

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

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

Студент гр.8304

\_\_\_\_\_

Воропаев А.О.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2019

## **Задание.**

Вариант №21

Построить синтаксический анализатор для понятия скобки.

скобки::=квадратные | круглые

квадратные::= [ [ квадратные ] ( круглые ) ] | В

круглые::=( ( круглые ) [ квадратные ] ) | А

## **Цель работы.**

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных функций на языке программирования C/C++.

## **Описание алгоритма.**

Сначала пользователю предлагается выбрать способ ввода, где 1 – ввод из файла, 2 – из консоли. После ввода строка передается функции check, которая проверяет введенную строку на соответствие условию. Из ф-ции check вызывается рекурсивная функция collapse, которая отвечает за «сворачивание» исходной строки для последующей проверки на соответствие поставленной задаче. Функция collapse принимает два аргумента: ссылку на исходную строку(source), и счетчик глубины рекурсии(n). С помощью метода find в исходной строке производится поиск подстроки [[B](A)]. Если такая подстрока была найдена, то сначала с помощью метода erase эта подстрока удаляется из исходной строки, а затем на её место вставляется символ В (по условию). Затем та же процедура происходит с подстрокой ((A)[B]). Когда в исходной строке не останется ни одной подстроки типа [[B](A)] либо ((A)[B]), ф-ция collapse завершает свою работу. Если введенная строка является скобками, то в итоге исходная строка будет иметь вид А либо В, и программа завершится с сообщением «Result: True», иначе – с результатом «Result: False».

## **Описание функций программы:**

```
1. void collapse(std::string& source, size_t n)
```

Функция предназначена для «сворачивание» исходной строки для последующей проверки на соответствие поставленной задаче.

Source – изначально, ссылка на строку, введенную пользователем. При рекурсивном вызове данной функции source содержит ссылку на изменённую строку.

N – счётчик глубины рекурсии.

```
2. void check(std::string& input)
```

Функция предназначена для проверки исходной строки на соответствие условию.

Input – ссылка, на строку введенную пользователем.

### Выводы.

Для решения полученной задачи целесообразно было использовать рекурсию.

### Тестирование:

1. Если программе не было подано аргументов, будет запрошено ввести строку для обработки.

```
Enter string:
(((A[B]))[[B]((A[B]))]))
Current collapsed string: ((A)[[B]((A[B]))]))
Depth of recursion: 0

Current collapsed string: ((A)[[B](A)])
Depth of recursion: 1

Current collapsed string: A
Depth of recursion: 2

Current collapsed string: A
Depth of recursion: 3

Result : True
```

2. Также есть возможность считывать данные из файла. Для этого требуется передать программе название файла с тестами в качестве

параметра. Файл должен содержать по одному тесту в каждой строке. Они будут проверяться последовательно.

```
Entered string: [[B](A)]

Current collapsed string: B
Depth of recursion: 0

-----

Current collapsed string: B
Depth of recursion: 1

-----

Result : True
#####
Entered string: (((A)[B]))[[[B](((A)[B]))]]

Current collapsed string: ((A)[[[B](((A)[B]))]])
Depth of recursion: 0

-----

Current collapsed string: ((A)[[[B](A)])]
Depth of recursion: 1

-----

Current collapsed string: A
Depth of recursion: 2

-----

Current collapsed string: A
Depth of recursion: 3

-----

Result : True
#####
Entered string: A

Current collapsed string: A
Depth of recursion: 0

-----

Result : True
#####
Entered string: ((A)[B])

Current collapsed string: A
Depth of recursion: 0

-----

Current collapsed string: A
Depth of recursion: 1

-----

Result : True
#####
Entered string: [[B](((A)[B]))]

Current collapsed string: [[B](A)]
Depth of recursion: 0

-----

Current collapsed string: B
Depth of recursion: 1

-----

Current collapsed string: B
Depth of recursion: 2

-----

Result : True
#####
```

Входные данные	Выходные данные
[[[[B](A)]][(A)[B]]]	Result : True
(((A)[B]))[[[B](A)]]	Result : True
qwerty	Entered string: qwerty contains invalid symbols
[[B][B]]	Result : False
[[B](((A)[B]))]	Result : True
1234	Entered string: 1234 contains invalid symbols
?><DDDQ	Entered string: ?><DDDQ contains invalid symbols
((B)[A])	Result : False
[[[[B](A)]](A)]	Result : True
B	Result : True

### Исходный код

```
#include <iostream>
#include <string>
#include <fstream>
#define Square "[[B](A)]"
#define Round  "((A)[B])"

// Функция, проверяющая входную строку на соответствие изначальному
// условию:
// ск ::= кв|кр
// кв ::= [[кв](кр)]|B
// кр ::= ((кр)[кв])|A

void collapse(std::string& source, size_t n){// source - ссылка на исходную строку, n - счетчик глубины рекурсии
    size_t s_ptr = source.find(Square);
    if (s_ptr != std::string::npos) {
        source.erase(s_ptr, 8);
        source.insert(s_ptr, "B");
    }

    size_t r_ptr = source.find(Round);
    if (r_ptr != std::string::npos) {
        source.erase(r_ptr, 8);
    }
}
```

```

        source.insert(r_ptr, "A");
    }
    std::cout << source << std::endl;
    std::cout << "Depth of recursion: " << n << std::endl;
    std::cout << " _____ "
<< std::endl;

    if (s_ptr == std::string::npos && r_ptr == std::string::npos)
        return;

    collapse(source, ++n);
}

void check(std::string& input)// input - строка,вводимая ползватаелем;
n - счетчик для подсчёта глубины рекурсии
{
    collapse(input, 0);

    if (input == "A" || input == "B") {
        std::cout << "Result : True" << std::endl;
    }
    else{
        std::cout << "Result : False" << std::endl;
    }
}

int main(int argc, char* argv[])
{
    std::string s;
    if (argc > 1) {
        std::string path = "Test/";
        path += argv[1];
        int flag = 0; //Flag to check the entered string(1 - true, 2-
false)
        std::ifstream str(path);
        while (std::getline(str, s)) {
            //удаление символа конца строки
            s.erase(s.end() - 1);

            //Checking for invalid symbols
            for (int i = 0; i < s.size(); i++) {
                if (s[i] != 'A' && s[i] != 'B' && s[i] != '(' && s[i]
!= ')' && s[i] != ' ' && s[i] != '[' && s[i] != ']')
                    flag = 0;
                else
                    flag = 1;
            }

            if (!flag) {
                std::cout << "Entered string: " + s + " contains inva-
lid symbols\n" << std::endl;
            }
        }
    }
}

```

```

        continue;
    }
    std::cout << "Entered string " + s + "\n"<<
std::endl;
    check(s);
}
}
else if(argc == 1){
    std::cout << "Enter string:\n" << std::endl;;
    std::cin >> s;

    for (int i = 0; i < s.size(); i++) {
        if (s[i] != 'A' && s[i] != 'B' && s[i] != '(' && s[i] !=
')' && s[i] != ' ' && s[i] != '[' && s[i] != ']') {
            std::cout << "Entered string: " + s + " contains inva-
lid symbols\n" << std::endl;
            return 0;
        }
    }

    check(s);

    return 0;
}

```