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 _____ Байков Е.С.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы

Изучение работы с динамическими структурами в языке C++.

Задание

Вариант 2

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

Структура *ListNode* предоставляет из себя элемент списка, содержащий указатель *mNext* на следующий элемент и числовое значение *mData.*

Стек реализован с помощью класса *CustomStack.* Защищенные поля: *mHead* — указатель на верхний элемент, и *mSize* — количество элементов в стеке. Для работы со стеком реализованы публичные методы.

void push(int val) – эта функция добавляет элемент в стек: создаёт объект структуры *ListNode*, куда записывается значение и указатель на предыдущий верхний элемент.

void pop() – эта функция удаляет верхний элемент стека: если элементов нет, выдаёт ошибку; если элемент один, то верхнему элементу присваивается *NULL*; иначе нынешний верхний элемент удаляется и верхним становится следующий элемент.

int top() – функция возвращает верхний элемент, если он существует, иначе выводит ошибку.

size_t size() – функция возвращает размер стека.

bool empty() – функция возвращает *true*, если стек не пуст, иначе *false*.

CustomStack() и *~CustomStack()* – конструктор и деструктор соответственно.

В функции main() создается экземпляр класса *CustomStack*. Также реализовано чтение с командной строки чисел и операций. Для начала считывается два символа: если первый символ – число либо если первый символ – минус, а второй – число, то эти символы возвращаются в поток ввода с помощью функции cin.putback(), а затем считывается число целиком и добавляется в стек. Иначе, если первый символ – символ операции ‘+’, ‘-‘, ‘*’ или ‘/’, а второй символ пробельный, то из стека берутся два верхних элемента и над ними проводится соответствующая операция, а результат записывается в стек. Если попался неизвестный символ, то выводится ошибка. Далее происходит проверка на наличие символа перевода строки, в этом случае цикл чтения завершается. В итоге, если элемент в стеке последний, то он выводится, иначе выдаётся ошибка.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 -10 — 2 *	22	Верно
2.	1 2 + 3 4 – 5 * +	-2	Верно
3.	1 + 10 2 -	error	Верно

Выводы

Была изучена работа с динамическими структурами данных на языке C++.

Разработана программа, в которой реализована работа со стеком на основе однонаправленного линейного списка целых чисел. Программа считывает последовательность чисел и операций, введённых пользователем; записывает числа в стек либо производит операцию над двумя верхними элементами стека; если в результате в стеке остаётся один элемент, то выводит его, иначе выдаёт ошибку.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Baykov_Egor_lb4.c

```
#include <iostream>
using namespace std;

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

class CustomStack{
    public:
        void push(int val){
            struct ListNode* node = new ListNode();
            node->mNext = mHead;
            node->mData = val;
            mHead = node;
            mSize++;
        };
        void pop(){
            ListNode* p = mHead->mNext;
            delete(mHead);
            if(mSize < 1){
                cout << "error";
                exit(0);
            }
            else if(mSize == 1) mHead = NULL;
            else mHead = p;
            mSize--;
        };
        int top(){
            if(mSize < 1){
                cout << "error";
                exit(0);
            }
            return mHead->mData;
        };
        size_t size() { return mSize; };
        bool empty() { return !mSize; };
        CustomStack() { return; };
        ~CustomStack(){  delete(mHead); };
    protected:
        ListNode* mHead = NULL;
        size_t mSize = 0;
};

int main(){
    int num;
```

```
    char operation, ch;
    CustomStack stack;
    for(int i = 0; i < 100; i++){
        operation = getc(stdin);
        ch = getc(stdin);
        if(isdigit(operation)   ||   (operation   ==   '-'   &&
isdigit(ch))){
            cin.putback(ch);
            cin.putback(operation);
            cin >> num;
            stack.push(num);
        }
        else if((operation == '+' || operation == '-' || operation
== '*' || operation == '/') && isspace(ch)){
            cin.putback(ch);
            int  second  =  stack.top(),  first  =  (stack.pop(),
stack.top());
            stack.pop();
            switch(operation){
                case '+':
                    stack.push(second+first);
                    break;
                case '-':
                    stack.push(first-second);
                    break;
                case '*':
                    stack.push(second*first);
                    break;
                case '/':
                    if(second  ==  0){cout  <<  "error"  <<  endl;
exit(0);}
                    stack.push(first/second);
                    break;
            }
        }
        else cout << "error" << endl;
        if(getc(stdin) == '\n') break;
        else if((operation = getc(stdin)) == '\n') break;
        else cin.putback(operation);
    }

    if(stack.size() == 1) cout << stack.top() << endl;
    else cout << "error" << endl;;

    return 0;
}
```