

**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

Старков С.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 (не более 100 элементов) из чисел и арифметических операций (+, -, \*, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

- Если очередной элемент входной последовательности - число, то положить его в стек,
- Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже),
- Если входная последовательность закончилась, то вывести результат (число в стеке).

Если в процессе вычисления возникает ошибка:

- например вызов метода **pop** или **top** при пустом стеке (для операции в стеке не хватает аргументов),
- по завершении работы программы в стеке более одного элемента,

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 и возвращает его значение, 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.

В области *private* находится размер стека *mSize*.

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

Описание основной части:

В `string mDataBuffer` происходит считывание строк через пробел до символа '\n'. Если строка является числом, то с помощью функции `stoi()` строка приводится к типу `int`, иначе над двумя верхними элементами стека производится операция, введенная пользователем. В конце, если в стеке больше одного элемента, выводится “error”, иначе значение в голове стека.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1.

№	Входные данные	Выходные данные	Комментарии
1.	1 2 + 3 4 - 5 * +	-2	Вывод соответствует ожиданиям.
2.	1 + 5 3 -	error	
3.	-12 -1 2 10 5 -14 17 17 * - - + - * +	304	

## **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.c

```
class CustomStack {
public:
    CustomStack()
    {
        mHead= nullptr;
        mSize = 0;
    }

    void push(int val){
        ListNode* node = new ListNode;
        node->mData = val;
        node->mNext = mHead;
        mHead = node;
        mSize++;
    }

    long long pop(){
        if(empty())
        {
            cout<< "error";
            exit(0);
        }
        long long data = mHead->mData;
        mHead = mHead->mNext;
        mSize--;
        return data;
    }

    long long top(){
        if(empty()){
            cout<<"error";
            exit(0);
        }
        return mHead->mData;
    }

    size_t size(){
        return mSize;
    }

    bool empty(){
        return mHead == nullptr;
    }
private:
    size_t mSize;
protected:
    ListNode* mHead;
};
```



```

CustomStack calc(CustomStack s, string op){
    int a = s.pop();
    int b = s.pop();
    if(op == "+"){
        s.push(b+a);
    }
    else if(op == "-"){
        s.push(b - a);
    }
    else if (op == "*")
        s.push(a * b);

    else if (op == "/")
        s.push(b / a);
    return s;
}

int main() {
    CustomStack stack;
    string dBuffer;
    while(cin.peek() != '\n'){

        cin>>dBuffer;
        if(dBuffer == "+" || dBuffer == "-" || dBuffer == "/" ||
dBuffer == "*")
        {
            stack = calc(stack,dBuffer);
        }
        else{
            stack.push(stoi(dBuffer));
        }

    }

    if(stack.size() > 1){
        cout << "error";
        return 1;
    }
    cout << stack.top();
    return 0;
}

```