

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

**ОТЧЕТ по лабораторной работе №3**  
**по дисциплине «Алгоритмы и структуры данных»**  
**Тема: «Стеки и очереди»**

Студент гр. 7381 \_\_\_\_\_

Аженилок В.А.

Преподаватель \_\_\_\_\_

Фирсов М. А.

Санкт-Петербург

2018

### **Цель работы.**

Ознакомиться с такими структурами данных, как стек и очередь, и научиться применять их на практике.

### **Основные теоретические положения.**

Стек – это структура данных, в которой хранятся элементы в виде последовательности, организованной по принципу LIFO (Last In – First Out).

Такую структуру данных можно сравнить со стопкой тарелок или магазином автомата. Стек не предполагает прямого доступа к элементам и список основных операций ограничивается операциями помещения элемента в стек и извлечения элемента из стека. Их принято называть PUSH и POP соответственно. Также, обычно есть возможность посмотреть на верхний элемент стека, не извлекая его (TOP), и несколько других функций, таких как проверка на пустоту стека и некоторые другие.

Очередь – эта структура данных, в которой хранятся элементы в виде последовательности, организованной по принципу FIFO (First In – First Out). Эта структура данных более естественна – например, очередь в магазине. Также, как и стек, очередь не предполагает прямого доступа к элементам, а основные операции: добавление ENQ (enqueue) и извлечение DEQ (dequeue). Также обычно есть функции получения первого элемента без его извлечения, определения размера очереди, проверки на пустоту и некоторые другие.

### **Задание.**

Вариант 7 - д:

В заданном текстовом файле F записана формула вида

$\langle \text{формула} \rangle ::= \langle \text{цифра} \rangle \mid M(\langle \text{формула} \rangle, \langle \text{формула} \rangle) \mid m(\langle \text{формула} \rangle, \langle \text{формула} \rangle)$

$\langle \text{цифра} \rangle ::= 0 \mid 1 \mid \dots \mid 9$

где  $M$  обозначает функцию  $\max$ , а  $m$  – функцию  $\min$ . Вычислить (как целое число) значение данной формулы. Например,  $M(5, m(6, 8)) = 6$ .

### Описание алгоритма.

В ходе выполнения алгоритма производится вычисление по формуле, которую вводит пользователь. Каждая введенная цифра помещается в стек, реализованный на базе списка. Обход ведется с конца строки для удобства работы программы. При встрече символа (буквы), соответствующего какой-либо математической операции, из стека вытаскиваются два элемента (цифры) и к ним применяется эта операция. Например, в данной работе могут быть операции вычисления минимальной и максимальной из двух цифр. Позже в стек заносится результат. Таким образом, по окончании прохода введенного выражения, в стеке должна остаться одна цифра, она и будет ответом на задачу.

### Функции и структуры данных.

В данной работе для поиска ответа на задачу был реализован стек на базе списка (динамической памяти) целых чисел. Головная функция считывает строку, содержащую условие. Потом начинается проход строки с конца (для удобства заполнения и разгрузки стека). Попутно выявляя ошибки условия, программа производит работу со стеком согласно алгоритма, описанного выше, после чего (при корректной записи условия) программа извлекает последний элемент стека и записывает его в ответ.

**struct Stack\_El** – структура, являющаяся элементом стека. Содержит поле с хранящимся числом и поле с указателем на следующий элемент стека. Если элемент находится в вершине стека, то это поле у него будет NULL.

**int main()** – головная функция программы, в которой происходит инициализация переменных, выделение и освобождение памяти, вызовы функций, необходимых

для построения и работы со стеком и проверки на возможные ошибки. **void push(int data, Stack\_El\* head, Stack\_El\* element)** – функция добавления элемента в стек.

**int pop(Stack\_El\* head)** – функция извлечения элемента из стека. Она возвращает значение элемента, который только что извлекла.

**int max(int x, int y)** – функция, вычисляющая максимум двух чисел. Вызывается при встрече символа “M”.

**int min(int x, int y)** – функция, вычисляющая минимум двух чисел. Вызывается при встрече символа “m”.

### Тестирование.

Таблица 1 - Примеры условий и ответов

Исходное выражение:	Результат:
M (5, m(6, 8))	6
M (m(4, 6), m(8, 0))	4
m (4, 0)	0
m ()	ERROR: There are 2 args should be in a function!
M (2, 1, 9)	ERROR: There are 2 args should be in a function!
m (2, M(0, 8))))))	ERROR: Counts of symbols '(' and ')' aren't equal!
m (3, U(8, 7))	ERROR: Unexpected symbol U!

### Вывод.

В ходе работы были получены необходимые теоретические знания по работе с динамическими структурами данных (стек и очередь), а также изучены различные варианты их реализации (на базе массива, или вектора, и на базе списка, или ссылочным методом).

### Приложение А. Код программы.

```
#include <stdio.h>
#include <stdlib.h>
```

```

#include <string.h>
#include <ctype.h>
#define N 1000

typedef struct Stack_El{
int data;
struct Stack_El* son;
} Stack_El;

int min(int x, int y) {
if(x <= y)
return x;
else
return y;
}

int max(int x, int y) {
if(x >= y)
return x;
else
return y;
}

void push(int data, Stack_El* head, Stack_El* element) {
element->data = data;
element->son = NULL;
Stack_El* tmp = head;
while(tmp->son)
tmp = tmp->son;
if(tmp != element) tmp->son = element;
printf("push: %d\n", data);
}

int pop(Stack_El* head) {
Stack_El* tmp = head;
Stack_El* tmp1 = NULL;
while(tmp->son){
tmp1 = tmp;
tmp = tmp->son;
}
int element = tmp->data;
if(tmp1)
tmp1->son = NULL;
printf("pop: %d\n", element);
return element;
}

int main() {
//Инициализация;
int stack_size = -1;

```

```

Stack_El* stack = (Stack_El*)malloc(sizeof(Stack_El));
char str[N];
printf("Please, enter the data line:\n");
fgets(str, N, stdin);
if(str[0] == '\n'){ //Поиск ошибки(пустая строка);
fprintf(stderr, "ERROR: You entered an empty line!\n");
free(stack);
return 0;
}
int i = strlen(str);
int count_of_brackets = 0;
printf("\n");

//Работа со стеком;
while(i >= 1){
i--;
if(str[i] == '('){
count_of_brackets++; //Поиск ошибки(название операции);
if(((i - 1) < 0) || (((i - 1) < 0) && ((str[i - 1] != 'm') || (str[i - 1] != 'M')))){
fprintf(stderr, "ERROR: Expected a name of operation before '('!\n");
free(stack);
return 0;
}
continue;
}
if(str[i] == ')'){
count_of_brackets--;
continue;
}
if((str[i] == ' ') || (str[i] == ',') || (str[i] == '\t') || (str[i] == '\n') || (str[i] == '\0'))
continue;

if(isdigit(str[i])){ //Пуш элемента в стек;
stack = (Stack_El*)realloc(stack, (stack_size + 2) * sizeof(Stack_El));
push((int)(str[i] - '0'), stack, &stack[++stack_size]);
continue;
}
if((str[i] == 'm') || (str[i] == 'M')){ //Вызов функции min или max;
if(stack_size < 1){ //Поиск ошибки(кол-во элементов);
fprintf(stderr, "mERROR: There are 2 args should be in a function!\n");
free(stack);
return 0;
}

int x = pop(stack);
stack_size--;
int y = pop(stack);
stack_size--;
}

```

```

if(str[i] == 'm'){
    stack = (Stack_El*)realloc(stack, (stack_size + 2) * sizeof(Stack_El));
    push(min(x, y), stack, &stack[++stack_size]);
}
if(str[i] == 'M'){
    stack = (Stack_El*)realloc(stack, (stack_size + 2) * sizeof(Stack_El));
    push(max(x, y), stack, &stack[++stack_size]);
}
continue;
}

fprintf(stderr, "ERROR: Unexpected symbol %c!\n", str[i]);
free(stack);
return 0;
}

//ошибка построения;
if(count_of_brackets != 0){
    fprintf(stderr, "ERROR: Counts of symbols '(' and ')' aren't equal!\n");
    free(stack);
    return 0;
}
if(stack_size != 0){ //Поиск ошибки(кол-во аргументов);
    fprintf(stderr, "ERROR: There are 2 args should be in a function!\n");
    free(stack);
    return 0;
}

//Результат;
printf("\nResult: %d\n", pop(stack));
free(stack);
return 0;
}

```