**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема:** Динамические структуры данных

Студент гр. 7383 _____ Зуев Д. В.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2018

# Содержание

# 1. ЦЕЛЬ РАБОТЫ

Цель работы: ознакомиться с динамическими структурами данных и их реализацией.

Формулировка задачи:

На вход программе подается последовательность (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 (для операции в стеке не хватает аргументов или по завершении работы программы в стеке более одного элемента), то вместо результата следует вывести "error".

Стек требуется реализовать самостоятельно на базе массива.

# 2. РЕАЛИЗАЦИЯ

В файле func.c реализованы функции, необходимые для работы со стеком:

- count — возвращает количество элементов в стеке.
- initstack — создает пустой стек.
- pop — удаляет верхний элемент стека и возвращает его значение.
- push — добавляет элемент в стек.

В файле func.h объявлены вышеперечисленные функциию

В функции main.c считывается строка, разделяется на элементы. Если элемент — арифметическая операция, то она, если это возможно, выполняется над двумя верхними элементами стека. Если элемент — число, то оно добавляется в стек. При ошибке выводит «error», при правильной работе выводит результат операций.

Makefile – make-файл, отвечающий за сборку проекта.

Коды функций представлены в приложении Б.

# 3. ТЕСТИРОВАНИЕ

Использовался компилятор gcc. Программа запускалась на Ubuntu 16.04 LTS.

Тестовые случаи представлены в приложении А

По результатам тестирования выявлено, что поставленная задача успешно реализована.

# 4. ВЫВОД

В ходе данной работы были получены навыки работы с динамическими структурами данных. Был реализован стек на базе массива, разработан алгоритм обработки данных и выполнения над ними арифметических операций.

# ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ.

| Ввод | Вывод |
|---|---|
| - 2 4 + 5 4 | error |
| 57 23 / 5 * 10 - | 0 |
| 1 2 + 3 4 - 5 * + | -2 |
| 45 65 - 5 * 110 + | 10 |

# ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

**main.c:**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include"func.h"

int main(){
    char c[100];
    fgets(c, 100, stdin);
    stack *stck = initstack();
    char* a[100];
    int i = 0, k;
    a[i] = strtok(c, " \n\0");
    while(a[++i] = strtok(NULL, " \n\0"));
    for(int j = 0; j < i; j++)
    {
        if(strcmp(a[j],"+") == 0)
        {
            if(count(stck)<2)
            {
                printf("error");
                return 0;
            }
            push(stck, pop(stck)+pop(stck));
            continue;
        }
        if(strcmp(a[j],"*") == 0)
        {
            if(count(stck)<2)
            {
                printf("error");
                return 0;
            }
            push(stck, pop(stck)*pop(stck));
            continue;
        }
        if(strcmp(a[j],"-") == 0)
        {
            if(count(stck)<2)
            {
                printf("error");
```

```
                return 0;
            }
            k =pop(stck);
            push(stck, pop(stck) - k);
            continue;
        }
        if(strcmp(a[j],"/") == 0)
        {
            if(count(stck)<2)
            {
                printf("error");
                return 0;
            }
            k = pop(stck);
            push(stck, pop(stck)/k);
            continue;
        }
        push(stck,atoi(a[j]));
    }
    if(count(stck) != 1)
        printf("error");
    else
        printf("%d", pop(stck));
    return 0;
}
```

**func.c:**

```
#include<stdlib.h>
#include"func.h"

int count(stack* mass)
{
    return mass->size;
}

stack* initstack()
{
    stack* mass;
    mass->size =0;
    return mass;
}

int pop(stack* mass)
```

```
{
    mass->size--;
    return mass->i[mass->size];
}

void push(stack* mass, int c)
{
    mass->i[mass->size] = c;
    mass->size++;
}
```

**func.h:**

```
#pragma once

typedef struct stack{
    int i[100];
    int size;
}stack;

int count(stack* mass);
stack* initstack();
int pop(stack* mass);
void push(stack* mass, int c);
```

**Makefile:**

```
OBJ = func.o main.o

all: $(OBJ)
    gcc $(OBJ) -o main
%.o: %.c func.h
    gcc -c $<
clear:
    rm *.o main
```