МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3 (вариант 9-в)

по дисциплине «Алгоритмы и структуры данных»

Тема: «Стеки и очереди»

Студент гр. 7381 _____ Адамов Я.В.

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 9-в.

В заданном текстовом файле F записан текст, сбалансированный по круглым скобкам:

< текст > ::= < пусто > | < элемент > < текст >

< элемент > ::= < символ > | (< текст >)

где < символ > – любой символ, кроме (,). Для каждой пары соответствующих открывающей и закрывающей скобок вывести номера их позиций в тексте, упорядочив пары в порядке возрастания номеров позиций:

а) закрывающих скобок; б) открывающих скобок.

Например, для текста A + (45 - F (X) * (B - C)) надо напечатать:

а) 8 10; 12 16; 3 17; б) 3 17; 8 10; 12 16.

Ход работы.

Программа написана на языке C.

Исходный файл: main.c

В начале работы происходит ввод строки, например: A+(45-F(X)*(B-C)), после чего строка передаётся функции error_checking(), которая проверяет её на валидность. В случае ошибок или отсутствия скобок в строке на экран выводится соответствующее сообщение, в ином случае строка передаётся функции output_brackets_indexes(), которая для каждой пары соответствующих открывающей и закрывающей скобок выводит номера их позиций в тексте, упорядочив пары в порядке возрастания закрывающих и открывающих скобок. Для работы используется стек.

Описание алгоритма работы функции output_brackets_indexes(): функция принимает указатель на строку. Поочередно проверяются все символы от первого до последнего. Если очередным символом является ‘(‘, то его индекс заносится в стек, если же встретился символ ‘)‘, то из стека извлекается значение и пара индексов выводится на экран. В конце работы на экране будут находится пары индексов, упорядоченные в порядке возрастания индексов закрывающих скобок. Этот же алгоритм используется функцией error_checking(): если очередным символом является ‘)‘, а стек пуст, значит для скобки отсутствует пара, если же в конце работы в стеке остались значения, значит для этих открывающих скобок отсутствуют пара. Для вывода пар, упорядоченных в порядке возрастания индексов открывающих

скобок используется более длинный алгоритм: основной цикл также идёт по всем символам строки. Если встречаемся открывающая скобка, то запускается второй цикл, который перебирает все символы из конца в начало. Индексы закрывающих скобок заносятся в стек, если же встречается открывающая скобка – значение из стека удаляется. В момент, когда второй цикл доходит до скобки, «вызвавшей» этот цикл, в стеке должно находится одно значение, которое и будет парным. Если стек к этому моменту пуст, значит открывающая скобка не имеет пары, если же в стеке больше одного значения, значит у закрывающих скобок нет пар.

Для демонстрации работы было написано несколько тестов, а также скрипт perform_tests.sh, который запускает все эти тесты. Полная демонстрация работы программы, а также её тестирование находится в Приложении А.

Описание функций и структур.

1) *struct Stack*

Структура стек. Для работы со стеком используются следующие функции: initStack, push, pop, top, size, isEmpty.

Поля структуры:

- **int arr[]**: массив, хранящий элементы стека.
- **int topIndex**: индекс, указывающий, куда долежен помещаться новый элемент.

2) *Stack initStack()*

Инициализация стека.

Возвращаемое значение: пустой стек.

3) *void push(Stack * stack, int c)*

Функция добавляет переданное значение в стек.

Параметры:

- **stack**: указатель на стек, в который добавляется элемент.

- **c**: значение, которое нужно поместить в стек.

4) *int pop(Stack * stack)*

Функция извлекает значение из стека.

Параметры:

- **stack**: указатель на стек, из которого извлекается значение.

Возвращаемое значение: верхний элемент стека.

5) *int top(Stack * stack)*

Функция возвращает верхний элемент стека.

Параметры:

- **stack**: указатель на стек, верхнее значение которого будет возвращено.

Возвращаемое значение: верхний элемент стека.

6) *int size(Stack * stack)*

Функция возвращает количество элементов стека.

Параметры:

- **stack**: указатель на стек, размер которого необходимо узнать.

Возвращаемое значение: размер стека.

7) *int isEmpty(Stack * stack)*

Функция проверяет стек на наличие элементов.

Параметры:

- **stack**: указатель на проверяемый стек.

Возвращаемое значение: 1 – если стек пуст, 0 – если стек содержит элементы.

8) *int error_checking(char * str)*

Функция принимает указатель на строку, которую необходимо проверить на сбалансированность по круглым скобкам.

Параметры:

- **str**: указатель на строку, которую необходимо проверить.

Возвращаемое значение: 1 – если присутствую ошибки, 0 – если строка сбалансирована.

9) *void output_brackets_indexes(char * str)*

Функция принимает указатель на строку, содержащую круглые скобки. Индексы пар скобок выводятся на экран.

Параметры:

- **str**: указатель на строку.

Тестирование программы.

Было создано несколько тестов для проверки работы программы. Помимо тестов, демонстрирующих работу алгоритма, были написаны тесты, содержащие некорректные данные, для демонстрации вывода сообщений об ошибках введенных данных (см. Приложение А).

Вывод.

В ходе выполнения работы была изучена новая структура данных: стек.

Приложение А. Тестирование.

Демонстрация работы программы:

input:

```
a+(45-f(x)*(b-c))
```

output:

```
Программа принимает строку и для каждой пары соответствующих открывающей
и закрывающей скобок выводит номера их позиций в строке.

Введите строку (не больше 500 символов): a+(45-f(x)*(b-c))

Сортировка по возрастания номеров позиций открывающих скобок.
Ход работы алгоритма:
  Символ №3 - '('. Поиск парной скобки:
    Символ №17 - ')', push 17
    Символ №16 - ')', push 16
    Символ №12 - '(', pop (16)
    Символ №10 - ')', push 10
    Символ №8 - '(', pop (10)
    Символ №3 - '(', pop (17), пара скобок: | 3  17 |
  Символ №8 - '('. Поиск парной скобки:
    Символ №17 - ')', push 17
    Символ №16 - ')', push 16
    Символ №12 - '(', pop (16)
    Символ №10 - ')', push 10
    Символ №8 - '(', pop (10), пара скобок: | 8  10 |
  Символ №12 - '('. Поиск парной скобки:
    Символ №17 - ')', push 17
    Символ №16 - ')', push 16
    Символ №12 - '(', pop (16), пара скобок: | 12  16 |
Результат:
| 3  17 | 8  10 | 12  16 |

Сортировка по возрастания номеров позиций закрывающих скобок.
Ход работы алгоритма:
```

```
  Символ №3 - '(', push 3
  Символ №8 - '(', push 8
  Символ №10 - ')', pop (8), пара скобок: | 8  10 |
  Символ №12 - '(', push 12
  Символ №16 - ')', pop (12), пара скобок: | 12  16 |
  Символ №17 - ')', pop (3), пара скобок: | 3  17 |
Результат:
| 8  10 | 12  16 | 3  17 |
```

Тестирование:

Входные данные	Выходные данные
a+(45-f(x)*(b-c))	| 3 17 | 8 10 | 12 16 | | 8 10 | 12 16 | 3 17 |
aaa((c(()b)d)e)aaaa	| 4 15 | 5 13 | 7 11 | 8 9 | | 8 9 | 7 11 | 5 13 | 4 15 |
a+c*(13*(10*(12+4)*(8-3)+1)*66+13/(1+1))	| 5 40 | 9 27 | 13 18 | 20 24 | 35 39 | | 13 18 | 20 24 | 9 27 | 35 39 | 5 40 |
(((()()))()((()))(()(()))	| 1 8 | 2 7 | 3 4 | 5 6 | 9 10 | 11 16 | 12 15 | 13 14 | 17 24 | 18 19 | 20 23 | 21 22 | | 3 4 | 5 6 | 2 7 | 1 8 | 9 10 | 13 14 | 12 15 | 11 16 | 18 19 | 21 22 | 20 23 | 17 24 |
a+b-c*3+d/5	В строке отсутствуют скобки.
()()()((()))()()()	| 1 2 | 3 4 | 5 6 | 7 12 | 8 11 | 9 10 | 13 14 | 15 16 | 17 18 | | 1 2 | 3 4 | 5 6 | 9 10 | 8 11 | 7 12 | 13 14 | 15 16 | 17 18 |
((a+b)*c)+(a*((b+c)/13)	Ошибка: символ №11 - '(', парная закрывающая скобка отсутствует.
(a+b)*(a+e/(2+g)+8))	Ошибка: символ №20 - ')', парная открывающая скобка отсутствует.

Приложение Б. Код программы.

```c
#include <stdio.h>
#include <string.h>


#define STACKSIZE 100
#define N 501


// стек
typedef struct Stack{
    int arr[STACKSIZE];
    int topIndex;
} Stack;


// инициализация стека
Stack initStack(){
    Stack stack;
    stack.topIndex=0;
    return stack;
}


// проверка на наличие элементов в стеке
int isEmpty(Stack * stack){
    return !stack->topIndex;
}


// размер стека
int size(Stack * stack){
    return stack->topIndex;
}


// добавление элемента в стек
void push(Stack * stack, int c){
    if (size(stack) < STACKSIZE){
        stack->arr[stack->topIndex] = c;
        stack->topIndex++;
    } else {
```

```
        printf("Ошибка: стек заполнен. Новый элемент не был добавлен.");
    }
}


// извлечение элемента из стека
int pop(Stack * stack){
    if (!isEmpty(stack)){
        stack->topIndex--;
        return stack->arr[stack->topIndex];
    } else {
        printf("Ошибка: стек пуст.");
        return 0;
    }
}


// верхний элемент стека
int top(Stack * stack){
    if (!isEmpty(stack)){
        return stack->arr[stack->topIndex - 1];
    } else {
        printf("Ошибка: стек пуст.");
        return 0;
    }
}




// проверка строки на валидность
// возвращает 0, если ошибок нет
int error_checking(char * str){
    Stack stack = initStack(); // стек
    int presence_of_brackets = 0; // наличие скобок
    for (int i = 0; i < strlen(str); i++){
        if (str[i] == '('){
            presence_of_brackets = 1;
            if (size(&stack) == STACKSIZE){
                printf("\nОшибка: стек переполнен.\n");
                printf("Вы используете слишком большое количество скобок.\n");
                return 1;
            }
            push(&stack, i+1);
        }
```

```
        if (str[i] == ')'){
            if (size(&stack) == 0){
                printf("\nОшибка: символ №%d - ')', парная открывающая скобка отсутствует.\n",
i+1);
                return 1;
            }
            pop(&stack);
        }
    }
    if (!isEmpty(&stack)){
        printf("\nОшибка: символ №%d - '(', парная закрывающая скобка отсутствует.",
pop(&stack));
        return 1;
    }
    if (presence_of_brackets == 0){
        printf("\nВ строке отсутствуют скобки.");
        return 1;
    }

    return 0;
}


// вывод индексов пар скобок
void output_brackets_indexes(char * str){
    Stack stack = initStack(); // стек

    printf("\nСортировка по возрастания номеров позиций открывающих скобок.\n");
    // демонстрация работы алгоритма
    printf("Ход работы алгоритма:\n");
    for (int i = 0; i < strlen(str); i++){
        if (str[i] == '('){
            printf("  Символ №%d - '('. Поиск парной скобки:\n", i+1);
            for (int j = strlen(str)-1; j >= 0; j--){
                if (str[j] == ')'){
                    printf("    Символ №%d - ')', push %d\n", j+1, j+1);
                    push(&stack, j+1);
                }
                if (str[j] == '('){
                    if (j > i){
                        printf("    Символ №%d - '(', pop (%d)\n", j+1, pop(&stack));
                    } else{
                        printf("    Символ №%d - '(', pop (%d), пара скобок: | %d  %d |\n",
j+1, pop(&stack), j+1, top(&stack));
                        //printf(" %d  %d |", i+1, pop(&stack));
```

```
                    break;
                }
            }
        }
    }
    // вывод результата
    printf("Результат:\n|");
    for (int i = 0; i < strlen(str); i++){
        if (str[i] == '('){
            for (int j = strlen(str)-1; j >= 0; j--){
                if (str[j] == ')')
                    push(&stack, j+1);
                if (str[j] == '('){
                    if (j > i){
                        pop(&stack);
                    } else{
                        printf(" %d  %d |", i+1, pop(&stack));
                        break;
                    }
                }
            }
        }
    }


    printf("\n\nСортировка по возрастания номеров позиций закрывающих скобок.\n");
    // демонстрация работы алгоритма
    printf("Ход работы алгоритма:\n");
    for (int i = 0; i < strlen(str); i++){
        if (str[i] == '('){
            printf("  Символ №%d - '(', push %d\n", i+1, i+1);
            push(&stack, i+1);
        }
        if (str[i] == ')'){
            printf("  Символ №%d - ')', pop (%d), пара скобок: | %d  %d |\n", i+1, pop(&stack),
top(&stack), i+1);
        }
    }
    // вывод результата
    printf("Результат:\n|");
    for (int i = 0; i < strlen(str); i++){
        if (str[i] == '(')
            push(&stack, i+1);
        if (str[i] == ')'){
```

```
            printf(" %d  %d |", pop(&stack), i+1);
        }
    }
    printf("\n\n");
}



int main(void)
{
    char str[N]; // строка

    printf("\nПрограмма принимает строку и для каждой пары соответствующих открывающей\n");
    printf("и закрывающей скобок выводит номера их позиций в строке.\n");
    printf("\nВведите строку (не больше %d символов):\n", N-1);
    fgets(str, N, stdin);

    if (error_checking(str) == 0)
        output_brackets_indexes(str);

    return 0;
}
```