1 Лабораторная работа «Построение лексического анализатора»

1.1 Основные понятия лексического анализа

Лексический анализ всегда предшествует синтаксическому анализу и заключается в предварительной обработке программы и ее перекодировании к виду, удобному для синтаксического анализа. Лексический анализ принято отделять от синтаксического анализа для сокращения времени компиляции, поскольку синтаксис (морфология) слов всегда проще, чем синтаксис программ.

Лексический анализ сводится к разбиению текста программы на лексические единицы (слова, лексемы) – конструкции, которые выступают в качестве терминальных символов для синтаксического анализа. Лексемы еще иногда называют символами или атомами.

Большинство лексем в языках программирования можно сгруппировать в следующие классы:

- класс идентификаторов;
- класс служебных слов;
- класс констант (числовых или символьных);
- класс операций (одно-, дву- или многолитерных);
- класс разделителей (однолитерных или двулитерных).

В лексическом анализе лексема представляется в виде пары (класс, значение). Значение может быть непосредственным (литеральным) или представлять собой ссылку на некоторую таблицу, в которой хранятся значения лексем.

1.2 Лексемы простого PL-подобного языка программирования

Для каждого из классов можно построить автоматную грамматику, полностью описывающую правила построения его лексем. Например, грамматика, описывающая идентификаторы, может быть построена следующим образом:

<Идентификатор>::= *Буква* | <Идентификатор> *Буква* |

| <Идентификатор> *Цифра*

Для чисел с фиксированной точкой:

<Число>::= <Целое> | <Смешанное число>

<Смешанное число>::= <Точка> ***Цифра*** | <Целое> . |

<Смешанное число> ***Цифра***

<Целое>::= ***Цифра*** | <Целое> ***Цифра***

<Точка>::= .

Для разделителей {„/“ , „//“ , „/*“ , „„“ , „;“}:

<Разделитель>::= / | / <Второй символ> | , | ;

<Второй символ>::= / | *

Как выше указывалось, во внутреннем представлении все лексемы имеют вид пары, в которой первый элемент – это признак принадлежности к классу (буква или цифра), а второй – значение в описанном выше понимании.

Рассмотрим пример построения лексического анализатора для языка, являющегося некоторым подмножеством языка PL/1. Определим его конструкции:

1. *Идентификатор* – любая последовательность букв и цифр, начинающаяся с буквы;
2. *Описание данных* (ограничимся только одним типом числовых данных DEC FIXED, то есть вещественное число с фиксированной точкой) имеет вид:

```
DCL Список переменных DEC FIXED;
```

где *Список переменных* – это идентификатор или перечисление идентификаторов через запятую в круглых скобках.

3. *Описание процедур* имеет вид

```
PROC Имя процедуры;
   Тело процедуры
END;
```

4. *Операторы*:
   - присваивания =;
   - безусловного перехода имеет вид

     GOTO ***Идентификатор;***
   - условный логический оператор

     IF ***Условие*** THEN ***Оператор;***
   - оператор конца процедуры END;
5. *Выражения*:
   - операции

     сложения +;

     умножения *;

     отношений <,>, <>;
   - скобки (,);
6. *Константы*:
   - числовые (целые и вещественные с фиксированной точкой);
   - символьные (произвольная последовательность символов, заключенная в апострофы);
7. *Метки* вида

   Идентификатор: ;
8. *Разделители*:

   пробел, конец строки, **« , » « ; » « . »**
9. *Комментарии*: произвольная последовательность символов от знака // до конца строки.

Построим таблицу классов лексем для рассматриваемого языка (табл. 1.1).

Таблица 1.1. Классы лексем подмножества языка PL/1.

Типы лексем	Обозначение
Служебные слова	W
Идентификаторы	I
Операции	O
Разделители	R
Константы	N
	C

Для оптимизации и ускорения поиска данных часто таблицу констант разделяют на две: таблицу N (для числовых констант) и таблицу C (для символьных констант).

1.3 Функции и таблицы лексического анализа

Основная функция лексического анализатора (сканера) состоит в выделении лексем из исходной программы и передаче их синтаксическому анализатору в некотором внутреннем представлении. При этом лексический анализатор использует таблицы, соответствующие классам лексем. Таблицы служебных слов, разделителей и операций определяются входным языком и должны быть сформированы при построении сканера, они являются постоянными и неизменяемыми. Таблицы идентификаторов и констант являются временными и создаются непосредственно в процессе лексического разбора исходной программы.

Комментарии в процессе лексического анализа игнорируются, так как они специфицируют исходную программу только для программиста и не влияют на семантику программы.

Для рассматриваемого языка сформированные таблицы служебных слов (табл. 1.2), таблица операций (табл. 1.3) и таблица разделителей (табл. 1.4) имеют вид:

Таблица 1.2

Служебное слово	Код
DCL	2
END	3
FIXED	4
GOTO	5
IF	6
PROC	7
THEN	8
MAIN	9

Таблица 1.3

Операция	Код
+	1
*	2
<	3
>	4
=	5
:	6
<>	7

Таблица 1.4

Разделитель	Код
пробел	1
,	2
;	3
(	4
)	5
.	6
конец строки	7

Каждая лексема, обработанная сканером, преобразуется к виду:

<буква><код>,

где: <буква> – это признак класса лексемы (W, I, O, R, N или C),

<код> – номер лексемы в соответствующей таблице.

Например, если на вход сканера поступила цепочка вида

```
IF C > 2.79 THEN B = A * E3;
```

то на выходе сканера будет сформирована следующая последовательность кодов лексем:

Код лексемы	Класс лексемы	Значение лексемы
W6	Служебное слово	IF
I1	Идентификатор	C
O4	Операция	>
N1	Число	2.79
W8	служебное слово	THEN
I2	Идентификатор	B
O5	Операция	=
I3	Идентификатор	A
O2	Оператор	*
I4	Идентификатор	E3
R3	Разделитель	;

При лексическом разборе входной цепочки пробелы не кодируются как разделители и в выходную цепочку не попадают. Это связано с тем, что в исходном тексте программы пробелы служат только для структурирования текста программы и не несут дополнительной смысловой нагрузки.

Рассмотрим работу сканера на примере исходной программы, представленной ниже:

```
//Вычисление суммы и произведения
PROC MAIN;
    DCL (A1, A2) DEC FIXED;
    A1=378;

    A2=.73;
```

```
        PROC CALC;
             DCL (SUM, MULT) DEC FIXED;
             IF A1+A2<>3.2 THEN GOTO P;
             SUM=(A1+A2)*A2;
        P:   MULT=A1*A2;
        END;
    END.
```

Для данной исходной программы лексический анализатор сформирует следующую выходную последовательность лексем:

Входная последовательность текста исходной программы	Выходная последовательность лексем во внутреннем представлении
//Вычисление суммы и произведения	
PROC MAIN;	W7 W9 R3
DCL (A1, A2) DEC FIXED;	W2 R4 I1 R2 I2 R5 W1 W4 R3
A1=378;	I1 O5 1 R3
A2=.73;	I2 O5 N2 R3
PROC CALC;	W7 I3 R3
DCL SUM, MULT DEC FIXED;	W2 R4 I4 R2 I5 R5 W1 W4 R3
IF A1+A2<>3.2 THEN GOTO P;	W6 I1 O1 I2 O7 N3 W8 W5 I6 R3
SUM=A1+A2;	I4 O5 I2 O1 I1 R3
P: MULT=A1*(A2+SUM);	I6 O6 I5 O5 I1 R4 I2 O1 I4 R5 R3
END;	W3 R3
END.	W3 R6

И, кроме того, лексический анализатор сформирует таблицу чисел N (табл. 1.5) и таблицу идентификаторов I (табл. 1.6).

Таблица 1.5

Код	Число
1	378
2	.73
3	3.2

Таблица 1.6

Код	Идентификатор	Номер процедуры	Уровень процедуры	Номер идентификатора в процедуре	Тип идентификатора	Объем памяти
1	A1					
2	A2					
3	CALC					
4	SUM					
5	MULT					
6	P					

1.4 Диаграмма состояний лексического процессора

Лексический анализатор представляет собой процессор или, иначе говоря, сканер (конечный автомат) для разбора и классификации лексем, связанный с некоторыми семантическими процедурами. На вход такого процессора последовательно подаются символы исходной программы, причем каждый входной символ вызывает изменение состояния сканера.

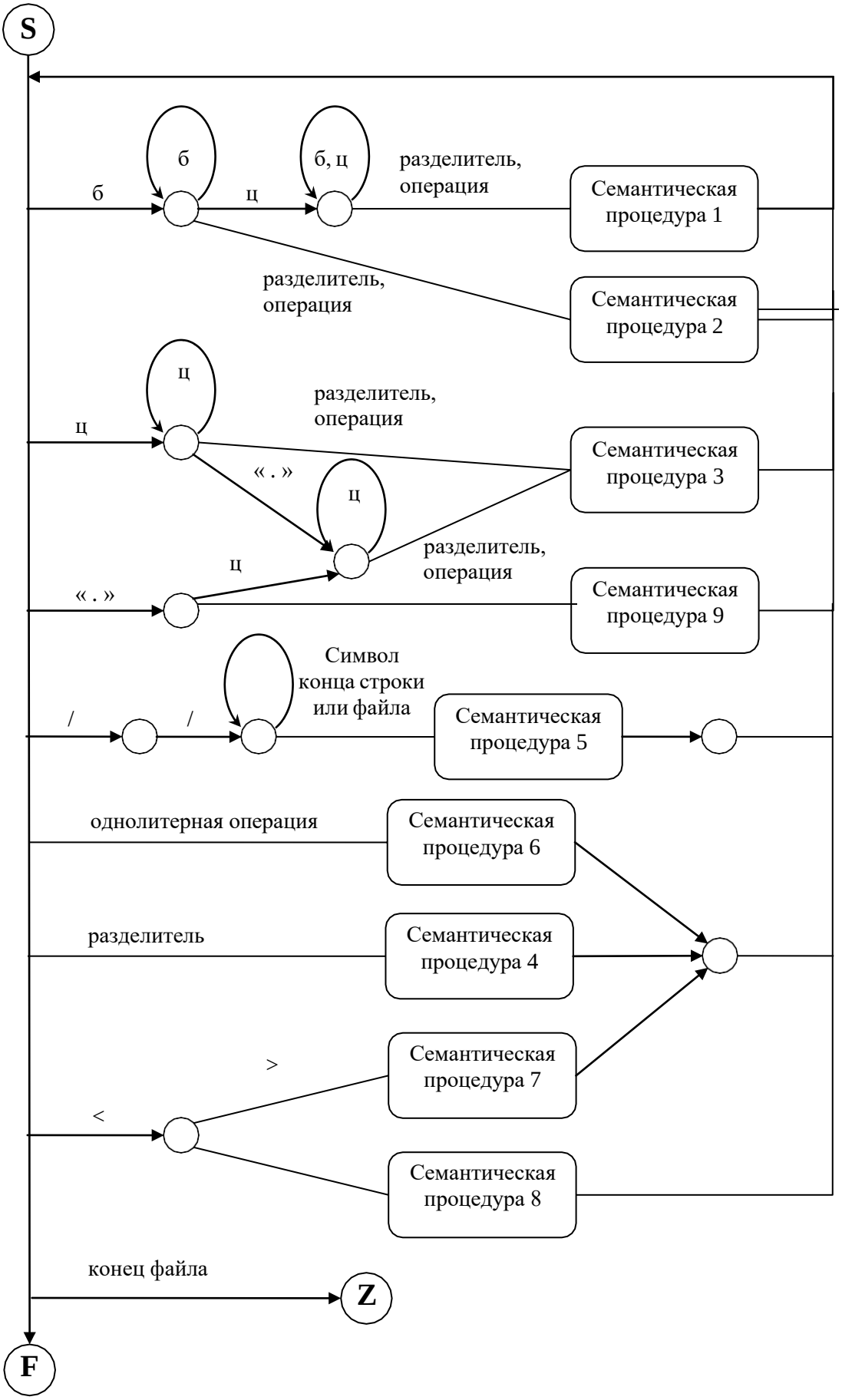
Если анализируемый символ означает конец разбираемой лексемы, то с переходом связывается некоторая семантическая процедура, позволяющая либо определить код лексемы по таблице (в случае служебных слов, разделителей, операций), либо пополнить таблицы (для констант и идентификаторов), а затем в выходную строку выдать очередной код лексемы во внутреннем представлении.

Для эффективного разбора и оптимизации сканера, а также для его программной реализации в алфавите автомата выделяют несколько подмножеств символов, по которым сканер выполняет одинаковые переходы и действия. В нашем случае такими подмножествами являются:

- буквы {A, …, Z};
- цифры {0, …, 9};
- символы однолитерных операций { + , * , > , = , : };
- символы, служащие началом двулитерных операций {<};
- разделители { пробел , конец строки , « , » , « ; » };
- точка (она играет в нашем примере двойную роль: десятичная точка в числе и точка в конце программы).

В некоторых языках программирования точка не имеет роли самостоятельного разделителя.

С учетом вышесказанного диаграмма состояний сканера для рассматриваемого языка имеет вид, представленный на рисунке 1.1.


S
б
б
б, ц
ц
разделитель,
операция
Семантическая
процедура 1
разделитель,
операция
Семантическая
процедура 2
ц
ц
разделитель,
операция
Семантическая
процедура 3
« . »
ц
ц
разделитель,
операция
« . »
Семантическая
процедура 9
Символ
конца строки
или файла
/
/
Семантическая
процедура 5
однолитерная операция
Семантическая
процедура 6
разделитель
Семантическая
процедура 4
>
Семантическая
процедура 7
<
Семантическая
процедура 8
конец файла
Z
F

На данной диаграмме состояния сканера представлены вершинами, а переходы между состояниями - дугами (направленными линиями). Каждый переход связан с чтением очередного символа их текста входной программы. Поэтому дуга взвешена (помечена) символом или множеством символов, которые вызывают данный переход. Если в диаграмме состояний есть невзвешенная дуга, ведущая из какого-либо состояния, то считается, что она взвешена любыми символами кроме тех, которыми взвешены другие дуги, исходящие из данного состояния. Это позволяет не перегружать диаграмму лишними символами. Скругленный прямоугольник в разрыве дуги указывает на семантическую процедуру, выполняемую при данном переходе. Если переход не сопровождается семантической процедурой, то текущий символ добавляется к буферу, в котором формируется лексема. Работа каждой семантической процедуры завершается очищением буфера.

Сканер всегда содержит три стандартных состояния:

S – начальное состояние сканера. Лексический разбор всегда начинается из этого состояния;

Z – заключительное состояние сканера. Если в процессе разбора достигнуто данное состояние, это означает, что разбор успешно завершен;

F – состояние ошибки. Если встретился символ, не входящий во входной алфавит, то сканер переходит в состояние ошибки и прекращает разбор. Кроме того, если в каком-либо состоянии на входе появился символ, по которому не предусмотрен переход из этого состояния, сканер также переходит в состояние ошибки (на диаграмме эти переходы не изображаются, чтобы избежать лишнего загромождения рисунка).

За семантическими процедурами процессора закреплены следующие функции:

Семантическая процедура 1: Провести поиск сформированного слова в таблице идентификаторов. Если такое слово в таблице

идентификаторов не найдено, то занести сформированное слово в таблицу идентификаторов. Сформировать и выдать в выходную последовательность лексему идентификатора во внутреннем представлении.

Семантическая процедура 2: Провести поиск сформированного слова в таблице служебных слов. Если такое слово в таблице служебных слов не найдено, то выполнить Семантическую процедуру 1, иначе сформировать и выдать в выходную последовательность лексему служебного слова во внутреннем представлении.

Семантическая процедура 3: Занести сформированное слово в таблицу констант. Сформировать и выдать в выходную последовательность лексему константы во внутреннем представлении.

Семантическая процедура 4: Провести поиск текущего символа в таблице разделителей. Сформировать и выдать в выходную последовательность лексему разделителя во внутреннем представлении.

Семантическая процедура 5: Удалить сформированную последовательность символов.

Семантическая процедура 6: Провести поиск текущего символа в таблице операций. Сформировать и выдать в выходную последовательность лексему операции во внутреннем представлении.

Семантическая процедура 7: Добавить текущий символ к сформированному слову. Провести поиск сформированного слова в таблице операций. Если такое слово в таблице операций не найдено, то перейти в состояние ошибки. Если поиск успешен, то сформировать и выдать в выходную последовательность лексему операции во внутреннем представлении.

Семантическая процедура 8: Провести поиск сформированного слова в таблице операций. Сформировать и выдать в выходную последовательность лексему операции во внутреннем представлении.

Семантическая процедура 9: Провести поиск сформированного слова в таблице разделителей. Сформировать и выдать в выходную последовательность лексему разделителя во внутреннем представлении.