МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 12 2017г.

КОНСТРУИРОВАНИЕ КОМПИЛЯТОРОВ

Методические указания и задания к выполнению
лабораторных работ
для студентов направления 09.04.04

Курск 2017

УДК 004.2

Составители: В.В. Ефремов, И.Н. Ефремова

Рецензент
Кандидат технических наук, доцент *Е.И.Аникина*

Конструирование компиляторов: методические указания к выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: В.В.Ефремов, И.Н. Ефремова. Курск, 2017. - 16 с.: 3 табл.

Содержат описание четырех лабораторных работ по дисциплине «Конструирование компиляторов».

Предназначены для студентов направления 09.04.04

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл. печ. л. 1,8. Уч.-изд. л. 1,7. Тираж 100 экз. Заказ 48. Бесплатно
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Лабораторная работа №1.

Построение лексического анализатора: формирование таблицы лексем.

Цель работы: Ознакомление с назначением и принципами работы лексических анализаторов. Получение практических навыков построения сканеров.

Задачи: Научиться определять границы лексем, проектировать лексический анализатор.

Содержание отчёта

1. Титульный лист.
2. Формулировка задачи.
3. Диаграмма состояний.
4. Листинг программы сканера.
5. Таблица лексем для фрагмента обрабатываемой входной программы.
6. Ответы на контрольные вопросы.

Контрольные вопросы

1. Назначение лексического анализатора?
2. Роль сканера?
3. Как строится таблица лексем?
4. Использование конечных автоматов при лексическом анализе.
5. Что такое лексема?
6. Какие бывают лексемы?

Задание

Разработать сканер компилятора с подмножества одного из языков программирования. Написать программу, которая порождает таблицу лексем с указанием их предварительного типа. Программа должна выдавать сообщения о наличии ошибок (при их присутствии) во входном тексте.

Теоретические и справочные сведения

Транслятор – программа, которая переводит программу, написанную на одном языке, в эквивалентную ей программу, написанную на другом языке.

Компилятор – транслятор с языка высокого уровня на машинный язык или язык ассемблера.

Интерпретатор – программа, которая преобразует исходную программу и выполняет ее, не создавая программы на другом языке.

Процесс компиляции предполагает распознавание конструкций исходного языка (анализ) и сопоставление каждой правильной конструкции семантически эквивалентной конструкций другого языка (синтез). Он включает несколько этапов:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- распределение памяти;
- генерация и оптимизация объектного кода.

Лексический анализ – процесс преобразования исходного текста программы в строку однородных символов. Каждый символ результирующей строки – токен соответствует слову языка – лексеме и характеризуется набором атрибутов, таких как тип, адрес и т. п., поэтому строку токенов часто представляют таблицей, строка которой соответствует одному токену.

Существует 2 типа лексем:

а) лексемы, соответствующие символам алфавита языка, такие как «Служебные слова» и «Служебные символы»;

б) лексемы, соответствующие базовым понятиям языка, такие как «Идентификатор» и «Литерал».

Язык описания лексических единиц в большинстве случаев является регулярным и может быть описан с помощью регулярных грамматик. Распознавание регулярных грамматик удобней всего производить с помощью конечных автоматов. Работа сканера (начальный этап лексического анализа) заключается в моделировании конечных автоматов для распознавания различных лексем и их типов.

Для выделения и распознавания лексемы устанавливаем начальное состояние автомата, затем (пока не будет достигнуто конечное состояние или состояние «Ошибка») считываем очередной символ строки, анализируем его и переходим в соответствующее состояние автомата.

Результатом работы сканера является таблица лексем.

Пример. В Таблице 1 представлен результат разбора входной строки:

for i:=1 to 10 do x:=x+1;

Таблица 1. Таблица лексем.

Лексема	Предварительный тип
for	идентификатор
i	идентификатор
:=	разделитель
1	литерал
to	идентификатор
10	литерал
do	идентификатор
x	идентификатор
:=	разделитель
x	идентификатор
+	разделитель
1	литерал
;	разделитель

Лабораторная работа №2.

Построение лексического анализатора: формирование таблицы стандартных символов.

Цель работы: Ознакомление с назначением и принципами работы лексических анализаторов. Получение практических навыков построения сканеров.

Задачи: Классификация и сохранения информации об обнаруженной лексеме.

Содержание отчёта

1. Титульный лист.
2. Формулировка задачи.
3. Таблицы служебных символов, разделителей.
4. Листинг программы сканера.
5. Таблица лексем для фрагмента обрабатываемой входной программы с результатом классификации.
6. Ответы на контрольные вопросы.

Контрольные вопросы

1. Общий принцип работы лексического анализатора?
2. Принцип работы сканера?
3. Как строится таблица лексем и отнесение их к типам?
4. Какие таблицы используются при работе лексического анализатора?
5. Что такое таблица идентификаторов?
6. Какие бывают спецификации лексем?

Задание

Разработать сканер компилятора с подмножества одного из языков программирования. Написать программу, которая порождает таблицу лексем с классификацией типа.

Теоретические и справочные сведения

На этапе лексического анализа идентификаторы и константы, имеющие разную длину, заменяются символами одинаковой длины. Это позволяет облегчить работу на остальных этапах компиляции.

Сканер работает со следующими таблицами:

- таблица терминальных символов (служебных слов; разделителей);
- таблица идентификаторов (имен);
- таблица констант (литералов).

Таблицы идентификаторов и констант создаются в процессе работы сканера.

Таблица терминальных символов — это заранее определенная фиксированная таблица.

На этапе классификации лексем проводится поиск по указанным таблицам. Это долгий процесс, для ускорения которого используются различные методы поиска. Лучшие результаты можно достичь с использованием методов, связанных на использовании хэш-функций.

Результат работы сканера на этом этапе — таблица стандартных символов.

В Таблице 2 приведены сведения о таблице идентификаторов.

В таблице 3 приведены сведения о формируемой на данном этапе таблице стандартных символов для входной строки исходной программы:

for i:=1 to 10 do x:=x+1;

Таблица 2. Таблица идентификаторов.

Таблица служебных слов		Таблица разделителей	
1	var	1	;
2	int	2	,
3	boolean	3	:
4	begin	4	:=
5	end	5	+
6	for	6	*
7	to	7	<
8	do	8	<=
9	…	9	...

Таблица 3. Таблица стандартных символов.

Входные данные	Результат
for	1,6
i	4,1
:=	2,4
1	3,1
to	1,7
10	3,2
do	1,8
x	4,2
...	...

Лабораторная работа №3.

Построение синтаксического анализатора.

Цель работы: Ознакомление с назначением и принципами работы синтаксического анализатора. Получение практических навыков построения синтаксического анализатора.

Задачи: Изучить методы работы синтаксического анализатора.

Содержание отчёта

1. Титульный лист.
2. Формулировка задачи.
3. Грамматика подмножества языка.
4. Преобразования и выводы.
5. Граф состояний.
6. Детерминированная таблица действий.
7. Ответы на контрольные вопросы.

Контрольные вопросы

1. Что такое формальная грамматика?
2. Что такое КС- грамматика?
3. Понятия языка, алфавита...
4. Что такое БНФ?
5. Принцип нисходящего разбора?

Задание

Для выбранного в лабораторной работе №1 языка построить контекстно-свободную грамматику. Проверить принадлежность грамматики классу LL(k). Выполнить преобразование грамматик к классу LL(k). Определить k. Построить граф состояний и детерминированную таблицу действий.

Теоретические и справочные сведения

Синтаксический анализ – процесс распознавания конструкций языка в строке токенов.

Главным результатом является информация об ошибках в выражениях, операторах и описаниях программы.

Для написания программ используется формальная грамматика и формальный язык.

Предложения любого языка строятся из символов алфавита.

Алфавит – непустое конечное множество символов, используемых для записи предложений языка. Например, множество арабских цифр, знаки «+» и «-»: A = {0,1,2,3,4,5,6,7,8,9,+,-} – алфавит для записи целых десятичных чисел, например: «-365», «78», «+45».

Строка – любая последовательность символов алфавита, расположенных один за другим. Строки в теории формальных языков обозначаются строчными греческими буквами. Строка, содержащая 0 символов, называется пустой.

Множество всех составленных из символов алфавита A строк, в которое входит пустая строка, обозначают A*. Множество всех составленных из символов алфавита A строк, в которое не входит пустая строка, обозначают A+. Откуда: A* = A+ E {e}.

Формальным языком L в алфавите A называют произвольное подмножество множества A*. Таким образом, язык определяется как множество допустимых предложений.

Задать язык L в алфавите A значит либо перечислить все включаемые в него предложения, либо указать правила их образования. Перечисление бесконечно большого количества предложений невозможно. Поэтому задаются правила их вывода. Определение правил образования предложений осуществляют с использованием абстракций формальных грамматик.

Формальная грамматика – математическая система, определяющая язык посредством порождающих правил – правил продукции. Она определяется как четверка:

G = (VT, VN, P, S),

где VT – алфавит языка или множество терминальных (незаменяемых) символов;

VN – множество нетерминальных (заменяемых) символов – вспомогательный алфавит, символы которого обозначают допустимые понятия языка;

V – словарь грамматики;

P – множество порождающих правил,

S — начальный символ (аксиома) грамматики.

Для описания синтаксиса языков с бесконечным количеством различных предложений используют рекурсию. При этом если нетерминал в порождающем правиле расположен справа, то рекурсию называют правосторонней, если слева, то – левосторонней, а, если между двумя подстроками, то – вложенной.

Для записи правил продукции обычно используют более компактные и наглядные формы (модели): форму Бэкуса-Наура (БНФ) или синтаксические диаграммы.

Формальная грамматика, таким образом, определяет правила определения допустимых конструкций языка, т. е. его синтаксис.

Распознавание принадлежности строки конкретному языку осуществляется его выводом из аксиомы. Вывод представляет собой последовательность подстановок, при выполнении которых левая часть правила заменяется правой.

Исходная строка, строки, получаемые в процессе вывода и аксиома, называются сентенциальными формами. Сентенциальная форма, содержащая только терминальные символы, называется допустимой и представляет собой предложение языка.

Вывод можно представить графически, в виде синтаксического дерева, корнем которого является аксиома, а листья – образуют предложение языка. Существуют грамматики, в которых для одной строки можно построить несколько деревьев. Такие грамматики называются неоднозначными. Разбор неоднозначных грамматик затруднен, поэтому их, если возможно, преобразуют в однозначные или ограничивают правилами.

Грамматический разбор – процедура построения синтаксического дерева для конкретного предложения языка. Построение такого дерева позволяет однозначно доказать, что анализируемая строка языка является допустимой, т.е. принадлежит конкретному языку.

Грамматический разбор может осуществляться в разной последовательности, причем дерево можно строить как «сверху» – от аксиомы, так и «снизу» – от предложения.

Соответственно существуют нисходящий и восходящий методы разбора. При этом предложения языка можно рассматривать слева направо и наоборот. Соответственно различают левосторонний и правосторонний разбор. Левосторонний разбор используют чаще, т. к. мы читаем слева направо.

Синтаксический анализатор – это часть компилятора, которая отвечает за выявление основных синтаксических конструкций входного языка. Его задачи: найти, выделить, проверить правильность каждой синтаксической конструкции и представить результат в форме удобной для дальнейшего применения на этапах оптимизации и генерации кода.

В основе синтаксического анализатора лежит распознаватель текста входной программы на основе грамматики входного языка. Распознаватель даёт ответ на вопрос принадлежит или нет цепочка входных символов заданному языку. Но его функции несколько шире: передать последующим фазам компилятора информацию о найденных и разобранных синтаксических структурах.

Таким образом в основе синтаксического анализатора лежит распознаватель с магазинной памятью. Результатом работы распознавателя контекстно-свободного языка (КС-языка) является последовательность правил грамматики, применённых для построения входной цепочки.

Среди всего множества распознавателей можно выделить следующие, наиболее часто используемые:

– нисходящие распознаватели с подбором альтернатив (на основе LL(k) грамматик, например метод рекурсивного спуска);

– восходящие распознаватели на основе алгоритма сдвиг-свёртка (на базе LR(k) грамматик или грамматик простого предшествования).

В общем виде процесс построения синтаксического анализатора можно описать следующим образом:

1. Построить грамматику языка.
2. Выполнить преобразования над построенной грамматикой.

3. Проверить принадлежность КСГ, получившейся в результате преобразований одному из известных классов КСГ, для которых существуют линейные распознаватели.

4. Если соответствующий класс найден, взять за основу для построения распознавателя алгоритм разбора входных цепочек, известный для этого класса.

5. Если соответствующий класс не найден или найденный класс не устраивает разработчика – попытаться выполнить преобразования под интересующий класс и вернуться к пункту 3.

6. Если ни к одному из классов привести грамматику не удалось воспользоваться алгоритмами одного из универсальных распознавателей. В этом случае вместо линейной зависимости от длины входной цепочки можно надеяться в лучшем случае на квадратичную зависимость.

7. Определить в какой форме будет синтаксический распознаватель передавать результаты своей работы другим фазам компилятора.

Имея грамматику входного языка, необходимо выполнить ряд формальных преобразований над этой грамматикой, облегчающих построение распознавателя.

После этого надо проверить относится ли полученная грамматика к одному из известных классов КС-языков, для которых существуют линейные распознаватели. Часто одна и та же КСГ может быть отнесена не к одному, а сразу к нескольким классам, допускающим построение линейных распознавателей. Тогда необходимо решить какой из нескольких возможных распознавателей выбрать для практической реализации.

Лабораторная работа №4.

Программирование синтаксического анализатора.

Цель работы: Получение практических навыков программирования синтаксического анализатора.

Задачи: Научиться программировать синтаксический анализатор.

Содержание отчёта

1. Титульный лист.
2. Формулировка задачи.
3. Листинг программы.
4. Результаты тестирования программы.
5. Ответы на контрольные вопросы.

Контрольные вопросы

1. Назначение синтаксического анализатора?
2. Способы построения синтаксического анализатора?
3. Место синтаксического анализатора в задаче компиляции.
4. Восходящий и нисходящий способы синтаксического анализа.
5. Метод рекурсивного спуска.

Задание

Написать программу, которая реализует синтаксический анализатор, разработанный в ходе выполнения лабораторной работы №3. Протестировать работу программы.

Список используемых источников

1. Малявко А. А. Системное программное обеспечение. Формальные языки и методы трансляции : учебное пособие : в 3-х ч. / А. А. Малявко. - Новосибирск : НГТУ. Ч. 1. - 2010. - 104 с.
2. Гагарина Л. Г . Введение в теорию алгоритмических языков и компиляторов: учебное пособие / Л. Г. Гагарина, Е. В. Кокорева. - М. : Форум, 2011. - 176 с.