به نام خدا

طراحی و پیادهسازی تحلیلگر لغوی (lexer)



بهار 1403

پوريا اسداللهي(**400405008**)

محمد دوكوهكى(400407009)

آيدا حافظيان(99407007)

استاد: دكتر فاطمه يوسفى نژاد

درس: طراحی کامپایلر

تحلیلگر لغوی با زبان ++C و با (Clion(minGW کامپیال شده.

فایل اجرایی(exe) برنامه در پوشهی cmake-build-debug قرار دارد.

برنامه کد برنامهی ورودی را از فایل command.txt خوانده و در صورت عدم وجود خطا خروجی را در فایل output.txt قرار میدهد.

*در صورتی که برای اجرای کد برنامه را از دوباره کامپایل کردید بسته به نوع IDE ممکن است فایل تکست ورودی و خروجی در جای متفاوتی ایجاد شوند.

طراحي

کلاس های توکن زبان مورد نظر را تعریف میکنیم:

- -identifier
- -int
- -double
- -operator
- -keyword
- -string
- -delimiter
- -comment
- -semicolon
- -error

این توکنها به صورت یک enum class در اول برنامه تعریف شدهاند

حالا به ازای هر کلاس، یک عبارت منظم مینویسیم:

 $\Sigma = \{a, b, c, ..., z, A, B, ..., Z, 0, 1, 2, ..., 9, (,), \{, \}, ; , ", .., +, -, *, /, \, = , >, <, !\}$ Letter = 'a' + 'b' + 'c' + ... + 'z' + 'A' + ... + 'Z'

Operator = '+' + '-' + '/' + '*' + '=' + '<' + '>' + '!'

NumbersNoZero = '1' + ... + '9'

Numbers = '0' + NumberNoZero

Delimiter = '(' + ')' + '{' + '}'

Space = ' ' + '\n' + '\t'

Keywords = 'if' + 'else' + 'print' + 'get' + 'int' + 'double' + 'for'

Operators = '++' + '+' + '*' + '-' + '/' + '=' + '<' + '<=' + '>' + '>=' + '==' + '!='

Identifier = Letter.(Letter + Number)*

Int = NumberNoZero . Number*

Double = Int . '.' . Int

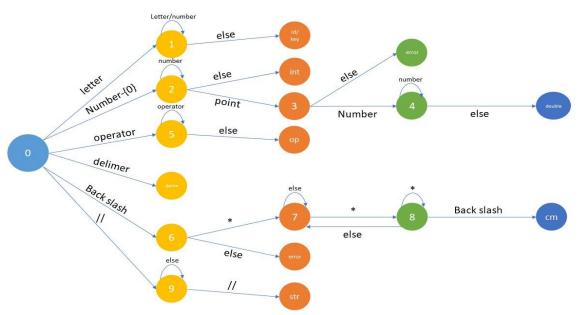
Comment = '\' . '*' . Σ^* -{"*\"}. '*' . '\'

Delta = Σ – {""}

String = "" + Delta" + ""

Semicolon = ';'

اکنون به ازای عبارات منظم توصیف کنندهی زبان، دیاگرام زیر را طراحی میکنیم:



پیادهسازی

پیاده سازی با زبان ++C و برنامهنویسی شی گرا انجام شده است.

کلاس توکن(Token class):

- یک توکن را با یک نوع(type) و یک مقدار (value) تعریف می کند.
 - متد هایی برای چاپ کردن توکن و مقایسه توکن فراهم میکد.

کلاس لکسر(Lexer class):

- تبدیل یک رشته به توکنها را بر عهده دارد.
- کلمات کلیدی(Keywords) و عملگرها(Operators) در این کلاس تعریف شدهاند.
 - متدی برای انجام عملیات توکن سازی(tokenize) رشته ورودی دارد.

تابع های کمکی:

- IsOperator: تشخیص عملگر بودن کاراکتر

- IsDelimiter: تشخیص دلیمیتر بودن کاراکتر

تابع main:

- 1- کد برنامه را از فایل command.txt خوانده و در یک رشته قرار میدهد(در اصل در یک ISS).
 - 2- با رشته ورودی یک شی Lexer ساخته و متد tokenize را روی آن صدا میزند.
 - 3- در صورت وجود error در ترمینال عبارت "error" را چاپ میکند.
 - 4- در صورت عدم وجود error خروجی برنامه را در فایل output.txt چاپ میکند.