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. 1304 _____ Макки К.Ю

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Научиться работать с динамическими структурами данных и реализовывать классы на языке C++.

Задание.

Вариант 2

Стековая машина.

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.

Класс CustomStack()

Объявляются метод public – метод и поля класса, которые будут доступны любым функциям, работающим с объектом данного класса. push() позволяет добавлять новый элемент в стек. pop() позволяет удалять элемент из стека, который является верхним. top() позволяет получить значение самого верхнего элемента списка. С помощью size() можно узнать количество элементов, которые находятся в стеке. Проверку на пустоту стека осуществляет empty(). Чтобы узнать, содержится в строке число или математическая операция, реализован NumCheck(). В check() происходит проверка на наличие элемента в стеке с помощью описанного ранее метода empty(), если нет, происходит вызов метода error(), выводящий сообщение об ошибке и осуществляющий завершение работы программы. Удаление всех элементов деструктором осуществляется методом pop(). В спецификаторе protected, в котором следует методы и классы поля, доступные как внутри класса, так и в наследуемых, содержит единственное поле указатель на голову списка mHead.

Main()

Выделяется память классу CustomStack. С помощью функции стандартной библиотеки scanf() считывается строка со значениями. Цикл while() осуществляет работу до тех пор, пока не будет достигнут конец строки. Далее проверяем является ли текущей элемент строки числом. Если да, то данное число будет добавлено в стек, иначе с верху стека извлекаются при возможности первые два элемента и согласно данному символу - операции с помощью математических действий создается новый элемент, который будет добавлен в стек. Далее считывается новый элемент. По

завершении цикла происходит проверка: сколько элементов осталось в стеке. Если число элементов не равно 1, на экран выводится сообщение об ошибке, в противном случае будет выведено значение элемента.

Тестирование

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2+ 3 4 — 5 * +	-2	Программа работает корректно.
2.	1 -10 - 2 *	22	Программа работает корректно.

Выводы.

В ходе лабораторной работы научился работать с динамическими структурами данных и реализовывать классы на языке C++. Написана программа, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе списка.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
class CustomStack
{
public:
    ~CustomStack()
    {
```

```
        while (mHead)
        {
            pop();
        }
    }
    void error()
    {
        printf("error\n");
        delete this;
        exit(0);
    }
    void check()
    {
        if (empty())
        {
            error();
        }
    }
    int NumCheck(char *s)
    {
        int len = strlen(s);
        if (len == 1 && !isdigit(s[0]))
        {
            return 0;
        }
        return 1;
    }
    void push(int val)
    {
        ListNode *upd = new ListNode{mHead, val};
        mHead = upd;
    }
    void pop()
    {
        check();
        ListNode *temp = mHead;
        mHead = mHead->mNext;
        delete temp;
    }
    int top()
    {
        check();
        return mHead->mData;
    }
    size_t size()
    {
        ListNode *cur = mHead;
        size_t n;
        for (n = 0; cur; n++)
        {
            cur = cur->mNext;
        }
        return n;
    }
    bool empty()
    {
        return !((bool)mHead);
    }

protected:
    ListNode *mHead;
```

```
};

int main()
{
    CustomStack *stack = new CustomStack();
    char cur[100];
    int first, secound;
    scanf("%s", cur);
    while (!feof(stdin))
    {
        if (stack->NumCheck(cur))
        {
            stack->push(atoi(cur));
        }
        else
        {
            first = stack->top();
            stack->pop();
            secound = stack->top();
            stack->pop();
            if (!strcmp(cur, "+"))
            {
                stack->push(first + secound);
            }
            else if (!strcmp(cur, "-"))
            {
                stack->push(secound - first);
            }
            else if (!strcmp(cur, "*"))
            {
                stack->push(first * secound);
            }
            else if (!strcmp(cur, "/"))
            {
                stack->push(secound / first);
            }
        }
        scanf("%s", cur);
    }
    if (stack->size() != 1)
    {
        stack->error();
    }
    first = stack->top();
    printf("%d\n", first);
    return 0;
}
```