

**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**

Студент гр. 7382

Преподаватель

Токарев А.П.

Фирсов М. А.

Санкт-Петербург

2018

### **Задание.**

16. Построить синтаксический анализатор для понятия *скобки*.

*скобки* ::= A | B (*скобки скобки*)

### **Пояснение задания.**

Требуется, чтобы программа считывала выражение с консоли , затем, используя рекурсивную функцию, проверяющую соответствие переданного выражения понятию «скобки» получить ответ и вывести его на экран.

### **Описание алгоритма.**

1. Происходит получение выражения от пользователя;
2. Строка передается в функцию *brackets*;
3. В функции *brackets* строка проверяется посимвольно на соответствие алфавиту, если на очередной итерации встречается '(', строка передаётся в функцию *round*
4. В функции *round* строка так же посимвольно проверяется, функция запоминает, сколько закрывающих скобок должно встретиться
5. Когда функция *brackets* доходит до конца строки, рекурсия завершается и выдаётся окончательный ответ.

### Описание функций.

bool round(string str (обрабатываемая строка), string cur\_str (обработанная строка));  
bool brackets(string str(обрабатываемая строка), string cur\_str (обработанная строка));

### Тестирование.

№	Исходное выражение:	Результат:
1	«A»	These are brackets
2	«AB»	These are brackets
3	«DED»	These aren't brackets
4	«(A(A(AA)))»	These are brackets
5	«(A(AB))»	These aren't brackets

<b>6</b>	<b>«A(AB)»</b>	<b>см. рис. 1</b>
<b>7.</b>	<b>«AA(A((AA)(BB)))»</b>	<b>см. рис. 2</b>
<b>8.</b>	<b>«A(A(A(AA)))»</b>	<b>см. рис. 3</b>

```

Консоль отладки Microsoft Visual Studio
Input your string:
A(AB)
In function brackets. Deep = 1
Cur str: A
In function brackets. Deep = 2
Cur str: A(
In function round. Deep = 3
Cur str: A(A
In function brackets. Deep = 4
Cur str: A(AB)
These are brackets

```

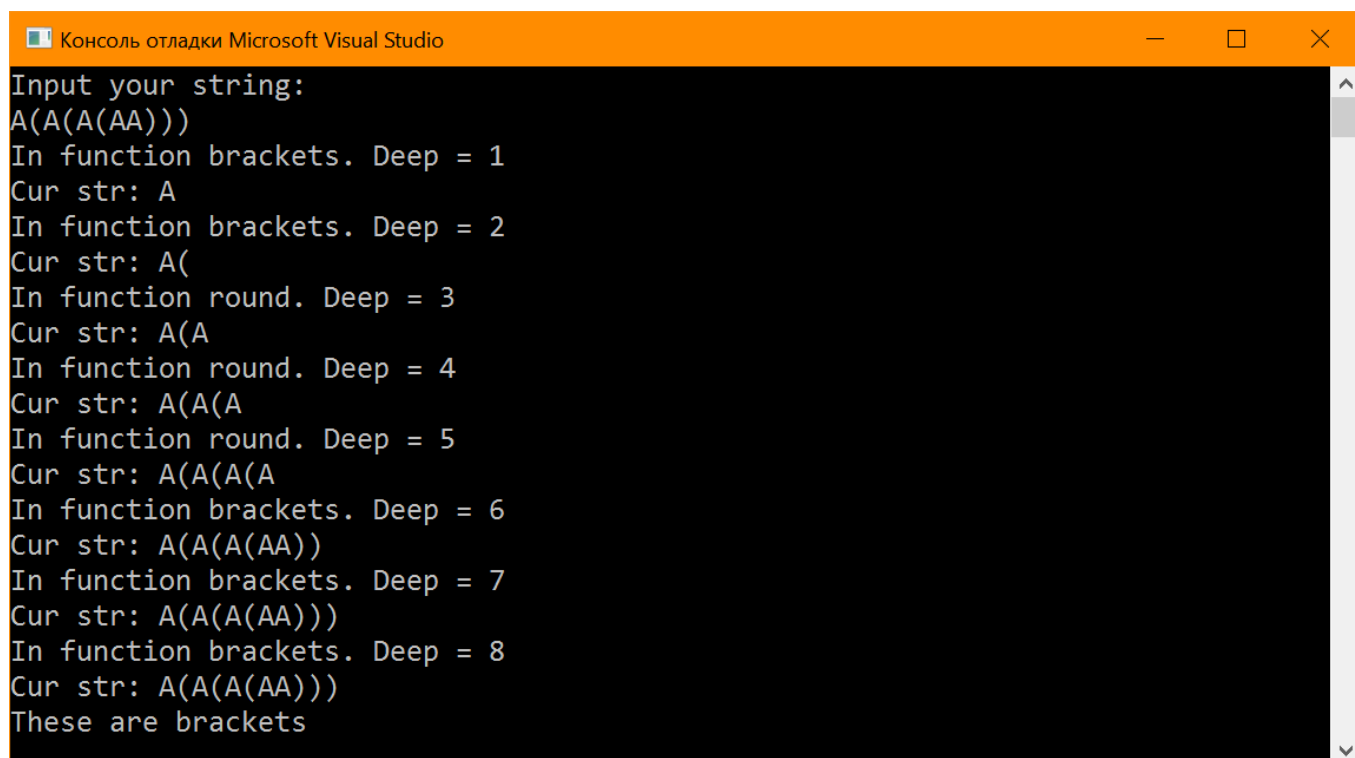
**Рисунок 1-Тест 6**

```

Консоль отладки Microsoft Visual Studio
Input your string:
AA(A((AA)(BB)))
In function brackets. Deep = 1
Cur str: A
In function brackets. Deep = 2
Cur str: AA
In function brackets. Deep = 3
Cur str: AA(
In function round. Deep = 4
Cur str: AA(A
In function round. Deep = 5
Cur str: AA(A((
In function round. Deep = 6
Cur str: AA(A((A
In function brackets. Deep = 7
Cur str: AA(A((AA)(
In function round. Deep = 8
Cur str: AA(A((AA)(B
In function brackets. Deep = 9
Cur str: AA(A((AA)(BB))
In function brackets. Deep = 10
Cur str: AA(A((AA)(BB)))
In function brackets. Deep = 11
Cur str: AA(A((AA)(BB)))
These are brackets

```

**Рисунок 2-Тест 7**



```
Консоль отладки Microsoft Visual Studio
Input your string:
A(A(A(AA)))
In function brackets. Deep = 1
Cur str: A
In function brackets. Deep = 2
Cur str: A(
In function round. Deep = 3
Cur str: A(A
In function round. Deep = 4
Cur str: A(A(A
In function round. Deep = 5
Cur str: A(A(A(A
In function brackets. Deep = 6
Cur str: A(A(A(AA))
In function brackets. Deep = 7
Cur str: A(A(A(AA)))
In function brackets. Deep = 8
Cur str: A(A(A(AA)))
These are brackets
```

**Рисунок 3-Тест 8**

### **Вывод.**

В ходе выполнения данной лабораторной работы было проведено ознакомление с основными понятиями и приёмами рекурсивного программирования, получены навыки программирования рекурсивных функций на языке программирования C++. В результате выполненной работы был создан синтаксический анализатор, который определяет: относится ли введённое выражение к понятию «скобки».

## Приложение 1. Код программы.

```
#include "pch.h"
#include <iostream>
#include <string>

using namespace std;

bool round(string str, string cur_str);

int deep = 0, iter = 0, round_in=0;

bool brackets(string str, string cur_str) { //beginning of recursion
    deep++;
    cur_str += str[iter];
    cout << "In function brackets. Deep = " << deep << endl;
    cout << "Cur str: " << cur_str << endl;
    while (str[iter]) {
        if (str[iter] == '(') {
            iter++;
            return round(str, cur_str); //enter to the second recursion
        }
        else if ((str[iter] == 'A') || (str[iter] == 'B')) {
            iter++;
            return brackets(str, cur_str);
        }
        else if (str[iter] == ')' && round_in>0) {
            iter++;
            return brackets(str, cur_str);
        }
        else
            return false; //exit when wrong symbol is encountered
    }
    return true; //exit when all symbols are checked
}

bool round(string str, string cur_str) { //second recursion
    deep++;
    cur_str += str[iter];
    cout << "In function round. Deep = " << deep << endl;
    cout << "Cur str: " << cur_str << endl;
    round_in++; //this var marked how many ')' should we meet at the exit of the second
recursion
    if (str[iter] == '(') {
        iter++;
        return round(str, cur_str);
    }
    else if (str[iter] == 'A' || str[iter] == 'B') {
        if ((str[iter + 1] == 'A' || str[iter + 1] == 'B') && str[iter + 2] == ')') {
            cur_str += str[iter+1];
            cur_str += str[iter+2];
            iter += 3;
            round_in--;
            return brackets(str, cur_str); //exit from the second recursion when nested
brackets are ended
        }
        if (str[iter + 1] == '(') {
            cur_str += str[iter + 1];
            iter+=2;
            return round(str, cur_str);
        }
        return false; //exit when the wrong symbol is encountered after 'A' || 'B'
    }
    else
```

```

        return false; //exit when the wrong symbol is encountered after '('
    }

int main() {
    string str, cur_str;
    cout << "Input your string: " << endl;
    getline(cin, str);
    if (!brackets(str, cur_str)) {
        cout << "These aren't brackets" << endl; //brackets = false
        return 0;
    }
    cout << "These are brackets" << endl; //brackets = true
    return 0;
}

```