

МЛиТА-2023 ИД3-2

Фомин Кирилл, группа 1305, вариант 23

Задание

Правильная скобочная запись арифметических выражений с двумя видами скобок. После круглой скобки в строке может стоять только квадратная, после квадратной - не обязательно. Каждая бинарная операция вместе с операндами берется в скобки. В правильной записи могут присутствовать “лишние” (двойные) скобки, но одна буква не может браться в скобки.

Пример. Правильная запись: $(((c+(a+b))*[(c-d)*a])/d)/([a-b])/[(c+(d-a))*a])]$
 Неправильная запись $[(a+b)*[a-b]-[c+a*b]/(a)+b]$

1. Уточнение языка

Примеры

$(a+[])$	В условии сказано, что одна буква не может браться в скобки, но ничего не уточнено о пустых скобках, тем более, в языке разрешены лишние скобки, так что пустые скобки можем считать разрешенными.
$((a-b))$	Формулировка “После круглой скобки в строке может стоять только квадратная, после квадратной - не обязательно.” может быть истрактована по разному, но в данном случае под словом “скобка” подразумевается не “(“ или “)”, а выражение вида “(<выражение>)”. Таким образом, разрешены конструкции вида $(([,]))$ и т.п., что также не противоречит условию “В правильной записи могут присутствовать “лишние” (двойные) скобки”.
$([])$	Запись, в которой присутствуют только пустые двойные скобки также не противоречит ни одному из условий.
	Пустая запись не противоречит ни одному из условий, так что может считаться правильной.
$((()*[])*[])$	Бинарные операции над пустыми скобками также не противоречат условиям.

Контрпримеры

$a+b$	Каждая бинарная операция вместе с операндами должна быть заключена в скобки.
$(a+)$	У бинарной операции должно быть 2 операнда.

$(\text{ } * \text{ })$	
$[a]$	Выражение в скобках не может содержать одну букву.
$(a+b)*(a+b)$	После круглой скобки в строке может стоять только квадратная
$[a*b])$	Поскольку под словом “скобка” имеется в виду запись вида $\langle \text{выражение} \rangle$ или $[\text{выражение}]$, скобочная последовательность должна быть правильной.

Рефлексия

Прочитав задание, было вообще неясно, что может вызвать рефлексия, однако начав придумывать граничные примеры, я столкнулся с рядом проблем, которые даже привели к полемике с одноклассниками.

Первые проблемы возникли с пониманием терминологии. Довольно долго пришлось разбираться, почему в условии сказано “После круглой скобки в строке может стоять только квадратная, после квадратной - не обязательно.”, а в примере правильной записи присутствует запись вида $(([a-b])/[(c+(d-a))*a])$, где, очевидно, две круглые скобки идут друг за другом. В результате выяснилось, что скорее всего под термином “скобка” имелась в виду запись вида $\langle \text{выражение} \rangle$ или $[\text{выражение}]$.

После этого возникла главная проблема: пустой символ. Сначала я даже забыл про его присутствие, но потом, вспомнив, понял, что его интерпретация может в корне изменить язык. Проблема была в том, как его воспринимать: как “присутствие отсутствия” или “отсутствие присутствия” (0 и null в программировании). То есть, может ли пустой символ считаться буквой (хоть и пустой, как пробел) или он говорит о полном отсутствии буквы. Ведь в первом случае, запись $()$ не подходит под условие, так как тогда в скобках на месте $\langle \text{выражения} \rangle$ у нас есть пустая буква, а одну букву в скобки помещать нельзя. С другой стороны, имели бы место записи вида $(a+)$, $(b/)$, $(/c)$ так как тогда в качестве операнда выступал бы пустой символ. Пришлось даже вспомнить теорию множеств, чтобы попытаться представить пустой символ как пустое множество.

Однако, мы пришли к выводу, что пустой символ не является буквой, то есть обозначает отсутствие присутствия и не может быть использован как операнд в выражениях. С другой стороны, мы приняли, что “скобка” ($\langle \text{выражение} \rangle$) является операндом, и даже пустые скобки - операнд.

В общем, задание оказалось очень интересным, но хотелось бы уточнений в условии, потому что на данный момент мое решение является верным только для интерпретации, где “скобка” - всегда операнд. В дополнение к этому, я ввел бы отдельный термин для обозначения записей вида “скобка” (хотя бы “скобки”).

Примеры (исправление)

$((a+b)*[b-a])$	Все бинарные операции с операндами должны быть окружены скобками
-----------------	--

$(((a+b)^*[b-a]))$	Лишние скобки разрешены
$[[a+b]+[b-c]]$.После квадратной скобки может стоять квадратная

Задание 2. КС-грамматика языка

$\langle \text{язык} \rangle ::= \langle \text{выражение} \rangle \mid \wedge$
 $\langle \text{выражение} \rangle ::= \langle \text{круглые скобки} \rangle \mid \langle \text{квадратные скобки} \rangle$
 $\langle \text{круглые скобки} \rangle = (\langle \text{операция} \rangle)$
 $\langle \text{квадратные скобки} \rangle = [\langle \text{операция} \rangle]$
 $\langle \text{операция} \rangle ::= \langle \text{круглые скобки} \rangle \langle \text{после круглых скобок} \rangle \mid \langle \text{квадратные скобки} \rangle \langle \text{после квадратных скобок} \rangle \mid \langle \text{буква} \rangle \langle \text{после буквы} \rangle$
 $\langle \text{после круглых скобок} \rangle = \langle \text{знак} \rangle \langle \text{набор операндов 1} \rangle \mid \wedge$
 $\langle \text{после квадратных скобок} \rangle = \langle \text{знак} \rangle \langle \text{операнд} \rangle \mid \wedge$
 $\langle \text{после букв} \rangle = \langle \text{знак} \rangle \langle \text{операнд} \rangle$
 $\langle \text{набор операндов 1} \rangle ::= \langle \text{квадратные скобки} \rangle \mid \langle \text{буква} \rangle$
 $\langle \text{операнд} \rangle ::= \langle \text{круглые скобки} \rangle \mid \langle \text{квадратные скобки} \rangle \mid \langle \text{буква} \rangle$
 $\langle \text{буква} \rangle ::= a \mid b \mid c \mid \dots \mid z$
 $\langle \text{знак} \rangle ::= + \mid - \mid * \mid /$

Задание 3. Разбор примера

$(((a+b)^*[b-a]))$
 $\langle \text{язык} \rangle$
 $\langle \text{выражение} \rangle$
 $\langle \text{квадратные скобки} \rangle$
 $[\langle \text{операция} \rangle]$
 $[\langle \text{круглые скобки} \rangle \langle \text{после круглых скобок} \rangle]$
 $[(\langle \text{операция} \rangle) \langle \text{после круглых скобок} \rangle]$
 $[(\langle \text{операция} \rangle) \wedge]$
 $[(\langle \text{круглые скобки} \rangle \langle \text{после круглых скобок} \rangle)]$
 $[(\langle \text{операция} \rangle) \langle \text{после круглых скобок} \rangle]$
 $[(\langle \text{операция} \rangle) \langle \text{знак} \rangle \langle \text{набор операндов 1} \rangle]$
 $[(\langle \text{операция} \rangle)^* \langle \text{набор операндов 1} \rangle]$
 $[(\langle \text{операция} \rangle)^* \langle \text{квадратные скобки} \rangle]$
 $[(\langle \text{операция} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{буква} \rangle \langle \text{после букв} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{после букв} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{знак} \rangle \langle \text{операнд} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{операнд} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{операция} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{буква} \rangle \langle \text{после букв} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{буква} \rangle \langle \text{знак} \rangle \langle \text{операнд} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{буква} \rangle \langle \text{знак} \rangle \langle \text{операнд} \rangle]]$
 $[(\langle \text{а} \rangle \langle \text{буква} \rangle)^* [\langle \text{буква} \rangle \langle \text{знак} \rangle \langle \text{операнд} \rangle]]$

$[(a+b)^*[b-a]]$

Задание 4. Принадлежность классу LL(1)

Обозначим как W_1, W_2, W_3, W_4 значения, которые может принимать тег <язык>. $L(W)$ - это множество символов, которые могут являться первыми в каждом значении тега.

Пусть $\langle \text{язык} \rangle ::= W_1 \mid W_2 \mid W_3 \mid W_4$, тогда, так как $\langle \text{язык} \rangle ::= \langle \text{выражение} \rangle \mid ^\wedge$

$L(W_1) = \{ \langle \text{"("} \rangle, \langle \text{"["} \rangle \}$

Пустой символ не является терминальным. Так проверять нельзя.

Всего один вариант.

$\langle \text{выражение} \rangle ::= \langle \text{круглые скобки} \rangle \mid \langle \text{квадратные скобки} \rangle$

$L(W_1) = \{ \langle \text{"("} \rangle \}$

$L(W_2) = \{ \langle \text{"["} \rangle \}$

Видим, что $L(W_1) \cap L(W_2) = \emptyset$, значит однозначность ветвления по первому символу для тега <выражение> выполняется.

$\langle \text{операция} \rangle ::= \langle \text{круглые скобки} \rangle \langle \text{после круглых скобок} \rangle \mid \langle \text{квадратные скобки} \rangle \langle \text{после квадратных скобок} \rangle \mid \langle \text{буква} \rangle \langle \text{после буквы} \rangle$

$L(W_1) = \{ \langle \text{"("} \rangle \}$

$L(W_2) = \{ \langle \text{"["} \rangle \}$

$L(W_3) = \{ \langle \text{"a"}, \langle \text{"b"}, \dots, \langle \text{"z"} \rangle \}$

$L(W_1) \cap L(W_2) \cap L(W_3) = \emptyset$

нужно проверять попарные пересечения !

$L(W_1) \cap L(W_2) = \emptyset$

$L(W_1) \cap L(W_3) = \emptyset$

$L(W_2) \cap L(W_3) = \emptyset$

$\langle \text{после круглых скобок} \rangle = \langle \text{знак} \rangle \langle \text{набор операндов 1} \rangle \mid ^\wedge$

$L(W_1) = \{ \langle \text{"+"}, \langle \text{"-"} \rangle, \langle \text{"/"} \rangle, \langle \text{"*"} \rangle \}$

$L(W_2) = \{ \langle \text{"^"} \rangle \}$

$L(W_1) \cap L(W_2) = \emptyset$

$\langle \text{после квадратных скобок} \rangle = \langle \text{знак} \rangle \langle \text{операнд} \rangle \mid ^\wedge$

$L(W_1) = \{ \langle \text{"+"}, \langle \text{"-"} \rangle, \langle \text{"/"} \rangle, \langle \text{"*"} \rangle \}$

$L(W_2) = \{ \langle \text{"^"} \rangle \}$

$L(W_1) \cap L(W_2) = \emptyset$

<набор операндов 1> ::= <квадратные скобки> | <буква>

$L(W_1) = \{ "[", "]" \}$

$L(W_2) = \{ "[", "a", "b", \dots, "z" \}$

$L(W_1) \cap L(W_2) = \emptyset$

<операнд> ::= <круглые скобки> | <квадратные скобки> | <буква>

$L(W_1) = \{ "(" \}$

$L(W_2) = \{ "[", "]" \}$

$L(W_3) = \{ "a", "b", \dots, "z" \}$

$L(W_1) \cap L(W_2) \cap L(W_3) = \emptyset$

$L(W_1) \cap L(W_2) = \emptyset$

$L(W_1) \cap L(W_3) = \emptyset$

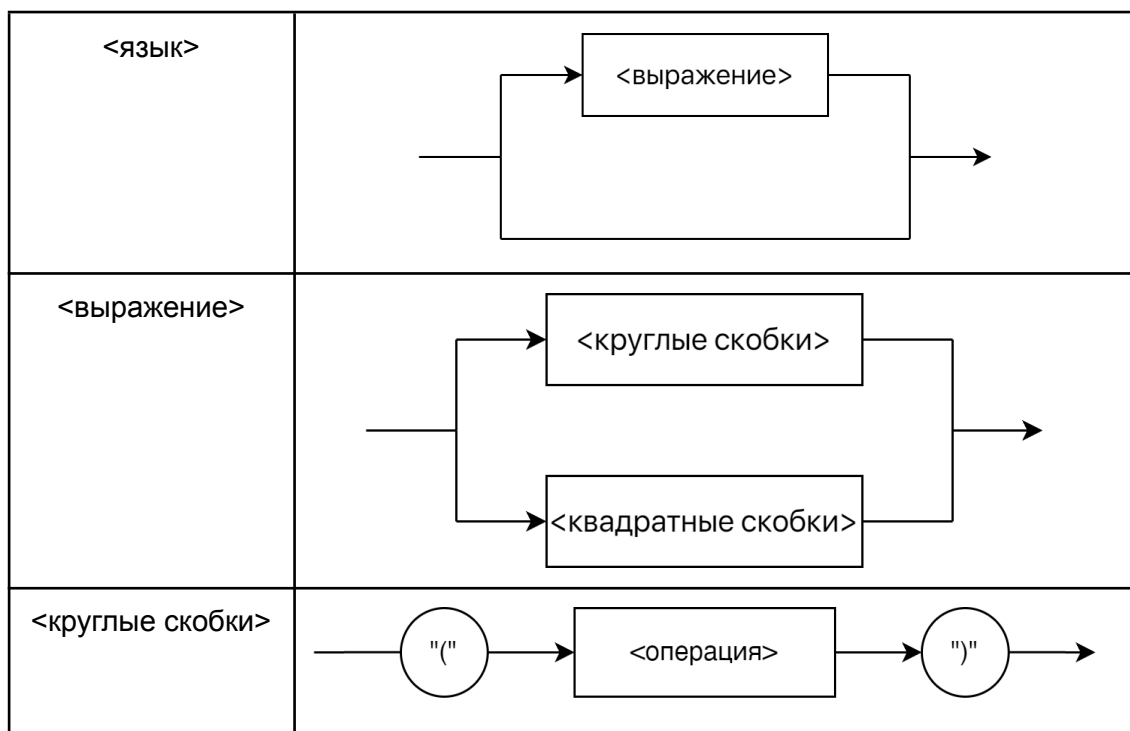
$L(W_2) \cap L(W_3) = \emptyset$

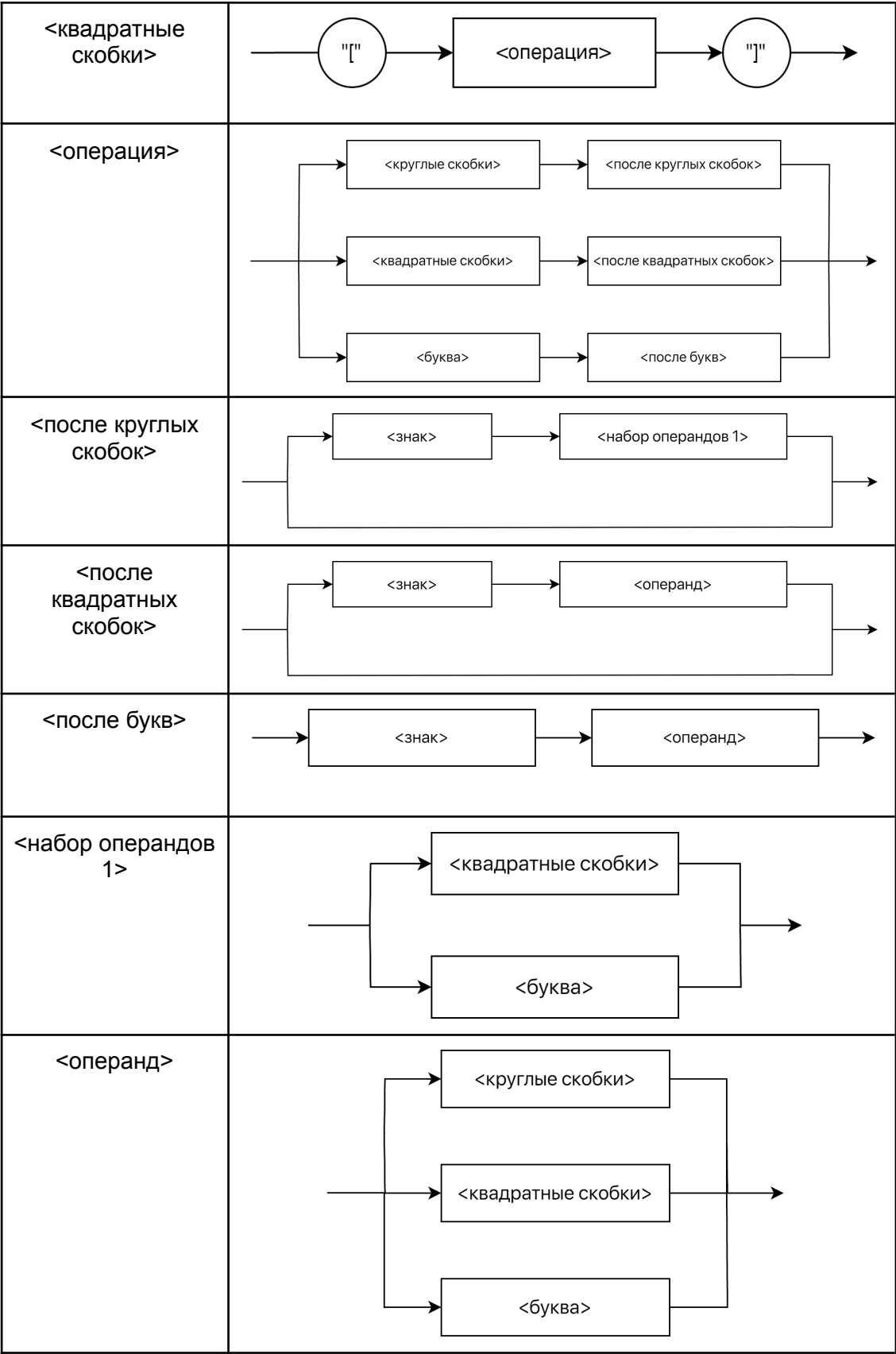
Вывод: исходная КС-грамматика принадлежит к классу LL(1).

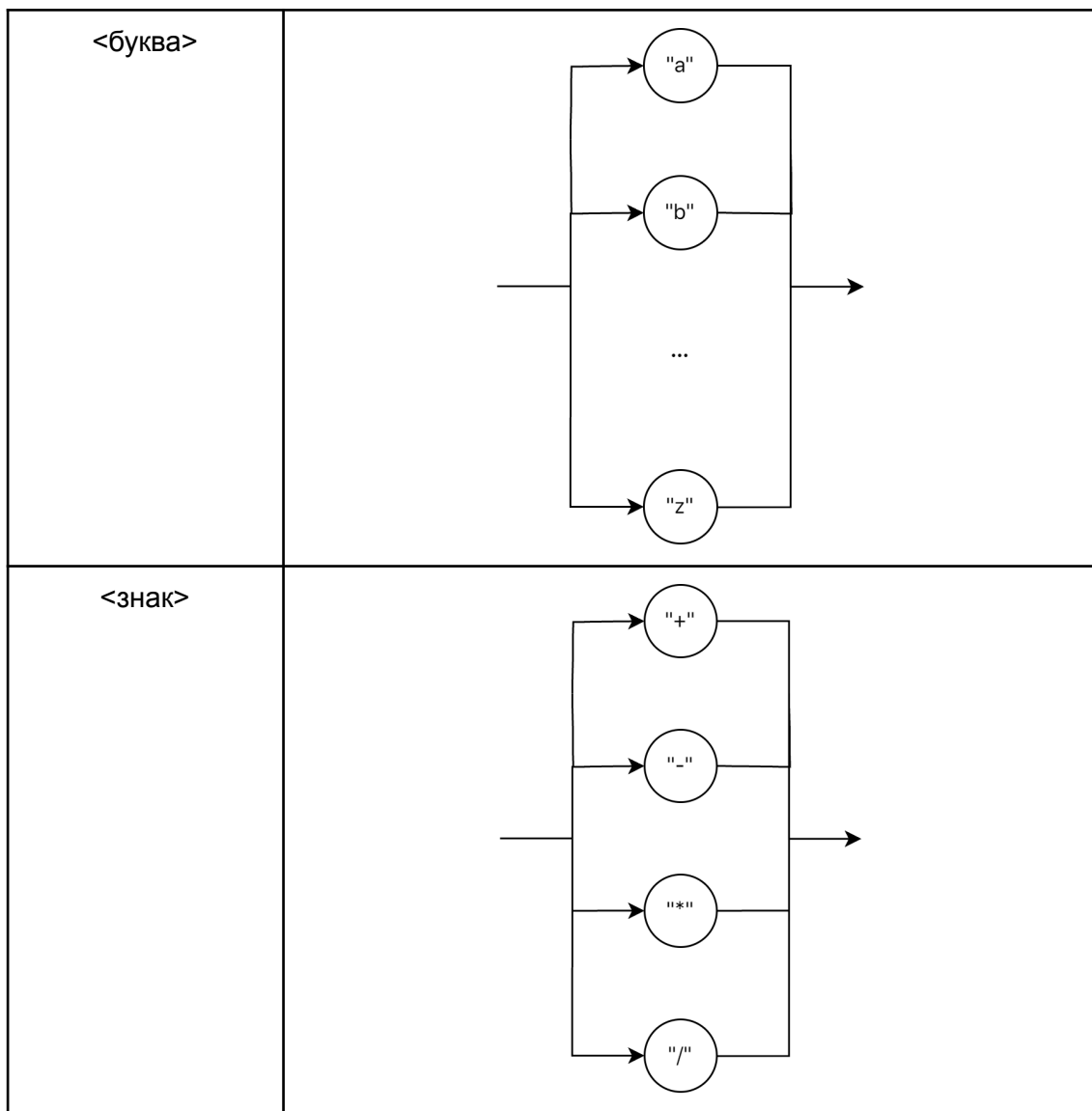
Задание 5. Модифицированная грамматика (LL(1))

Исходная грамматика является LL(1) грамматикой.

Задание 6. Таблица перевода языка в диаграммы. Оптимизация числа диаграмм подстановкой

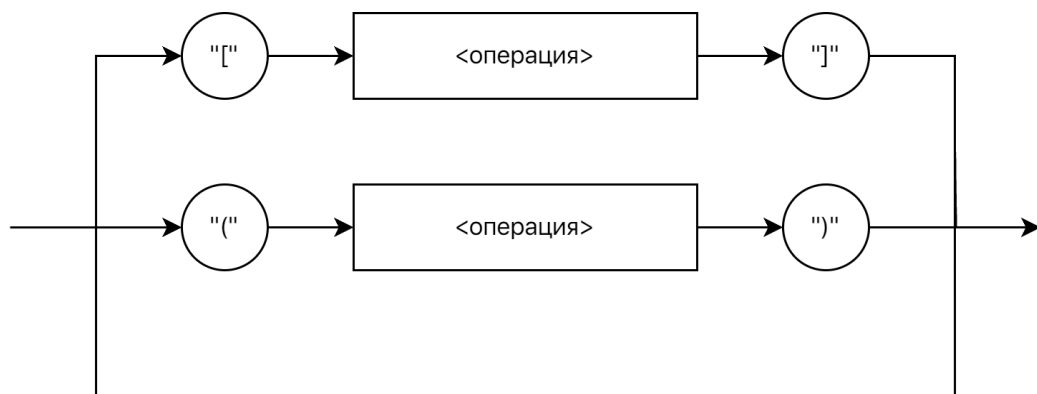




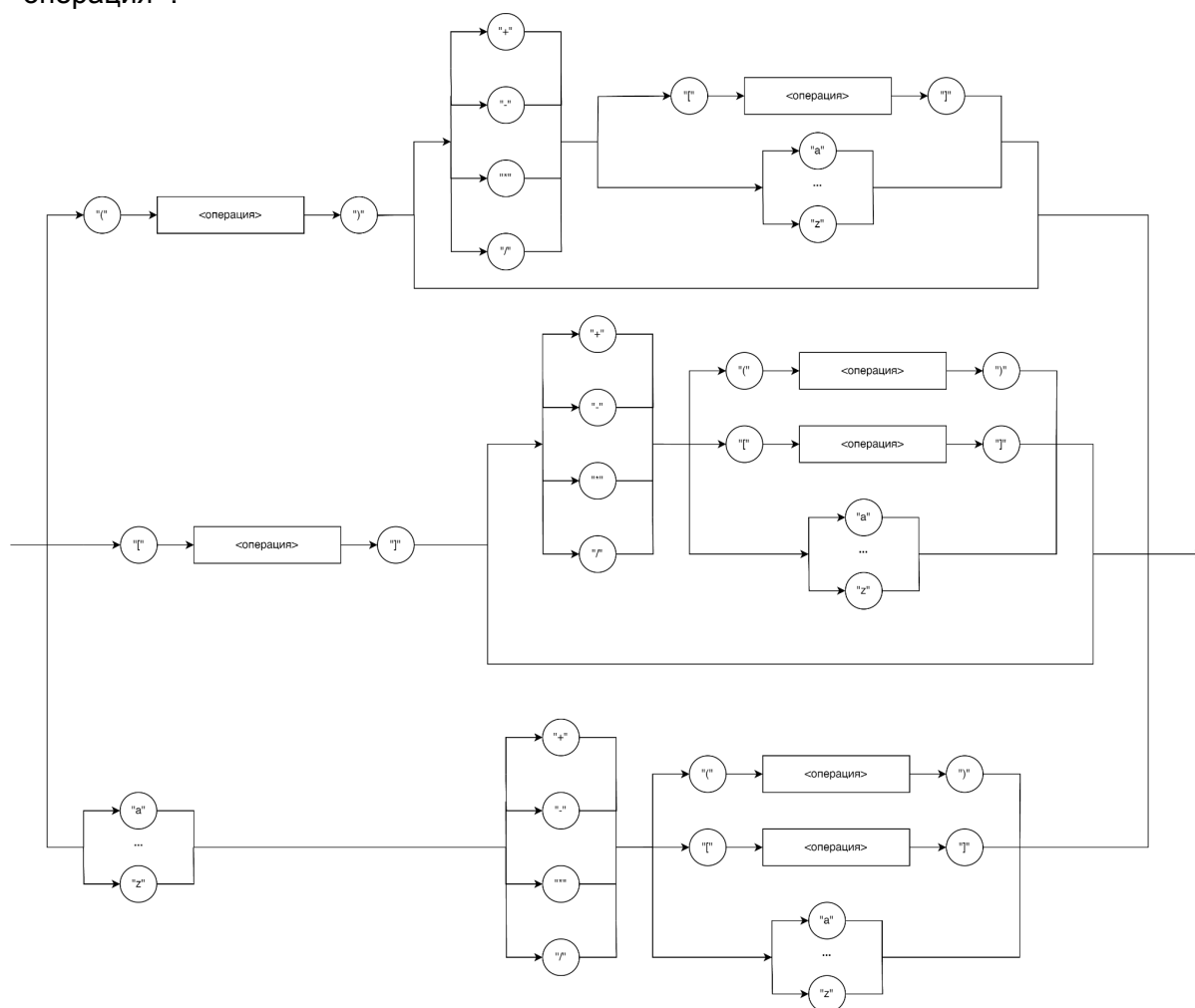


Оптимизация:

<язык>:



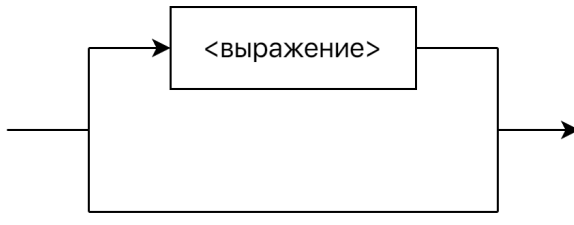
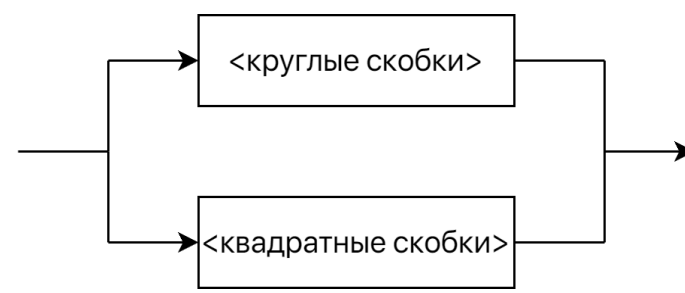
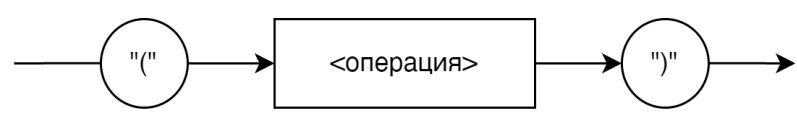
<операция>:

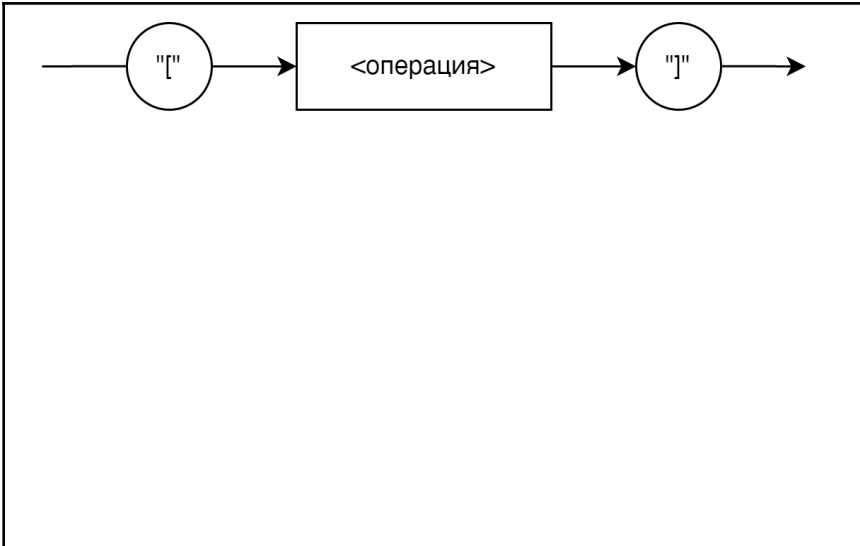


[Ссылка на диаграмму в высоком разрешении](#)

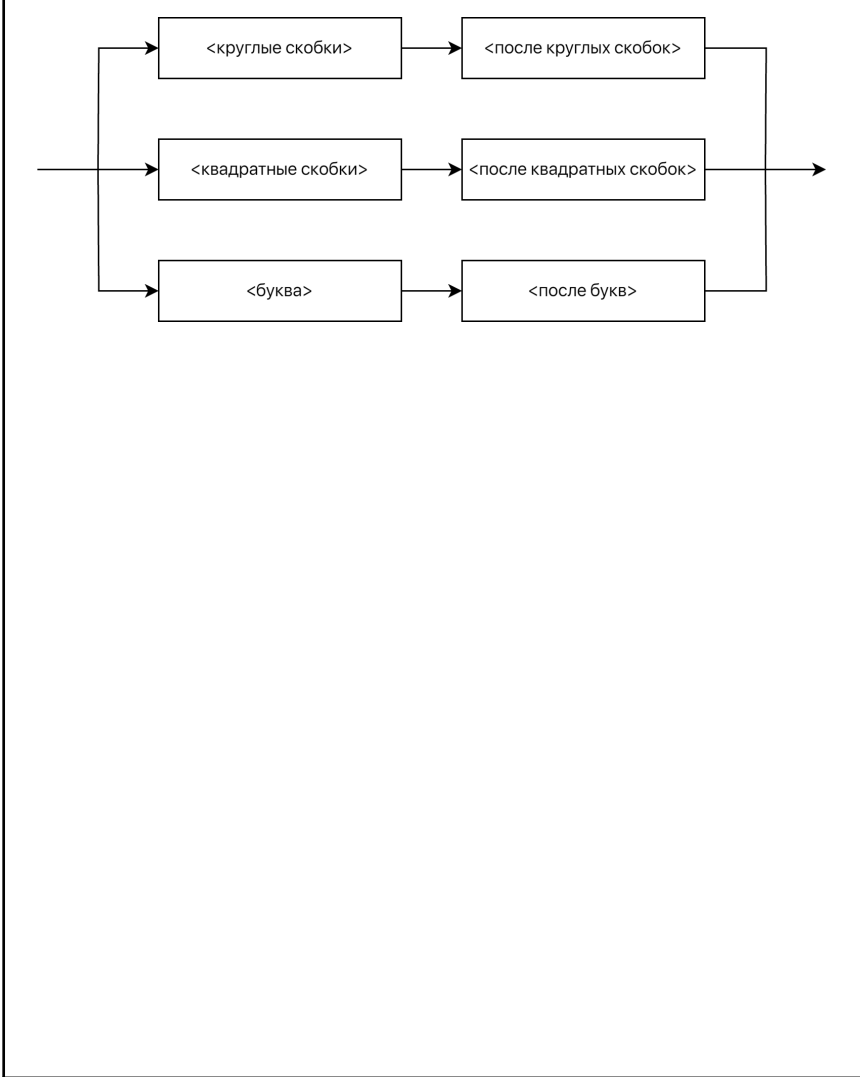
Задание 7. Таблица перевода синтаксических диаграмм в алгоритм синтаксического анализа

Будем использовать диаграммы без сокращения, так как иначе код будет нечитабельным и в нем будет много повторений.

 <pre> graph LR In(()) --> Box1[<выражение>] In --> Out(()) Box1 --> Out </pre>	<pre> функция language { иначе если ch = '(' то { expression() } иначе если ch = '[' то { expression() } } функция language { если (ch = '(' или ch = '[') то { expression() } } </pre>
 <pre> graph LR In(()) --> Box1[<круглые скобки>] In --> Box2[<квадратные скобки>] Box1 --> Out(()) Box2 --> Out </pre>	<pre> функция expression { если ch = '(' то { roundBrackets() } иначе если ch = '[' то { squareBrackets() } иначе { ошибка } } </pre>
 <pre> graph LR In(()) --> Circle1((("(")) Circle1 --> Box1[<операция>] Box1 --> Circle2(("))") Circle2 --> Out(()) </pre>	<pre> функция roundBrackets() { если ch = "(" то { ch := read() operation() если ch = ")" то { ch := read() } иначе { ошибка } } иначе { ошибка } } </pre>



```
функция squareBrackets() {  
    если ch = "[" то {  
        ch := read()  
        operation()  
        если ch = "]" то {  
            ch := read()  
        }  
        иначе {  
            ошибка  
        }  
    }  
    иначе {  
        ошибка  
    }  
}
```

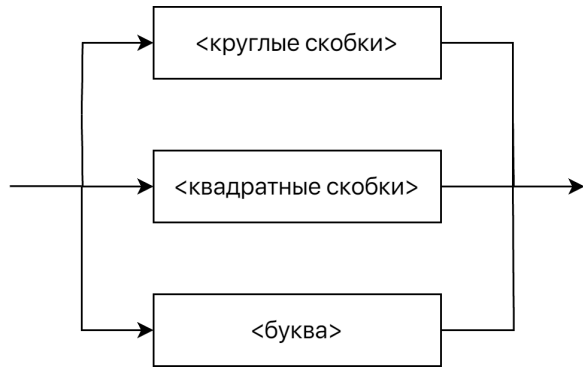


```
функция operation() {  
    если ch = "(" то {  
        roundBrackets()  
        если ch ∈ {+, -, *, =}  
        то {  
            afterRoundBrackets()  
        }  
    иначе если ch ∈ {a..z} то {  
        letter()  
        если ch ∈ {+, -, *, /}  
        то {  
            afterLetters()  
        }  
    иначе  
        ошибка  
    }  
    иначе если ch = "[" то {  
        squareBrackets()  
        если ch ∈ {+, -, *, /}  
        то {  
            afterSquareBrackets()  
        }  
    }  
    иначе  
        ошибка  
}
```

псевдокод не соответствует диаграмме

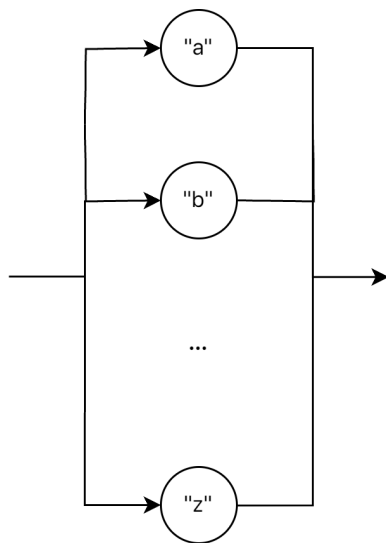
на этом месте проверка остановлена, так как
ведется до первой серьезной ошибки

<pre> graph LR In(()) --> J1(()) J1 --> A1[<круглые скобки>] J1 --> A2[<квадратные скобки>] J1 --> A3[<буква>] A1 --> B1[<после круглых скобок>] A2 --> B2[<после квадратных скобок>] A3 --> B3[<после букв>] B1 --> J2(()) B2 --> J2 B3 --> J2 J2 --> Out(()) </pre>	<pre> функция operation() { если ch = "(" то { roundBrackets() afterRoundBrackets() } иначе если ch ∈ {a..z} то { letter() afterLetters() } иначе если ch = "[" то { squareBrackets() afterSquareBrackets() } иначе ошибка } </pre>
<pre> graph LR In(()) --> J1(()) J1 --> A1[<знак>] J1 --> B1[<набор операндов 1>] A1 --> J2(()) B1 --> J2 J2 --> Out(()) </pre>	<pre> функция afterRoundBrackets(){ если ch ∈ {+, -, *, /} то { sign() setOfOperands1() } } </pre>
<pre> graph LR In(()) --> J1(()) J1 --> A1[<знак>] J1 --> B1[<операнд>] A1 --> J2(()) B1 --> J2 J2 --> Out(()) </pre>	<pre> функция afterSquareBrackets(){ если ch ∈ {+, -, *, /} то { sign() operand() } } </pre>
<pre> graph LR In(()) --> A1[<знак>] A1 --> B1[<операнд>] B1 --> Out(()) </pre>	<pre> функция afterLetters(){ если ch ∈ {+, -, *, /} то { sign() operand() } иначе ошибка } </pre>
<pre> graph LR In(()) --> J1(()) J1 --> A1[<квадратные скобки>] J1 --> A2[<буква>] A1 --> J2(()) A2 --> J2 J2 --> Out(()) </pre>	<pre> функция setOfOperands1(){ если ch = "[" то { squareBrackets() } иначе если ch ∈ {a..z} то { letter() } иначе ошибка } </pre>



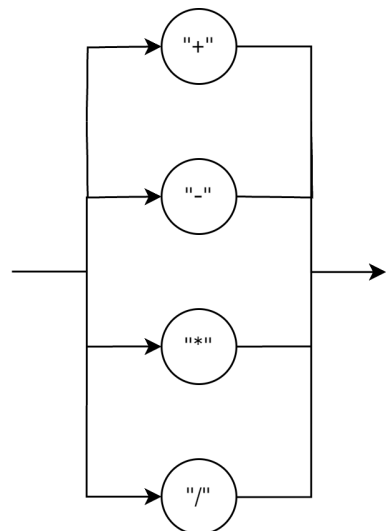
```

функция operand(){
  если ch = "(" то {
    roundBrackets()
  }
  иначе если ch = "[" то {
    squareBrackets()
  }
  иначе если ch ∈ {a..z} то {
    letter()
  }
  иначе
    ошибка
}
  
```



```

функция letter{
  если ch ∈ {a..z} то {
    ch := read()
  }
  иначе
    ошибка
}
  
```



```

функция sign{
  если ch ∈ {+, -, *, /} то {
    ch := read()
  }
  иначе
    ошибка
}
  
```

Задание 8. Таблица перевода алгоритма в программу

<pre> функция language { если ch = '(' то { expression() } иначе если ch = '[' то { expression() } } </pre>	<pre> void SyntaxAnalyzer::language() { if ((ch == '(') (ch == '[')) { expression(); } else throw valueError(); } </pre>
<pre> функция expression { если ch = '(' то { roundBrackets() } иначе если ch = '[' то { squareBrackets() } иначе { ошибка } } </pre>	<pre> void SyntaxAnalyzer::expression() { if (ch == '(') { roundBrackets(); } else if (ch == '[') { squareBrackets(); } else throw valueError(); } </pre>
<pre> функция roundBrackets() { если ch = "(" то { ch := read() operation() если ch = ")" то { ch := read() } иначе { ошибка } } иначе { ошибка } } </pre>	<pre> void SyntaxAnalyzer::roundBrackets() { if (ch == '(') { read(); operation(); if (ch == ')') { read(); } else throw valueError(); } else throw valueError(); } </pre>
<pre> функция squareBrackets() { если ch = "[" то { ch := read() operation() } } </pre>	<pre> void SyntaxAnalyzer::squareBrackets() { if (ch == '[') { </pre>

<pre> если ch = "]" то { ch := read() } иначе { ошибка } } иначе { ошибка } </pre>	<pre> read(); operation(); if (ch == ']') { read(); } else throw valueError(); } else throw valueError(); } </pre>
<pre> функция operation() { если ch = "(" то { roundBrackets() afterRoundBrackets() } иначе если ch ∈ {a..z} то { letter() afterLetters() } иначе если ch = "[" то { squareBrackets() afterSquareBrackets() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::operation() { if (ch == '(') { roundBrackets(); afterRoundBrackets(); } else if (std::isalpha(ch)) { letter(); afterLetters(); } else if (ch == '[') { squareBrackets(); afterSquareBrackets(); } else throw valueError(); } </pre>
<pre> функция afterRoundBrackets(){ если ch ∈ {+, -, *, /} то { sign() setOfOperands1() } } </pre>	<pre> void SyntaxAnalyzer::afterSquareBrackets() { if (isSign(ch)) { sign(); operand(); } } </pre>

	<pre> } } </pre>
<pre> функция afterSquareBrackets(){ если ch ∈ {+, -, *, /} то { sign() operand() } } </pre>	<pre> void SyntaxAnalyzer::afterSquareBrackets() { if (isSign(ch)) { sign(); operand(); } } </pre>
<pre> функция afterLetters(){ если ch ∈ {+, -, *, /} то { sign() operand() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::afterLetters() { if (isSign(ch)) { sign(); operand(); } else throw valueError(); } </pre>
<pre> функция setOfOperands1(){ если ch = "[" то { squareBrackets() } иначе если ch ∈ {a..z} то { letter() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::setOfOperands1() { if (ch == '[') { squareBrackets(); } else if (std::isalpha(ch)) { letter(); } else throw valueError(); } </pre>

<pre> функция operand(){ если ch = "(" то { roundBrackets() } иначе если ch = "[" то { squareBrackets() } иначе если ch ∈ {a..z} то { letter() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::operand() { if (ch == '(') { roundBrackets(); } else if (ch == '[') { squareBrackets(); } else if (std::isalpha(ch)) { letter(); } else throw valueError(); } </pre>
<pre> функция letter{ если ch ∈ {a..z} то { ch := read() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::letter() { if (std::isalpha(ch)) { read(); } else throw valueError(); } </pre>
<pre> функция sign{ если ch ∈ {+, -, *, /} то { ch := read() } иначе ошибка } </pre>	<pre> void SyntaxAnalyzer::sign() { if (isSign(ch)) { read(); } else throw valueError(); } </pre>

Задание 9. Код программы

Программа написана на языке C++ с использованием стандартной библиотеки. В случае ошибки, она уведомляет пользователя, выводит обработанную часть (так можно понять, на каком символе возникла ошибка) и показывает стек вызова функций.

```

#include <iostream>
#include <string>
#include <windows.h>
#include <vector>

```

```

class valueError : public std::exception {
public:
    valueError() : std::exception() {};
};

```

```

class SyntaxAnalyzer {

```



```

public:
    SyntaxAnalyzer();
    void readInitialExpression();
    void analyze();
private:
    std::string initialExpression;
    char ch;
    std::string initialExpressionProcessedPart;
    std::vector<std::string> callStack;
    void read();
    void language();
    void expression();
    void roundBrackets();
    void squareBrackets();
    void operation();
    void afterRoundBrackets();
    void afterSquareBrackets();
    void afterLetters();
    void setOfOperands1();
    void operand();
    void letter();
    void sign();

    bool isSign(const char ch);
    void tryAgain();
    void showCallStack();
};

SyntaxAnalyzer::SyntaxAnalyzer() {
    initialExpression = "";
    initialExpressionProcessedPart = "";
    callStack.push_back("");
}

void SyntaxAnalyzer::readInitialExpression() {
    std::cout << "Правильная скобочная запись арифметических
выражений с двумя видами скобок.\n"
        "После круглой скобки в строке может стоять только
квадратная, после квадратной - не обязательно.\n"
        "Каждая бинарная операция вместе с операндами берется в
скобки.\n"
        "В правильной записи могут присутствовать “лишние”
(двойные)\n"
        "скобки, но одна буква не может браться в скобки.\n"
        "\n"
        "Пример.\n"
        "Правильная запись:
[[((c+(a+b))*[(c-d)*a])/d]/([(a-b)])/[(c+(d-a))*a]]\n"

```

```

        "Неправильная запись [(a+b)*([a-b]-([c+a*b]/(a))+b)]\n"
        "\n"
        "Please, enter initial string: ";
std::cin >> initialExpression;
read();
}

void SyntaxAnalyzer::read()
{
    callStack.push_back("read(); ");
    if (!initialExpression.empty()) {
        ch = initialExpression[0];
        initialExpressionProcessedPart += ch;
        initialExpression.erase(0, 1);
    }
    else {
        ch = '!';
    }
}

void SyntaxAnalyzer::analyze() {
    try {
        language();
        std::cout << "OK";
    }
    catch (valueError error) {
        showCallStack();
    }
    tryAgain();
}

void SyntaxAnalyzer::language()
{
    callStack.push_back("language(); ");
    if ((ch == '(') || (ch == '[')) {
        expression();
    }
    else throw valueError();
}

void SyntaxAnalyzer::expression()
{
    callStack.push_back("expression(); ");
    if (ch == '(') {
        roundBrackets();
    }
    else if (ch == '[') {

```

```

        squareBrackets();
    }
    else throw valueError();
}

void SyntaxAnalyzer::roundBrackets()
{
    callStack.push_back("roundBrackets(); ");
    if (ch == '(') {
        read();
        operation();
        if (ch == ')') {
            read();
        }
        else throw valueError();
    }
    else throw valueError();
}

void SyntaxAnalyzer::squareBrackets()
{
    callStack.push_back("squareBrackets(); ");
    if (ch == '[') {
        read();
        operation();
        if (ch == ']') {
            read();
        }
        else throw valueError();
    }
    else throw valueError();
}

void SyntaxAnalyzer::operation()
{
    callStack.push_back("operation(); ");
    if (ch == '(') {
        roundBrackets();
        afterRoundBrackets();
    }
    else if (std::isalpha(ch)) {
        letter();
        afterLetters();
    }
    else if (ch == '[') {
        squareBrackets();
        afterSquareBrackets();
    }
}

```

```

        }
        else throw valueError();
    }

void SyntaxAnalyzer::afterRoundBrackets()
{
    callStack.push_back("afterRoundBrackets(); ");
    if (isSign(ch)) {
        sign();
        if (std::isalpha(ch) || ch == '[') {
            setOfOperands1();
        }
        else throw valueError();
    }
}

void SyntaxAnalyzer::afterSquareBrackets()
{
    callStack.push_back("afterSquareBrackets(); ");
    if (isSign(ch)) {
        sign();
        operand();
    }
}

void SyntaxAnalyzer::afterLetters()
{
    callStack.push_back("afterLetters(); ");
    if (isSign(ch)) {
        sign();
        operand();
    }
    else throw valueError();
}

void SyntaxAnalyzer::setOfOperands1()
{
    callStack.push_back("setOfOperands1(); ");
    if (ch == '[') {
        squareBrackets();
    }
    else if (std::isalpha(ch)) {
        letter();
    }
    else throw valueError();
}

```

```

void SyntaxAnalyzer::operand()
{
    callStack.push_back("operand(); ");
    if (ch == '(') {
        roundBrackets();
    }
    else if (ch == '[') {
        squareBrackets();
    }
    else if (std::isalpha(ch)) {
        letter();
    }
    else throw valueError();
}

void SyntaxAnalyzer::letter()
{
    callStack.push_back("letter(); ");
    if (std::isalpha(ch)) {
        read();
    }
    else throw valueError();
}

void SyntaxAnalyzer::sign()
{
    callStack.push_back("sign(); ");
    if (isSign(ch)) {
        read();
    }
    else throw valueError();
}

bool SyntaxAnalyzer::isSign(const char ch)
{
    if (ch == '+' || ch == '-' || ch == '*' || ch == '/')
        return true;
    else
        return false;
}

void SyntaxAnalyzer::tryAgain()
{
    initialExpressionProcessedPart.clear();
    callStack.clear();
    std::string tryAgain;
    std::cout << "\nTry again?: (Y/N) ";
}

```

```

        std::cin >> tryAgain;
        if (tryAgain == "Y") {
            readInitialExpression();
            analyze();
        }
    }

void SyntaxAnalyzer::showCallStack()
{
    std::cout << "Error!\nProcessed part : " <<
initialExpressionProcessedPart << std::endl << std::endl << "Call stack:
";
    for (std::string call : callStack) {
        std::cout << call;
    }
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    SyntaxAnalyzer syntaxAnalyzer;
    syntaxAnalyzer.readInitialExpression();
    syntaxAnalyzer.analyze();
}

```

```

C:\Users\neko\Desktop\Lab\MLiTA\SyntaxAnalyzer\SyntaxAnalyzer\Debug\SyntaxAnalyzer.exe
Правильная скобочная запись арифметических выражений с двумя видами скобок.
После круглой скобки в строке может стоять только квадратная, после квадратной - не обязательно.
Каждая бинарная операция вместе с операндами берется в скобки.
В правильной записи могут присутствовать "лишние" (двойные)
скобки, но одна буква не может браться в скобки.

Пример.
Правильная запись: [(((c+(a+b))*[(c-d)*a])/d)/([a-b])/[(c+(d-a))*a])]
Неправильная запись [(a+b)*([a-b]-([c+a*b]/(a))+b)]

Please, enter initial string: [(((c+(a+b))*[(c-d)*a])/d)/([a-b])/[(c+(d-a))*a])]
OK
Try again?: (Y/N)

Please, enter initial string: ((a))
Error!
Processed part : ((a))

Call stack: read(); language(); expression(); roundBrackets(); read(); operation(); roundBrackets(); read(); operation();
; letter(); read();
Try again?: (Y/N)

```

```
Please, enter initial string: a+b
Error!
Processed part : a

Call stack: read(); language();
Try again?: (Y/N)
```

```
Please, enter initial string: ((a+b)*(a+b))
Error!
Processed part : ((a+b)*(

Call stack: read(); language(); expression();
; letter(); read(); afterLetters(); sign(); re
ead();
Try again?: (Y/N)
```

```
Please, enter initial string: ((a+b)*[a+b])
OK
Try again?: (Y/N)
```

```
Please, enter initial string: (a+b]
Error!
Processed part : (a+b]

Call stack: read(); language(); expression
gn(); read(); operand(); letter(); read();
Try again?: (Y/N)
```

```
Please, enter initial string: (a+b
Error!
Processed part : (a+b

Call stack: read(); language(); expression(
gn(); read(); operand(); letter(); read();
Try again?: (Y/N)
```

```
Please, enter initial string: )a
Error!
Processed part : )

Call stack: read(); language();
Try again?: (Y/N)
```

```
Please, enter initial string: [[[[[
Error!
Processed part : [[[[[

Call stack: read(); language(); expressio
()); squareBrackets(); read(); operation()
Try again?: (Y/N)
```

Задание 10. Исполняемый файл

[Ссылка на исполняемый файл](#)

Требования:

OS Windows 10/11 x64