Федеральное агентство по образованию РФ
Пермский Национальный Исследовательский Политехнический Университет
Кафедра Информационных Технологий и Автоматизированных Систем

Кузнецов Д.Б., Вагин Д.А.

Теория языков программирования и методы трансляции

Методические указания по выполниению лабораторных работ
для специальности ПОВТ

г. Пермь 2011

1 Введение

Предлагаемый лабораторный практикум предназначен для приобретение знаний и навыков в области использования и разработки как трансляторов в целом, так и отдельных элементов — лексического, синтаксического и семантического анализаторов, генератора выходного кода.

1.1 Требования к аппаратному обеспечению

Вариант 1: Выполненние работ на локальном компьютере Компьютер должен обеспечивать возможность запуска операционной системы, текстового редактора, и средств компиляции исходного кода. Необходимо иметь 200МБ свободного места на жёстком диске для установки необходимого по и хранения исходных текстов программ.

Вариант 2: Выполненние работ на удалённом сервере Сервер должен иметь технические характеристики достаточные для запуска операционной системы и сервисов для удалённого подключения. Более точные аппартные характеристики зависят от версии ядра и системных библиотек. Для запуска последних версий без графического режима вполне достаточно процессор с частотой от 300 МГц, ОЗУ от 512 МБ, свободно дисковое пространство от 500 МБ. Локальное рабочее место должно быть оборудовано терминалом, подключенным к серверу.

1.2 Требования к системному программному обеспечению

Рассмариваемое ниже системное программное обеспечение должно быть установлено на локальном компьютере (Вариант 1 аппаратного обеспечения) или на сервере (Вариант 2 аппаратного обеспечения).

Для выполнения лабораторных работ потребуется инструментарий для сборки компиляции программ на языке C — GNU gcc, а так же консольные программы yacc и lex.

2 Лабораторная работа №1
Построение лексического анализатора на основании автомат грамматики

2.1 Цель

- Научитсья строить лексические анализаторы на основе автоматной грамматики

2.2 Порядок выполнения

1 Построить автоматную грамматику

2 Построить автомат

3 Привести к детерминированному автомату

4 Реализовать автомат программно на языке программирования C

5 Написать отчет

2.3 Рекомендации по выполнению

- Используйте массивы фиксированной длины
- Задавайте данные внутри исходного кода

2.4 Состав отчета

- Титульный лист (фамилия, группа, номер варианта, наименование работы, задание)
- Текст задания
- Грамматика
- Диаграмма переходов
- Автоматы
- Текст программы

2.5 Варианты заданий

В задании указано содержательное описание грамматики и простейший пример для облегчения понимания.

1 Одинаковые символы стоят парами: *aabbaabbbbaa*

2 В начале строки *a*: *ababbbabb*

3 Первый символ — не важен, далее одни *a*: *baaaaaaaaa*, *aaaaaaaaaa*

4 Либо одни *a*, либо одни *b*: *aaaaaaaaaa*, *bbbbbbbbb*

5 В конце строки *b*: *ababbbabb*

6 Одинаковые символы не должны стоять рядом: *abababababab*, *bababababa*

7 В строке должна встретиться хотя бы одна буква a: *bbbbbabbb*, *aaaaaaaa*

8 Предпоследним символом строки должна быть b, *abbabaabb*

9 Вторым символом строки должна быть a: *baaaaabbb*

10 Два последних символа должны быть b: *abababb*, *bbbbbb*

11 Первый и третий символы должны быть разными: *aabbbabab*, *baaabbbab*

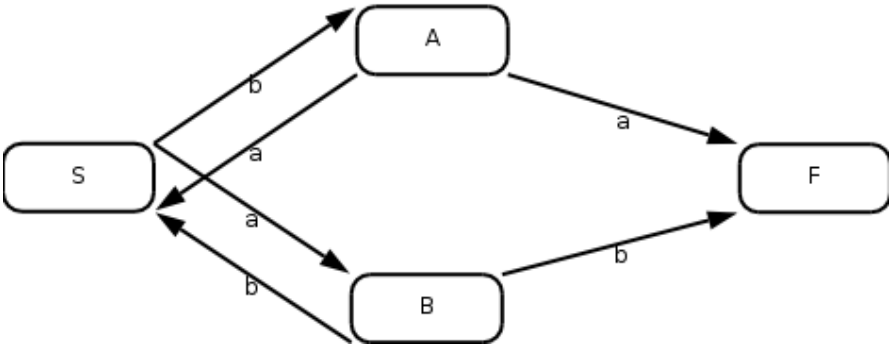
12 Первый и последний символы должны быть одинковыми: *ababababa*, *babbbabab*

2.6 Пример

Задание Символы a и b стоят парами: abbaabab, baabbaba.

Грамматика Построим грамматику по заданию:
$S \to aB$
$S \to bA$
$B \to bF$
$A \to aF$
$B \to bS$
$A \to aS$
$F \to \dashv$

Диаграмма переходов



	−	−	−	+
	S	A	B	F
a	B	S, F		
b	A		S, F	

Детерминированный автомат

	−	−	−	+	−	+
	S	A	B	F	$\{\}$	$\{S, F\}$
a	B	$\{S, F\}$	$\{\}$	$\{\}$	$\{\}$	B
b	A	$\{\}$	$\{S, F\}$	$\{\}$	$\{\}$	A

Программа Напишем программу на языке программирования C. Откомпилировать её можно компилятором GNU gcc так: *gcc -o program lab1.c*

```c
#include <stdio.h>

#define S  0 /* S */
#define A  1 /* A */
#define B  2 /* B */
#define F  3 /* F */
#define H  4 /* {S,F} */
#define W  5 /* {} */

#define P  1 /* + */
#define M  0 /*   */

#define a 0 /* a */
#define b 1 /* b */

struct action
{
    int sos; /* состояние */
    int out; /* вывод */
};

int main()
{
    int table[2][6];
    table[a][S] = B;
    table[b][S] = A;
    table[a][A] = H;
    table[b][A] = W;
    table[a][B] = W;
    table[b][B] = H;
    table[a][F] = W;
    table[b][F] = W;
    table[a][W] = W;
    table[b][W] = W;
    table[a][H] = B;
    table[b][H] = A;

    int output[6];
    output[S] = M;
    output[A] = M;
    output[B] = M;
    output[F] = P;
    output[W] = M;
    output[H] = P;


    int input[]={b,a,a,b};
    int n = 4, i;
    int isos = S;

    for( i=0; i<n; ++i)
    {
        isos=table[input[i]][isos];
    }

    printf("%d\n", output[isos]);
    return 0;
}
```

Listing 1: анализатор

3 Лабораторная работа №2 Построение лексического анализатора с использованием lex

3.1 Цели

- Познакомиться с «регулярными выражениями»
- Познакомиться с программой lex
- Научиться строить лексические анализаторы с использованием lex

3.2 Порядок выполнения

1 Построить регулярное выражение

2 Составить файл для lex

3 Получить программу на Си

4 Откомпилировать и запустить

5 Написать отчет

3.3 Состав отчёта

- Титульный лист (фамилия, группа, номер варианта, наименование работы, задание)
- Текст задания
- Текст программы на lex
- Результаты работы программы

3.4 Варианты заданий

В задании указано содержательное описание грамматики и простейший пример для облегчения понимания.

1 Одинаковые символы стоят парами: *aabbaabbbbaa*

2 В начале строки *a*: *ababbbabb*

3 Первый символ — не важен, далее одни *a*: *baaaaaaaaa*, *aaaaaaaaaa*

4 Либо одни *a*, либо одни *b*: *aaaaaaaaaa*, *bbbbbbbbb*

5 В конце строки *b*: *ababbbabb*

6 Одинаковые символы не должны стоять рядом: *abababababab*, *bababababa*

7 В строке должна встретиться хотя бы одна буква *a*: *bbbbbabbb*, *aaaaaaaaa*

8 Предпоследним символом строки должна быть *b*, *abbabaabb*

9 Вторым символом строки должна быть *a*: *baaaaabbb*

10 Два последних символа должны быть *b*: *abababb*, *bbbbbb*

11 Первый и третий символы должны быть разными: *aabbbabab*, *baaabbbab*

12 Первый и последний символы должны быть одинковыми: *ababababa*, *babbbabab*

3.5 Пример

Задание Символы *a* и *b* стоят парами: *abbaabab*, *baabbaba*.

Регулярное выражение Регулярное выражение будет иметь вид: (ab|ba)+

Программа на lex Создадим файл lab3.l со следующим содержимым:

```
%%
(ab|ba)+ { printf("Yes");}
.* { printf("No"); }
%%

yyerror(char *str)
{ printf(str); }

main()
{ yylex(); }
```

Listing 2: lab3.l

Программу на C получим простой командой. lab3.c - имя выходного файла, lab3.l имя входного файла на lex.

```
flex -o lab3.c lab3.l
```

Для компиляции воспользуемся компилятором gcc.

```
gcc -o lab3 lab3.c -lfl
```

Запуск осуществляется так:

```
./lab3
ababbababa
Yes

bbbaa
No
```

4 Лабораторная работа №3 Построение синтаксического анализатора на основании LL(1) грамматики

4.1 Порядок выполнения

1 Построить LL(1) грамматику

2 Определить множества выбора для каждого правила

3 Изобразить низходящую схему разбора

4 Разработать лексический анализатор на базе lex

5 Построить МП-автомат по грамматике

6 Реализовать МП-автомат

7 Реализовать синтаксический анализ методом рекурсивного спуска

4.2 Варианты заданий

В задании указано содержательное описание грамматики и простейший пример для облегчения понимания.

1 конструкция for языка c++.

```
for(
    a=0, b=76;
    i<10 && b>0 ;
    ++i , b=i+1
){
    b++;
    i++;
}
```

Listing 3: for

2 конструкция if языка c++.

```
if(a==0 && b<=0)
{
    a=b; b++;
}
else
{
    a++;
    b+=2;
}
```

Listing 4: if

3 конструкция switch языка c++.

```
switch(a){
    case 1: a=1;
    case 3:
    case 2: a++; break;
    default: a=0;
}
```

Listing 5: switch

4 тэг img языка разметки HTML:

```
<img src="/images/a.png" width="20" height="30"
     alt="картинка" />
```

Listing 6: img

5 Объявление функции в c++:

```
int funcname(int a, char b, float c = 0.1);
```

Listing 7: func

6 тэг table языка разметки HTML:

```
<table>
    <tr>
        <th>заголовок 1</th>
        <th>заголовок 2</th>
    </tr>
    <tr>
        <td>ячейка 1</td>
        <td>ячейка 2</td>
    </tr>
    <tr>
        <td>ячейка 3</td>
        <td>ячейка 4</td>
    </tr>
</table>
```

Listing 8: table

7 тэг ul (ненумерованный список) языка разметки HTML:

```
<ul>
    <li>item 1</li>
    <li>item 2</li>
    <li>item 3</li>
</ul>
```

Listing 9: ul

8 конструкция while языка c++.

```
while( a>0 || b<76){
    b++;
    a++;
}
```

Listing 10: while

9 конструкция do-while языка c++.

```
do{
    b++;
    a++;
}while( a>0 || b<76);
```

Listing 11: do-while

10 конструкция insert языка запросов SQL:

```
insert into tablename(pk,column1,column2)
values (1, 2 , 'varchar');
```

Listing 12: insert

11 конструкция select языка запросов SQL:

```
select pk, column1, column2
from tablename
where column1 = 2
    and column2 like 'xxx%';
```

Listing 13: select

12 конструкция delete языка запросов SQL:

```
delete from tablename
where column1 = 1
    or column2 like 'xxx%';
```

Listing 14: delete

13 тэг form языка разметки HTML:

```
<form methid="post" action="/registration/">
    <input type="text" name="login" />
    <input type="text" name="email" />
    <input type="password" name="password" />
    <input type="submit"/>
</form>
```

Listing 15: form

14 конструкция if языка Pascal:

```
if (a <= 10) or (b >= 2) then
begin
    a := 10;
    b := 10;
end else
    a := b;
```

Listing 16: if

15 конструкция for языка Pascal:

```
for i:= 0 to 10 step 2 do
begin
    a := i;
end;
```

Listing 17: for

4.3 Пример реализации

- Пример МП автомата: http://www.softcraft.ru/translat/lect/t07-07.shtml
- Пример программы с использованием рекурсивного спуска: http://www.softcraft.ru/translat/lect
08.shtml

5 Лабораторная работа №4 Построение синтаксического анализатора на основании LR грамматики

5.1 Порядок выполнения

1 Построить грамматику с предшествованием

2 Определить отношения предшествования

3 Изобразить восходящую схему разбора

4 Выполнить свертку заданного примера

5 Разработать синтаксический анализатор на базе yacc

6 Разработать лексический анализатор на базе lex

7 Выполнить синтаксический анализ заданного примера

5.2 Варианты заданий

В задании указано содержательное описание грамматики и простейший пример для облегчения понимания.

1 конструкция for языка c++.

```
for(
    a=0, b=76;
    i<10 && b>0 ;
    ++i , b=i+1
){
    b++;
    i++;
}
```

Listing 18: for

2 конструкция if языка c++.

```
if(a==0 && b<=0)
{
    a=b; b++;
}
else
{
    a++;
    b+=2;
}
```

Listing 19: if

3 конструкция switch языка c++.

```
switch(a){
    case 1: a=1;
    case 3:
    case 2: a++; break;
    default: a=0;
}
```

Listing 20: switch

4 тэг img языка разметки HTML:

```
<img src="/images/a.png" width="20" height="30"
     alt="картинка" />
```

Listing 21: img

5 Объявление функции в c++:

```
int funcname(int a, char b, float c = 0.1);
```

Listing 22: func

6 тэг table языка разметки HTML:

```
<table>
    <tr>
        <th>заголовок 1</th>
        <th>заголовок 2</th>
    </tr>
    <tr>
        <td>ячейка 1</td>
        <td>ячейка 2</td>
    </tr>
    <tr>
        <td>ячейка 3</td>
        <td>ячейка 4</td>
    </tr>
</table>
```

Listing 23: table

7 тэг ul (ненумерованный список) языка разметки HTML:

```
<ul>
    <li>item 1</li>
    <li>item 2</li>
    <li>item 3</li>
</ul>
```

Listing 24: ul

8 конструкция while языка c++.

```
while( a>0 || b<76){
    b++;
    a++;
}
```

Listing 25: while

9 конструкция do-while языка c++.

```
do{
    b++;
    a++;
}while( a>0 || b<76);
```

Listing 26: do-while

10 конструкция insert языка запросов SQL:

```
insert into tablename(pk,column1,column2)
values (1, 2 , 'varchar');
```

Listing 27: insert

11 конструкция select языка запросов SQL:

```
select pk, column1, column2
from tablename
where column1 = 2
    and column2 like 'xxx%';
```

Listing 28: select

12 конструкция delete языка запросов SQL:

```
delete from tablename
where column1 = 1
    or column2 like 'xxx%';
```

Listing 29: delete

13 тэг form языка разметки HTML:

```
<form methid="post" action="/registration/">
    <input type="text" name="login" />
    <input type="text" name="email" />
    <input type="password" name="password" />
    <input type="submit"/>
</form>
```

Listing 30: form

14 конструкция if языка Pascal:

```
if (a <= 10) or (b >= 2) then
begin
    a := 10;
    b := 10;
end else
    a := b;
```

Listing 31: if

15 конструкция for языка Pascal:

```
for i:= 0 to 10 step 2 do
begin
    a := i;
end;
```

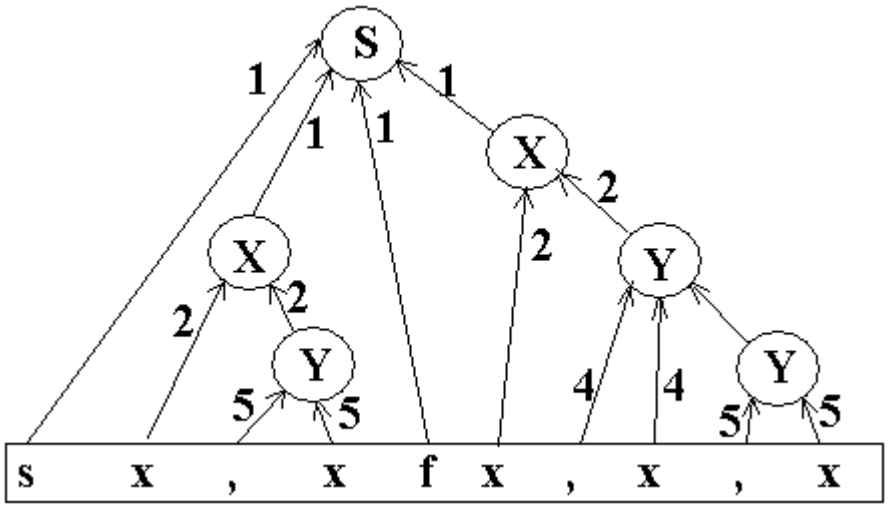
Listing 32: for

5.3 Пример реализации

Построим грамматику
$S \to sXfX$
$X \to xY$
$X \to x$
$Y \to, xY$

$Y \rightarrow, x$

	s	X	Y	x	f	,
s		$=$		$<$		
X					$=$	
Y					$>$	
x			$=$		$>$	$<$
f		$=$		$<$		
,				$=$		



Обозначим условно ^как начало, а $ как конец строки.

$sx, xfx \Rightarrow$

$< s < x <, \doteq x > f < x > \Rightarrow$

$< s < x \doteq Y > f \doteq X > \Rightarrow$

$< s < X \doteq f \doteq > X > \Rightarrow < S >$

```
%%
x               { return x; }
f               { return f; }
,               { return comma; }
.               { return yytext; }
%%
```

Listing 33: lab4.l

```
%start start

%token f x comma

%%

start:  X f X { printf("%s\n", "OK");}
        ;
X :     x
        | x Y
        ;
Y :     comma x
        | comma x Y
        ;

%%
/* start of programs */
#include "lex.yy.c"
```

```
main() { return  yyparse();}

yyerror(char *s) { fprintf(stderr,"%s\n",s); }
```

Listing 34: lab4.y

Выполняем в shell следующие комманды:

```
flex lab4.l
yacc lab4.y
gcc -o lab4 y.tab.c -ll
./lab4
```