کامپایلر
پویشگر ساز لکس

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

```
while (count<=100) {
  count++;
  // ...
```

```
while
(
count
<=
100
)
{
count
++
;
...
```

لکس

مولد پویشگر لغت

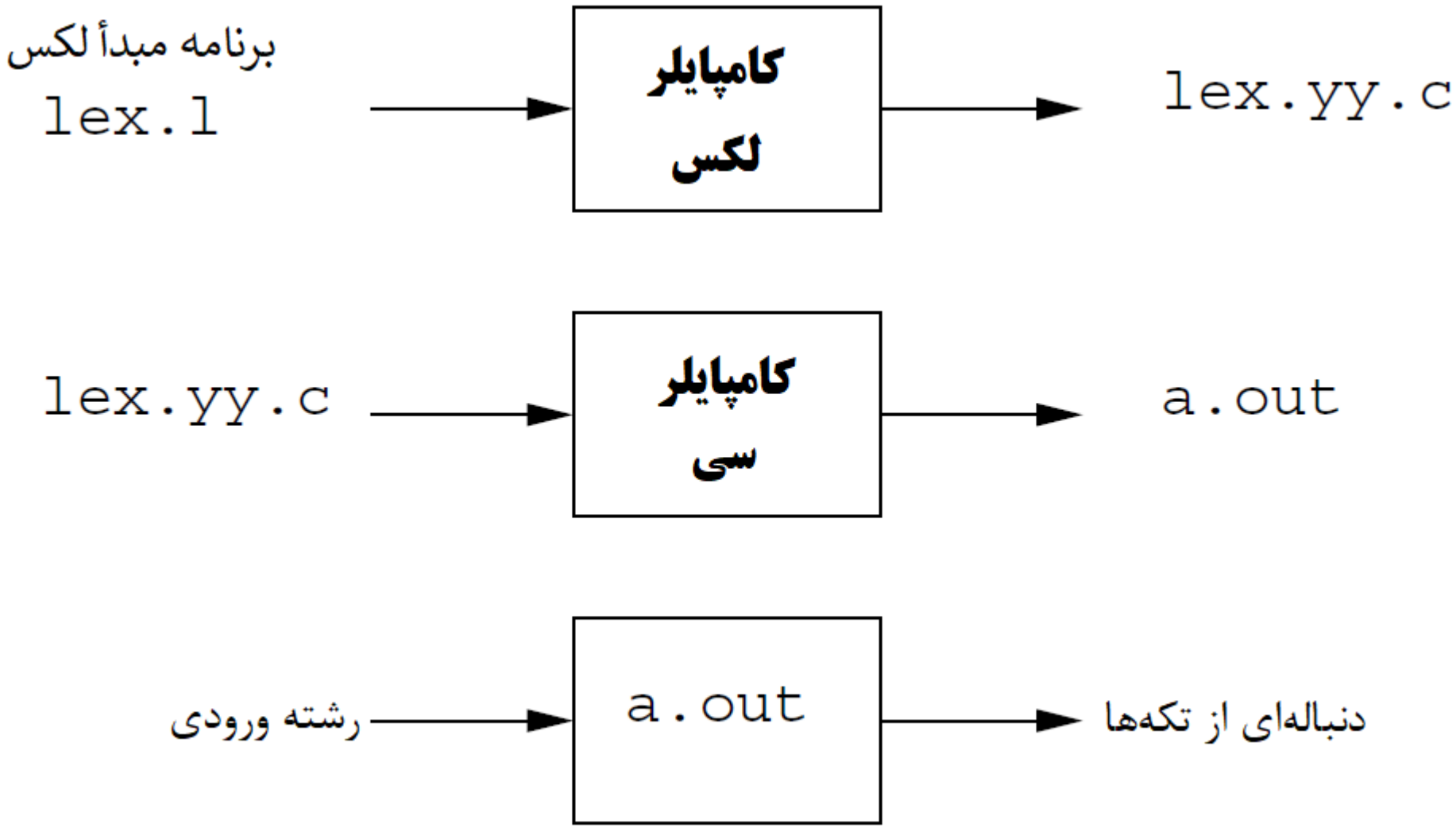
- با استفاده از برنامه c از مشخصات لغوی
- ورودی
  - مشخصات هر نوع تکه
  - دارای عبارت منظم
  - اقدام متناظر
- خروجی
  - پویشگر جدول‌گرا
  - Lex.yy.c

میم لسک و ا اشمیت

- ازمایشگاه ات و ت بل
- ۱۹۷۵

فلکس

«چه» نه «چگونه»


برنامه مبدأ لکس
lex.l
کامپایلر
لکس
lex.yy.c
lex.yy.c
کامپایلر
سی
a.out
رشته ورودی
a.out
دنباله‌ای از تکه‌ها

نحوهٔ نوشتن برنامهٔ لکس

Nam.l

دارای سه بخش

- اعلان و تعریف
  - اعلان متغیرها
- الگوها و اقدام‌های متناظر
  - در بخش دوم
  - الگوها در قالب عبارات منظم
  - اقدام‌ها کد: معمولا در قالب سی
- توابع کمکی و بخش‌های اختیاری
  - بخش سوم
  - توابع کمکی مورداستفاده در اقدام‌ها

```
%{
بخش اول
%}

%%
الگو                اقدام
%%
بخش سوم
```

نحوهٔ نوشتن برنامهٔ لکس

Nam.l

دارای سه بخش

- اعلان و تعریف
  - اعلان متغیرها
- الگوها و اقدام‌های متناظر
  - در بخش دوم
  - الگوها در قالب عبارات منظم
  - اقدام‌ها کد: معمولا در قالب سی
- توابع کمکی و بخش‌های اختیاری
  - بخش سوم
  - توابع کمکی مورداستفاده در اقدام‌ها

```
%%
“compiler”      printf(“99001”);
.               ;
%%
```

نحوهٔ نوشتن برنامهٔ لکس

Nam.l

دارای سه بخش

- اعلان و تعریف
  - اعلان متغیرها
- الگوها و اقدام‌های متناظر
  - در بخش دوم
  - الگوها در قالب عبارات منظم
  - اقدام‌ها کد: معمولا در قالب سی
- توابع کمکی و بخش‌های اختیاری
  - بخش سوم
  - توابع کمکی مورداستفاده در اقدام‌ها

```
>lex nam.l
>cc lex.yy.c –ll
>./a.out
Compiler
99001
```

چگونگی کار

در تعامل و هماهنگی با تجزیه‌گر

در هر مرحله درخواست تجزیه‌گر برای تکه جدید

یافتن طولانی‌ترین انطباق کلمه

- در صورت فاصله از سرگیری یافت کلمه
- پس از یافتن الگو
  - اجرای اقدام متناظر
  - نام
  - yylval

الگوهای مورد استفاده در عبارات منظم -لکس

علامت	کاربرد
.	هر نویسه به جز خط جدید
\n	خط جدید
*	هیچ یا چند بار تکرار عبارت
+	یک یا چند بار تکرار عبارت
?	هیچ یا یک بار تکرار عبارت
$	پایان فایل
a|b	a یا b
^	به جز
\	
(	
{	
[	

الگوهای مورد استفاده در عبارات منظم -لکس

نمونه	انطباق با
123	رشته ۱۲۳
[a-zA-Z]	حرف کوچک یا بزرگ
Com.*piler	هر ترکیبی از نویسه‌ها در وسط
(ab)+	
[^a-z]+	
[+-]?[0-9]+	

متغیرهای از پیش تعریف شده

نام	کارکرد
int yylex(void)	فراخوانی لکس جهت برگرداندن تکه
char *yytext	اشاره‌گر به رشته انطباق یافته
yyleng	طول رشتهٔ انطباق یافته
yylval	مقدار متناظر با تکه
int yywrap(void)	۱ در صورت انجام، صفر در صورت عدم انجام
FILE *yyout	فایل خروجی
FILE *yyin	فیل ورودی
INITIAL	
BEGIN	
ECHO	نوشتن رشته انطباق یافته

کـلـمـه	نام تکه	مقدار
هر فاصله	–	–
if	if	–
then	then	–
else	else	–
هر شناسه	id	اشاره به مدخل جدول
هر عدد	number	اشاره به مدخل جدول
<	relop	LT
<=	relop	LE
=	relop	EQ
<>	relop	NE
>	relop	GT
>=	relop	GE

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%

{ws}      {/* no action and no return */}
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID);}
{number}  {yylval = (int) installNum(); return(NUMBER);}
"<"       {yylval = LT; return(RELOP);}
"<="      {yylval = LE; return(RELOP);}
"="       {yylval = EQ; return(RELOP);}
"<>"      {yylval = NE; return(RELOP);}
">"       {yylval = GT; return(RELOP);}
">="      {yylval = GE; return(RELOP);}

%%

int installID() {/* function to install the lexeme, whose
                    first character is pointed to by yytext,
                    and whose length is yyleng, into the
                    symbol table and return a pointer
                    thereto */
}

int installNum() {/* similar to installID, but puts numer-
                     ical constants into a separate table */
}
```

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

عینا قرارگرفتن هر چیزی بین %{ و %} در lex.yy.c

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

```
#define LT 1
#define LE 2
#define EQ 3
...
```

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

یا استفاده از «سرآغاز»

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

یا تعریف در yacc

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

تعاريف منظم

```
%{
    /* definitions of manifest constants
    LT, LE, EQ, NE, GT, GE,
    IF, THEN, ELSE, ID, NUMBER, RELOP */
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

%%
```

جهت
نمایش بسط یافته

```
int installID() {/* function to install the lexeme, whose
                    first character is pointed to by yytext,
                    and whose length is yyleng, into the
                    symbol table and return a pointer
                    thereto */
}

int installNum() {/* similar to installID, but puts numer-
                     ical constants into a separate table */
}
```

عینا قرارگرفتن در lex.yy.c

امکان استفاده در اقدام‌ها

```
%%

{ws}      {/* no action and no return */}
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID);}
{number}  {yylval = (int) installNum(); return(NUMBER);}
"<"       {yylval = LT; return(RELOP);}
"<="      {yylval = LE; return(RELOP);}
"="       {yylval = EQ; return(RELOP);}
"<>"      {yylval = NE; return(RELOP);}
">"       {yylval = GT; return(RELOP);}
">="      {yylval = GE; return(RELOP);}

%%
```

```
%%

{ws}        {/* no action and no return */}
if          {return(IF);}
then        {return(THEN);}
else        {return(ELSE);}
{id}        {yylval = (int) installID(); return(ID);}
{number}    {yylval = (int) installNum(); return(NUMBER);}
"<"         {yylval = LT; return(RELOP);}
"<="        {yylval = LE; return(RELOP);}
"="         {yylval = EQ; return(RELOP);}
"<>"        {yylval = NE; return(RELOP);}
">"         {yylval = GT; return(RELOP);}
">="        {yylval = GE; return(RELOP);}

%%
```

برگرداندن هیچی و
جستجو به دنبال کلمه بعدی

```
%%

{ws}      {/* no action and no return */}
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID);}
{number}  {yylval = (int) installNum(); return(NUMBER);}
"<"       {yylval = LT; return(RELOP);}
"<="      {yylval = LE; return(RELOP);}
"="       {yylval = EQ; return(RELOP);}
"<>"      {yylval = NE; return(RELOP);}
">"       {yylval = GT; return(RELOP);}
">="      {yylval = GE; return(RELOP);}

%%
```

تک الگو

اگر بدون عدد یا حرف بعد از آن

```
%%

{ws}        {/* no action and no return */}
if          {return(IF);}
then        {return(THEN);}
else        {return(ELSE);}
{id}        {yylval = (int) installID(); return(ID);}
{number}    {yylval = (int) installNum(); return(NUMBER);}
"<"         {yylval = LT; return(RELOP);}
"<="        {yylval = LE; return(RELOP);}
"="         {yylval = EQ; return(RELOP);}
"<>"        {yylval = NE; return(RELOP);}
">"         {yylval = GT; return(RELOP);}
">="        {yylval = GE; return(RELOP);}

%%
```

مشابه تکهٔ «اگر»

```
%%

{ws}      {/* no action and no return */}
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID);}
{number}  {yylval = (int) installNum(); return(NUMBER);}
"<"       {yylval = LT; return(RELOP);}
"<="      {yylval = LE; return(RELOP);}
"="       {yylval = EQ; return(RELOP);}
"<>"      {yylval = NE; return(RELOP);}
">"       {yylval = GT; return(RELOP);}
">="      {yylval = GE; return(RELOP);}

%%
```

مشابه تکهٔ «اگر»

```
%%

{ws}      {/* no action and no return */}
if        {return(IF);}
then      {return(THEN);}
else      {return(ELSE);}
{id}      {yylval = (int) installID(); return(ID);}
```

شناسه

الف) استفاده از تابع installID() جهت قرار دادن کلمه در جدول علامت

ب) اشاره‌گر به جدول علامت yylval

مورد استفاده در تجزیه‌گر یا مراحل بعد

ایجاد دو متغیر دیگر

yytext اشاره‌گر به آغاز کلمه

yyleng طول کلمهٔ یافت شده

ج) برگرداندن نام تکه (ID) به تجزیه‌گر

```
%%

{ws}        {/* no action and no return */}
if          {return(IF);}
then        {return(THEN);}
else        {return(ELSE);}
{id}        {yylval = (int) installID(); return(ID);}
{number}    {yylval = (int) installNum(); return(NUMBER);}
"<"         {yylval = LT; return(RELOP);}
"<="        {yylval = LE; return(RELOP);}
"="         {yylval = EQ; return(RELOP);}
"<>"        {yylval = NE; return(RELOP);}
">"         {yylval = GT; return(RELOP);}
">="        {yylval = GE; return(RELOP);}

%%
```

مشابه شناسه
installNUM()

حل تصادم (گشودن تصادم)

دو قانون مورداستفادهٔ لکس جهت انتخاب کلمهٔ مناسب

- مورداستفاده در زمان مطابقت چند کلمه
- الف) ترجیح همیشگی پیشوند طولانی‌تر
- در صورت انطباق طولانی‌ترین پیشوند با دو یا چند الگو
  - انتخاب نخستین در فهرست لکس

- مثال
  - =>
  - then

```
if                                          {return IF;}
[a-z][a-z0-9]*                              {return ID;}
[0-9]+                                      {return NUM;}
([0-9]+"."[0-9]*)|([0-9]*"."[0-9]+)         {return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+           {  /* do nothing */ }
.                                           {error();}
```

```
%{
/* C Declarations: */
#include "tokens.h"    /* definitions of IF, ID, NUM, ... */
#include "errormsg.h"
union {int ival; string sval; double fval;} yylval;
int charPos=1;
#define ADJ  (EM_tokPos=charPos, charPos+=yyleng)
%}
/* Lex Definitions: */
digits  [0-9]+
%%
/* Regular Expressions and Actions: */
if                        {ADJ; return IF;}
[a-z][a-z0-9]*            {ADJ; yylval.sval=String(yytext);
                            return ID;}
{digits}                 {ADJ; yylval.ival=atoi(yytext);
                            return NUM;}
({digits}"."[0-9]*)|([0-9]*"."{digits})     {ADJ;
                            yylval.fval=atof(yytext);
                            return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+    {ADJ;}
.                         {ADJ; EM_error("illegal character");}
```

```
%{
/* C Declarations: */
#include "tokens.h"    /* definitions of IF, ID, NUM, ... */
#include "errormsg.h"
union {int ival; string sval; double fval;} yylval;
int charPos=1;
#define ADJ  (EM_tokPos=charPos, charPos+=yyleng)
%}
/* Lex Definitions: */
digits  [0-9]+
%%
/* Regular Expressions and Actions: */
if                          {ADJ; return IF;}
[a-z][a-z0-9]*              {ADJ; yylval.sval=String(yytext);
                              return ID;}
{digits}                   {ADJ; yylval.ival=atoi(yytext);
                              return NUM;}
({digits}"."[0-9]*)|([0-9]*"."{digits})     {ADJ;
                              yylval.fval=atof(yytext);
                              return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+    {ADJ;}
.                           {ADJ; EM_error("illegal character");}
```

yytext رشته منطبق با عبارت منظم

Yylength طول رشته منطبق با عبارت منظم

```
%{
/* C Declarations: */
#include "tokens.h"    /* definitions of IF, ID, NUM, ... */
#include "errormsg.h"
union {int ival; string sval; double fval;} yylval;
int charPos=1;
#define ADJ  (EM_tokPos=charPos, charPos+=yyleng)
%}
/* Lex Definitions: */
digits  [0-9]+
%%
/* Regular Expressions and Actions: */
if                        {ADJ; return IF;}
[a-z][a-z0-9]*            {ADJ; yylval.sval=String(yytext);
                            return ID;}
{digits}                 {ADJ; yylval.ival=atoi(yytext);
                            return NUM;}
({digits}"."[0-9]*)|([0-9]*"."{digits})     {ADJ;
                            yylval.fval=atof(yytext);
                            return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+    {ADJ;}
.                         {ADJ; EM_error("illegal character");}
```

char-pos نگهداری مکان هر تکه

Yylength طول رشته منطبق با عبارت منظم

بخش اول

- بین }% و {%
- قبل از %%
- شامل اعلان‌های محتملا مورد استفاده جهت ایجاد مقادیر تکه‌ها
- همچنین تعاریف و خلاص‌ها و محفف‌های مورداستفاده
- digits [0-9]+
  - نمایشگر اینکه به جای عبارت منظم اعداد طبیعی بی‌علامت می‌توان **digit**را قرار داد.
- بخش سوم
  - شامل عبارات منظم و اقدام‌ها
  - هر اقدام باید برگرداننده مقداری از نوع اینت **int**
    - نمایشگر نوع تکه تشخیص‌داده‌شده

در قطعه اقدام

- yytext
  - رشته یافت‌شدهٔ منطبق با عبارت منظم
- yyleng
  - طول رشته یافت شده

طراح پویشگر در جاوا

جاواسی‌سی

سمور!

- سیبل‌سی‌سی

جاواسی‌سی JAVACC

```
PARSER_BEGIN(MyParser)
   class MyParser {}
PARSER_END(MyParser)

/* For the regular expressions on the right, the token on the left will be returned: */
TOKEN : {
    < IF: "if" >
  | < #DIGIT: ["0"-"9"] >
  | < ID: ["a"-"z"] (["a"-"z"]|<DIGIT>)* >
  | < NUM: (<DIGIT>)+  >
  | < REAL: ( (<DIGIT>)+ "." (<DIGIT>)* ) |
          ( (<DIGIT>)* "." (<DIGIT>)+ )>
}

/* The regular expressions here will be skipped during lexical analysis: */
SKIP : {
    <"--" (["a"-"z"])* ("\n" | "\r" | "\r\n")>
  |  " "
  |  "\t"
  |  "\n"
}

/* If we have a substring that does not match any of the regular expressions in TOKEN or SKIP,
   JavaCC will automatically throw an error. */
void Start() :
{}
{  ( <IF> | <ID> | <NUM> | <REAL> )*  }
```

SABLECC سمور

```
Helpers
    digit = ['0'..'9'];
Tokens
    if = 'if';
    id = ['a'..'z'](['a'..'z'] | (digit))*;
    number = digit+;
    real =  ((digit)+ '.' (digit)*) |
            ((digit)* '.' (digit)+);
    whitespace = (' ' | '\t' | '\n')+;
    comments = ('--' ['a'..'z']* '\n');
Ignored Tokens
    whitespace,
    comments;
```

پیتون

```
import ply.lex as lex

#list of tokens
tokens = (
        'LANGLE', #<
        'LANGLESLASH', #</
        'RANGLE', #>
        'EQUAL', # =
        'STRING', # "hello"
        'WORD', # Welcome!
    )
```

پیتون

```
#start of comment token
def t_htmlcomment(token):
    r'<!--[^]*-->'
    token.lexer.lineno += token.value.count('\n')
    pass

#new line counter
def t_newline(token):
    r'\n'
    token.lexer.lineno +=1
    pass
```

پیتون

```
#left angle, slash token
def t_LANGLESLASH(token):
    r'</'
    return token
#left angle token
def t_LANGLE(token):
    r'<'
    return token
#right angle token
def t_RANGLE(token):
    r'>'
    return token
```

پیتون

```
#equal token
def t_EQUAL(token):
    r'='
    return token

#string token
def t_STRING(token):
    r'"[^"]*"'
    token.value = token.value[1:-1]
    return token

#word token
def t_WORD(token):
    r'[^ <>\n]+'
    return token
```

پیتون

```
#how to test
webpage = "this is <b>my</b> webpage!"
htmllexer = lex.lex()
htmllexer.input(webpage)

while True:
    tok = htmllexer.token()
    if not tok: break
    print tok
```

مثال

توضیح با علامت // را بنویسید

def t_LINECOMMENT(token)

- ?
- pass

منابع

[اژدها]

[ببرسبز]

[ببرسرخ]

[فیشر]

https://gist.github.com/mohamed-ali/8255392