МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

Лабораторная работа №3

**по дисциплине «Теория формальных языков и компиляторов»**

«Лексика языков программирования. Конечные автоматы без памяти для обнаружения слов в тексте программы»

Вариант № 11213131

Группа: АВТ-709

Студент: Антонов К. О.

Преподаватель: Малявко А.А.

Новосибирск, 2020 г.

Оглавление

[Цель работы 3](#_Toc37006757)

[Задание 3](#_Toc37006758)

[Дополнения к описанию языка 4](#_Toc37006759)

[Обновленное описание конечного автомата 6](#_Toc37006760)

[Обработка фрагмента тестового примера 7](#_Toc37006761)

[Краткое описание метаязыка формальных грамматик 8](#_Toc37006762)

[Свойства грамматик из примеров 10](#_Toc37006763)

[Часть грамматики языка для курсовой работы 15](#_Toc37006764)

[Вывод 15](#_Toc37006765)

# Цель работы

Изучение основных понятий метаязыка формальных грамматик, свойств грамматик и нетерминальных символов, рекурсивности и однозначности грамматик, недостижимости, бесплодности, аннулируемости и рекурсивности нетерминальных символов, отношений предшествования и последования между символами, приобретение навыков эквивалентных преобразований формальных грамматик, освоение технологий разработки формальных грамматик.

# Задание

* Изучить интерфейс пакета ВебТрансЛаб, предназначенный для работы с синтаксическими правилами.
* На примерах 3LL1 и 4SLR1GrammarArithmExpr (грамматики ***Ga1*** и ***Ga2*** из учебника) изучить и освоить:

-  ввод и редактирование правил грамматики;

-  построение синтаксических акцепторов с целью вычисления свойств грамматики и ее символов – пункт меню «Построить» с выбором нужного шаблона;

-  понятия множеств предшественников и множеств последователей для нетерминальных символов;

-  понятие множеств выбора правил грамматики и их пересечений для правил с одинаковыми нетерминалами в левой части;

-  просмотр свойств символов грамматики (следует достичь понимания того, почему те или иные символы грамматики имеют свой конкретный набор свойств – пункт меню «Показать» и подпункты «Правила грамматики», «Отношение предшествования», «Отношение последования» и «Множества выбора правил».

* Изучить способы эквивалентных преобразований грамматики из примера 4SLR1GrammarArithmExpr для устранения пересечений множеств выбора правил с одинаковыми нетерминалами в левой части, т.е. превращения этой грамматики в LL(1)-грамматику.
* Ориентируясь на свой вариант задания на курсовую работу, модифицировать (расширить) грамматики  ***Ga1*** и ***Ga2***  до грамматик операторов присваивания, условного оператора и оператора цикла, добавляя новые нетерминалы и правила по технологии разработки «снизу вверх». Изучить свойства полученных грамматик и их символов.
* Используя полученные навыки работы с грамматиками и программным обеспечением, начать поэтапную разработку грамматики по технологии «сверху вниз» для заданного варианта курсовой работы.
* Оценить свои субъективные проблемы и трудности разработки формальных грамматик при использовании разных технологий («снизу вверх» и «сверху вниз»). Выбрать технологию для выполнения задания на курсовую работу (рекомендуется технология «сверху вниз»).
* Подготовить, сдать и защитить отчет к лабораторной работе.
* Требования к содержанию отчета.

Отчет должен содержать:

-  цель работы;

-  краткое описание метаязыка формальных грамматик;

-  свойства грамматик из примеров и символов этих грамматик;

-  матричное представление отношений предшествования и последования для символов изученных грамматик и описание того, для чего нужны эти отношения;

-  расширение одной из изученных грамматик до части языка, заданного на курсовую работу, включающего операторы присваивания, условные операторы и операторы цикла;

-  краткое описание разработанной грамматики, как часть расчетно-пояснительной записки к курсовой работе;

-  выводы и заключение.

# Дополнения к описанию языка

Программа на данном языке может состоять из нескольких файлов. Чтобы добавить файл в проект, необходимо использовать следующую команду: *include <file> .*

В данном языке передача аргументов в функции происходит всегда по

ссылке. Это позволяет манипулировать самим объектом, а не только его значением. Передача по ссылке позволяет возвращать из функции сразу несколько значений.



Рисунок 1 – Обновленные правила лексики

# Обновленное описание конечного автомата

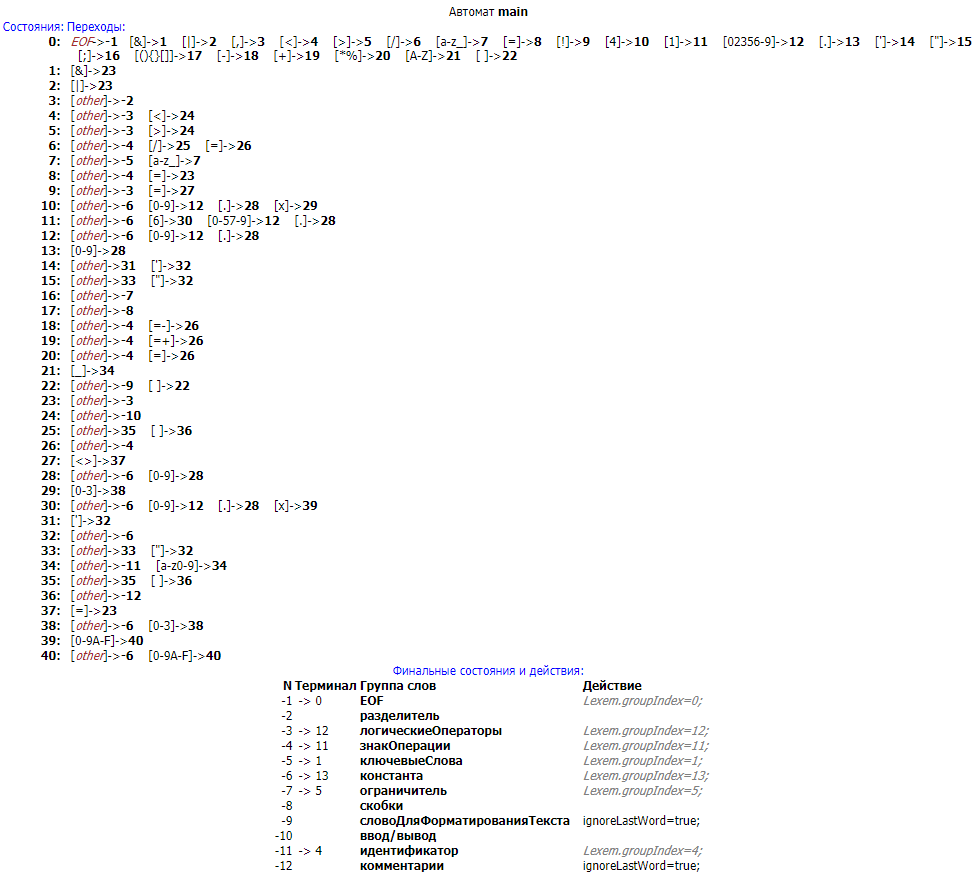


Рисунок 1 – Конечный автомат, заданный графом

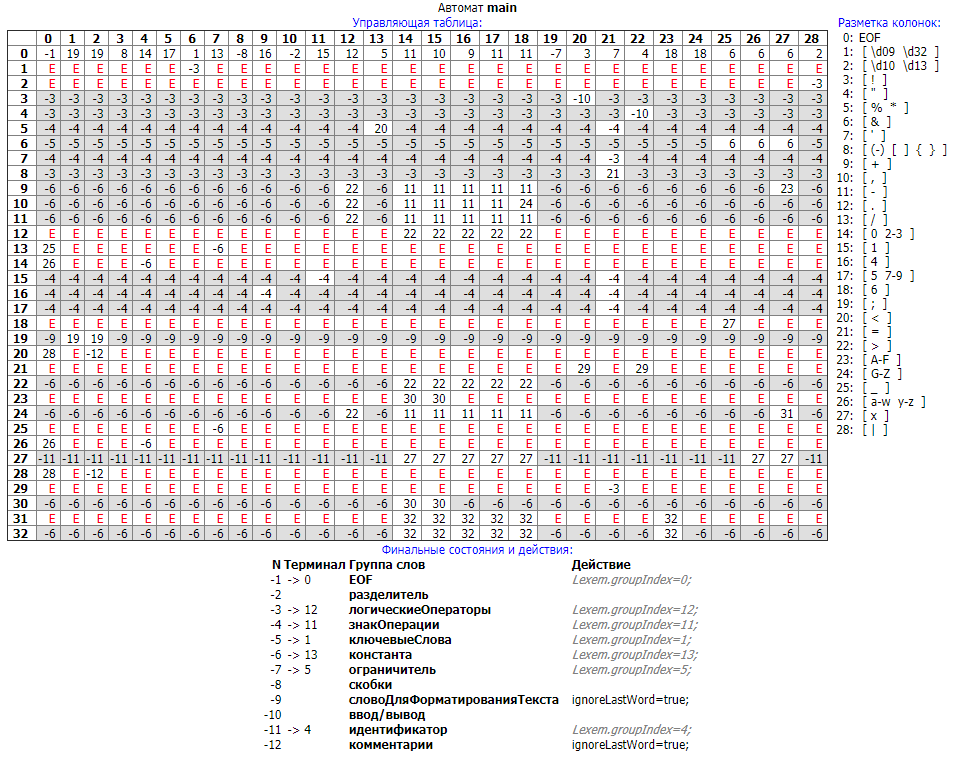


Рисунок 2 – Конечный автомат, заданный таблицей

# Обработка фрагмента тестового примера

*const int I\_size16 = 16xA;*

Конечный автомат, заданный графом:

0 → 7 → 7 → 7 → 7 → 7 → -5 (ключевыеСлова)

0 → 22 → -9 (словоДляФорматированияТекста)

0 → 7 → 7 → 7 → -5 (ключевыеСлова)

0 → 21 → 34 → 34 → 34 → 34 → 34 →34 → 34 → -11 (идентификатор)

0 → 22 → -9 (словоДляФорматированияТекста)

0 → 8 → -4 (знакОперации)

0 → 22 → -9 (словоДляФорматированияТекста)

0 → 11 → 30 → 39 → 40 → -6 (константа)

0 → 16 → -7 (ограничитель)

Конечный автомат, заданный таблицей:

0 → [0][26] → [6][26] → [6][26] → [6][26] → [6][26] → [6][0] → -5 (ключевыеСлова)

0 → [0][1] → [19][0] → -9 (словоДляФорматированияТекста)

0 → [0][26] → [6][26] → [6][26] → [6][0] → -5 (ключевыеСлова)

0 → [0][24] → [18][25] → [27][26] → [27][26] → [27][26] → [27][26] → [27][15] → [27][15] → [27][0] → -11 (идентификатор)

0 → [0][1] → [19][0] → -9 (словоДляФорматированияТекста)

0 → [0][21] → [7][0] → -4 (знакОперации)

0 → [0][1] → [19][0] → -9 (словоДляФорматированияТекста)

0 → [0][15] → [10][18] → [24][27] → [31][23] → [32][0] → -6 (константа)

0 → [0][19] → -7 (ограничитель)

# Краткое описание метаязыка формальных грамматик

Формальной грамматикой *G* называется совокупность

состоящая из:

* алфавита терминальных символов ;
* алфавита нетерминальных символов ;
* начального нетерминального символа S;
* системы правил подстановки P.

*Алфавит терминальных символов* , есть конечное множество всех слов языка, порождаемого данной грамматикой. Понятие «терминальный» в данном случае обозначает неразложимость таких символов с точки зрения синтаксических правил.

*Алфавит нетерминальных символов* есть конечное множество названий синтаксических конструкций, например: <предложение>, <выражение>, <список аргументов>, <условный оператор>, <тело функции>. Нетерминальные символы используются только в метаязыке, на котором описывает язык программирования.

*Начальный нетерминальный символ* S есть один из нетерминальных символов. Этим символом обычно обозначается наиболее общая синтаксическая конструкция, например: <правильная программа>.

*Система правил подстановки* P (система порождающих правил или продукций) есть конечное множество пар цепочек вида α : β, причем цепочка α (левая часть правила) должна содержать хотя бы один нетерминальный символ.

Все грамматики можно разделить на четыре основных класса по Хомскому:

1. Класс 0 – *общие*. К этому классу относятся грамматики, на порождающие правила которых не накладывается дополнительных ограничений, кроме сформулированных ранее при определении понятия грамматик.
2. Класс 1 – *контекстно-зависимые*. К этому классу относятся грамматики, у которых длина цепочки символов левой части каждого правила не больше длины цепочки правой части.
3. Класс 2 – *контекстно-свободные*. К этому классу относятся грамматики, у которых левая часть каждого порождающего правила состоит из одного нетерминального символа.
4. Класс 3 – *регулярные*. Грамматики этого класса содержат правила, левая часть каждого из которых – один нетерминал, а правая часть содержит не более одного нетерминала. Другое определение: в правой части может находиться либо пустая цепочка, либо один терминал и один нетерминал.

В данной работе используются контекстно-свободные грамматики, позволяющие определять синтаксис формальных языков, в том числе языков программирования. Под синтаксисом понимается совокупность правил построения предложений языка из слов языка.

Независимость от контекста означает возможность подстановки правой части любого из правил для замены нетерминала из левой части правила в любой цепочке, содержащей этот нетерминал. Отсюда следует невозможность включения смысловых ограничений на построение синтаксически правильных конструкций в рамках контекстно-свободных грамматик.

Множество предшественников – множество символов, с которых начинаются правила, выводимые из заданного нетерминала. Множество предшественников используется для восстановления дерева синтаксического разбора «сверху вниз»: на каждом шаге синтаксического анализа множество предшественников определяет, возможна ли замена текущего нетерминала на соответствующее ему правило (и выбор правила, если таковых несколько). Аналогично для восходящего акцепта используется множество последователей – множество символов, непосредственно следующих за данным нетерминалом.

Множество выбора – множество терминальных символов, содержащее первые символы, с которых начинается правая часть правил для заданного нетерминала.

# Свойства грамматик из примеров

Пример 3LL1.xml

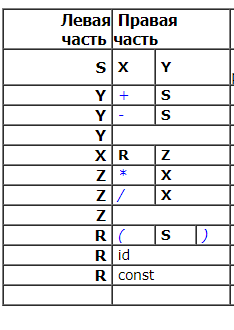
**

Рисунок 3 – Правила грамматики языка

**

Рисунок 4 – Множество выбора правил языка

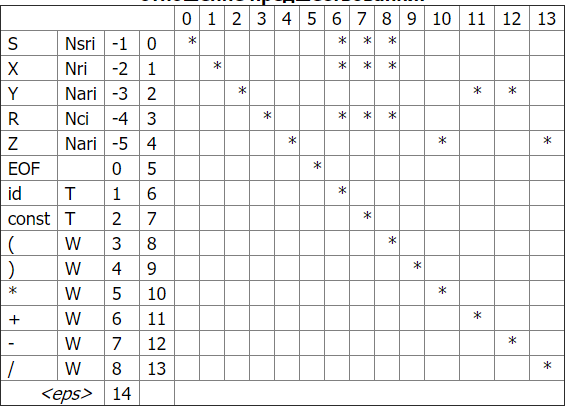


Рисунок 5 – Множество предшественников языка

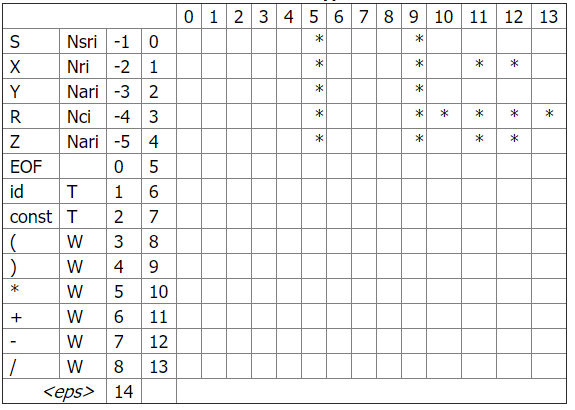


Рисунок 6 – Множество последователей языка

Система правил данного языка не содержит бесплодных и недостижимых правил, но содержит два аннулируемых правила (Z и Y). Множества выбора данного языка попарно не пересекаются для всех нетерминалов, поэтому язык принадлежит классу LL1 пригоден для построения как восходящего, так и нисходящего синтаксического акцептора.

Пример 4SLR1grammarArithmExpr.xml



Рисунок 7 – Правила грамматики языка



Рисунок 8 – Множество выбора правил языка

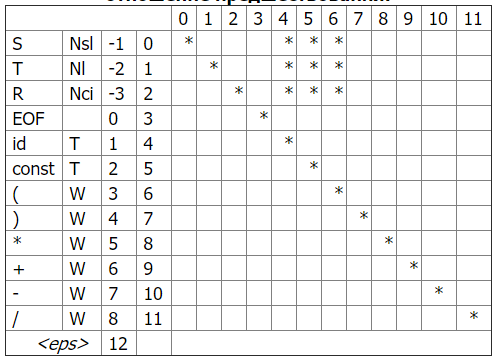
**

Рисунок 9 – Множество предшественников языка

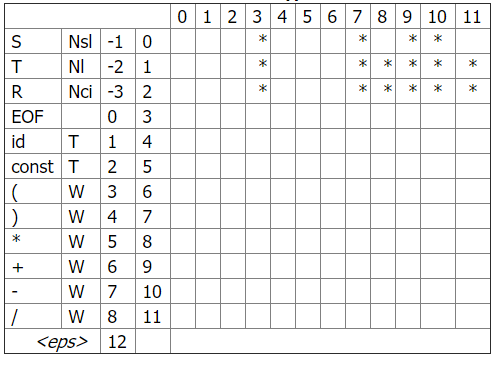
**

Рисунок 10 – Множество последователей языка

Множества данного языка имеют пересечения, поэтому грамматика не является LL1 и не пригодна для нисходящего безоткатного восстановления дерева разбора.

# Часть грамматики языка для курсовой работы

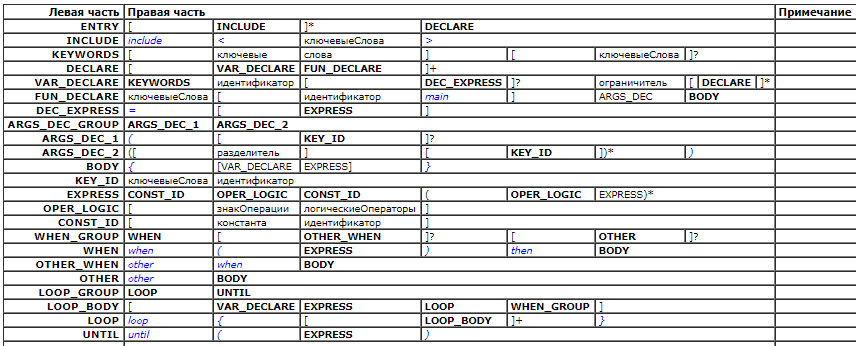


Рисунок 11 – Часть грамматики языка

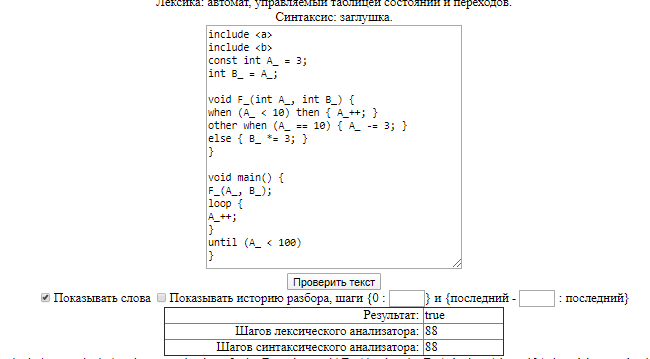


Рисунок 12 – Результат работы лексического и синтаксического анализатора

# Вывод

В результате выполнения лабораторной работы были изучены основные понятия метаязыка формальных грамматик, а также было доработано описание языка. Были продемонстрированы истории работы графового и табличного автоматов при обработке фрагмента тестового примера.