LEXICAL ANALYSIS PHASE

Prince Thomas 19/Nov/20

- Lexical Analyzer ultimately generates set of tokens from source file & maintains data structure of tokens.

Q) What is Token?

Token is basically a Data Structure or container that contains an information about a valid word.

A token is consist of 3 parts,

```
class --> TOKEN {

    CP(Class Part),
    VP(Value Part),
    line #
};
```

↳ is use for error reporting purpose.

1. CS → C#
2. Class, abstract
3. Multiple Inheritance
4. this, base
5. Arrays → 1D
6. Struct

- CP & VP has to be a string value and line# will be a integer value.
- A proper data structure is require to hold the output from LA phase because that ~~phase~~ output is used in syntax & semantic phase.

Project: To design a compiler.

→ You need to have a language and its specifications.
→ Your language consist of words, keywords, operators, punctuators, constants, specification of identifiers.

LANGUAGE SPECIFICATIONS

·) List of punctuators:

1) .
2) ,
3) ;
4) :
5) ( )
6) [ ]
7) { }

SL, ML Comments

// → SL
/* ↔ ML

·) List of operators:

Arithmetic Operators: +, −, *, /

Relational Operators: <, >, !=, ==, <<, >>, <=, >=

Logical Operators: AND (&), OR (||), NOT (!)

Increment - Decrement: ++, --

Assignment Operator : = -

) List of keywords / reserved words:

Keywords	Class
main	main
void	void
break	break
continue	continue
for	for
do	do
while	while
if	if
else	else
switch	switch
case	case
default	default
return	return
int	data-type
float	data-type
bool	data-type
char	data-type
string	data-type
class	class
interface	interface
static	static
abstract	abstract
public	access-modifiers

Loops
for
do-while

private	access-modifiers
protected	access-modifiers
new	new
this	this

•) Identifiers Specifications:
In our language:

•) Identifier will start with an alphabet (either lowercase or uppercase).
•) It can be followed by any combination of alphabets & digits.

A: alphabets [A-Z & a-z]
D: digits [0-9]

Regular Expression (R.E):
$A(A+D)^*$

•) Constants:
1) Int Constant = $('+' + '-' + \Lambda)\, D^+$

2) Float Constant = $('+' + '-' + \Lambda)\, D^* \cdot D^+$

3) String Constant = $"(\backslash(A+C) + B + C)^*"$

A: with \ { ' , " , \ }
B: without \ { @ , + , ... }
C: with & without \ { n, t, b, r, 0 }