Оглавление

Лабораторная работа №1

Формальные грамматики. Выводы

Цель работы: изучить основные понятия теории формальных языков и грамматик.

Задание

1. Написать программу, выполняющую левый вывод в заданной КС-грамматике.

Исходные данные: КС-грамматика.

Выполнение: на каждом шаге вывода отображается промежуточная цепочка (на первом шаге она представляет собой начальный нетерминал, на последнем — терминальную цепочку) и правила, которые можно применить на данном шаге. Пользователь выбирает одно из предложенных правил и процесс повторяется.

Результат: терминальная цепочка, последовательность номеров правил, участвовавших в её выводе, дерево вывода в линейной скобочной форме (ЛСФ ДВ).

Пример выполнения программы

КС-грамматика:
1. $S \rightarrow aAbS$
2. $S \rightarrow b$
3. $A \rightarrow SAc$
4. $A \rightarrow \varepsilon$

Левый вывод:

Шаг 1.
Промежуточная цепочка: S
Можно применить:
1. $S \rightarrow aAbS$
2. $S \rightarrow b$
Применяем правило 1

Шаг 2.
Промежуточная цепочка: aAbS
Можно применить:
3. $A \rightarrow SAc$
4. $A \rightarrow \varepsilon$
Применяем правило 3

Шаг 3.

Промежуточная цепочка: aSAcbS

Можно применить: 1. $S \rightarrow aAbS$

2. $S \rightarrow b$

Применяем правило 2

Шаг 4.

Промежуточная цепочка: abAcbS

Можно применить: 3. $A \rightarrow SAc$

4. $A \rightarrow \varepsilon$

Применяем правило 4

Шаг 5.

Промежуточная цепочка: abcbS

Можно применить: 1. $S \rightarrow aAbS$

2. $S \rightarrow b$

Применяем правило 2

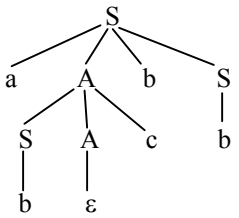
Шаг 6.

Терминальная цепочка: авсвв

Последовательность правил: 1 3 2 4 2

ЛСФ ДВ: S(aA(S(b)A()c)bS(b))

Примечание. Дерево вывода имеет вид:



2. Выполнить левый (правый) вывод терминальной цепочки в заданной грамматике (см. варианты заданий п.1), построить дерево вывода. Определить, существует ли неэквивалентный вывод полученной цепочки и, если существует, представить его деревом вывода.

3. Написать программу, определяющую, можно ли применить заданную последовательность правил при левом выводе цепочки в заданной КС-грамматике.

Исходные данные:
— КС-грамматика;
— последовательность номеров правил.
Результат: “да” или “нет”.

Пример выполнения программы

КС-грамматика: 1. $S \to aAbS$
2. $S \to b$
3. $A \to SAc$
4. $A \to \varepsilon$
Последовательность равил: 1 3 2 1 4
Результат: “нет”.

4. Для каждой последовательности правил (см. варианты заданий п.2) определить, можно ли её применить при левом (правом) выводе терминальной цепочки в заданной КС-грамматике, и, если можно, построить дерево вывода.

5. Написать программу, определяющую, можно ли применить заданную последовательность правил при выводе цепочки в заданной КС-грамматике.

Исходные данные:
— КС-грамматика;
— последовательность номеров правил.
Результат: “да” или “нет”.

Пример выполнения программы

КС-грамматика: 1. $S \to aAbS$
2. $S \to b$
3. $A \to SAc$
4. $A \to \varepsilon$
Последовательность равил: 1 3 2 1 4
Результат: “да”.

6. Для каждой последовательности правил (см. варианты заданий п.2) определить, можно ли её применить при выводе терминальной цепочки в заданной КС-грамматике, и, если можно, построить дерево вывода и записать эквивалентные левый и правый вывод.

Варианты заданий

Вариант 1

1. КС-грамматика

1. S→aSSa	5. A→Ba
2. S→bSb	6. A→ε
3. S→cAb	7. B→bB
4. A→Aa	8. B→aA

2. Последовательности правил вывода

1. 1, 1, 3, 4, 6, 3, 5, 7, 8, 6, 2, 3, 6
2. 1, 1, 2, 3, 3, 3, 4, 5, 6, 6, 7, 8, 6
3. 1, 3, 3, 5, 7, 4, 6, 8, 6, 2, 3, 6, 6
4. 1, 2, 3, 6, 1, 3, 5, 7, 8, 6, 3, 4, 6

Вариант 2

1. КС-грамматика

1. S→Aba	5. A→BaB
2. S→bS	6. A→ε
3. S→cAbA	7. B→b
4. A→AaS	8. B→aA

2. Последовательности правил вывода

1. 1, 4, 7, 5, 3, 7, 7, 6, 6, 4, 6, 2, 3, 6, 6
2. 1, 7, 4, 3, 6, 6, 5, 7, 4, 2, 3, 6, 6, 6, 7
3. 1, 4, 5, 7, 7, 4, 6, 2, 1, 6, 7, 3, 6, 6, 7
4. 1, 4, 3, 5, 7, 6, 6, 7, 7, 4, 2, 6, 6, 7, 6

Вариант 3

1. КС-грамматика

1. S→Ssa	5. A→BaB
2. S→b	6. A→S
3. S→Ab	7. B→b
4. A→AaA	8. B→aA

2. Последовательности правил вывода

1. 1, 2, 3, 4, 5, 6, 7, 8, 6, 2, 2
2. 1, 2, 3, 4, 5, 7, 8, 6, 2, 6, 2
3. 1, 3, 4, 6, 2, 5, 8, 6, 2, 7, 2
4. 1, 2, 3, 4, 6, 6, 2, 2, 7, 8, 6

Вариант 4

1. КС-грамматика

1. S→AaBaA	5. A→a
2. S→bSb	6. A→ε
3. A→B	7. B→bS
4. A→Sa	8. B→A

2. Последовательности правил вывода

1. 2, 1, 6, 8, 3, 8, 6, 4, 1, 5, 8, 6, 5
2. 2, 1, 8, 3, 8, 6, 4, 1, 8, 6, 5, 5, 6
3. 2, 1, 4, 1, 5, 8, 6, 5, 8, 3, 8, 6, 6
4. 2, 1, 8, 6, 4, 3, 8, 6, 8, 6, 3, 4, 6

Вариант 5

1. КС-грамматика

1. S→SS	5. A→Ba
2. S→A	6. A→ε
3. A→AbA	7. B→bB
4. A→B	8. B→aA

2. Последовательности правил вывода

1. 1, 1, 2, 6, 2, 3, 6, 6, 2, 4, 8, 6
2. 1, 2, 4, 8, 6, 1, 2, 6, 2, 3, 6, 6
3. 1, 1, 2, 3, 6, 6, 2, 6, 2, 4, 8, 6
4. 1, 2, 1, 4, 2, 2, 8, 6, 3, 6, 6, 6

Вариант 6

1. КС-грамматика

1. S→SaS	5. A→b
2. S→A	6. B→bB
3. A→aA	7. B→bBaB
4. A→AB	8. B→a

2. Последовательности правил вывода

1. 1, 2, 4, 5, 8, 2, 4, 4, 5, 8, 6, 8
2. 1, 2, 4, 4, 5, 8, 6, 8, 2, 4, 5, 8
3. 1, 2, 4, 6, 8, 4, 8, 5, 2, 4, 8, 5
4. 1, 2, 2, 4, 4, 4, 6, 5, 8, 5, 8, 8

Вариант 7

1. КС-грамматика

1. S→aSbS	5. A→Ba
2. S→aS	6. A→b
3. S→A	7. B→b
4. A→Sa	8. B→aA

2. Последовательности правил вывода

1. 1, 2, 3, 5, 8, 4, 3, 6, 2, 3, 5, 8, 6
2. 2, 1, 3, 5, 8, 4, 3, 6, 2, 3, 5, 8, 6
3. 1, 2, 2, 3, 3, 5, 5, 8, 8, 4, 6, 3, 6
4. 2, 1, 2, 3, 5, 8, 6, 3, 5, 8, 4, 3, 6

Вариант 8

1. КС-грамматика

1. S→aSbA	5. A→aB
2. S→aB	6. A→S
3. S→A	7. B→b
4. A→aAbS	8. B→aA

2. Последовательности правил вывода

1. 1, 3, 4, 5, 7, 2, 7, 5, 7
2. 1, 1, 2, 7, 5, 7, 3, 2, 7
3. 1, 5, 7, 3, 4, 2, 7, 5, 7
4. 1, 3, 1, 5, 2, 2, 7, 7, 7

Вариант 9

1. КС-грамматика

1. S→SbSa	5. A→aB
2. S→Sa	6. A→b
3. S→A	7. B→b
4. A→aS	8. B→Aa

2. Последовательности правил вывода

1. 1, 2, 3, 5, 8, 6, 2, 3, 5, 8, 4, 3, 6
2. 2, 1, 3, 5, 8, 4, 3, 6, 2, 3, 5, 8, 6
3. 1, 2, 2, 3, 3, 5, 5, 8, 8, 4, 6, 3, 6
4. 2, 1, 2, 3, 5, 8, 6, 3, 5, 8, 4, 3, 6

Вариант 10

1. КС-грамматика

1. $S \to SaS$	5. $A \to b$
2. $S \to A$	6. $B \to Bb$
3. $A \to Aa$	7. $B \to BaBb$
4. $A \to BA$	8. $B \to a$

2. Последовательности правил вывода
   1. 1, 2, 4, 5, 8, 2, 4, 4, 5, 8, 6, 8
   2. 1, 2, 4, 4, 5, 8, 6, 8, 2, 4, 5, 8
   3. 1, 2, 4, 6, 8, 4, 8, 5, 2, 4, 8, 5
   4. 1, 2, 2, 4, 4, 4, 6, 5, 8, 5, 8, 8

К о н т р о л ь н ы е в о п р о с ы

1. Что такое формальный язык?
2. Назовите способы задания формальных языков.
3. Что такое формальная грамматика?
4. Какие классы грамматик выделяют по классификации Хомского?
5. Приведите примеры грамматик различных классов.
6. Какие классы языков выделяют по классификации Хомского?
7. Приведите примеры языков различных классов.
8. Что такое терминальная цепочка?
9. В чём заключается вывод терминальной цепочки в грамматике.
10. Что такое промежуточная цепочка вывода?
11. Приведите примеры различных способов представления вывода. Что такое дерево вывода?
12. Какие выводы называются эквивалентными?
13. Чем различаются неэквивалентные выводы?
14. Приведите примеры эквивалентных выводов?
15. Приведите примеры неэквивалентных выводов?
16. Чем различаются левые и правые выводы?
17. Определите класс КС-грамматик, в которых левый и правый выводы одной и той же цепочки совпадают.
18. Как определить, существуют ли два неэквивалентных вывода заданной выводимой цепочки?
19. Какие грамматики называются неоднозначными?
20. Приведите примеры неоднозначных грамматик.

Лабораторная работа №2

Преобразования КС-грамматик

Цель работы: изучить основные эквивалентные преобразования КС-грамматик и научиться применять их для получения КС-грамматик, обладающих заданными свойствами.

Задание

1. Преобразовать исходную грамматику G (см. варианты заданий) в грамматику G_1 без лишних символов.

2. Преобразовать грамматику G_1 в грамматику G_2 без ε-правил.

3. Преобразовать грамматику G_1 в грамматику G_3 без цепных правил.

4. Преобразовать грамматику G_1 в грамматику G_4 без левой рекурсии.

5. Преобразовать грамматику G_1 в грамматику G_5 без несаморекурсивных нетерминалов.

6. Получить грамматику G_6, эквивалентную грамматике G_1, в которой правая часть каждого правила состоит либо из одного терминала, либо двух нетерминалов.

7. Получить грамматику G_7, эквивалентную грамматике G_1, в которой правая часть каждого правила начинается терминалом.

8. Получить грамматику G_8, эквивалентную грамматике G_1, в которой правая часть каждого не ε-правила начинается терминалом и любые два правила с одинаковой левой частью различаются первым символом в правой части.

9. Получить грамматику G_9, эквивалентную грамматике G_1, в которой правая часть каждого правила не содержит двух стоящих рядом нетерминала.

10. Получить грамматику G_{10}, эквивалентную грамматике G_1, в которой любой символ (нетерминал) занимает либо только крайнюю правую позицию в правых частях правил, либо находится левее самого правого символа (нетерминала) в правых частях правил.

Варианты заданий

Вариант 1	Вариант 2	Вариант 3	Вариант 4	Вариант 5
1. A→aBBb	1. T→aRMb	1. E→EabE	1. S→EabS	1. B→EabB
2. A→B	2. T→b	2. E→BaDC	2. S→SaDB	2. E→BaDb
3. B→BabB	3. T→ε	3. D→ε	3. S→ε	3. E→B
4. B→AaCD	4. M→Q	4. D→aEAb	4. D→EbAb	4. A→aEAb
5. B→ε	5. M→Sbb	5. D→b	5. D→S	5. A→b
6. C→aBaE	6. S→aRQb	6. D→C	6. D→SB	6. A→E
7. C→bb	7. S→RQ	7. C→BCa	7. E→Ba	7. D→BDa
8. C→ε	8. R→RabR	8. C→CEb	8. E→Eab	8. D→DEb
9. D→Ada	9. R→SaTQ	9. A→abC	9. A→ab	9. B→abC
10. D→DEb	10. R→ε	10. A→Bbb	10. A→Bab	10. B→DaB
11. E→abD	11. Q→SQa	11. B→aECb	11. B→aEb	11. B→D
12. E→Ab	12. Q→Qb	12. B→D	12. B→Ea	12. B→ε
Вариант 6	Вариант 7	Вариант 8	Вариант 9	Вариант 10
1. A→abBb	1. A→EabE	1. T→abETP	1. E→EabE	1. S→RMb
2. A→C	2. A→BDC	2. T→aDC	2. E→BaDC	2. S→ab
3. B→BabB	3. B→ε	3. T→D	3. D→ε	3. S→T
4. B→AaCD	4. B→AEAb	4. D→DTAb	4. D→aEAb	4. M→Q
5. B→ε	5. D→bDD	5. D→b	5. D→b	5. M→Sbb
6. C→BaE	6. D→C	6. E→ε	6. D→C	6. T→aRQb
7. C→ab	7. C→BCa	7. P→BCa	7. C→Bca	7. T→RQ
8. D→aEA	8. C→CEb	8. P→Cb	8. C→ε	8. R→RabR
9. D→ADa	9. E→abC	9. C→abC	9. A→abC	9. R→STQ
10. D→Eb	10. E→Bbb	10. A→Bbb	10. A→DBb	10. R→ε
11. E→DaE	11. E→aECb	11. B→aECb	11. B→Beb	11. Q→SQa
12. E→A	12. E→D	12. B→D	12. B→DCB	12. Q→Qb

Контрольные вопросы

1. Какие символы грамматики называются лишними? Как их устранить?

2. Что такое аннулирующий нетерминал? Как найти множество аннулирующих нетерминалов?

3. Какие правила называются цепными? В чём заключается «опасность» цепных правил? Как их устранить?

4. Какие правила называются самолеворекурсивными? Приведите пример леворекурсивной грамматики без самолеворекурсивных правил.

5. Дайте определения грамматикам, заданным в нормальных формах Хомского и Грейбах.

6. Дайте определение операторной грамматики.

Лабораторная работа №3

Регулярные языки и конечные распознаватели

Цель работы: изучить основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.

Задание

1. Построить минимальный детерминированный конечный распознаватель заданного языка (см.варианты заданий).

2. Написать программу-распознаватель компиляционного и интерпретационного типа.

Исходные данные: строка.
Результат: “допустить” — если строка представляет собой цепочку заданного языка;
“отвергнуть” — в противном случае.

3. Написать программу, которая оставляет в исходном текстовом файле только те строки, которые представляют собой цепочки заданного языка.

4. Написать программу, которая исключает из исходного текстового файла строки, являющиеся цепочками заданного языка.

Варианты заданий

Вариант 1

Язык строковых констант.

Строковая константа может состоять из одной или более частей, соединённых между собой знаком + . Каждая часть начинается и заканчивается апострофом. Между апострофами могут быть любые символы, но если нужно включить апостроф, то его нужно повторить дважды. Строковая константа может содержать коды символов. Код символа — не более чем трёхзначное беззнаковое восьмеричное число, записанное после знака # . Последовательность кодов — часть константы.

Вариант 2

Язык строковых констант.

Строковая константа — последовательность символов, заключённая в кавычки. Последовательности символов \n , \t , \b , \r , \f , \' , \" , \\ и символ \ , за которым следует трёхразрядное восьмеричное число, считаются одним символом, а одиночный символ \ недопустим.

Вариант 3

Язык прилагательных.

Язык прилагательных содержит прилагательные, независимо от рода и падежа. Принадлежность слова множеству прилагательных определяется окончанием. Кроме этого слова должны содержать не менее одной гласной и не менее одной согласной в части слова до окончания, последовательности согласных не должны превышать трёх букв, а последовательности гласных, не включая окончание — двух букв.

Вариант 4

Язык управляющих строк.

Управляющая строка начинается и заканчивается кавычками, содержит обычные символы и спецификации. Спецификация начинается символом % и заканчивается символом преобразования. Между % и символом преобразования может находиться формат, представляющий собой последовательность цифр, возможно, разделённых на две части точкой, перед которой или за которой может находиться знак минус. Символами преобразования являются: D , O , X , U , C , S , E , F , G . Символом % может начинаться только спкцификация.

Вариант 5

Язык параметров оператора вывода.

Параметры оператора вывода заключаются в круглые скобки и разделяются запятыми. Параметр может представлять собой целочисленную или строковую константу или переменную, за которыми может следовать формат.

Строковая константа представляет собой последовательность символов, заключённую в кавычки. Целочисленная константа может быть десятичной, которая начинается не с нуля, восьмеричной, которая начинается с нуля, или шестнадцатеричной, которая начинается с бук-

вы H. Константе может предшествовать знак + или – . Формат, следующий за константой, представляет собой двоеточие, за которым беззнаковое целое. Формат, следующий за переменной, может представлять собой формат, следующий за константой, или формат, следующий за константой, за которым следует такой же формат.

Вариант 6

Язык адресов.

Адрес представляет собой последовательность объектов, разделённых запятой.

Объект “область” представляет собой название области (слово), за которым следует слово “обл.” .

Объект “край” представляет собой название края (слово), за которым следует слово “край”.

Объект “город” представляет собой слово “г.”, за которым следует название города (последовательность слов).

Объект “район” представляет собой название района (слово), за которым следует слово “район”.

Объект “село” представляет собой слово “с.”, за которым следует название села (последовательность слов).

Объект “улица” представляет собой слово “ул.”, за которым следует название улицы (последовательность слов).

Объект “дом” представляет собой слово “д.”, за которым следует число и, возможно, буква.

Объект “квартира” представляет собой слово “кв.”, за которым следует число.

Каждый объект в адресе встречается только один раз. Начинаться адрес может объектами : “область” , “край” или “город”. Если адрес начинается объектами “область” или “край” , то за ними могут следовать объекты “город” или “район”. После объекта “район” может следовать “город” или “село”. За объектами “город” и “село” всегда следуют “улица” и “дом”, после чего может быть “квартира”.

Вариант 7

Язык директив компилятора.

Цепочка языка начинается парой символов {$, заканчивается символом } и содержит последовательность директив, разделённых запятой. Директива представляет собой один из символов A , B , D , E , F , L , N , R , S , V , за которым следует символ + или – , или символ I ,

за которым следует имя файла (полное, начиная с имени диска и заканчивая расширением, или сокращённое), или символ M , за которым следует три беззнаковых целых числа, разделённых запятыми.

Вариант 8

Язык вещественных констант.

Вещественная константа может состоять из мантиссы и порядка, разделённых символом E . Мантисса может состоять из целой и дробной частей, разделённых точкой. Целая часть мантиссы и порядок — целые знаковые десятичные числа (знак может отсутствовать). Если отсутствует порядок и символ E , то могут отсутствовать целая или дробная часть мантиссы, но не обе сразу. Если присутствует порядок и символ E , то может отсутствовать дробная часть и точка.

Вариант 9

Язык слов, правильно разбитых на две части.

Части слова разделяются символом - . Слово считается правильно разбитым на две части, если:

— первая часть заканчивается последовательностью состоящей из согласной и гласной буквы, а вторая часть начинается гласной, за которой идёт хотя бы одна буква (буква й при этом рассматривается вместе с предшествующей гласной как единое целое);

— первая часть заканчивается последовательностью, состоящей из гласной и согласной буквы, а вторая часть начинается согласной и в оставшейся части имеется хотя бы одна гласная (буквы ь и ъ вместе с предшествующей согласной рассматриваются как единое целое).

Вариант 10

Язык числовых констант.

Числовая константа может быть вещественной или целочисленной. Константа может начинаться со знака + или – . Вещественная константа представляется только в десятичной системе счисления, а целочисленная — в задаваемой системе счисления.

Вещественная константа может быть представлена в форме с фиксированной или с плавающей точкой. В форме с фиксированной точкой константа состоит из двух частей — целой и дробной, разделённых точкой. Одна из частей может отсутствовать, но не обе сразу. В форме с плавающей точкой константа состоит из двух частей — ман-

тиссы и порядка, разделённых символом E . Мантисса — вещественное число в форме с фиксированной точкой, целая часть которой представляет собой одну цифру, не равную нулю. Порядок — целое число со знаком + или –, не более чем четырёхзначное.

Целочисленная константа в десятичной системе представляет собой последовательность десятичных цифр. Целочисленная константа начинается с точки, после которой следует число $n \in \{2, 8, 10\}$, представляющее собой основание системы счисления, точка и последовательность цифр соответствующей системы счисления.

Контрольные вопросы

1. Какой язык называется регулярным?
2. Дайте определение правосторонней и левосторонней грамматики.
3. Опишите алгоритм преобразования правосторонней грамматики в автоматную правостороннюю.
4. Опишите алгоритм преобразования левосторонней грамматики в автоматную левостороннюю.
5. Приведите пример конечного распознавателя с ε-переходами. Преобразуйте его в конечный распознаватель без ε-переходов.
6. Приведите пример недетерминированного конечного распознавателя без ε-переходов. Преобразуйте его в детерминированный конечный распознаватель.
7. Постройте конечный распознаватель по правосторонней грамматике.
8. Постройте конечный распознаватель по левосторонней грамматике.
9. Приведите пример двух эквивалентных детерминированных конечных распознавателей с различным числом состояний. Проверьте их эквивалентность и минимизируйте их.
10. Приведите пример регулярного выражения. Постройте по нему детерминированный конечный распознаватель.
11. Какие регулярные выражения называются эквивалентными? Как определить эквивалентность регулярных выражений?
12. Представьте регулярным выражением язык, заданный правосторонней грамматикой.
13. Представьте левосторонней грамматикой язык, заданный регулярным выражением.
14. Напишите программу, которая табличное представление конечного детерминированного распознавателя преобразует в компиляционную программу реализации конечного распознавателя.

Лабораторная работа №4

Лексический анализатор

Цель работы: изучить и научиться применять методы разработки и реализации лексических анализаторов.

Задание

1. Построить конечные распознаватели лексем заданного языка (см. варианты заданий).

2. Определить метод обнаружения границы лексемы.

3. Определить действия, выполняемые лексическим анализатором. Тип лексем определяется вариантом задания. Если значение лексемы не определено заданием, то значением является имя лексемы.

4. Выбрать метод и программно реализовать лексический анализатор в виде подпрограммы. При обнаружении лексической ошибки лексический анализатор должен выдавать сообщение о типе ошибки и её расположении (номер строки) в исходном тексте.

5. Написать программу, преобразующую исходный текст программы на заданном языке в таблицу лексем.

6. Обработать исходные тексты, содержащие и не содержащие лексические ошибки.

Варианты заданий

Вариант 1

1. Идентификаторы целочисленных переменных представляют собой букву, за которой следуют буквы или цифры. Все идентификаторы имеют один и тот же тип лексемы.
2. Целочисленные константы представляют собой последовательность десятичных цифр. Все целочисленные константы имеют один и тот же тип лексемы.
3. Строковая константа состоит из одной или нескольких частей, соединённых символом + . Каждая часть представляет собой последовательность символов, не содержащую апострофы, заключённую в

апострофы, или последовательность символов, не содержащую кавычки, заключённую в кавычки. Все строковые константы имеют один и тот же тип лексемы, значение лексемы — последовательность символов, образованная конкатенацией всех частей лексемы без ограничивающих символов, заключённая в апострофы. Апостроф внутри части лексемы заменяется парой апострофов.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, IF, THEN, ELSE. Каждое слово имеет свой тип лексемы.

5. Арифметические операции: + и – . Каждая операция имеет свой тип лексемы.

6. Логические операции: NOT, AND, OR. Каждая операция имеет свой тип лексемы.

7. Операции отношения: =, <>, <, >, <=,>=. Все операции имеют один тип лексемы.

8. Знак оператора присваивания: := .

9. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

10. Комментарии заключаются в фигурные скобки { и }. Могут располагаться после любой лексемы, но не могут находиться внутри лексемы.

Вариант 2

1. Идентификаторы вещественных переменных представляют собой букву, за которой следуют буквы или цифры. Все идентификаторы имеют один и тот же тип лексемы, значение лексемы — ссылка на элемент таблицы идентификаторов.

2. Числовые константы представляют собой вещественные числа с фиксированной точкой. Целая часть и точка обязательны, дробная может отсутствовать. Все числовые константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, заключённую в апострофы. Если строковая константа должна содержать в себе апостроф, то в последовательности символов такой апостроф дублируется (записывается дважды). Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, WHILE, DO. Каждое слово имеет свой тип лексемы.

5. Аддитивные операции: + и – . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * и / . Операции имеют один и тот же тип лексемы.

7. Операции отношения: =, <>, <, >, <=,>=. Все операции имеют один тип лекемы.

8. Знак оператора присваивания: := .

9. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

10. Строчные комментарии начинаются символами // вначале строки и продолжаются до конца строки.

Вариант 3

1. Идентификаторы вещественных переменных представляют собой букву, за которой следуют буквы или цифры. Все идентификаторы имеют один и тот же тип лексемы.

2. Числовые константы представляют собой вещественные числа с фиксированной точкой. Целая и дробная часть должны содержать хотя бы одну цифру. Все числовые константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, заключённую в апострофы. Строковая константа не может содержать в себе апостроф. Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, REPEAT, UNTIL. Каждое слово имеет свой тип лексемы.

5. Арифметические операции: + , – , * , / . Каждая операция имеет свой тип лексемы.

6. Операции отношения: =, <>, <, >, <=,>=. Все операции имеют один тип лекемы.

7. Знак оператора присваивания: = .

8. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

9. Комментарии заключаются в фигурные скобки { и } или ограничиваются с обеих сторон парой символов // . Могут располагаться после любой лексемы, но не могут находиться внутри лексемы.

Вариант 4

1. Идентификаторы целочисленных переменных представляют собой букву, за которой могут следовать цифры. Все идентификаторы имеют один и тот же тип лексемы.

2. Числовые константы представляют собой целые числа. Все числовые константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, заключённую в апострофы. Если строковая константа должна содержать в себе апостроф, то в последовательности символов такой апостроф дублируется (записывается дважды). Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: BEGIN, END, READ, WRITE, FOR, TO, DO. Каждое слово имеет свой тип лексемы.

5. Аддитивные операции: + и – . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * и / . Операции имеют один и тот же тип лексемы.

7. Знак оператора присваивания: := .

8. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

9. Комментарии начинаются и заканчиваются символами //.

Вариант 5

1. Идентификаторы целочисленных переменных представляют собой букву, за которой может следовать буква или цифры. Все идентификаторы имеют один и тот же тип лексемы.

2. Числовые константы представляют собой целые числа. Все числовые константы имеют один и тот же тип лексемы.

3. Ключевые слова: VAR, BEGIN, END, READ, WRITE, FOR, TO, STEP, DO. Каждое слово имеет свой тип лексемы.

4. Унарная операция not .

5. Аддитивные операции: + , – , or , xor . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * , / , and . Операции имеют один и тот же тип лексемы.

7. Знак оператора присваивания: := .

8. Специальные знаки: (,) . Каждый знак имеет свой тип лексемы.

9. Комментарии начинаются символами /* и заканчиваются символами */.

Вариант 6

1. Идентификаторы вещественных переменных представляют собой букву, за которой могут следовать буквы или цифра. Все идентификаторы имеют один и тот же тип лексемы, значение лексемы — ссылка на элемент таблицы идентификаторов.

2. Числовые константы представляют собой вещественные числа с плавающей точкой в нормализованной форме. Все числовые константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, заключённую в апострофы. Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, WHILE, DO. Каждое слово имеет свой тип лексемы

5. Аддитивные операции: + и – . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * и / . Операции имеют один и тот же тип лексемы.

7. Знаки оператора присваивания: <= и => . Каждый оператор имеет свой тип лексемы.

8. Специальные знаки: , и ; . Каждый знак имеет свой тип лексемы.

9. Комментарии начинаются символами // вначале строки и продолжаются до конца строки или ограничиваются такими же символами.

Вариант 7

1. Идентификаторы вещественных переменных представляют собой букву, за которой следуют буквы или цифры. Все идентификаторы имеют один и тот же тип лексемы.

2. Числовые константы представляют собой вещественные числа с фиксированной точкой. Целая и дробная часть могут отсутствовать, но не обе сразу. Все числовые константы имеют один и тот же тип лексемы.

3. Ключевые слова: VAR, BEGIN, END, READ, WRITE, REPEAT, UNTIL. Каждое слово имеет свой тип лексемы.

4. Логические операции: or , and и not . Каждая операция имеет свой тип лексемы.

5. Аддитивные операции: + и – . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * и / . Операции имеют один и тот же тип лексемы.

7. Операции сравнения: =, <>, <, >, <=, >=. Все операции имеют один тип лексемы.

8. Знак оператора присваивания: = .

9. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

11. Комментарии заключаются в фигурные скобки { и } или ограничиваются с одной стороны парой символов (*, а с другой — парой символов *). Могут располагаться после любой лексемы, но не могут находиться внутри лексемы.

Вариант 8

1. Идентификаторы целочисленных переменных представляют собой последовательность букв, за которой могут следовать цифры. Все идентификаторы имеют один и тот же тип лексемы.

2. Целочисленные константы представляют собой последовательность десятичных цифр. Все целочисленные константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, не содержащую кавычки и апострофы, заключённую в кавычки. Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, IF, THEN, ELSE. Каждое слово имеет свой тип лексемы.

5. Арифметические операции: + , – , * , / . Все операции имеют один тип лексемы. Значение лексемы — номер операции.

6. Операции сравнения: =, <>, ><, <, >, <=, =<, >=, =>. Все операции имеют один тип лексемы. Пары лексем, обозначающие одинаковые операции, имеют одинаковые значения, а пары лексем, обозначающие различные операции, имеют различные значения.

7. Знак оператора присваивания: := .

9. Специальные знаки: (,), , , ; . Каждый знак имеет свой тип лексемы.

10. Комментарии заключаются в фигурные скобки { и }. Могут располагаться после любой лексемы, но не могут находиться внутри лексемы.

Вариант 9

1. Идентификаторы вещественных переменных представляют собой букву, за которой могут следовать буквы или цифра. Все идентификаторы имеют один и тот же тип лексемы, значение лексемы — ссылка на элемент таблицы идентификаторов.

2. Числовые константы представляют собой знаковые вещественные числа с плавающей точкой в нормализованной форме. Все числовые константы имеют один и тот же тип лексемы.

3. Строковая константа представляет собой последовательность символов, заключённую в апострофы. Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, WHILE, DO. Каждое слово имеет свой тип лексемы.

5. Операции: + , – , * , / , or , xor , and , not , = , <> , < , > , <= , >= . Все операции имеют один тип лекемы.

6. Знак оператора присваивания: := .

7. Специальные знаки: , и ; . Каждый знак имеет свой тип лексемы.

9. Комментарии начинаются символами // и продолжаются до конца строки или ограничиваются такими же символами.

Вариант 10

1. Идентификаторы целочисленных переменных представляют собой букву, за которой следуют буквы или цифры. Заканчивается идентификатор обязательно буквой. Все идентификаторы имеют один и тот же тип лексемы, значение лексемы — ссылка на элемент таблицы идентификаторов.

2. Числовые константы представляют собой последовательность десятичных цифр. Все числовые константы имеют один и тот же тип лексемы

3. Строковая константа представляет собой последовательность символов, заключённую в кавычки. Если строковая константа должна содержать в себе апостроф, то в последовательности символов такой апостроф дублируется (записывается дважды). Все строковые константы имеют один и тот же тип лексемы.

4. Ключевые слова: VAR, BEGIN, END, READ, WRITE, CASE, DO. Каждое слово имеет свой тип лексемы.

5. Аддитивные операции: + и – . Операции имеют один и тот же тип лексемы.

6. Мультипликативные операции: * и / . Операции имеют один и тот же тип лексемы.

7. Знак оператора присваивания: := .

8. Специальные знаки: (,), , , ; , : . Каждый знак имеет свой тип лексемы.

10. Строчные комментарии начинаются символами // вначале строки и продолжаются до конца строки.

Контрольные вопросы

1. Из каких основных блоков состоит транслятор?

2. Что такое лексема? Какие типы лексем существуют в языках программирования? Приведите примеры.

3. Назовите основные функции лексического анализатора.

4. Чем различаются лексемы одного типа?

5. Что такое таблица лексем? Что представляет собой элемент таблицы лексем? Сколько элементов в таблице лексем?

6. В чём разница между таблицей лексем и таблицей идентификаторов?

7. Как могут взаимодействовать лексический и синтаксический анализаторы?

8. Что такое проход компилятора? От чего зависит количество проходов?

9. В каком случае таблица лексем может отсутствовать?

10. Как работает прямой лексический анализатор?

11. Как работает непрямой лексический анализатор?

12. Как лексический анализатор обрабатывает комментарии?

13. Какие синтаксические ошибки может обнаруживать лексический анализатор?

14. Как лексический анализатор может отличить унарный минус от бинарного?

15. Как лексический анализатор обнаруживает границу лексемы?

16. В чём заключается принцип выбора лексемы наибольшей длины?

17. Какими способами могут распознаваться ключевые слова?

18. Постройте конечный распознаватель заданного множества ключевых слов?

19. Как можно объединить распознаватель ключевых слов и распознаватель идентификаторов?

Лабораторная работа №5

Синтаксически управляемая трансляция

Цель работы: изучить принципы синтаксически управляемой трансляции и научиться применять их при проектировании и реализации трансляторов.

Задание

1. Получить транслирующую грамматику, задающую перевод текста программы на исходном языке, грамматика которого представлена в вариантах заданий (см. ниже), в семантически эквивалентную программу на языке Паскаль. В программе на языке Паскаль не должно быть выражений (арифметических и логических), содержащих более одной операции, и структурированных операторов управления. Управление в программе на языке Паскаль задается только условными и безусловными переходами на метки.

2. Преобразовать транслирующую грамматику так, чтобы грамматика входного языка принадлежала классу LL(1)-грамматик. Если устранением левой рекурсии и выполнением факторизации не удается получить грамматику указанного класса, то допускается внесение изменений в грамматику не изменяющих или незначительно изменяющих исходный язык.

3. Определить множества первых и следующих для каждого нетерминала грамматики.

4. Определить множества первых для правой части каждого правила грамматики.

5. Определить множества выбора для каждого правила грамматики.

6. Построить нисходящий МП-транслятор.

7. Выбрать метод реализации транслятора.

8. Выбрать способ взаимодействия лексического и синтаксического анализаторов.

9. Написать программу, реализующую заданный транслятор.

10. Транслировать программу на исходном языке, не содержащую лексических, синтаксических и семантических ошибок, в семантически эквивалентную программу на языке Паскаль. Используя компилятор языка Паскаль выполнить результат трансляции.

11. Обработать транслятором программы на исходном языке, содержащие ошибки.

Варианты заданий

Вариант 1

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <операторы> ; <оператор>
   | <оператор>
4. <оператор> → READ (строка , идентификатор)
   | WRITE (строка , идентификатор)
5. <оператор> → идентификатор := <выражение>
6. <оператор> → IF <условие> THEN <оператор> ELSE <оператор>
7. <выражение> → идентификатор
   | число
   | <выражение> + <выражение>
   | <выражение> – <выражение>
8. <условие> → <условие> OR <терм>
   | <терм>
9. <терм> → <терм> AND <множитель>
   | <множитель>
10. <множитель> → NOT (<условие>)
    | (<условие>)
    | <отношение>
11. <отношение> → <выражение> операция_отношения <выражение>

Комментарии:

— идентификатор — имя переменной;
— все переменные — целочисленные;
— число — целочисленная константа;
— строка — строковая константа.

Вариант 2

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <оператор> ; <операторы>
   | <оператор>

4. <оператор> → READ (идентификатор)
   | WRITE (строка)
   | WRITE (идентификатор)
5. <оператор> → идентификатор := <выражение>
6. <оператор> → WHILE <отношение> DO BEGIN <операторы> END
7. <выражение> → <выражение> аддитивная_операция <терм>
   | <терм>
8. <терм> → <терм> мультипликативная_операция <множитель>
   | <множитель>
9. <множитель> → идентификатор
   | число
   | (<выражение>)
10. <отношение> → <выражение> операция_отношения <выражение>

Комментарии:

— идентификатор — имя переменной;
— все переменные — вещественные;
— число — вещественная константа;
— строка — строковая константа.

Вариант 3

1. <программа> → BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <оператор> <операторы>
   | <оператор>
4. <оператор> → VAR <переменные>;
5. <оператор> → READ идентификатор ;
   | WRITE идентификатор ;
6. <оператор> → <переменные> := <выражение>;
7. <оператор> → REPEAT <операторы> UNTIL <отношение>;
8. <выражение> → <выражение> + <терм>
   | <выражение> – <терм>
   | <терм>
9. <терм> → <терм> * <множитель>
   | <терм> / <множитель>
   | <множитель>

10. <множитель> → идентификатор
| число
| (<выражение>)
| – (<выражение>)
11. <отношение> → <идентификатор> операция_отношения <выражение>

Комментарии:

— идентификатор — имя переменной;
— все переменные — вещественные;
— число — вещественная константа;
— оператор READ перед вводом значения выводит имя переменной и символ = ;
— оператор WRITE перед выводом значения выводит имя переменной и символ = .

Вариант 4

1. <программа> → <переменные> ; <операторы>
2. <переменные> → идентификатор , <переменные>
| ε
3. <операторы> → <оператор> ; <операторы>
| <оператор>
4. <оператор> → READ идентификатор
| WRITE строка
| WRITE идентификатор
5. <оператор> → идентификатор := <выражение>
6. <оператор> → FOR идентификатор := <выражение> TO <выражение> BEGIN <операторы> END
7. <выражение> → <выражение> аддитивная_операция <терм>
| <терм>
8. <терм> → <терм> мультипликативная_операция <множитель>
| <множитель>
9. <множитель> → идентификатор
| число
| (<выражение>)

Комментарии:

— идентификатор — имя переменной;

— все переменные — целые;
— число — целочисленная константа;
— строка — строковая константа.

Вариант 5

1. <программа> → VAR <переменные> BEGIN <операторы>
2. <переменные> → идентификатор <переменные>
   | ε
3. <операторы> → <оператор> <операторы>
   | <оператор>
4. <оператор> → READ <переменные>
   | WRITE <переменные>
5. <оператор> → идентификатор := <выражение>
6. <оператор> → FOR идентификатор := <выражение> TO число STEP число BEGIN <операторы> END
7. <выражение> → <выражение> аддитивная_операция <терм>
   | <терм>
8. <терм> → <терм> мультипликативная_операция <множитель>
   | <множитель>
9. <множитель> → идентификатор
   | число
   | унарная_операция (<выражение>)

Комментарии:
— идентификатор — имя переменной;
— все переменные — целые;
— число — целочисленная константа;
— оператор READ перед вводом значения выводит имя переменной и символ = ;
— оператор WRITE перед выводом значения выводит имя переменной и символ = .

Вариант 6

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор

3. <операторы> → <оператор> ; <операторы>
   | <оператор>
4. <оператор> → READ (идентификатор)
   | WRITE (строка)
   | WRITE (идентификатор)
5. <оператор> → идентификатор <= <выражение>
6. <оператор> → <выражение> => идентификатор
7. <оператор> → WHILE <выражение> DO BEGIN <операторы> END
8. <выражение> → <выражение> аддитивная_операция <терм>
   | <терм>
9. <терм> → <терм> мультипликативная_операция <множитель>
   | <множитель>
10. <множитель> → идентификатор
    | число

Комментарии:

— идентификатор — имя переменной;

— все переменные — вещественные;

— число — вещественная константа;

— строка — строковая константа;

— операторы тела цикла выполняются, если значение выражения не равно нулю;

— операторы присваивания (<= и =>) записывают значение выражения в переменную, которая может находиться в левой или в правой части оператора.

Вариант 7

1. <программа> → BEGIN VAR <переменные>;<операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <операторы> <оператор>
   | <оператор>
4. <оператор> → READ <переменные> ;
   | WRITE <переменные> ;
5. <оператор> → идентификатор := <выражение>;
6. <оператор> → REPEAT <операторы> UNTIL <лог_выражение>;
7. <лог_выражение> → <лог_выражение> or <лог_терм>
   | <лог_терм>

8. <лог_терм> → <лог_терм> and <лог_множитель>
   | <лог_множитель>
9. <множитель> → (<отношение>)
   | not (<отношение>)
10. <отношение> → <выражение> сравнение <выражение>
11. <выражение> → <выражение> аддитивная_операция <терм>
   | <терм>
12. <терм> → <терм> мультипликативная_операция <множитель>
   | <множитель>
13. <множитель> → идентификатор
   | число

Комментарии:

— идентификатор — имя переменной;

— все переменные — вещественные;

— число — вещественная константа;

— оператор READ перед вводом значения выводит имя переменной и символ = ;

— оператор WRITE перед выводом значения выводит имя переменной и символ = .

Вариант 8

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <операторы> ; <оператор>
   | <оператор>
4. <оператор> → READ (строка , идентификатор)
   | WRITE (строка , идентификатор)
5. <оператор> → идентификатор := <выражение>
6. <оператор> → IF <условие> THEN <операторы> ELSE <операторы> END
7. <оператор> → IF <условие> THEN <операторы> END
8. <выражение> → идентификатор
   | число
   | операция (<выражение> , <выражение>)
9. <условие> → <выражение> сравнение <выражение>

Комментарии:

— идентификатор — имя переменной;
— все переменные — целочисленные;
— число — целочисленная константа;
— строка — строковая константа.

Вариант 9

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → идентификатор , <переменные>
   | идентификатор
3. <операторы> → <оператор> ; <операторы>
   | <оператор>
4. <оператор> → READ идентификатор
   | WRITE строка
   | WRITE <выражение>
5. <оператор> → идентификатор <выражение> :=
6. <оператор> → WHILE <выражение> DO <операторы> END
7. <выражение> → <выражение> <выражение> операция
   | идентификатор
   | число

Комментарии:

— идентификатор — имя переменной;
— все переменные — вещественные;
— число — вещественная константа;
— строка — строковая константа.

Вариант 10

1. <программа> → VAR <переменные> BEGIN <операторы> END
2. <переменные> → <переменные> , идентификатор
   | идентификатор
3. <операторы> → <оператор> ; <операторы>
   | <оператор>
4. <оператор> → READ (<переменные>)
   | WRITE (строка)
   | WRITE (<переменные>)

5. <оператор> → идентификатор := <выражение>
6. <оператор> → CASE <выражение> DO <варианты> END
7. <варианты> → вариант <варианты>
| вариант
8. <вариант> →число : <операторы> END
8. <выражение> → <выражение> аддитивная_операция <терм>
| <терм>
9. <терм> → <терм> мультипликативная_операция <множитель>
| <множитель>
10. <множитель> → идентификатор
| число
| (<выражение>)

Комментарии:

— идентификатор — имя переменной;
— все переменные — целочисленные;
— число — целочисленная константа;
— строка — строковая константа.

Контрольные вопросы

1. Дайте определение класса КС-языков.
2. Какие методы обработки КС-языков вы знаете?
3. Какие классы КС-языков можно обрабатывать нисходящим методом?
4. Какие классы КС-языков можно обрабатывать восходящим методом?
5. Как определяется множество ПЕРВ для символов грамматики?
6. Как определяется множество ПЕРВ для цепочки символов грамматики?
7. Как определяется множество СЛЕД для нетерминалов грамматики?
8. Что такое множество выбора и как его найти?
9. Что такое МП-распознаватель (МП-транслятор)?
10. Дайте определение транслирующей грамматики.
11. Как построить нисходящий МП-распознаватель (МП-транслятор) по заданной грамматике?
12. В чем заключается метод рекурсивного спуска?
13. Как построить восходящий МП-распознаватель (МП-транслятор) по заданной грамматике?

Библиографический список

1. *Ахо, А.* Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж. Ульман — М.: Мир, 1978. — Т. 1, 612 с. Т.2, 487 с.

2. *Ахо, А.* Компиляторы: принципы, технологиии и инструменты / А. Ахо, Р. Сети, Дж. Ульман — М.: Издательский дом “Вильямс”, 2003. — 768 с. : ил.

3. *Ахо, А.* Компиляторы: принципы, технологиии и инструменты / А. Ахо, Р. Сети, С.М. Лам, Дж. Ульман — М.: Издательский дом “Вильямс”, 2008. — 1184 с. : ил.

4. *Гордеев, А.В.* Системное программное обеспечение / А.В. Гордеев, А.Ю. Молчанов — СПб.: Питер, 2003. — 736 с.: ил.

5. *Карпов, В.Э.* Классическая теория компиляторов. Учебное пособие / В.Э. Карпов — Московский государственный институт электроники и математики. М., 2003. — 79 с.

6. *Карпов, Ю.Г.* Теория и технология программирования. Основы построения трансляторов / Ю.Г. Карпов — СПб.: БХВ-Питербург, 2005. — 284 с.: ил.

7. *Льюис, Ф.* Теоретические основы проектирования компиляторов / Ф. Льюис, Д. Розенкранц, Р. Стирнз — М.: Мир, 1979. — 656 с.

8. *Молчанов, А.Ю.* Системное программное обеспечение. Лабораторный практикум / А.Ю. Молчанов — СПб.: Питер, 2005. — 284 с.: ил.

9. *Молчанов, А.Ю.* Системное программное обеспечение: Учебник для вузов / А.Ю. Молчанов — СПб.: Питер, 2003. — 396 с.: ил.

10. *Опалева, Э.А.* Языки программирования и методы трансляции / Э.А. Опалева, В.П. Самойленко — СПб.: БХВ-Питербург, 2005. — 480 с.: ил.

11. *Серебряков, В.И.* Лекции по конструированию компиляторов / В.И. Серебряков — М.: МГУ, 1997. — 171 с.

12. *Соколов, А.П.* Системы программирования: теория, методы, алгоритмы: Учеб. пособие / А.П. Соколов — М.: Финансы и статистика, 2004. — 320 с.: ил.