

**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

Лобанов Е.А.

Преподаватель

Чайка К.В.

Санкт-Петербург

2022

## Цель работы.

Научиться работать с динамическими структурами и ознакомиться с основами языка C++.

## Задание.

### Вариант 4. Стековая машина.

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

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()** - доступ к верхнему элементу
- **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 (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 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.

#### **Пример:**

Исходная последовательность: 1 -10 - 2 \*

Результат: 22

#### **Выполнение работы.**

#### **Класс CustomStack :**

Поля класса:

- `int m_size` — количество переменных находящихся в стеке.
- `int m_max_size` — максимально доступный размер стека.
- `int* mData` — указатель на массив данных (на сам стек).

Методы класса :

- `CustomStack()` - конструктор класса, в котором выделяется память для массива `mData`.
- `~CustomStack()` - деструктор класса, в котором очищается память, выделенная для массива `mData`.
- `void extend(int n)` - расширяет существующий массив на `n` ячеек .
- `bool empty()` - проверяет отсутствие элементов в стеке.
- `int size()` - возвращает количество элементов в стеке.
- `int top()` - возвращает значение верхнего элемента в стеке.
- `void pop()` - удаляет верхний элемент из стека.
- `void push(int val)` - добавляет новый элемент `val` в стек.

### **Функция `main ( )` :**

Начинается с конструкции `try-catch`, которая отлавливает ошибки и выводит «error» в консоль при их наличии. Далее в теле `try` происходит инициализация стека, считывание данных из консоли и разбиение строки на символы при помощи `strtok()`. Если полученный символ является математической операцией, то она применяется для двух верхних чисел в стеке при помощи соответствующих методов класса стека. Если же полученный символ является числом, то оно кладется в стек. Далее следует проверка на наличие единственного числа в стеке, которое необходимо вывести в консоль и завершить программу. Если в стеке присутствует несколько чисел, то при помощи `throw` программа вызывает ошибку и завершает работу.

Разработанный программный код см. в приложении А.

## 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 2 + + +	error	Верно

## Вывод

Были изучены принципы создания динамических структур, работы с ними и основы написания программы на языке C++.

Разработана программа, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе массива. В процессе работы над программой были использованы классы, методы и поля классов, происходило считывание входных данных при помощи консоли и вывод в консоль. Также была реализована функция отлавливания ошибок при помощи конструкции try-catch.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

class CustomStack{
public:
    CustomStack(){
        mData = new int[m_max_size];
    }
    ~CustomStack(){
        delete[] mData;
    }
    void push(int val){
        if (m_size + 1 >= m_max_size)
            extend(10);
        mData[m_size++] = val;
    }
    void pop(){
        if (empty())
            throw 1;
        m_size--;
    }
    int top(){
        if (empty())
            throw 1;
        return mData[m_size - 1];
    }

    int size(){
        return m_size;
    }

    bool empty(){
        if (m_size)
            return false;
        else
            return true;
    }

    void extend(int n){
        m_max_size += n;
        int *newData = new int[m_max_size];
        memcpy(newData, mData, m_size * sizeof(int));
        delete[] mData;
        mData = newData;
    }

private:
    int m_size = 0;
```

```

        int m_max_size = 100;

protected:
    int *mData;
};

int main(){
    try{
        CustomStack stack;
        char input[100] = {0};
        fgets(input, 100, stdin);
        char *p = strtok(input, " \n");
        while (p != NULL){
            if (!strcmp(p, "+")){
                int top = stack.top();
                stack.pop();
                int bottom = stack.top();
                stack.pop();
                stack.push(top + bottom);
            }
            if (!strcmp(p, "-")){
                int top = stack.top();
                stack.pop();
                int bottom = stack.top();
                stack.pop();
                stack.push(bottom - top);
            }
            if (!strcmp(p, "/")){
                int top = stack.top();
                stack.pop();
                int bottom = stack.top();
                stack.pop();
                stack.push(bottom / top);
            }
            if (!strcmp(p, "*")){
                int top = stack.top();
                stack.pop();
                int bottom = stack.top();
                stack.pop();
                stack.push(bottom * top);
            }
            if (atoi(p)){
                stack.push(atoi(p));
            }
            p = strtok(NULL, " \n");
        }
        if (stack.size() != 1)
            throw 1;
        else
            cout << stack.top() << endl;
    }
    catch (...){
        cout << "error" << endl;
    }
    return 0;
}

```