Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 13

По дисциплине «Конструирование программного обеспечения»

На тему «Подготовка к разработке лексического распознавателя»

Выполнил:

Рауба Арсений

Студент 1 курса 8 группы

Преподаватель: Север А.С.

Минск, 2024

Вариант №9

if(□)+((const+ab+count)□+)\* □\*{return};

3. Примеры цепочек:

if (const ab count )return

if(const ab)return

if(ab count)return

if(ab)return

if(ab ab )return

if(count)return

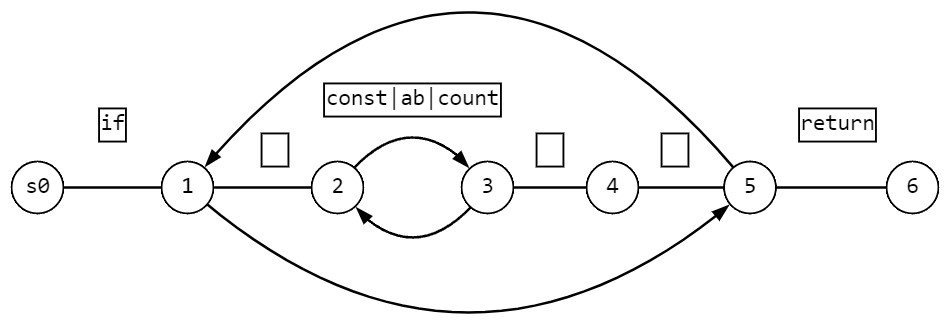
if(const const ab ab count)return

4. Диаграмма мгновенных состояний конечного автомата:

if const return → if return → return → λ

5. Граф конечного автомата

if = a; □ = b; const = c; ab = d; count = e; return = f

a(b) + ((c + d + e) b +) \* b \* f

**Что такое алфавит 𝐼?**

Алфавит ***I*** — это конечное множество символов, используемых для построения цепочек или строк. Например, для двоичного алфавита 𝐼={0,1}.

**Обозначения**

* 𝜆: обозначает пустую строку, то есть строку, не содержащую ни одного символа.
* ∗: оператор Клини, обозначает множество всех возможных строк (включая пустую строку), которые можно составить из символов алфавита 𝐼. Например, 𝐼∗ — это множество всех строк над алфавитом 𝐼.
* 𝐼+: множество всех непустых строк, которые можно составить из символов алфавита 𝐼. То есть, 𝐼+=𝐼∗∖{𝜆}.
* 𝐼𝑛: множество всех строк длины 𝑛, составленных из символов алфавита 𝐼.

**Что такое язык 𝐿(𝐼) над алфавитом 𝐼?**

**Язык** 𝐿(𝐼) — это множество строк, составленных из символов алфавита 𝐼. Например, для 𝐼={𝑎,𝑏}*,*  язык может быть 𝐿(𝐼)={𝑎,𝑏,𝑎𝑏,𝑏𝑎,𝑎𝑎𝑏,𝑏𝑏𝑎,…}*.*

**Определение формальной грамматики 𝐺**

**Формальная грамматика** 𝐺 — это кортеж 𝐺=(𝑁,𝑇,𝑃,𝑆) где:

* 𝑁 — конечное множество нетерминальных символов,
* 𝑇 — конечное множество терминальных символов (алфавит),
* 𝑃 — конечное множество правил продукций (переписей),
* 𝑆 — стартовый символ (нетерминал), откуда начинается генерация языка.

**Обозначения 𝛼⇒𝛽 и 𝛼⇒∗𝛽 для цепочек символов 𝛼 и 𝛽**

* 𝛼⇒𝛽: существует одно применение правила продукции, которое преобразует строку 𝛼 в строку 𝛽.
* 𝛼⇒∗𝛽: существует конечная последовательность применений правил продукции, которая преобразует строку 𝛼 в строку 𝛽.

**Что такое язык 𝐿(𝐺), порождаемый грамматикой 𝐺?**

Язык 𝐿(𝐺), порождаемый грамматикой 𝐺, — это множество всех строк, которые можно вывести из начального символа 𝑆 с использованием правил грамматики 𝑃. Формально, если 𝐺=(𝑁,𝑇,𝑃,𝑆), то 𝐿(𝐺) — это множество всех строк, состоящих из терминалов 𝑇, которые могут быть получены из 𝑆 с применением последовательности продукций из 𝑃.

**Что такое форма Бэкуса-Наура?**

ФормаБэкуса-Наура (BNF, Backus – Naur Form) —это нотация для описания синтаксиса языков программирования и других формальных языков. В БНФ используются правила продукции для определения структуры строк в языке. Правила продукции записываются в виде <𝑛𝑜𝑛𝑡𝑒𝑟𝑚𝑖𝑛𝑎𝑙>::=<𝑒𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛>, где <𝑛𝑜𝑛𝑡𝑒𝑟𝑚𝑖𝑛𝑎𝑙> — это нетерминальный символ, а <𝑒𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛> — последовательность терминальных и нетерминальных символов.

**Поясните понятие «регулярная грамматика»**

Регулярная грамматика — это тип формальной грамматики, правила которой ограничены определенными формами. Правила продукции регулярной грамматики имеют одну из следующих форм:

* 𝐴→𝑎
* 𝐴→𝑎𝐵

где 𝐴 и 𝐵 — нетерминальные символы, а 𝑎 — терминальный символ. Регулярные грамматики могут описывать только регулярные языки.

**Поясните понятие «регулярное множество»**

Регулярное множество — это множество строк, которое может быть описано регулярным выражением, распознано конечным автоматом или описано регулярной грамматикой. Примеры включают множества строк, которые соответствуют определенным шаблонам, например, строки, содержащие только символы 'a' и 'b'.

**Поясните понятие «регулярный язык»**

Регулярный язык — это язык, который можно описать с помощью регулярного выражения, регулярной грамматики или распознать конечным автоматом. Регулярные языки включают, например, множества строк, которые соответствуют шаблону, определенному регулярным выражением.

**Что такое лексический анализ?**

Лексический анализ — это процесс разбивки входной строки на лексемы (токены), которые являются минимальными значимыми элементами языка. Лексический анализ является первым этапом компиляции, где исходный код программы преобразуется в токены.

**Что такое лексический анализатор?**

Лексический анализатор (лексер) — это программа или компонент компилятора, который выполняет лексический анализ, преобразуя входную строку в последовательность токенов. Лексический анализатор читает исходный код и выделяет из него ключевые слова, идентификаторы, литералы и другие значимые элементы.

**Входная и выходная информация для лексического анализатора**

* Входная информация: исходный текст программы или строки.
* Выходная информация: последовательность токенов (лексем), каждый из которых представляет собой элемент исходного текста с определенным значением и типом.

**Различия между последовательным и параллельным лексическими анализаторами**

* Последовательный лексический анализатор обрабатывает входную строку последовательно, по одному символу за раз. Он анализирует строку линейно, от начала до конца.
* Параллельный лексический анализатор может обрабатывать несколько частей входной строки одновременно, используя параллельные вычисления для увеличения скорости анализа. Он разбивает входную строку на сегменты и анализирует их независимо друг от друга.

**Определение регулярного выражения над алфавитом 𝐼**

Регулярное выражение над алфавитом 𝐼 — это формальное описание множества строк, составленных из символов 𝐼. Регулярные выражения используют символы и операторы (например, конкатенация, альтернация, звезда Клини) для описания паттернов строк. Примеры операторов включают:

* Конкатенация: 𝑎𝑏 означает строку, состоящую из 'a', за которой следует 'b'.
* Альтернация: 𝑎∣𝑏 означает строку, содержащую либо 'a', либо 'b'.
* Звезда Клини: 𝑎∗ означает строку, содержащую ноль или более символов 'a'.

**Определение конечного автомата 𝑀=(𝑆,𝐼,𝛿,𝑠0,𝐹)**

Конечный автомат 𝑀 определяется как кортеж (𝑆,𝐼,𝛿,𝑠0,𝐹), где:

* 𝑆 — конечное множество состояний,
* 𝐼 — алфавит,
* 𝛿 — функция переходов 𝛿:𝑆×𝐼→𝑆,
* 𝑠0​ — начальное состояние,
* 𝐹 — множество конечных состояний.

**Отличие между детерминированным и недетерминированным автоматом**

* Детерминированный конечный автомат (DFA) имеет в каждом состоянии ровно один переход для каждого символа алфавита.
* Нетерминированный конечный автомат (NFA) может иметь несколько переходов для одного символа или переходы без символов (ε-переходы). NFA может находиться в нескольких состояниях одновременно.

**Мгновенное состояние конечного автомата**

Мгновенное состояние конечного автомата — это пара (𝑠,𝑤), где 𝑠 — текущее состояние, а 𝑤 — оставшаяся часть входной строки. Это состояние описывает текущее положение автомата в процессе обработки входной строки.

**Поясните обозначения (𝑠,𝑎𝑤)⊢(𝑠′,𝑤) и (𝑠𝑖,𝑤)⊢∗(𝑠𝑘,𝑤𝑘)**

* (𝑠,𝑎𝑤)⊢(𝑠′,𝑤) означает, что автомат в состоянии 𝑠 при чтении символа 𝑎 и строки 𝑤 переходит в состояние 𝑠′ и остается с оставшейся строкой 𝑤.
* (𝑠𝑖,𝑤)⊢∗(𝑠𝑘,𝑤𝑘) означает, что автомат переходит из состояния 𝑠𝑖​ в состояние 𝑠𝑘 за несколько (возможно, ноль) шагов, обрабатывая строку 𝑤 и оставаясь с остатком строки 𝑤𝑘.

**Соотношение между регулярной грамматикой, регулярным языком, регулярным выражением, конечным автоматом и графом состояний конечного автомата**

Регулярная грамматика, регулярный язык, регулярное выражение, конечный автомат и граф состояний конечного автомата тесно связаны между собой и описывают одни и те же классы языков — регулярные языки:

* Регулярные выражения могут быть преобразованы в конечные автоматы (DFA или NFA) и наоборот.
* Регулярные грамматики описывают те же языки, что и конечные автоматы.
* Граф состояний конечного автомата визуально представляет переходы между состояниями при обработке входных символов.
* Любой регулярный язык может быть описан регулярным выражением, распознан конечным автоматом или описан регулярной грамматикой.