t size() {
        int count = 0;
        auto current = mHead;
        while (current) {
            current = current->mNext;
            count++;
        }
        return count;
    }

    bool empty() {
        return mHead == nullptr;
    }

protected:
    ListNode *mHead;
};

int main() {
```

```
    CustomStack stack;
    string command;
    int value;

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