

**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: Рекурсивные алгоритмы**

Студент гр. 7383

\_\_\_\_\_

Лосев М. Л.

Преподаватель

\_\_\_\_\_

Размочаева Н.

Санкт-Петербург

2018

## СОДЕРЖАНИЕ

|                           |    |
|---------------------------|----|
| 1. ЦЕЛЬ РАБОТЫ.....       | 3  |
| 2. РЕАЛИЗАЦИЯ ЗАДАЧИ..... | 4  |
| 3. ТЕСТИРОВАНИЕ.....      | 6  |
| 4 . ВЫВОД.....            | 6  |
| ПРИЛОЖЕНИЕ А.....         | 7  |
| ПРИЛОЖЕНИЕ Б.....         | 15 |

## **1. ЦЕЛЬ РАБОТЫ**

Цель работы: ознакомление с основными понятиями и приемами рекурсивного программирования, получение навыков программирования рекурсивных процедур и функций на языке программирования C++

Формулировка задачи: Построить синтаксический анализатор для понятия скобки. скобки::=квадратные | круглые | фигурные квадратные::=[круглые фигурные] | + круглые::=(фигурные квадратные) | } фигурные::={квадратные круглые} | 0

## 2. РЕАЛИЗАЦИЯ ЗАДАЧИ

### *Используемые функции:*

***bool Bracket(char \*buf);*** – функция, получающая на вход строку (выражение) и возвращающая true, если выражение является скобками, или false, если не является

***bool Round (char \*\*buf, char s);*** – функция, получающая на вход строку (выражение), и возвращающая true, если выражение является круглыми скобками, или false, если не является.

***bool Square (char \*\*buf, char s );*** – функция, получающая на вход строку (выражение), и возвращающая true, если выражение является квадратными скобками, или false, если не является.

***bool Figure (char \*\*buf, char s );*** – функция, получающая на вход строку (выражение), и возвращающая true, если выражение является фигурными скобками, или false, если не является.

***void Error (short k);*** – функция, получающая на вход номер ошибки и выводящая соответствующее сообщение об ошибке

Описание алгоритма:

***void Load ( char \*filename, char \*buf );*** – функция, получающая на вход имя файла и строку, и загружающая из этого файла выражение в строку.

***void UserInterface ();*** – функция, выводящая пользовательское меню и обрабатывающая ввод (выбор опций меню).

### *Описание алгоритма функции Bracket:*

Функция Bracket проверяет первый символ полученной строки: если он является открывающей скобкой или +, - или 0, то она вызывает функцию, проверяющую, является ли выражение соответствующими скобками. Потом она проверяет, не осталось ли в строке лишних символов, и возвращает true, если выражение является скобками, или false, если не является. Например, если входная строка - «(0+)», то будет вызвана функция Round (потому что

первый символ „(“ соответствует круглым скобкам), которая вернет true, потом будет проверено, что в строке нет лишних символов, и, так как их не осталось, будет возвращено true.

#### ***Описание алгоритма функции Round:***

Функция Round проверяет первый символ полученной строки: если он „-“ то она вернет true (так как „-“ по определению является круглыми скобками). Если он „(“ , то она проверит, идут ли дальше фигурные скобки, вызывая Figure: если не идут, то Round тоже вернет false (т. к. внутри круглых скобок должны быть сначала фигурные, а потом квадратные), а если идут, то Round проверит, идут ли после фигурных скобок квадратные, вызывая Square: если не идут, то Round тоже вернет false (т. к. внутри круглых скобок после фигурных должны быть квадратные), а если идут, то Round проверит, идет ли дальше „)“: если да, то она вернет true, а если нет, то вызовет Error (которая выведет соответствующее сообщение), и вернет false. Если первый символ не „-“ и не „(“, то вызовется Error (которая выведет соответствующее сообщение). Если строка кончится раньше, чем встретится „)“, то функция вызовет Error (которая выведет соответствующее сообщение) и вернет false.

Например, если Round получит на вход строку «(0+)», то она проверит первый символ: он окажется „(“. Поэтому она проверит, идут ли дальше фигурные скобки с помощью функции Figure. Figure вернет true (потому что 0 – это фигурные скобки по определению). Поэтому дальше Round проверит, идут ли после фигурных скобок квадратные с помощью функции Square. Square вернет true (потому что „+“ – это квадратные скобки по определению). Поэтому дальше Round проверит, идет ли после квадратных скобок закрывающаяся круглая скобка, и вернет true, потому что закрывающаяся круглая действительно идет после „+“.

#### ***Описание алгоритма функций Square и Figure:***

Функции Square и Figure полностью аналогичны функции Round.

#### ***Описание алгоритма функции Error:***

Функция `Error` принимает целое число — код ошибки. Она выводит соответствующее сообщение об ошибке и не возвращает ничего.

### **3. ТЕСТИРОВАНИЕ**

#### **3.1 ПРОЦЕСС ТЕСТИРОВАНИЯ**

Программа собрана в операционной системе Ubuntu 17.04 компилятором `g++`. В других ОС тестирование не проводилось.

#### **3.1 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ**

Тестовые случаи в приложении Б. По результатам тестирования было показано, что задача выполнена.

### **4 . ВЫВОД**

В ходе работы была написана программа на языке C++, решающая поставленную задачу с помощью сложной рекурсии. Был получен опыт в написании такого рода программ.

## ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <iomanip>
#include <unistd.h>

using namespace std;

bool Bracket(char *buf);
bool Round (char **buf, char s);
bool Square (char **buf, char s );
bool Figure (char **buf, char s );
void Error (short k);
void Load ( char *filename, char *buf );
void UserInterface ();

int main ( )
{
    UserInterface();
    return 0;
}

bool Square (char **buf, char s)
// квадратные:=[круглые фигурные] | +
// s — текущий символ входной строки
{
    bool k;
    if (s == '+') { return true;}
    else if ( s == '[' )
    {
        if ((*buf)++)
        {
            cout << **buf;
            k = Round (buf, **buf);
            if (k)
            {
                if ((*buf)++)
                {
                    cout << **buf;
                    k = Figure (buf, **buf);}
                else {Error (5); return false;} // квадрат - пуст!
            }
            else return false; //первый квадрат ошибочен
            if (k) // оба квадрат правильны
            {
                if ((*buf)++)
                {
                    if (**buf != ']') {Error(3 ); return false;}
                    cout << **buf;
                    return true;
                }
                else {Error (3); return false;}
            }
            else return false;
        }
        else { Error (5); return false;} // квадрат — пуст!
    }
    else { Error(4); return false;} // не + и не [
}
// end of Square

bool Round (char **buf, char s)
// круглые:=(фигурные квадратные) | -
// s — текущий символ входной строки
```

```

{   bool k;
    if (s == '-') { return true;}
    else if ( s == '(' )
        {   if ((*buf)++)
                {   cout << **buf;
                    k = Figure (buf, **buf);
                    if (k)
                        {   if ((*buf)++)
                                {   cout << **buf;
                                    k = Square (buf, **buf);}
                                else {Error (8); return false;} // кругл - пуст!
                            }
                        else return false; //первый квадрант ошибочен
                    if (k) // оба круга правильны
                        if ((*buf)++)
                            {   if (**buf != ')') {Error(6); return false;}
                                cout << **buf;
                                return true;
                            }
                        else {Error (6); return false;}
                    else return false;
                }
            else { Error (8); return false;} // кругл — пуст!
        }
    else { Error(7); return false;} // не - и не (
}
// end of Round

```

```

bool Figure (char **buf, char s)
// фигурные:={квадратные круглые} | 0
// s — текущий символ входной строки
{   bool k;
    if (s == '0') { return true;}
    else if ( s == '{' )
        {   if ((*buf)++)
                {   cout << **buf;
                    k = Square (buf, **buf);
                    if (k)
                        {   if ((*buf)++)
                                {   cout << **buf;
                                    k = Round (buf, **buf);}
                                else {Error (11); return false;} // фиг - пуст!
                            }
                        else return false; //первый квадрант ошибочен
                    if (k) // оба квадранта правильны
                        if ((*buf)++)
                            {   if (**buf != '}') {Error(9); return false;}
                                cout << **buf;
                                return true;
                            }
                        else {Error (9); return false;}
                    else return false;
                }
            else { Error (11); return false;} // фиг — пуст!
        }
    else { Error(10); return false;} // не - и не (
}
// end of Square

```

```

bool Bracket(char *buf)

```



```

{   char s;
    bool b;
    b = false;
    s = *(buf);
    if (s)
    {   cout << s;
        if ((s == '+') || (s == '[')) b = Square (&buf, s);
        else if ((s == '-') || (s == '(')) b = Round (&buf, s);
            else if ((s == '0') || (s == '{')) b = Figure (&buf, s);
                else Error(2); //недопустимый начальный символ
        s = ((*buf)++);
        if (b && !(*buf)) Error(1); // лишние символы
        b = (b && *buf);
    }
    else Error (0); // пустая входная строка
    return b;
}

void Error (short k)
{
    cout << endl << "err#" << k << endl;
    switch (k) {
        case 0: cout << "! - Пустая входная строка" << endl; break;
            //{Bracket}
        case 1: cout << "! - Лишние символы во входной строке" << endl; break;
            //{Bracket}
        case 2: cout << "! - Недопустимый начальный символ" << endl; break;
            //{Bracket}
        case 3: cout << "! - Отсутствует ']'." << endl; break;
            //{Square}
        case 4: cout << "! - Отсутствует '+' или '['." << endl; break;
            //{Square}
        case 5: cout << "! - Очередной квад — пуст." << endl; break;
            //{Round}
        case 6: cout << "! - Отсутствует ')'." << endl; break;
            //{Round}
        case 7: cout << "! - Отсутствует — или (." << endl; break;
            //{Round}
        case 8: cout << "! - Очередной кругл — пуст." << endl; break;
            //{Round}
        case 9: cout << "! - Отсутствует '}'." << endl; break;
            //{Figure}
        case 10: cout << "! - Отсутствует 0 или {." << endl; break;
            //{Figure}
        case 11: cout << "! - Очередной фиг — пуст." << endl; break;
            //{Figure}
        default : cout << "! - ...";break;
            // ?
    };
}
// end of Error

void Load ( char *filename, char *buf )
{
    ifstream infile (filename);
    if (!infile) cout << "Входной файл не открыт" << endl;
    else {
        cout << "Входной файл открыт" << endl;
        int i = 0;
        char s;
        while (infile >> s) {

```

```

        buf[i++] = s;
    }
}

void UserInterface ()
{
    int key;
    char exp[100] = {0};
    while (true) {
        cout << "Выбери действие:" << endl;
        cout << "0. Завершить выполнение;" << endl;
        cout << "1. Загрузить выражение из пользовательского файла;" << endl;
        cout << "2. Загрузить выражение из файла по умолчанию;" << endl;
        cout << "3. Ввести выражение с клавиатуры;" << endl;
        cout << "4. Анализировать выражение;" << endl;
        cin >> key;
        switch (key) {
            case 0:
                return;
                break;
            case 1:
            {
                cout << "input file name: ";
                char filename[20];
                cin >> filename;
                Load(filename, exp);
            }
            break;
            case 2:
            {
                Load("in_seq5.txt", exp);
            }
            break;
            case 3:
            {
                cout << "input expression: ";
                cin >> exp;
            }
            break;
            case 4:
            {
                cout << "Анализатор скобок:" << endl;
                bool b = Bracket (exp);
                if (b) cout << endl << "ЭТО СКОБКИ!" << endl;
                else cout << "НЕТ, ЭТО НЕ СКОБКИ!" << endl;
            }
            break;
            //

            default : cout << "! - ...";
            break;
            // ?
        };
    }
}

```

## ПРИЛОЖЕНИЕ Б. ТЕСТОВЫЕ СЛУЧАИ

| Входное выражение      | Вывод программы  | Корректность выполнения |
|------------------------|--|-------------------------|
| (                      | Анализатор скобок:<br>(<br>err#10<br>! - Отсутствует 0 или {.<br>НЕТ, ЭТО НЕ СКОБКИ!         | Да                      |
| (0                     | Анализатор скобок:<br>(0<br>err#4<br>! - Отсутствует '+' или '['.<br>НЕТ, ЭТО НЕ СКОБКИ!     | Да                      |
| (d                     | Анализатор скобок:<br>(d<br>err#10<br>! - Отсутствует 0 или {.<br>НЕТ, ЭТО НЕ СКОБКИ!        | Да                      |
| abracadabre            | Анализатор скобок:<br>a<br>err#2<br>! - Недопустимый начальный символ<br>НЕТ, ЭТО НЕ СКОБКИ! | Да                      |
| (0+)                   | Анализатор скобок:<br>(0+)<br>ЭТО СКОБКИ!  | Да                      |
| (0asd                  | Анализатор скобок:<br>(0a<br>err#4<br>! - Отсутствует '+' или '['.<br>НЕТ, ЭТО НЕ СКОБКИ!    | Да                      |
| {[(0[(0+)0)]{+-}](0+)} | Анализатор скобок:<br>{[(0[(0+)0)]{+-}](0+)}   | Да                      |

|         |  |    |
|---------|--|----|
|         | ЭТО СКОБКИ!  |    |
| (0[12]) | Анализатор скобок:<br>(0[1<br>егг#7<br>! - Отсутствует – или (.<br>НЕТ, ЭТО НЕ СКОБКИ! | Да |