

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

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Алгоритмы и структуры данных»**  
**Тема: Рекурсия**

Студентка гр. 9381

\_\_\_\_\_

Москаленко Е.М.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

## Цель работы.

Ознакомиться с понятием рекурсивной функции и взаимно-рекурсивных функций, построить синтаксический анализатор на языке программирования С.

## Задание.

Вариант 10.

Построить синтаксический анализатор для определяемого далее понятия *константное\_выражение*.

*константное\_выражение::=ряд\_цифр|*

*константное\_выражение знак\_операции константное\_выражение*

*знак\_операции::=+ | - | \**

*ряд\_цифр::=цифра | цифра ряд\_цифр*

## Функции.

Используемые функции:

- 1) **int checkingSign(char sign);** - функция на проверку знака операции (+, - или \*).

char sign – проверяемый знак

Возвращает 1, если встречен знак +, - или \*, 0 – в противном случае.

- 2) **int numbersRow(char\* numbers, int\* start, int tab);** - функция проверки ряда цифра (numbers - выражение, start - индекс элемента, tab - значение табуляции)

Возвращает 1, если встречается крайнюю цифру и 0, если встречается НЕ цифру. Иначе рекурсивно вызывает саму себя.

- 3) **void constMessage(char\* message, int\* start);** - функция проверки константного выражения (message - выражение, start - индекс элемента)

**constMessage** завершается в зависимости от значения **numbersRow** и выводит информацию. Если цифра крайняя и больше знаков нет, то это константное выражение. Если встретила НЕ цифра, то это не

константное выражение. Если встретился знак операции, то **constMessage** рекурсивно вызывает саму себя.

**4) char\* readMessage();** - функция динамического считывания строки  
Возвращает считанную строку.

#### **Описание алгоритма.**

Пользователь может ввести данные с консоли или из файла. Чтобы считать данные из файла, необходимо ввести полный путь до файла. Для считывания строки данных используется функция **char\* readMessage()**. В функции строка считывается динамически с помощью функций **malloc** и **realloc**, а последним символом записывается **'\0'**.

Затем создается переменная **int s = 0**, а строка с данными и указатель на **s** передаются в рекурсивную функцию **constMessage**.

В **constMessage** инициализируется переменная **tab**, изначально равная 0. Эта переменная отвечает за отступы строк при промежуточном выводе.

В этой функции рекурсивно вызывается функция **numbersRow(char\* numbers, int\* start, int tab)**, в которую передается строка, указатель на индекс рассматриваемого символа и значение отступа.

Функция **numbersRow** предназначена для проверки части строки на цифру или ряд цифр. Сначала для удобства вывода на консоль выводятся отступы, количество которых равно **tab**. Затем происходит проверка символа на цифру. Если символ – не цифра, то об этом выводится информация и возвращаемое значение равно 0. Если символ – крайняя цифра или за ней следует знак операции, то индекс рассматриваемого символа увеличивается на 1 и возвращается значение 1. Если знак – цифра, и после нее нет знака операции и **/0**, то индекс символа увеличивается на 1 и функция **numbersRow** рекурсивно вызывает саму себя со значением **tab**, увеличенным на 1.

Как только функция **numbersRow** вернет целочисленное значение, то **constMessage** выведет сообщение, константное это выражение или нет. Если же

встретился знак операции, то индекс символа увеличивается на 1 и проверка на выражение происходит дальше с рекурсивным вызовом `constMessage`.

Функция проверки на символ – `checkingSign(char sign)`. Возвращает значение 1, если символ равен \*, - или +. И 0 в обратном случае.

### Тестирование.

Тестирование программы представлено в таблице 1.

Таблица 1.

Номер теста	Входные данные	Вывод
1	12+78-1	<p>Вошли в функцию <code>constMessage</code></p> <p>Вошли в функцию <code>numbersRow</code></p> <p>Цифра ряд цифр 1</p> <p>Вошли в функцию <code>numbersRow</code></p> <p>Цифра 2</p> <p>Конец функции <code>numbersRow</code></p> <p>Знак операции +</p> <p>Вошли в функцию <code>constMessage</code></p> <p>Вошли в функцию <code>numbersRow</code></p> <p>Цифра ряд цифр 7</p> <p>Вошли в функцию <code>numbersRow</code></p> <p>Цифра 8</p> <p>Конец функции <code>numbersRow</code></p> <p>Знак операции -</p> <p>Вошли в функцию <code>constMessage</code></p> <p>Вошли в функцию <code>numbersRow</code></p> <p>Цифра 1</p>

		Конец функции numbersRow Выход из функции constMessage Константное выражение
2	56-9	Вошли в функцию constMessage Вошли в функцию numbersRow Цифра ряд цифр 5 Вошли        в        функцию numbersRow Цифра 6 Конец функции numbersRow Знак операции - Вошли в функцию constMessage Вошли в функцию numbersRow Цифра 9 Конец функции numbersRow Выход из функции constMessage Константное выражение
3	234+87)1	Не константное выражение
4	6-88*9+15	
5	/Users/elizaveta/test1.c в файле: 123_4)	Не константное выражение
6	15+5a	Не константное выражение

### **Выводы.**

Была написана программа синтаксического анализатора для определения понятия «константное\_выражение». В алгоритме использованы две взаимно-рекурсивные функции отдельно для понятий «константное\_выражение» и

«ряд\_цифр». Освоены принципы работы рекурсивных функций и их написание на языке программирования C.

## ПРИЛОЖЕНИЕ А

### Файл main.c

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

int checkingSign(char sign);
int numbersRow(char* numbers, int* start, int tab);
void constMessage(char* message, int* start);
char* readMessage();

int checkingSign(char sign){
    if (sign == '+' || sign == '-' || sign == '*')           //функция на
    проверку знака операции (+, - или *)
        return 1;
    else return 0;
}

int numbersRow(char* numbers, int* start, int tab){          printf("Вошли в
функцию numbersRow\n");
    for(int i = 0; i < tab; i++)
        printf("\t");

    if (!isdigit(numbers[*start])) {                         //если знак - не
    цифра, то возвращается 0.
        printf("Не цифра %c\n", numbers[*start]);
        return 0;
    }
    if (isdigit(numbers[*start]) && (numbers[(*start)+1] == '\0' || /*
    если цифра крайняя(перед знаком операции или /0),
    то
    возвращается 1. */
        checkingSign(numbers[(*start)+1]))) {
        printf("Цифра %c\n", numbers[*start]);
        (*start)++;
        return 1;
    }

    if (isdigit(numbers[*start]) && (numbers[(*start)+1] != '\0') &&
    //если цифра не крайняя, то далее рекурсивный вызов
        (!checkingSign(numbers[(*start)+1]))) {
```

```

        printf("Цифра ряд цифр %c\n", numbers[*start]);
        (*start)++;
        return numbersRow(numbers, start, tab+1);
    }
    return 0;
}

void constMessage(char* message, int* start){ //функция проверки
константного выражения
    printf("Вошли в функцию constMessage\n");
    int tab = 0;
    if (!numbersRow(message, start, tab)){
        printf("\033[1;31m" "Не константное выражение\n"); //вызов
NumbersRow. если встретилась НЕ цифра, то выводим сообщение и выходим
        return;
    }
    else{
        if (message[*start] == '\0') { //если
дальше знаков нет, то это const. выходим
            printf("\033[1;31m" "Константное выражение\n");
            return;
        }
        if (checkingSign(message[*start])){ //если
встретился знак операции, то выводим сообщение и сдвигаемся на знак далее
            printf("\033[1;34m" "Знак операции" %c\n "\033[0m"
, message[*start]);
            (*start)++;
            return constMessage(message, start);
        }
    }
}

```

```

char* readMessage(){ //функция динамического считывания строки
    char* n = malloc(10);
    int size = 0;
    char c = fgetc(stdin);
    while (1){
        c = fgetc(stdin);
        if (c == '\n')
            break;
        n = (char*)realloc(n, size+1);
        n[size++] = c;
    }
    n[size] = '\0';
    return n;
}

```

```

int main(){
    char choiceRead;
    printf("Нажмите А при вводе данных с клавиатуры, \n В - при считывания
с файла\n");
    scanf("%c", &choiceRead);
    char* n;
    switch(choiceRead) {

        case 'a':
            printf("Введите строку для тестирования\n");
            n = readMessage();
            break;

        case 'b':

```

```

        printf("Введите полный путь файла с данными\n");
        char *name = readMessage();
//при данном выборе считываются данные с файла и далее происходит работа с ними
        FILE *fileTask = fopen(name, "r");
        if (!fileTask){
            printf("Ошибка при чтении файла. Вероятно, файла с таким
названием нет.\n");
            return 0;
        }
        n = malloc(500);
        char c;
        int i;
        for(i = 0; (c = getc(fileTask)) != '\n' && (c != EOF); ++i) {
            n[i] = c; // заполнение строки
        }
        if(c == EOF){
            puts("\ndостигнут конец файла, выход.");
        }
        n[i] = '\0';
        if (fclose(fileTask) == EOF) printf ("ошибка\n");
        else printf ("Считывание выполнено\n");
        n = realloc(n, strlen(n)+1);
        free(name);
        break;

    default:
        printf("Такого варианта нет. Вы должны ввести А или В.\n");
        return 0;
    }
    int s = 0;
    int* z = &s;
    constMessage(n, z); //вызов функции
    free(n);
    return 0;
}

```