МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИиСП

ОТЧЕТ ПО ВЫПОЛНЕНИЮ   
ПРОГРАММЫ

**По дисциплине: ТАиФЯ**

Выполнили: студент 3 курса

специальности ПС

группы 31

Смирнов Фёдор Сергеевич

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

Проверил: Зав. каф. ИиСП

Нехорошкова Л. Г.

Подпись\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2021

## Содержание

[Содержание](#_heading=h.gjdgxs) **2**

[СОДЕРЖАНИЕ ЗАДАНИЯ](#_heading=h.30j0zll) **3**

[Задание 1](#_heading=h.ahdt8iqarkqi) 5

[**Задание 2**](#_heading=h.2et92p0) **5**

[Задание 4](#_heading=h.at9kcpd78bbl) 7

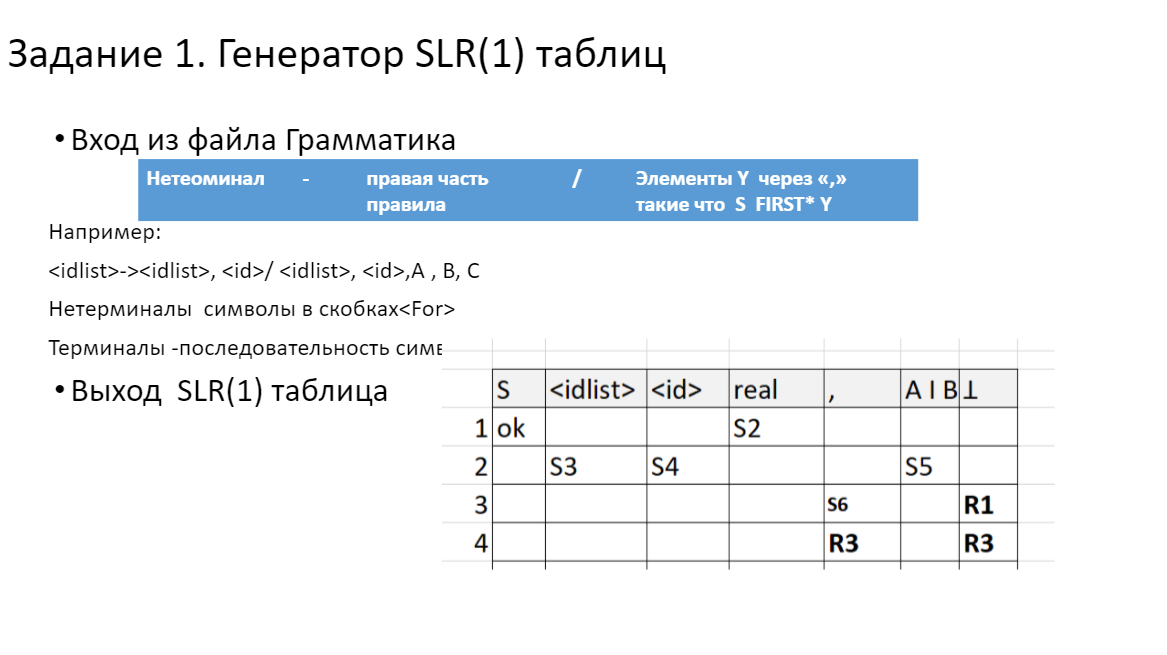
[**ВЫВОД**](#_heading=h.2s8eyo1) **8**

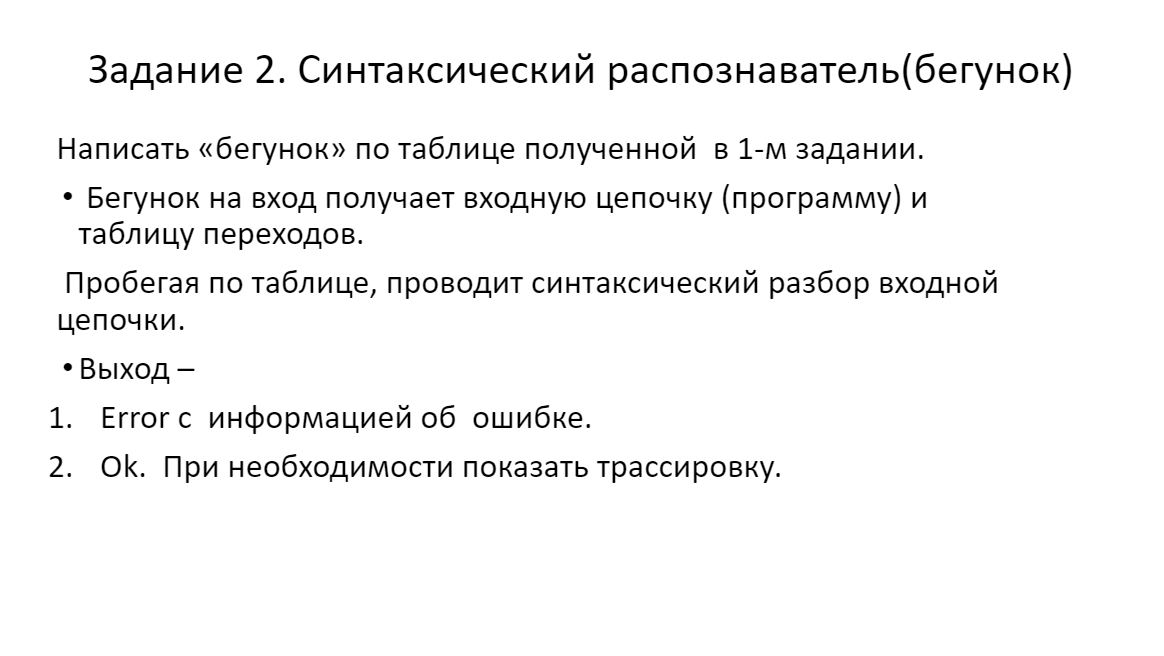
[ПРИЛОЖЕНИЯ](#_heading=h.3dy6vkm) **9**

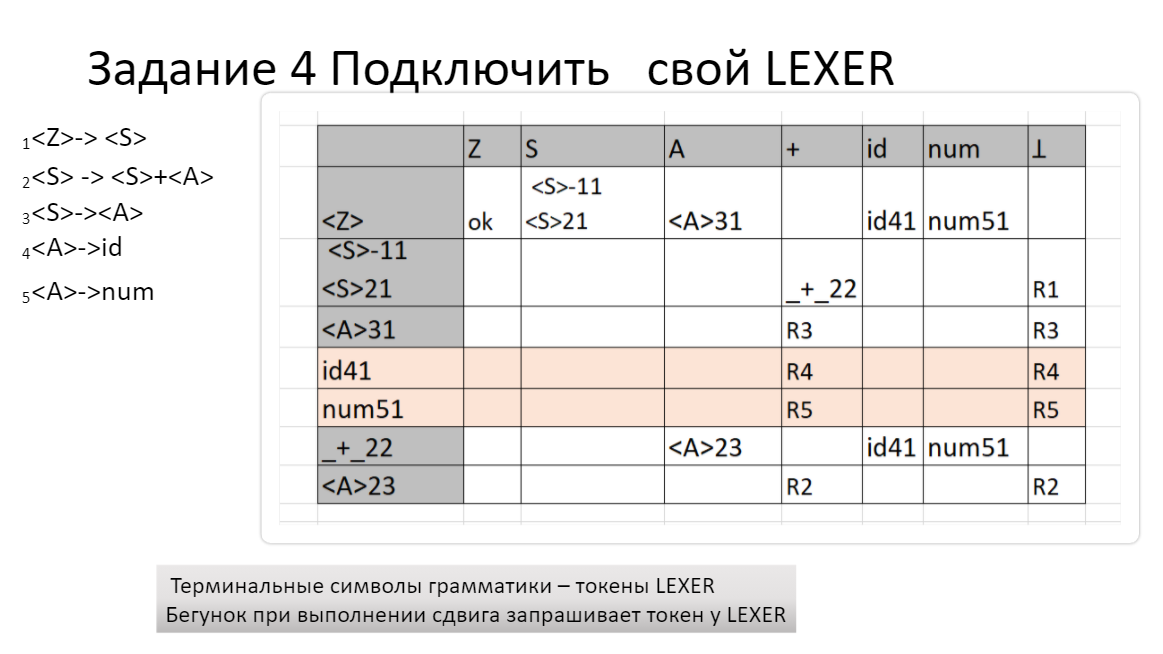
[**Приложение 1**](#_heading=h.1t3h5sf) **9**

[Приложение 2](#_heading=h.vfavk8v774oi) 10

## СОДЕРЖАНИЕ ЗАДАНИЯ







РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАНИЯ

### Задание 1

Запускаем генератор с входной грамматикой на входе (grammar.txt)

<Z> => <S> $

<S> => ( <S> )

<S> => e

Вызываем: generator.exe grammar.txt table.txt

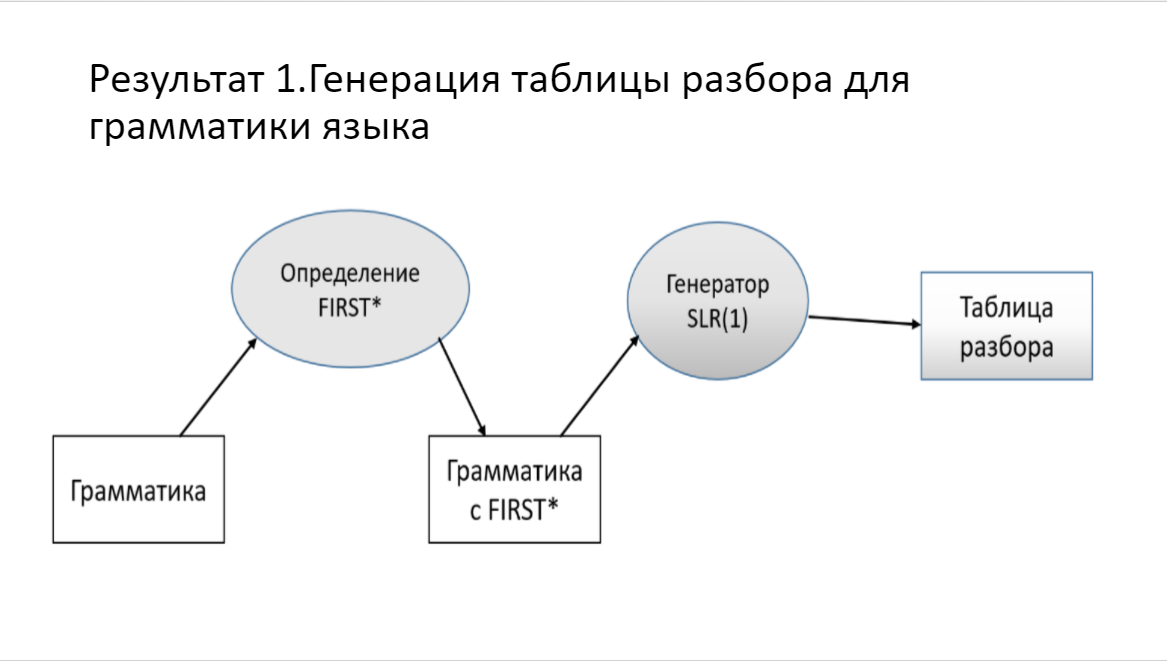
В выходном файле(table.txt) получаем таблицу

| <Z> | <S> | CloseParenthesis | OpenParenthesis | $ |
| --- | --- | --- | --- | --- |
| OK | 1 | - | 2 | <Z>|1 |
| - | - | - | - | <Z>|2 |
| - | 4 | 3 | 2 | - |
| - | - | <S>|2 | - | <S>|2 |
| - | - | 5 | - | - |
| - | - | <S>|3 | - | <S>|3 |

Таблица 1

Числа рядом с нетерминальн. символом явл. количеством элементов для удаления из стека, а без - правилом перехода

Методы в приложении 1



### Задание 2

на вход подадим цепочку ( ( ) ) (input.txt)

1. stack [0]  
   cursor at element 1 ( ( ) )  
   position 0
2. stack [ 0, 0 ]

cursor at element 2 ( ( ) )

position 2

1. stack [ 0, 0, 2 ]

cursor at element 3 ( ( ) )

position 2

1. stack [ 0, 0, 2, 2 ]

cursor at element 4 ( ( ) )

position 3

1. stack [ 0, 0, 2 ]

cursor at element 4 ( ( ) )

position 4

1. stack [ 0, 0, 2, 4 ]

cursor at element 5 ( ( ) ) \_

position 5

1. stack [ 0 ]

cursor at element 5 ( ( ) ) \_

position 0

1. stack [ 0, 0 ]

cursor at element 5 ( ( ) ) \_

position 1

1. stack [ 0, 0 ]

cursor at element 5 ( ( ) ) \_

position 1

1. stack [ ]

cursor at element 5 ( ( ) ) \_

position 0 (transfer throw Z)

Well Done!

Запускаем SyntaxAnalyzer.exe table.txt input.txt

Получаем сообщение об успешном проходе по таблице (table.txt)

Методы в приложении 2

### Задание 4

Лексер используется в 1 таблице при наименовании скобок

Метод, подключающий лексер

Grammar LexemizeGrammar(const Grammar& grammar)

{

auto grammarCopy(grammar);

if constexpr (Settings::USE\_LEXER)

{

for (auto& [left, right] : grammarCopy)

{

std::transform(right.cbegin(), right.cend(), right.begin(), [](const auto& ch) {

return IsNonTerminal(ch)

? ch

: (IsEndRule(ch) ? ch : LexemeTypeToString(ClassifyLexeme(ch)));

});

}

}

return grammarCopy;

}

## ВЫВОД

В течении семестра изучалась дисциплина ТАиФЯ, где были получены знания о синтаксических анализатора, методах разборов и сценариев использования, что в дальнейшем поможет в трудовой деятельности. Курс был непростым, но очень интересным. В подтверждение усвоения материала мною написан синтаксический анализатор типа SLR(1), и соединен вместе с Lexer, написанным нами ранее в предыдущей РГР.

## ПРИЛОЖЕНИЯ

### **Приложение** **1**

Table<std::optional<std::variant<Shift, Reduce>>> GetTableSLR(const std::vector<Rule>& grammar)

{

const auto chars = GetUniqueCharacters(grammar);

Table<std::optional<std::variant<Shift, Reduce>>> table(chars, [](const auto& cell) {

return cell

? std::visit([](auto&& arg) {

return ToString(arg);

}, \*cell)

: "-";

});

size\_t rowNum = 0;

auto mainColumn = std::make\_shared<std::vector<std::set<std::pair<size\_t, size\_t>>>>();

auto transitions = ColdStart(grammar);

TransitionsToTable(grammar, chars, mainColumn, transitions, table, rowNum);

transitions.clear();

auto nextToProcess = GetNextToProcess(table, mainColumn);

do

{

for (const auto& next : nextToProcess)

{

ProcessNext(next, grammar, transitions);

mainColumn->emplace\_back(next);

TransitionsToTable(grammar, chars, mainColumn, transitions, table, rowNum);

transitions.clear();

}

nextToProcess = GetNextToProcess(table, mainColumn);

} while (!nextToProcess.empty());

return table;

### Приложение 2

void ValidateSentence(const std::vector<std::string>& characters, const Table<std::optional<std::variant<Shift, Reduce>>>& table, const std::vector<Lexeme>& lexemes)

{

std::list<Lexeme> sentence(lexemes.cbegin(), lexemes.cend());

const auto axiom = characters.front();

std::unordered\_map<std::string, size\_t> getChIndex;

for (size\_t i = 0; i < characters.size(); ++i)

{

getChIndex[characters[i]] = i;

}

auto currData = table[0];

std::stack<std::remove\_reference<decltype(table)>::type::Row> st;

st.push(currData);

while (!sentence.empty() || !st.empty())

{

if (st.empty() && (sentence.size() == 2) && (PrecariousLexeme(sentence.front()) == characters.front()) && (PrecariousLexeme(sentence.back()) == characters.back()))

{

break;

}

auto currLexeme = sentence.front();

sentence.pop\_front();

if (const auto& cell = currData[getChIndex[PrecariousLexeme(currLexeme)]]; cell)

{

std::visit([&table, &currData, &sentence, &st, &currLexeme](auto&& arg) {

using T = std::decay\_t<decltype(arg)>;

if constexpr (std::is\_same\_v<T, Shift>)

{

currData = st.emplace(table[arg.pointer]);

std::cout << "transit " << currData[0]. << "\n";

}

else if constexpr (std::is\_same\_v<T, Reduce>)

{

for (size\_t i = 0; i < arg.len; ++i) st.pop();

sentence.push\_front(currLexeme);

sentence.push\_front({ LexemeType::Error, "<" + arg.ch + ">", currLexeme.lineNum, currLexeme.linePos });

currData = st.empty() ? table[0] : st.top();

}

}, \*cell);

}

else

{

throw std::runtime\_error(std::string("[") + ToString(currLexeme.lineNum) + ":" + ToString(currLexeme.linePos) + "] Unexpected lexeme");

}

}

}