ТЕОРИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Задания на лабораторные работы

Лабораторная работа № 1, 2. Разработка лексического анализатора

Задание. Для предложенного преподавателем варианта слов разработать лексический анализатор. На вход подаются два вида слов в любом порядке, разделенные любым количеством пробелов. Текст может быть многострочным. Текст может содержать комментарии. Шаблон комментария разработать самостоятельно. Комментарии и пробелы должны пропускаться.

Варианты заданий

N	Первое слово	Второе слово	Условие для второго слова
1	(000)*001(010)*	$(a\|b\|c\|d)^+$	В алфавитном порядке
2	(000)*001(100)*	$(a\|b\|c\|d)^+$	Первые два символа всегда ab
3	(000)*010(011)*	$(a\|b\|c\|d)^+$	Нет подстроки bb
4	(000)*010(100)*	$(a\|b\|c\|d)^+$	Не должно начинаться с ab
5	(000)*100(010)*	$(a\|b\|c\|d)^+$	Первые два символа всегда ac
6	(000)*101(110)*	$(a\|b\|c\|d)^+$	Первые два символа всегда ad
7	(001)*000(010)*	$(a\|b\|c\|d)^+$	Первые два символа всегда ba
8	(001)*010(011)*	$(a\|b\|c\|d)^+$	Если начинается с a, то не должно встретиться b
9	(001)*010(100)*	$(a\|b\|c\|d)^+$	Первые два символа всегда bc
10	(001)*010(101)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться cd
11	(001)*011(000)*	$(a\|b\|c\|d)^+$	Первые два символа всегда bd
12	(001)*100(101)*	$(a\|b\|c\|d)^+$	Если начинается с a то $(a)^+$
13	(001)*101(110)*	$(a\|b\|c\|d)^+$	Длина не более 4
14	(010)*000(001)*	$(a\|b\|c\|d)^+$	Первые два символа всегда ca
15	(010)*001(000)*	$(a\|b\|c\|d)^+$	Первые два символа всегда cb
16	(010)*011(100)*	$(a\|b\|c\|d)^+$	Нет подстроки bc
17	(010)*100(000)*	$(a\|b\|c\|d)^+$	Первые два символа всегда cd
18	(010)*101(110)*	$(a\|b\|c\|d)^+$	Если начинается с b, то не должно встретиться a
19	(010)*110(111)*	$(a\|b\|c\|d)^+$	В порядке обратном алфавитному
20	(011)*000(001)*	$(a\|b\|c\|d)^+$	Вторые два символа всегда ab
21	(011)*001(010)*	$(a\|b\|c\|d)^+$	Вторые два символа всегда ac
22	(011)*010(100)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться aa
23	(011)*100(101)*	(a|b)*a(a|b)	Вторые два символа всегда ab

24	(011)*101(110)*	$(a\|b\|c\|d)^+$	Вторые два символа всегда ba
25	(011)*111(000)*	$(a\|b\|c\|d)^+$	Если начинается с d, то не должно встретиться a
26	(100)*000(001)*	$(a\|b\|c\|d)^+$	Только b и d могут повторяться
27	(100)*001(010)*	$(a\|b\|c\|d)^+$	Вторые два символа всегда bc
28	(100)*010(011)*	$(a\|b\|c\|d)^+$	Вторые два символа всегда bd
29	(100)*011(100)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться cc
30	(100)*101(110)*	(a|b)(a|b|0|1)*	Если начинается с b, то не должно встретиться c
31	(100)*110(100)*	$(a\|b\|c\|d)^+$	Нет подстроки bd
32	(101)*000(001)*	$(a\|b\|c\|d)^+$	Если начинается с b, то не должно встретиться d
33	(101)*001(010)*	$(a\|b\|c\|d)^+$	Повторение символов может быть только рядом
34	(101)*010(011)*	$(a\|b\|c\|d)^+$	Если начинается с d, то не должно встретиться a
35	(101)*011(111)*	$(a\|b\|c\|d)^+$	Первый символ в слове всегда a
36	(101)*100(110)*	$(a\|b\|c\|d)^+$	Первый символ в слове всегда b
37	(101)*110(111)*	$(a\|b\|c\|d)^+$	Если начинается с d, то не должно встретиться b
38	(110)*000(001)*	$(a\|b\|c\|d)^+$	Первый символ в слове всегда c
39	(110)*010(011)*	$(a\|b\|c\|d)^+$	Нет подстроки ac
40	(110)*011(100)*	$(a\|b\|c\|d)^+$	Если начинается с d, то не должно встретиться c
41	(110)*100(101)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться dd
42	(110)*101(111)*	$(a\|b\|c\|d)^+$	Первый символ в слове всегда d
43	(110)*111(000)*	$(a\|b\|c\|d)^+$	Нет подстроки ad
44	(111)*000(001)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться da
45	(111)*010(011)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться bd
46	(111)*011(100)*	$(a\|b\|c\|d)^+$	Нет подстроки aa
47	(111)*100(001)*	$(a\|b\|c\|d)^+$	Нет подстроки ab
48	(111)*101(110)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться bc
49	(111)*110(000)*	$(a\|b\|c\|d)^+$	Не должно заканчиваться cb

Обозначения

*- множество всех строк, включая, пустую строку.

$^{+-}$ множество всех строк, исключая пустую строку.

(,), | – символы метаязыка, не являются частью описываемого языка.

В дальнейшем слова первого вида будут называться также числами, а слова второго вида - идентификаторами.

Методические указания

1. В таблице вариантов заданий используются регулярные выражения. Построение регулярных выражений рассмотрено в разделе 4.2 конспекта лекций.
2. Реализация диалога с пользователем выполняется с помощью экранной формы. Экранная форма должна содержать следующие визуальные компоненты:

   а) поле класса TMemo для ввода и редактирования исходного текста;

   б) поле класса TEdit для вывода диагностических сообщений работающее в режиме «Только чтение»;

   в) кнопка класса TButton для запуска процесса трансляции;

   г) поля класса TLabel для размещения на форме названий (назначения) текстовых полей.
3. Диалог с пользователем оформляется в отдельном программном модуле на базе класса TForm.
4. Оформить лексический анализатор в виде класса, расположенного в отдельном программном модуле.
5. Рекомендуется включать в класс «Лексический анализатор» следующие свойства и функции:

__property TStrings *stgsPSourceCode={write = stgsFSourceCode}; - свойство содержит ссылку на поле TMemo, содержащее исходный текст.

__property TToken enumPCurrentToken={read=enumFToken}; -

__property AnsiString strPLexicalUnit={read=strFLexicalUnit}; - свойство содержит текст распознанного слова.

__property TState enumPState={read=enumFState, write=enumFState};

__property Cardinal cardPSourceCodeRowSelection = {read=cardFSourceCodeRowSelection, write=cardFSourceCodeRowSelection}; - свойство содержит текущий номер строки обрабатываемого исходного текста.

__property Cardinal cardPSourceCodeColSelection= {read=cardFSourceCodeColSelection, write=cardFSourceCodeColSelection}; - свойство содержит текущий номер колонки в текущей строке обрабатываемого исходного текста.

void NextToken(); - функция, которая обращается за очередным словом в исходном тексте, выделяет в тексте и классифицирует.

6. При обнаружении лексической ошибки целесообразно возбуждать исключительную ситуацию, которая будет обрабатываться в главной форме программы.

7. В дальнейшем объект «Лексический анализатор» удобно размещать в классе «Синтаксический анализатор» в виде локального поля.

План работы

1. Составить регулярную грамматику для каждого вида слов.
2. Построить конечные автоматы для каждого вида слов.
3. Построить детерминированные конечные автоматы для каждого вида слов.
4. Составить объединенный конечный автомат.
5. Спроектировать и отладить экранную форму для ввода исходных данных, вывода сообщений программы и управления программой.
6. Написать и отладить модуль лексического анализатора по алгоритму объединенного конечного автомата. Предусмотреть транслитератор (читатель литер). Обработчик лексических ошибок исходного текста.
7. Для отладки лексического анализатора временно включить в программу цикл чтения слов исходного текста и вывода результатов лексического анализа.

СОДЕРЖАНИЕ ОТЧЕТА

Во всех отчетах предусмотреть титульный лист и текст задания, включающий вариант задания.

а) описание регулярной грамматики;

б) описание недетерминированного конечного автомата;

в) описание детерминированного конечного автомата;

г) описание объединенного алгоритма;

д) исходный текст программы;

е) результаты тестирования.

ОЦЕНКА РЕЗУЛЬТАТОВ

В дисциплинах цикла «Программирование» конечным результатом обучения является приобретение профессиональных навыков разработки программ. Поэтому результаты лабораторных работ напрямую формируют аттестационные баллы.

Отчет считается недействительным при отсутствии исходного текста программы, за остальные недочеты снижаются баллы путем применения корректировочных коэффициентов.

Критерии оценки лабораторной работы № 1,2 «Разработка лексического анализатора»

Задание	**Оценка (балл)**
Построить конечный автомат для слова 1	5
Построить конечный автомат для слова 2	5
Реализовать и отладить лексический анализатор	10
Всего	20

Корректировочные коэффициенты при оценке программы

Степень выполнения	Полученные результаты	Коэффициент
1	Нет исходного текста программы	0,0
2	Задание сдается после соответствующей аттестации	0,0
3	Есть исходный текст программы, но нет чистой трансляции	0,2
4	Лабораторная работа сделана не самостоятельно	0,2

5	Имеются динамические ошибки	0,8
6	За каждое невыполненное требование к заданию	0,8
7	Имеются серьезные недоработки.	0,5-0,8
8	Имеются погрешности в работе программы	0,9
9	Программа работает верно, замечания по защите отсутствуют.	1,0
10	В программе реализованы не предусмотренные заданием возможности	1,1
11	Представлено оригинальное решение задачи	1,2