МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Ижевский государственный технический университет имени М.Т.Калашникова"
(ФГБОУ ВПО «ИжГТУ имени М.Т.Калашникова»)

Кучуганов В.Н., Касимов Д.Р.

МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА
МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №3
«ПОСТРОЕНИЕ СИНТАКСИЧЕСКОГО ДЕРЕВА»

Рекомендовано учебно-методическим советом ФГБОУ ВПО «ИжГТУ имени М.Т. Калашникова» для использования в учебном процессе в качестве
элемента ЭУМКД для студентов обучающихся по направлению 230100.62 «Информатика и вычислительная техника», профилям «Автоматизированные системы обработки информации и управления», «Системы автоматизированного проектирования» при изучении дисциплин «Математическая лингвистика», «Лингвистическое обеспечение САПР»

Ижевск 2013

Составители: Кучуганов Валерий Никонорович, доктор технических наук, профессор
Касимов Денис Рашидович, ассистент

УДК 681.3

Математическая лингвистика: методические указания к выполнению лабораторной работы №3 «Построение синтаксического дерева» по курсам «Математическая лингвистика», «Лингвистическое обеспечение САПР» профилей «Автоматизированные системы обработки информации и управления», «Системы автоматизированного проектирования» направления 230100.62 «Информатика и вычислительная техника».
Составители: Кучуганов В.Н., Касимов Д.Р., Ижевский государственный технический университет имени М.Т. Калашникова. Ижевск, 2013. – 11 с.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Синтаксические деревья используются в лингвистических процессорах для промежуточного представления текста между деревом разбора (конкретным синтаксическим деревом) и структурой данных, которая затем используется в качестве внутреннего представления текста на поздних фазах обработки (оптимизация, генерация выходного текста и т.п.).

Абстрактное синтаксическое дерево (АСД) – это синтаксическое дерево, в котором внутренние вершины сопоставлены с (помечены) операторами языка программирования, а листья – с соответствующими операндами. Таким образом, листья являются пустыми операторами и представляют только переменные и константы.

Абстрактное синтаксическое дерево отличается от дерева разбора тем, что в нем отсутствуют узлы и ребра для тех синтаксических правил, которые не влияют на семантику текста. Классическим примером такого отсутствия являются группирующие скобки, так как в АСД группировка операндов явно задается структурой дерева. Для языка, который описывается контекстно-свободной грамматикой, какими являются почти все языки программирования, создание абстрактного дерева в синтаксическом анализаторе является тривиальной задачей. Большинство правил в грамматике создают новую вершину, а символы в правиле становятся ребрами. Правила, которые ничего не привносят в АСД, такие, например, как группирующие правила, просто заменяются в вершине одним из своих символов. Кроме того, анализатор может создать полное дерево разбора и затем пройти по нему, удаляя узлы и ребра, которые не используются в абстрактном синтаксисе, чтобы получить АСД.

Цель описанной ниже лабораторной работы – ознакомиться с теоретическими и практическими основами визуализации грамматической структуры предложений формального языка.

1. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ №3

Тема работы: «Построение синтаксического дерева»

Цель работы: ознакомиться с теоретическими и практическими основами визуализации грамматической структуры предложений формального языка.

Используемые программные средства: система программирования Delphi 7.0 или выше; графический редактор Microsoft Visio.

Задание по лабораторной работе. Включить в синтаксический анализатор из лабораторной работы № 2 построение синтаксического дерева. Использовать атрибутный метод Кнута, т.е. преобразовать КС–грамматику из лабораторной работы № 2 в атрибутную грамматику добавлением атрибутов и правил построения синтаксического дерева.

Содержание отчета:

1) титульный лист;

2) текст задания, включающий вариант задания;

3) атрибутная грамматика с действиями по построению синтаксического дерева;

4) исходный текст синтаксического анализатора;

5) результаты тестирования (должны содержать отладочный вывод синтаксического дерева на экран).

Методические рекомендации к лабораторной работе

1. Расширить программу синтаксического анализатора из лабораторной работы № 2 введением действий по построению синтаксического дерева.

2. Для визуализации дерева использовать соответствующие визуальные компоненты, такие как TTreeView (Delphi).

Варианты индивидуальных заданий

Данная лабораторная работа выполняется на основе результатов выполнения лабораторной работы № 2, и должна соответствовать ее варианту.

Контрольные вопросы:

1. Синтаксические деревья.
2. Методы представления деревьев в памяти компьютера.
3. Однозначность и неоднозначность грамматики.

4. Стратегии вывода и редукции предложений языка.

5. Атрибутный метод Кнута. Атрибутная грамматика.

6. Документирование контекстных условий и семантики.

2. ДЕРЕВО СИНТАКСИЧЕСКОГО РАЗБОРА

Пусть G = (N, T, R, S) – контекстно-свободная грамматика. Синтаксическое дерево (в G) – это произвольное конечное ориентированное упорядоченное дерево, все вершины которого помечены символами алфавита ($N \cup T$), такое, что:

– все листья помечены символами терминального алфавита T или пустыми словами (ε);

– все внутренние вершины помечены символами нетерминального алфавита N, причём корень помечен стартовым символом S;

– для каждой нелистовой вершины если $A \in N$ – его пометка, а $X_1, \ldots, X_k \in N \cup T$ – перечисленные слева направо пометки непосредственных наследников этой вершины, то продукция $A \to X_1 \ldots X_k \in R$; в качестве специального случая продукции $A \to \varepsilon$ соответствует узел A с единственным дочерним узлом ε.

Пусть G – контекстно-свободная грамматика, Tr – какое-либо синтаксическое дерево в G. Если «прочитать» слева на право пометки всех листьев Tr, то получится некоторое слово α в терминальном алфавите; в таком случае принято говорить, что данное синтаксическое дерево Tr является деревом грамматического разбора этого слова α (в грамматике G).

Для любой контекстно-свободной грамматики G и слова α в алфавите терминальных символов имеет место следующая эквивалентность: α имеет дерево грамматического разбора в G тогда и только тогда, когда это слово α порождается грамматикой G.

3. ПОСТРОЕНИЕ СИНТАКСИЧЕСКИХ ДЕРЕВЬЕВ

Рассмотрим технологию построения синтаксических деревьев для арифметических выражений. Мы строим поддеревья для подвыражений путем создания узлов для каждой операции и операндов. Узлы, дочерние по отношению к узлу операции, являются корнями поддеревьев для подвыражений, образующих операнды данной операции.

Каждый узел в синтаксическом дереве может быть реализован как запись с несколькими полями. В узле операции одно поле идентифицирует операцию, а остальные поля содер-

жат указатели на узлы операндов. Знак операции часто называется меткой узла. Узлы синтаксического дерева, используемого при трансляции, могут иметь дополнительные поля для хранения значений (или указателей на значения) атрибутов, связанных с узлом.

Предлагается использовать следующие функции для создания узлов синтаксических деревьев арифметических выражений с бинарными операциями. Каждая функция возвращает указатель на вновь созданный узел.

1. mknode(op, left, right) создает узел операции с меткой op и двумя полями, содержащими указатели на left и right.

2. mkleaf(id, entry) создает узел идентификатора с меткой id и полем, содержащим entry – указатель на запись для этого идентификатора в таблице символов.

3. mkleaf(num, val) создает узел числа с меткой num и полем, содержащим val - значение числа.

Пример. Следующая последовательность вызовов функций строит синтаксическое дерево (снизу вверх) для выражения a–4+c на рисунке 3.1:

1) p_1 := mkleaf(id, entry_a);

2) p_2 := mkleaf(num, 4);

3) p_3 := mknode('-', p_1, p_2);

4) p_4 := mkleaf(id, entry_c);

5) p_5 := mknode('+', p_3, p_4).

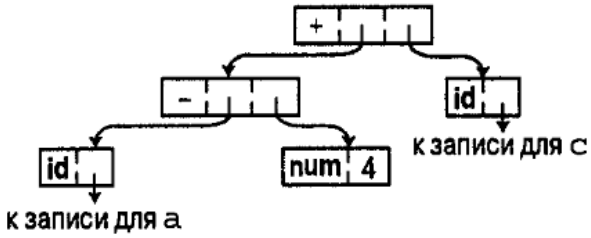

Рисунок 3.1. Синтаксическое дерево для a–4+c

Атрибутная грамматика для построения синтаксических деревьев

Атрибутная грамматика представляет собой обобщение КС-грамматики, в которой каждый грамматический символ может иметь связанное множество атрибутов, разделенное на два подмножества, – синтезируемые и наследуемые.

Атрибут может представлять собой все, что угодно, – строку, число, тип, адрес памяти и т.д. Значение атрибута в узле дерева разбора определяется семантическими правилами, связанными с используемой в данном узле продукцией. Значение синтезируемого атрибута в узле вычисляется по значениям атрибутов в дочерних по отношению к данному узлах; зна-

чения наследуемых атрибутов определяются значениями атрибутов соседних (т.е. узлов, дочерних по отношению к родительскому узлу данного) и родительских узлов.

Для указания типа атрибута рядом с его именем записывается стрелка: ↑ – для синтезируемого, ↓ – для наследуемого.

Дерево разбора, показывающее значения атрибутов в каждом узле, называется аннотированным, а процесс вычисления значений атрибутов в узлах дерева – аннотированием дерева разбора.

В таблице 3.1. содержится атрибутная грамматика для построения синтаксического дерева арифметического выражения, содержащего операции + и –. Это определение использует продукции грамматики, чтобы задать порядок вызовов функций mknode и mkleaf для построения дерева. Синтезируемый атрибут nptr у E и T используется для хранения указателей, возвращаемых вызовами функций.

Таблица 3.1. Атрибутная грамматика для построения синтаксического дерева арифметического выражения

Продукция	Семантические правила
$E \rightarrow E_1+T$	E.↑nptr := mknode('+', E_1.↑nptr, T.↑nptr)
$E \rightarrow E_1-T$	E.↑nptr := mknode('–', E_1.↑nptr, T.↑nptr)
$E \rightarrow T$	E.↑nptr := T.↑nptr
$T \rightarrow (E)$	T.↑nptr := E.↑nptr
$T \rightarrow id$	T.↑nptr := mkleaf(id, id.↑entry)
$T \rightarrow num$	T.↑nptr := mkleaf(num, num.↑val)

Аннотированное дерево разбора, изображающее построение синтаксического дерева для выражения a–4+c, показано на рисунке 3.2. Оно представлено на рисунке пунктирными линиями.

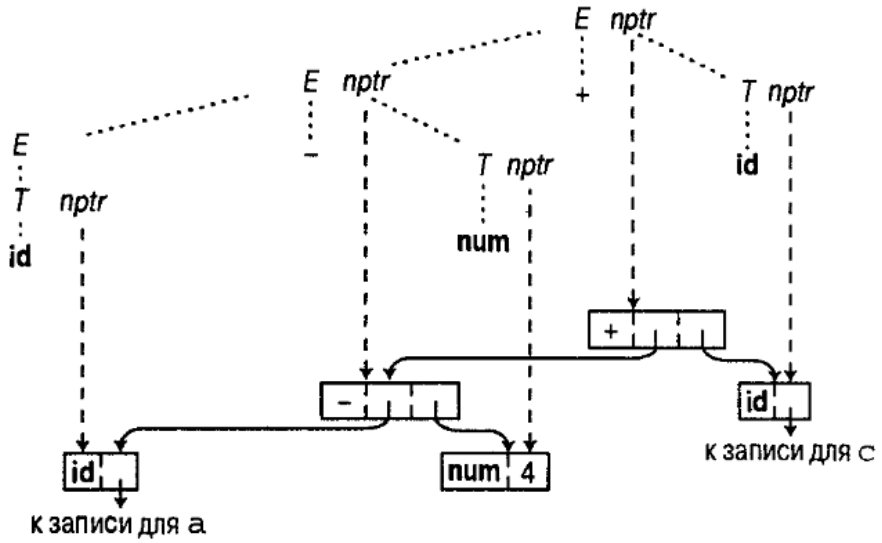

Рисунок 3.2. Построение синтаксического дерева для a–4+c

При интерпретации рисунка 3.2 важно осознавать, что нижнее дерево, построенное из записей, является “реальным” синтаксическим деревом, формирующим выход, в то время как пунктирное дерево над ним – дерево разбора, которое может существовать только в переносном смысле.

4. ПРИМЕР ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В данном разделе приводятся результаты выполнения лабораторной работы при следующей КС-грамматике:

$S \rightarrow (D\ !:\ S\ !\ S)$

$S \rightarrow <1>$

$D \rightarrow K \mid KE$

$E \rightarrow \vee\ K \mid \vee\ KE$

$K \rightarrow\ A \mid\ AX$

$X \rightarrow \wedge A \mid \wedge AX$

$A \rightarrow\ <2> \mid\ \neg\ A\ \mid\ (D)$

В таблице 4.1 приведена атрибутная грамматика для построения синтаксического дерева, в которой функция mknode принимает переменное число аргументов (первый – метка создаваемого узла, остальные – указатели на дочерние узлы), функция mkleaf создает листьевой узел с меткой, указанной в параметре.

Таблица 4.1. Атрибутная грамматика

Продукция	Семантические правила
$S \to (D ! : S_1 ! S_2)$	S.↑n := mknode('S', mkleaf('('), D.↑n, mkleaf('!'), mkleaf(':'), S_1.↑n, mkleaf('!'), S_2.↑n, mkleaf(')'))
$S \to <1>$	S.↑n := mknode('S', mkleaf(<1>))
$D \to K$	D.↑n := mknode('D', K.↑n)
$D \to K\ E$	D.↑n := mknode('D', K.↑n, E.↑n)
$E \to \vee K$	E.↑n := mknode('E', mkleaf('∨'), K.↑n)
$E \to \vee K\ E_1$	E.↑n := mknode('E', mkleaf('∨'), K.↑n, E_1.↑n)
$K \to A$	K.↑n := mknode('K', A.↑n)
$K \to A\ X$	K.↑n := mknode('K', A.↑n, X.↑n)
$X \to \wedge A$	X.↑n := mknode('X', mkleaf('∧'), A.↑n)
$X \to \wedge A\ X_1$	X.↑n := mknode('X', mkleaf('∧'), A.↑n, X_1.↑n)
$A \to <2>$	A.↑n := mknode('A', mkleaf(<2>))
$A \to \neg A_1$	A.↑n := mknode('A', mkleaf('¬'), A_1.↑n)
$A \to (D)$	A.↑n := mknode('A', mkleaf('('), D.↑n, mkleaf(')'))

На рисунке 4.1 приведен пример работы программы.

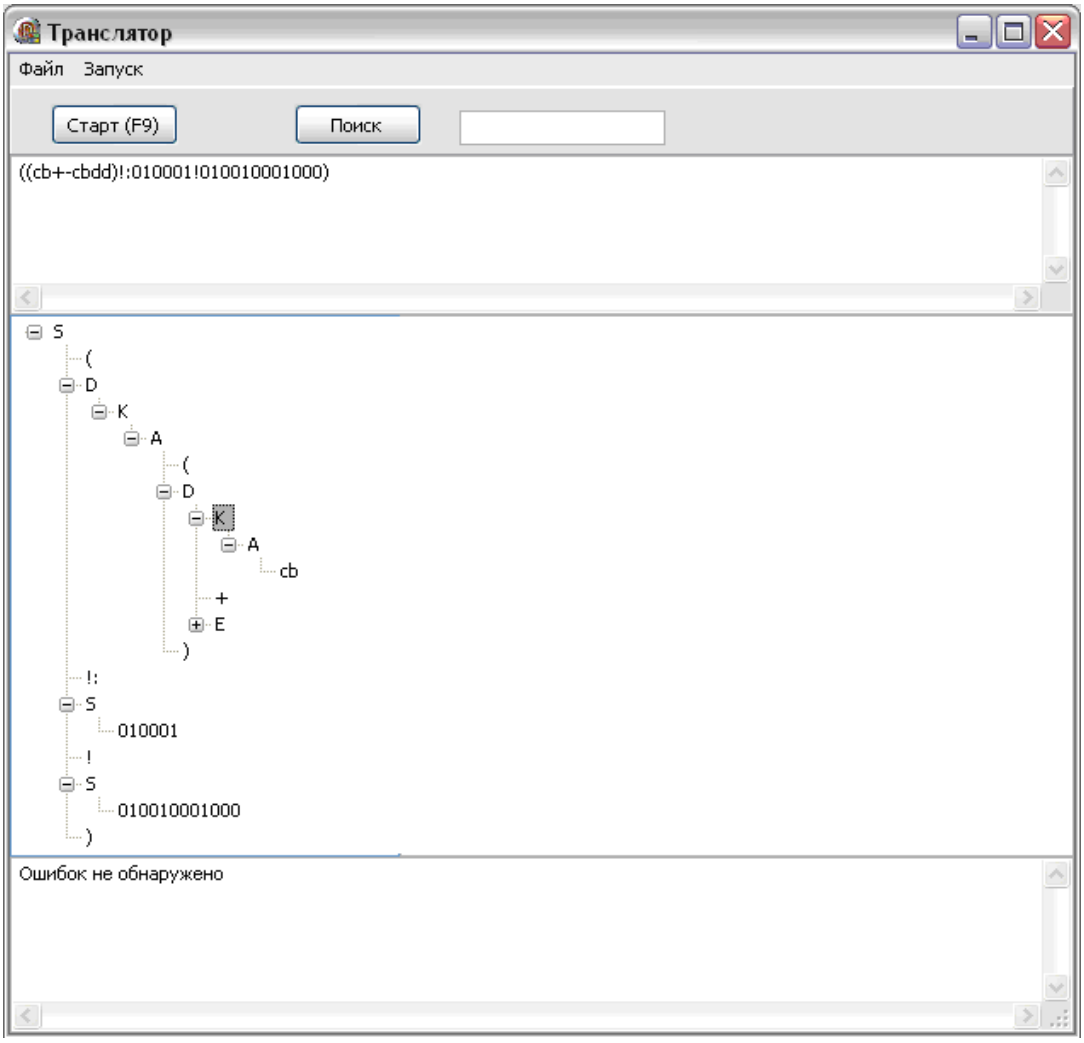

Рисунок 4.1. Пример работы программы

ЛИТЕРАТУРА

1. Ахо А., Лам М., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструментарий, 2-е изд. : Пер. с англ. – М. : ООО "И.Д. Вильямс", 2008. – 1184 с. : ил.
2. Knuth D.E. Examples of formal semantics // Lecture Notes in Mathematics. - N.Y., Springer-Verlag, 1971. -V. 188. – P. 212-235.
3. И.Г. Кревский, М.Н. Селиверстов, К.В. Григорьева. Формальные языки, грамматики и основы построения трансляторов. Учебное пособие (под ред. д.т.н., профессора А.М. Бершадского). – Пенза, 2003.
4. Свердлов С. З. Языки программирования и методы трансляции: Учебное пособие. — СПб.: Питер, 2007. — 638 с: ил.
5. Льюис Ф., Розенкранц Д., Стирнз Р. «Теоретические основы проектирования компиляторов». – М.: Мир, 1979.
6. Компаниец Р. И., Маньяков Е. В., Филатов Н. Е., «Системное программирование. Основы построения трансляторов». – СПб.: КОРОНА принт, 2000.
7. Мозговой М. В. «Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход». – СПб.: Наука и Техника, 2006.