اصول طراحی کامپایلر

۳

درس‌نامه‌ی

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۳ تولید خودکار تحلیل‌گر لغوی

AUTOMATIC GENERATION OF LEXICAL ANALYZER

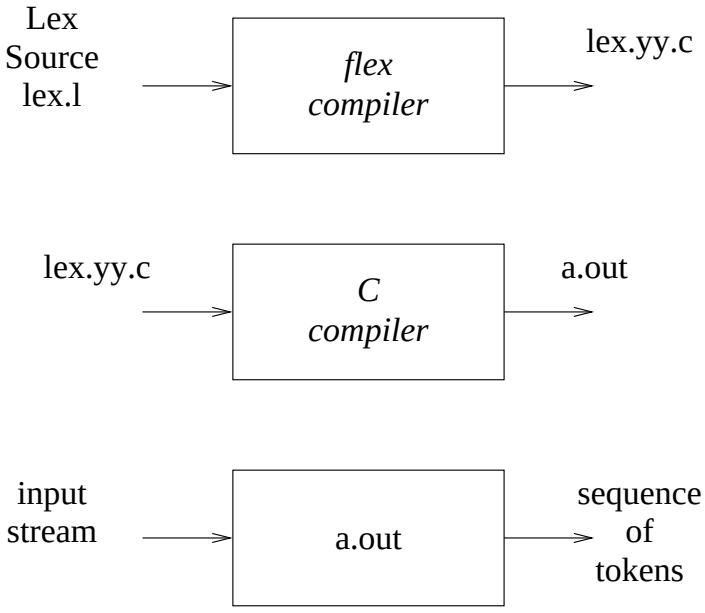
۳-۱ مولدهای پویشگر

تعداد زیادی ابزار با نام مولد پویشگر (scanner generator) وجود دارد.

نام مولد		زبان برنامه‌سازی	سیستم
LEX		C	UNIX
FLEX	GNU's fast LEX	C	UNIX
MKS-LEX		C	MS-DOS, OS/2
JLEX		JAVA	Princeton
JAVACC		JAVA	SUN

۳-۲ LEX: مولد تحلیل‌گر لغوی

برنامه‌ای که در LEX نوشته می‌شود، پس از کامپایل توسط کامپایلر LEX (مانند FLEX) به یک برنامه‌ی زبان C تبدیل می‌شود. برنامه‌ی C پس از کامپایل شدن توسط کامپایلر C به یک برنامه‌ی اجرایی تبدیل می‌شود. این برنامه‌ی اجرایی به عنوان پویشگر (scanner) می‌تواند عملیات تحلیل لغوی را انجام دهد.



۳-۳ قالب کد منبع

کد منبع یک برنامه‌ی LEX از سه قسمت تشکیل شده است که با %% از هم جدا می‌شوند:

- **اعلان‌ها (Declarations)**: مجموعه‌ای از تعاریف منظم (نام عبارت و عبارت منظم متناظر با آن)
- %%
- **قواعد ترجمه (Translation Rules)**: کنش‌هایی که باید در مواجهه با یک الگو انجام شود
- %%
- **روال‌های کمکی (Auxiliary Procedures)**

```
Declarations
%%
Translation Rules
%%
Auxiliary Procedures
```

۴-۳ متغیرهای سراسری در LEX و کاربرد آنها

متغیرهای سراسری که از آنها می‌توان در برنامه استفاده کرد عبارتند از:

- `yyleng`: طول رشته‌ی جاری
- `yytext`: رشته‌ی جاری
- `yylex()`: روتین اسکنر

۵-۳ قالب اجزای کد منبع

- **اعلان‌ها (Declarations)**:
  - متغیرها و ثوابت در قالب C
  - بازنمایی ثوابت با شناسه‌ها
  - عبارت‌های منظم
- **قواعد ترجمه (Translation Rules)**:

```
pattern   { action }
```

  اگر با عبارت منظم `pattern` مواجه شدیم، `action` انجام می‌شود.
- **روال‌های کمکی (Auxiliary Procedures)**:
  که با زبان C نوشته می‌شوند.

مثال

```
%{
%}
Digit       [0-9]
IntLit      {Digit}+
%%
[ \t] {/* skip white spaces */}
[\n] {return('\n');}
{IntLit}                          {return(NUMBER);}
+                                 {return('+');}
-                                 {return('-');}
*                                 {return('*');}
/                                 {return('/');}
.          {printf(error token <%s>\n,yytext); return(ERROR);}
%%
```

نکته: عبارت منظم نقطه (.) با هر چیزی غیر از کاراکتر خط جدید (\n) تطابق می‌یابد.

تمرین

۱. یک برنامه‌ی LEX بنویسید که یک فایل منبع زبان C++ را به عنوان ورودی دریافت کند و تمامی لغت‌های `float` را با `double` جایگزین کند و در یک فایل خروجی بنویسید.

۲. یک برنامه‌ی LEX بنویسید که یک فایل منبع زبان C++ را بخواند و فهرستی از تمام شناسه‌های (identifier) موجود در آن را در یک فایل خروجی بنویسید.

۳. یک برنامه‌ی LEX بنویسید که یک فایل متنی را بخواند و برچسب `mailto://` را به صورت زیر در اطراف کلیه‌ی آدرس‌های e-mail موجود در آن قرار دهد:

```
<a href=mailto://email-address>email-address</a>
```

۴. می‌خواهیم یک تحلیل‌گر لغوی را برای زبانی که به منظور محاسبه و چاپ عبارات ساده‌ی ریاضی طراحی شده است، پیاده‌سازی کنیم. این زبان شامل عملیات ریاضی بر روی اعداد و دستور چاپ نتیجه به همراه رشته‌ی مورد نظر است.

قواعد نحوی قواعد نحوی این زبان به شرح زیر است (گرامر و نمادگذاری آن در درس بعدی توضیح داده می‌شود):

$$
\begin{aligned}
program &\rightarrow stmt^* \\
stmt &\rightarrow \mathbf{id} = expr; \mid \mathbf{print}(pr); \\
pr &\rightarrow p(,p)^* \\
p &\rightarrow \mathbf{string} \mid \mathbf{id} \\
expr &\rightarrow expr + term \mid expr - term \mid term \\
term &\rightarrow term \times factor \mid term/factor \mid factor \\
factor &\rightarrow \mathbf{number} \mid (expr)
\end{aligned}
$$

قواعد لغوی قواعد لغوی این زبان به صورت زیر است:

آ) شناسه (**id**): لغتی شامل حروف، ارقام و زیرخط، که با عدد شروع نمی‌شود.

ب) رشته (**string**): مانند رشته‌های قابل قبول در زبان C++

پ) عدد (**number**): دنباله‌ای از ارقام که با 0 شروع نمی‌شود مگر عدد صفر که با 0 نشان داده می‌شود. برای مثال 000 و 016 عدد نیستند.

ت) توضیحات (**comment**): مانند توضیحات تک خطی و چندخطی در C++.

به وسیله‌ی FLEX برنامه‌ای بنویسید که متن برنامه‌ای از زبان فوق را گرفته و در فایل خروجی، شکل توکن‌بندی شده‌ی آن را برگرداند.

نمونه فایل ورودی

```
x = 1 + 27 * (33-5) ; // line1
/* Now,
we should print the result.
*/
print ( result = \n , x , !!!! ) ;
```

نمونه فایل خروجی

```
< id , x > < = > <number , 1 > <+> < number , 27 > <*> <(>
<number,33 > <-> <number ,5 > <;> < print , print > <(>
<string , result =\n> <,> < id , x > <,> <string , !!!> <)> <;>
```

۵. در زبان SQL، کلمات کلیدی و شناسه‌ها به حروف کوچک و بزرگ حساس نیستند .یک برنامه‌ی LEX بنویسید که کلمات کلیدی `SELECT`، `FROM` و `WHERE` (با هر ترکیبی از حروف کوچک و بزرگ) و توکن ID (شناسه) را شناسایی کند و آنها را به ترتیب چاپ کند.

۶. یک برنامه‌ی LEX بنویسید که هر دنباله از فضاهای خالی (white space) را با یک تک فاصله‌ی خالی (blank) جایگزین کند.