**Лекция 18**

**БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования**

**Лексический анализатор: проектирование, принципы построения и реализации**

Цель: получение практических навыков построения лексического анализатора на примере заданного простейшего входного языка

Процесс компиляции представляет собой последовательность фаз, каждая из которых преобразует одно из представлений исходной программы в другое.

На первой фазе лексический анализатор (сканер) читает входной поток символов, составляющих исходную программу, и разбивает ее на составные части, накладывает на них грамматическую структуру для создания промежуточного представления исходной программы (таблица лексем).

Общая схема работы лексического анализатора.

Лексический анализатор выполняет предварительную обработку текста, разбивая его на фразы. Входные символы, составляющие отдельную фразу (токен), называется лексемой. Примеры лексем: числа (десятичные целые, вещественные), строки символов, идентификаторы, ключевые слова, разделители, знаки операций.

Каждой лексеме сопоставляется ее тип и запись в таблице идентификаторов, в которой хранится дополнительная информация. Например, data– лексема, являющаяся идентификатором в языке SVV-2015. Обозначаем ее символом i. В таблице идентификаторов создадим запись для этой лексемы, сохранив имя идентификатора, тип и другую дополнительную информацию (например, отметим место его первого появления, определим область видимости).

Таблица лексем (ТЛ) и таблица идентификаторов (ТИ) являются входом для следующей фазы компилятора – синтаксического анализа (разбора, парсера).

1. Задачей лексического анализатора является замена элементов текста стандартными лексемами, которые в дальнейшем будет легче обрабатывать в других частях компилятора. Дополнительно лексический анализ выполняет исключение незначащих фрагментов текстов программ, например, комментариев, незначащих пробелов. Для тех языков, в которых имеются макросредства, дополнительно выполняется расширение макровызовов.

В лабораторной работе №14 приведена спецификация языка SVV-2015. В таблице п.6 дано соответствие фраз входного языка лексемам. Каждая лексема имеет свой тип.

Все символы входной последовательности разделяются на символы, принадлежащие каким-либо лексемам, и символы, разделяющие лексемы (разделители). Лексема, соответствующая ограничителю является результатом лексического анализа и заносится в ТЛ. Если выделенная лексема является ключевым словом, то выдается признак (тип лексемы) соответствующего ключевого слова. Если выделенная лексема является идентификатором ‑ выдается признак идентификатора, а сам идентификатор сохраняется в ТИ. Если выделенная лексема принадлежит какому-либо другому типу лексем (например, лексема представляет собой число, строку и т.д.), то выдается лексема соответствующего типа, а значение лексемы и другая дополнительная информация сохраняется в ТИ.

Если в процессе работы лексический анализатор не смог правильно определить тип лексемы, считается, что программа содержит ошибку. Информация об ошибке выдается пользователю.

Взаимодействие лексического и синтаксического анализаторов может быть последовательным и параллельным.

2. Последовательное взаимодействие лексического и синтаксического анализаторов.



3. Параллельное взаимодействие лексического и синтаксического анализаторов.



4. Для описания лексики языка программирования обычно применяются регулярные грамматики. С точки зрения лексического анализатора – язык программирования это набор лексем (токенов), которые распознаются (классифицируются) лексическим анализатором по шаблонам, описывающим вид соответствующих лексем. Множество слов, или строк символов, которые соответствуют шаблону, называется языком. Для описания шаблонов используются регулярные выражения. Язык программирования (на уровне лексического анализа) представляет собой регулярный язык, заданный регулярным выражением (язык типа 3 в иерархии Хомского).

Основной задачей лексического анализатора является разбиение входного текста, состоящего из последовательности символов алфавита, на минимально значимые части языка, называемые лексемами. Вторая задача лексического анализатора заключается в выполнении действий, связанных с обнаружением и распознаванием той или иной лексемы. Математический аппарат, лежащий в основе лексического анализатора – теория формальных языков и конечных автоматов. Автомат распознает правильную лексему на входе и выполняет такие действия, как запись выделенной лексемы в таблицу лексем, поиск ее в таблице идентификаторов, запись нового имени или новой константы в ТИ.

**Напоминание.**

1. Грамматика описывает множество правильных цепочек символов над заданным алфавитом. Как правило, для описания регулярных языков не применяют грамматики в виду громоздкости записи, а используют другую форму – регулярные выражения.
2. Регулярное выражение описывает множество цепочек – формальный язык. Для записи регулярного выражения используются метасимволы. Множество цепочек описанных регулярным выражением называется **регулярным множеством** (или регулярным языком).
3. Пусть - алфавит. Регулярные выражения над  и языки, представляемые ими, рекурсивно определяются следующим образом.
4.  - регулярное выражение и представляет пустое множество;
5.  – регулярное выражение и представляет множество ;
6. для каждого  символ  является регулярным выражением и представляет множество ;
7. если  - регулярное приложение, представляющее множество , если  - регулярное приложение, представляющее множество , то , ,  являются регулярными выражениями и представляют множества ,  (конкатенация множеств)и  соответственно.
8. 

Например.

Для каждой лексемы из лабораторной работы 14 надо разработать регулярное выражение. Пример: для лексемы t регулярное выражение integer соответствует строке integer.

При описании лексических структур полезно вводить имена некоторым регулярным выражениям и использовать эти имена для ссылки на них по имени.

Пример использования имен для регулярных выражений.

* Регулярное выражение для множества идентификаторов.

|  |  |
| --- | --- |
|  | Letter = a|b|c|...|x|y|z|\_ |
|  | Digit = 0|1|...|9 |
|  | Identifier = Letter(Letter|Digit)\* |

1. Символы, применяемые для описания регулярных выражений, называются **метасимволами** или **символами-джокерами**. В описанном выше языке джокерами являются символы: .
2. Схема работы лексического анализатора



Распознаватель – это алгоритм, позволяющий определить некоторое множество (входной язык).

Простейший распознаватель состоит из *входной* *ленты* (входная цепочка символов), *управляющего устройства* с конечной памятью.

Работа конечного автомата представляет собой последовательность шагов (или тактов). Такт определяется текущим состоянием управляющего устройства и входным символом, обозреваемым в данный момент входной головкой. Сам шаг состоит из изменения состояния и, возможно, сдвига входной головки на одну ячейку вправо.

1. Класс алгоритмов, соответствующих схеме (см. выше), может быть записан в форме конечного автомата (KA).

Регулярные выражения, введенные ранее, служат для описания регулярных множеств. Для распознавания регулярных множеств служат конечные автоматы.

1. КА это пятерка , где

 - конечное множество состояний устройства управления;

- алфавит входных символов;

- функция переходов, отображающая  в множество подмножеств : ;

- начальное состояние устройства управления;

- множество заключительных (допускающих) состояний устройства управления.

Если  и , то конечный автомат **детерминированный** (ДКА) иначе - конечный автомат **недетерминированный** (НКА).

Недетерминированность автомата заключается в том, что, во-первых, находясь в некотором состоянии и обозревая текущий символ, автомат может перейти в одно из нескольких возможных состояний, и во-вторых, автомат может делать переходы по λ (пустой цепочке).

1. Мгновенное описание КА является пара , где - состояние КА, - неиспользованная часть входной цепочки.
2. - начальное мгновенное описание КА,  –анализируемая цепочка.
3. - допускающее мгновенное описание КА.
4. Если  и , где , , ,то - непосредственно следует.
5. Если , то - следует.
6. Если ,  - начальное состояние,  - конечное состояние, то цепочка  допускается (распознается) КА.
7. **Графом переходов** конечного автомата  называется ориентированный граф , где - множество вершин графа совпадает с множеством состояний конечного автомата,

– множество ребер (направленных линий, соединяющих вершины),

ребро , если .

Метка ребра  – все , для которых .



1. Конечный автомат может быть однозначно задан своим графом переходов.
2. Доказаны 4 утверждения:
3. язык является регулярным множеством тогда и только тогда, когда он задан регулярной грамматикой;
4. язык может быть задан регулярной грамматикой (левосторонней или правосторонней) тогда и только тогда, когда язык является регулярным множеством;
5. язык является регулярным множеством тогда и только тогда, когда он задан конечным автоматом;
6. язык распознается с помощью конечного автомата тогда и только тогда, когда он является регулярным множеством.

Другими словами: любой регулярный язык может быть задан регулярной грамматикой, регулярным выражением или конечным автоматом.

Другими словами: любой конечный автомат задает регулярный язык, а значит грамматику или регулярное выражение.

1. Доказана теорема (А. Ахо, Дж. Хопкрофт, Дж. Ульман): пусть  - регулярное выражение, тогда найдется недетерминированный конечный автомат , допускающий автомат, представленный , и обладающий следующими свойствами:

1);

2);

3) .

1. Построение графа конечного автомата по регулярному выражению.

Автомат для выражения строится композицией из автоматов, соответствующих подвыражениям. На каждом шаге построения строящийся автомат имеет в точности одно заключительное состояние; в начальное состояние нет переходов из других состояний и нет переходов из заключительного состояния в другие.

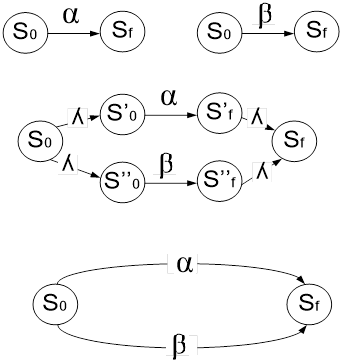
Автомат для выражения http://citforum.ru/programming/theory/serebryakov/msbm10-3f.gif, обозначающего множество http://citforum.ru/programming/theory/serebryakov/msbm10-3f.gif. Автомат для выражения λ. Автомат для выражения а.



Строим автомат для выражения α+β.

Здесь M’ и M’’ – автоматы для выражений α и β, S0’, S0’’ - новые начальные состояния и Sf’ и Sf’’ - новые заключительные состояния. Имеет место переход по λ из S0 в начальные состояния автоматов M’ и M’’ и переход по λ из заключительных состояний M’ и M’’ в Sf.

Автомат для выражения α+β:



1. Алгоритм для разбора с двумя массивами.



24. Программирование лексического анализатора.

Определение границ лексем.

Если говорить в терминах программы-сканера, то определение границ лексем ‑ это выделение тех строк в общем потоке входных символов, для которых надо выполнять распознавание. В общем случае эта задача может быть сложной, но для простейших входных языков границы лексем распознаются по заданным терминальным символам. Эти символы ‑ пробелы, знаки операций, символы комментариев, а также разделители (запятые, точки с запятой и др.). Набор таких терминальных символов может варьироваться в зависимости от входного языка. Важно отметить, что знаки операций сами также являются лексемами, и необходимо не пропустить их при распознавании текста.

Алгоритм работы простейшего сканера можно описать так:

1. проверяет входной поток символов программы на исходном языке на допустимость, удаляет лишние пробелы и добавляет сепаратор для вычисления номера строки для каждой лексемы;
2. для выбранной части входного потока выполняется функция распознавания лексемы;
3. при успешном распознавании информация о выделенной лексеме заносится в таблицу лексем и таблицу идентификаторов, и алгоритм возвращается к первому этапу;
4. формирует протокол работы;
5. при неуспешном распознавании выдается сообщение об ошибке, а дальнейшие действия зависят от реализации сканера - либо его выполнение прекращается, либо делается попытка распознать следующую лексему (идет возврат к первому этапу алгоритма).

Для каждой лексемы разрабатываем регулярное выражение, например для лексемы t регулярное выражение integer соответствует строке integer. Далее по заданию строим граф переходов и конечный автомат для этой и всех остальных лексем.

Выполняем п.7. задания. Текст программы на языке SVV-2015, символы которой проверены на допустимость, из текста удалены лишние пробелы и т.п., анализируется построенными конечными автоматами последовательно, и соответствующие лексемы заносятся в таблицу лексем. Дополнительную информацию для каждой лексемы заносим в таблицу идентификаторов.

На уровне лексического анализатора определяются только некоторые ошибки.