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

Кафедра «МОЭВМ»

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 7381 _____ Дорох С. В.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2018

Задание:

Требуется построить синтаксический анализатор для понятия *простое выражение.*

простое_выражение::=простой_идентификатор |

(*простое_выражение знак_операции простое_выражение*)

простой_идентификатор::= буква

знак_операции:: = – | + | *

Пояснение задания:

На вход подаётся строка. Требуется определить удовлетворяет ли она понятию «Простое выражение». Необходимо реализовать рекурсивную функцию проверки.

Описание алгоритма:

Входные данные поступают в программу, где вызывается рекурсивная функция для проверки поступивших данных.

Описание функции:

Функция `is_Valid(char* buffer, int depth).` Данная функция анализирует исходное выражение при помощи рекурсии, разбивая исходную строку на подстроки, пока это возможно, и проверяет их, выводя ответ, который зависит от вида выражения. Также данная функция сообщает о состоянии программы и ошибках, которые могут возникнуть в процессе анализа данных.

Функция `is_depth(int iter)`. Данная функция определяет глубину рекурсии и в соответствии с ней создаёт необходимое количество отступов.

Тестирование программы:

Тестирование данной программы состоит в том, чтобы ее проверить работоспособность путем запуска со всеми верными входными данными, а также убедится в корректном выводе при неверных введенных данных. В качестве примера тестировании данных рассмотрим пример “((a+b)*(m-k)”:

```
1 function call is_Valid for "((a+b)*(m-k))"
            2 function call is_Valid for "(a+b)"
                  3 function call is_Valid for "a"
```

```
                The end of the 3 call is_Valid
                3 function call is_Valid for "b"
                The end of the 3 call is_Valid
          The end of the 2 call is_Valid
          2 function call is_Valid for "(m-k)"
                3 function call is_Valid for "m"
                The end of the 3 call is_Valid
                3 function call is_Valid for "k"
                The end of the 3 call is_Valid
          The end of the 2 call is_Valid
     The end of the 1 call is_Valid
This expression is correct!
```

Вывод.

В ходе выполнения лабораторной работы получены знания по теме «рекурсия» и закреплены знания синтаксиса языка C, а также получены навыки по написанию bash-скрипта.

Приложение А. Код программы main.c:

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

void is_depth(int iter){
    for (int i = 0; i < iter; i++){
        printf("\t");
    }
}
int is_Valid(char *buffer, int depth) {
    depth++;
    is_depth(depth);
    printf("%d function call is_Valid for \"%s\"\n", depth, buffer);

    if (strlen(buffer) == 1){
        is_depth(depth);
        printf("The end of the %d call is_Valid\n", depth);
        return isalpha(buffer[0]);
    }
    else if (!strlen(buffer)){
        is_depth(depth);
        printf("The end of the %d call is_Valid\n", depth);
        return 0;
    }

    char *first_brackets;
    char *second_brackets;

    int bracket_count = 0;
    int i = 1;
    do {
        if (buffer[i] == '(')
            bracket_count++;
        if (buffer[i] == ')')
            bracket_count--;
        i++;
        if(buffer[i] == '\0' && bracket_count != 0){
            is_depth(depth);
            printf("\tDifferent number of brackets!\n");
            is_depth(depth);
            printf("The end of the %d call is_Valid.\n", depth);
            return 0;
        }
    } while (bracket_count != 0 && i <= strlen(buffer));

    if(buffer[i] != '+' && buffer[i] != '-' && buffer[i] != '*'){
        is_depth(depth);
        printf("\t%d element is an invalid character!\n", i);
```

```
        is_depth(depth);
        printf("The end of the %d call is_Valid\n", depth);
        return 0;
    }

    buffer[strlen(buffer)-1] = '\0';
    first_brackets = buffer + 1;
    second_brackets = buffer + i + 1;
    buffer[i] = '\0';

    if (is_Valid(first_brackets, depth) && is_Valid(second_brackets, depth)) {
        is_depth(depth);
        printf("The end of the %d call is_Valid\n", depth);
        return 1;
    }
    return 0;

}

int main(){
    char *buffer=malloc(sizeof(char)*50);
    fgets(buffer, 50, stdin);
    int depth = 0;
    if (buffer[strlen(buffer) - 1] == '\n')
        buffer[strlen(buffer) - 1] = '\0';
    printf(is_Valid(buffer, depth) ? "This expression is correct!\n" : "This
expression is incorrect!\n");
    free(buffer);
}
```