پروژه درس اصول طراحی کامپایلر

تحلیلگر واژه‌ای و دستوری

ترم اول سال
تحصیلی ۹۸ – ۹۷

هدف از این پروژه پیاده‌سازی زبانی به نام LULU نسخه 2.1 است. به این منظور باید از ابزار تولید کامپایلر ANTLR[1] (ANother Tool for Language Recognition) استفاده شود. ANTLR می‌تواند برنامه‌های پیاده‌سازی کامپایلر را به زبان‌های مختلفی از جمله Java، C++، C# و Python تولید کند. برای آشنایی بیشتر با نحوه نصب، بکارگیری و امکانات مختلف این ابزار به راهنمای اینترنتی آن[2] مراجعه کنید. کتاب مرجع کامل این ابزار نیز از گروه درس (در LMS) قابل دسترسی است.

در این مرحله از پروژه دانشجویان باید تحلیلگر واژه‌ای (Lexical Analyzer) و دستوری (Syntactic Analyzer) کامپایلر زبان LULU2.1 را به نحوی پیاده‌سازی کرده که بتواند با بررسی برنامه ورودی، دسته نمادهای (tokens) تشکیل دهنده آن را شناسایی و سپس با بررسی صحت ساختار برنامه، درخت تجزیه (parse tree) را در خروجی ارائه کند.

۱ – معرفی واژه های مورد استفاده در زبان

زبان LULU بسیار شبیه به زبان‌های آشنای C/C++ و Java است. یک برنامه به زبان LULU شامل تعدادی تابع و نوع کاربری (مشابه مفهوم کلاس در زبان‌های شیء گرا) است. شکل ۱ یک نمونه برنامه ساده به زبان LULU را نشان می‌دهد. در صورت نیاز می‌توان نام توابع و نوع‌های کاربری تعریف شده در برنامه به همراه متغیرهای سراسری مورد استفاده در آن را به صورت اختیاری در ابتدای برنامه اعلام (declare) کرد. توضیحات تکمیلی در مورد ویژگی‌های هر یک از این اجزا در مراحل بعدی ارائه خواهد شد. در این زبان حروف کوچک و بزرگ متفاوت از هم در نظر گرفته می‌شود (یک زبان case-sensitive). علاوه بر شناسه‌ها (identifiers)، سایر واژه‌های تشکیل دهنده برنامه‌ها شامل کلیدواژه‌ها (keywords)، عملگرها (operators) و مقادیر ثابت (literal constants) هستند.

۱ – ۱ – شناسه‌ها

شناسه‌ها در این زبان، که برای نام متغیرها، توابع و نوع‌های کاربری مورد استفاده قرار می‌گیرند، شامل دنباله‌ای از حروف انگلیسی (بزرگ و کوچک)، ارقام 0 تا 9، و نویسه‌های (character) خط زیر (‘_’) و علامت عدد (‘#’) است که با رقم شروع نشود.

[1] www.antlr.org
[2] https://github.com/antlr/antlr4/blob/master/doc/index.md

```
%% A sample program
declare      {
       int a;
       myType;
       (int, float) = f1(float b);
}
type myType {
       protected int x;
       public (float y) = function myFun(int z){
              if (z > this.x)     {
                     y = 23.5;    }
              else   {
                     y = 15.2;    }
              return;
       }
}
(int r) = function start() {
       const float c = 0.23;
       int s;
       read(s);
       float w;
       (r, w) = f1(s);
       return;
}
(int x, float y) = function f1(int b)   {
       myType a = allocate myType();
       (y) = a.myFun(b);
       x = 0;
       return;
}
```

شکل ۱: نمونه برنامه به زبان LULU

۱ – ۲ – کلیدواژه‌ها

کلیدواژه‌های زبان که رزرو بوده و نمی‌توانند به عنوان شناسه مورد استفاده قرار گیرند عبارتند از:

allocate	**bool**	**break**	**case**	**const**	**continue**	**declare**	**default**
destruct	**else**	**false**	**function**	**float**	**for**	**if**	**int**
nil	**of**	**private**	**protected**	**public**	**read**	**return**	**string**
super	**switch**	**this**	**true**	**type**	**while**	**write**	

۱ – ۳ – مقادیر ثابت

مقادیر ثابت در زبان LULU شامل مقادیر عددی صحیح (مرتبط با نوع int)، حقیقی (مرتبط با نوع float)، رشته‌ای (مرتبط با نوع string) و بولی (مرتبط با نوع bool) هستند. یک عدد صحیح شامل هر دنباله پیوسته از ارقام است. اعداد صحیح همچنین می‌توانند در مبنای ۱۶ (به صورت Hexadecimal) نوشته شوند که به صورت دنباله‌ای پیوسته از ارقام و مجموعه نویسه‌های {‘a’, ‘A’, ‘b’, ‘B’, ‘c’, ‘C’, ‘d’, ‘D’, ‘e’, ‘E’, ‘f’, ‘F’} نمایش داده می‌شوند که در ابتدای آن رقم صفر و یکی از نویسه‌های ‘x’ یا ‘X’ آمده باشد. به عنوان مثال عدد 0x23 که برابر با عدد 35 در مبنای ۱۰ است. نماد متناظر با اعداد صحیح در گرامر زبان int_const است.

اعداد اعشاری به صورت دنباله‌ای پیوسته از ارقام که دارای یک نقطه ممیز (نویسه ‘.’) باشد نوشته می‌شوند و می‌توانند دارای یک قسمت توانی (exponent) نیز باشند. قسمت توانی بعد از نویسه‌های ‘e’ یا ‘E’ به صورت یک عدد صحیح با امکان استفاده از نویسه‌های ‘+’ یا ‘-’ در ابتدای آن می‌آید (مثال: 2.1e+3). توجه داشته باشید که اعداد زیر اعشاری هستند:

- سمت چپ نویسه ’.‘ شامل یک یا بیشتر رقم باشد. مثال: .1
- سمت راست نویسه ’.‘ شامل یک یا بیشتر رقم باشد. مثال: 1.
- دو سمت نویسه ’.‘ شامل یک یا بیشتر رقم باشد. مثال: 1.1

همچنین قسمت قبل از نقطه ممیز می‌تواند به شکل توضیح داده شده در بالا در مبنای ۱۶ نوشته شود. نماد متناظر با اعداد اعشاری در گرامر زبان real_const است.

مقادیر رشته‌ای به صورت دنباله‌ای از نویسه‌ها نوشته می‌شوند که در نشانه‌های نقل قول تکی (single quotation – ’ ‘) قرار می‌گیرند (مثل ’A‘، ’b‘، ’1‘ و ’this is a\t valid String\’...\n‘). در نوشتن رشته‌ها می‌توان از کدهای فرار (escape characters) خاص استفاده کرد که بعد از نویسه ’\‘ قرار می‌گیرند. کدهای فرار قابل استفاده عبارتند از: ’\n‘، ’\r‘، ’\\‘، ’\t‘، ’\0‘ و ’\’‘. در کنار این کدهای فرار، کد فرار خاصی برای نوشتن کدهای اسکی (ASCII) نویسه‌ها در مبنای ۱۶ وجود دارد به صورت ’\x□□‘ یا ’\X□□‘ نوشته می‌شوند که در جای خالی می تواند هر یک از ارقام و مجموعه نویسه‌های { ’a‘, ’A‘, ’b‘, ’B‘, ’c‘, ’C‘, ’d‘, ’D‘, ’e‘, ’E‘, ’f‘, ’F‘} بیاید، مانند ’\x62‘ یا ’\X62‘ که همان نویسه ’2‘ است. نماد متناظر با مقادیر رشته‌ای در گرامر زبان string_const است.

مقادیر بولی یکی از دو کلیدواژه true و false هستند. نماد متناظر با مقادیر بولی در گرامر زبان bool_const است.

۱ – ۴ – عملگرها

عملگرهای موجود در زبان برای انجام عملیات مقایسه‌ای، منطقی، ریاضی و یا کنترل ساختار برنامه مورد استفاده قرار می‌گیرد. این عملگرها می‌توانند بر اساس تعداد عملوندها، تکی (unary) یا دوتایی (binary) باشند. جدول شماره ۱ عملگرهای مورد استفاده در زبان LULU را نشان می‌دهد. قوانین ناظر به اولویت (priority) و شرکت‌پذیری (associativity) این عملگرها مانند زبان‌های متداول برنامه‌نویسی است (به اسلایدهای درس مراجعه کنید).

۱ – ۵ – دستورات

مجموعه متنوعی از دستورات در زبان LULU وجود دارد که شامل انتساب (=)، فرخوانی توابع، دستورات شرطی و حلقه‌ای است و با کلیدواژه‌های خاص بر اساس گرامر داده شده قابل استفاده در برنامه‌ها هستند.

۱ – ۶ – توضیحات

توضیحات (comments) تک خطی با رشته ”%%“ شروع شده و تا پایان خط فعلی ادامه می‌یابند. مثال:

```
%%single line comment
```

توضیحات چندخطی با رشته ”%~“ شروع شده و تا رسیدن به رشته ”%~“ ادامه می‌یابند. مثال:

```
%~ multi line
Comment ~%
```

جدول 1: عملگرهای مورد استفاده در زبان LULU

عملگرهای ریاضی		عملگرهای بیتی و منطقی		عملگرهای مقایسه‌ای	
-	Subtraction\Unary Minus	~	Bitwise Negation	==	Equal
+	Addition	|	Bitwise Or	!=	Not Equal
*	Multiplication	&	Bitwise And	<=	Less than or Equal
/	Division	^	Bitwise\Logical Xor	<	Less than
%	Modulus	!	Logical Not	>	Bigger than
		||	Logical Or	>=	Bigger than or equal
		&&	Logical and		

عملگرهای کنترل ساختار برنامه

{ }	Opening and Closing Curly Braces	.	Dot
( )	Opening and Closing Parenthesis	,	Comma
[ ]	Opening and Closing Braces	:	Colon
		;	Semi-Colon

۲ – گرامر زبان

گرامر زبان LULU نسخه 2.1 در شکل ۲ نشان داده شده است. برای توصیف گرامر زبان از علامت گذاری‌های زیر استفاده شده است.

- نشانه‌های غیرپایانی (non-terminals) به شکل <a> مشخص شده‌اند.
- نشانه‌های پایانی (terminal) به شکل پررنگ (bold) مشخص شده‌اند مانند: **for**.
  - نشانه‌های پایانی تک‌نویسه‌ای در نشانه‌های نقل قول تکی قرار دارند مانند: **';'**.
- نشانه شروع گرامر <program> است.
- برای سهولت نمایش، از قواعد عبارت‌های منظم هم در گرامر استفاده شده است:
  - X? به معنای اختیاری بودن حضور X در عبارت است.
  - X^* و X^+ به ترتیب به معنای صفر یا بیشتر و یک یا بیشتر رخداد X در عبارت هستند.
  - X | Y به معنای انتخاب X یا Y است و برای تفکیک بدنه‌های قواعد تولید یک نشانه غیرپایانی نیز استفاده می‌شود.

۳ – نکات مهم

در انجام پروژه به نکات زیر توجه داشته باشید:

- برای ارزیابی این مرحله از پروژه، پیاده‌سازی شما از کامپایلر زبان LULU2.1 بر روی تعدادی برنامه آزمایشی که شامل ساختارها و دستورات مختلف زبان است اجرا شده و فقط در صورت صحت عملکرد، به آن نمره تعلق می‌گیرد. دقت کنید اگر در مورد هر تست، مشخص شود که برنامه شما اتفاقی جواب درست را تولید کرده، نمره‌ای به آن تعلق نخواهد گرفت.
- دانشجویان می‌توانند برای سهولت پیاده‌سازی، گرامر داده شده برای زبان LULU2.1 را به هر شکل دلخواه بازنویسی کنند به شرطی که زبان گرامر تغییر نکند.

```
<program> ::= <ft_dcl>? <ft_def>+
<ft_dcl> ::= declare '{' ( <func_dcl> | <type_dcl> | <var_def> )+ '}'
<func_dcl> ::= ( '(' <args> ')' '=' )? id '(' ( <args> | <args_var> )? ')' ';'
<args> ::= <type> ( '[' ']' )* | <args> ',' <type> ( '[' ']' )*
<args_var> ::= <type> ( '[' ']' )* id | <args_var> ',' <type> ( '[' ']' )* id
<type_dcl> ::= id ';'
<var_def> ::= const? <type> <var_val> ( ',' <var_val> )* ';'
<var_val> ::= <ref> ( '=' <expr>)?
<ft_def> ::= ( <type_def> | <fun_def> )
<type_def> ::= type id ( ':' id )? '{' <component>+ '}'
<component> ::= <access modifier>? ( <var_def> | <fun_def> )
<access_modifier> ::= private | public | protected
<fun_def> ::= ( '(' <args_var> ')' '=' )? function id '(' <args_var>? ')' <block>
<block> ::= '{' ( <var_def> | <stmt> )* '}'
<stmt> ::= <assign> ';' | <func_call> ';' | <cond_stmt> | <loop_stmt> | return ';' |
           break ';' | continue ';' | destruct ( '[' ']' )* id ';'
<assign> ::= ( <var> | '(' <var> ( ',' <var> )* ')' ) '=' <expr>
<var> ::= ( ( this | super ) '.' )? <ref> ( '.' <ref> )*
<ref> ::= id ( '[' <expr> ']' )*
<expr> ::= <expr> <binary_op> <expr> | '(' <expr> ')' | <unary_op> <expr>| <const_val> |
           allocate <handle_call> | <func_call> | <var> | <list> | nil
<func_call> ::= ( <var> '.' )? <handle_call> | read '(' <var> ')' | write '(' <var> ')'
<list> ::= '[' ( <expr> | <list> ) ( ',' ( <expr> | <list> ) )* ']'
<handle_call> ::= id '(' <params>? ')'
<params> ::= <expr> | <expr> ',' <params>
<cond_stmt> ::= if <expr> <block> ( else <block> )? |
           switch <var> '{' ( case int_const ':' <block> )* default ':' <block> '}'
<loop_stmt> ::= for ( <type>? <assign> )? ';' <expr> ';' <assign>? <block> |
           while <expr> <block>
<type> ::= int | bool | float | string | id
<const_val> ::= int_const | real_const | bool_const | string_const
<unary_op> ::= '-' | '!' | '~'
<binary_op> ::= <arithmetic> | <relational> | <bitwise> | <logical>
<arithmetic> ::= '+' | '-' | '*' | '/' | '%'
<bitwise> ::= '&' | '|' | '^'
<logical> ::= "||" | "&&"
<relational> ::= "==" | "!=" | "<=" | ">=" | '<' | '>'
```

شکل ۲: گرامر زبان LULU2.1

- پروژه در قالب گروه‌های حداکثر ۳ نفری انجام می‌شود. مهلت تعیین گروه‌ها **حداکثر تا ساعت ۸:۰۰ روز سه‌شنبه ۵ آبان ماه** (کلاس درس) خواهد بود. پس از این مهلت تغییر گروه‌بندی امکان‌پذیر نخواهد بود. افرادی که گروه آنها مشخص نشده باشد باید به صورت انفرادی پروژه را انجام دهند. اعضای هر گروه باید ضمن تقسیم مناسب کار بین خود با تمام

بخش‌های پروژه آشنا باشند. بسته به کیفیت کار گروهی صورت گرفته، نمره گروه‌های چند نفره تا ۱٫۲ برابر در نظر گرفته می‌شود.

- گروه‌ها باید **حداکثر تا ساعت ۲۳:۵۹ روز دوشنبه ۲۸ آبان ماه** فایلهای زیر را به صورت فشرده در قالب یک فایل zip به ایمیل professor.karshenas@gmail.com ارسال کنند.
  - فایل ورودی برنامه ANTLR
  - فایل توضیحات به شکل یک فایل pdf حاوی اطلاعات زیر:
    - نام اعضای گروه همراه با شماره دانشجویی آنها
    - توضیح نقش هر عضو در انجام پروژه و نحوه تقسیم کار
    - متن تعهدنامه اخلاقی زیر با درج نام دانشجویان

ما (*نام دانشجویان عضو گروه*) تعهد می‌نماییم که پروژه تحویل داده شده نتیجه کار گروهی ما بوده و در هیچ یک از بخشهای انجام شده از کار دیگران کپی برداری نشده است. در صورتی که مشخص شود که پروژه تحویل داده شده کار این گروه نبوده است، طبق ضوابط آموزشی با ما برخورد شده و حق اعتراض نخواهیم داشت.

    - هرگونه توضیحات اضافی مورد نیاز درباره پیاده‌سازی این مرحله از پروژه

برای نامگذاری عنوان ایمیل جهت ارسال فایل zip پروژه از فرمت زیر استفاده کنید در غیر این صورت به ایمیل ارسالی ترتیب اثر داده نمی‌شود.

Compiler97-phase(phase#)-group(group#)

مثلاً compiler97-phase1-group1، که در آن شماره گروه (group#) شماره یکتای اختصاص داده شده به هر گروه پس از مشخص شدن گروه‌بندی است که در گروه درس (در LMS) اعلام می‌شود.

- برای آگاهی از اطلاعیه‌های تکمیلی در مورد پروژه به صورت مداوم به گروه درس (در LMS) مراجعه کنید.

موفق باشید

کارشناس