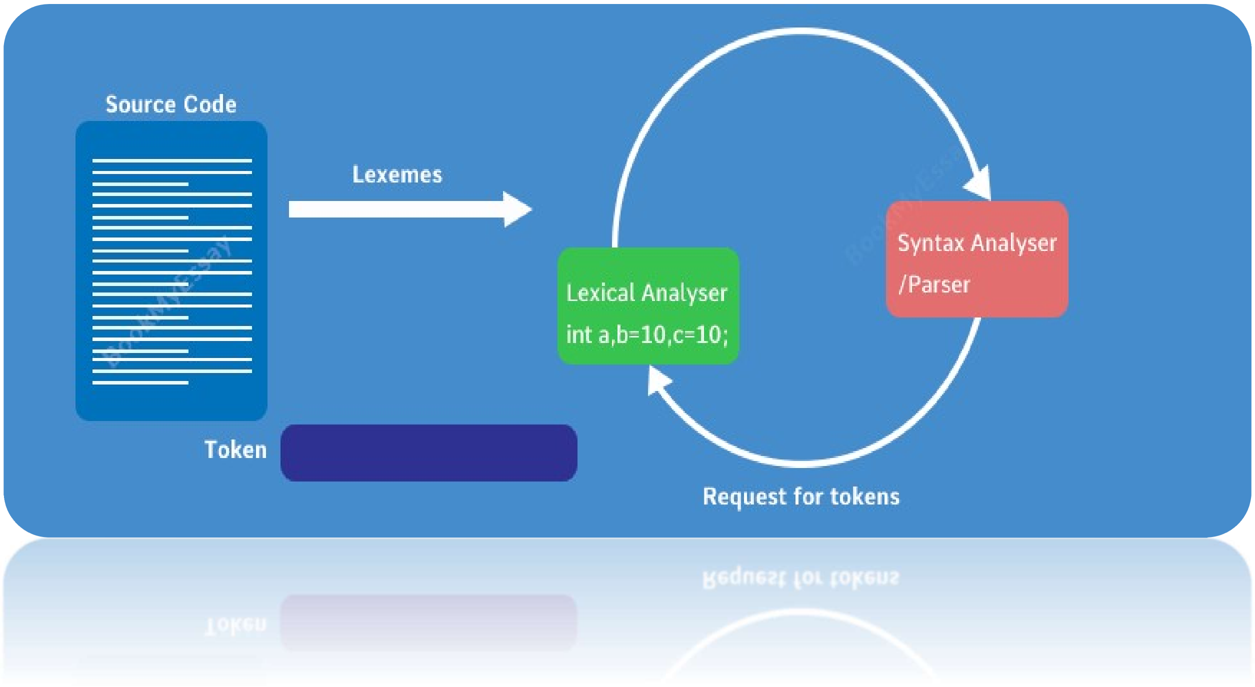


Compiler Project Phase 1

Lexical Analyzer (Scanner)

Due : 3/19/2020

Scanner :

As you must have learned until now, one of the most important parts of every Compiler is it's Lexical Analyzer or Scanner. At this step you are going to implement a simple one which a bit later, will be used to develop a complete compiler. Scanners can be used outside of compilers though, therefore to be more precise, your task is to make a Syntax highlighter. You need to read a code written in a C-like language and highlight it's Keywords as mentioned in following pages:

Token	Format
Reserved Key Words	Bold and Blue
Identifiers	Violate
Integer numbers	Orange
Real numbers	*Italic & Orange*
Strings & Characters	Green
Special Character (both in string or character)	*Italic & Green*
Comments	Gray
Other	Black

- **Reserved Key words:**

int	if
short	else
long	switch
float	case
double	default
char	auto
string	volatile
const	static
for	goto
foreach	signed
while	bool
do	void
in	return

break	record
continue	repeat
new	until
sizeof	function
do	println
true	false

- **Numbers:**

Type	Description
Decimal Integer	Either 32 bit (int) or 64 bit number (long). Long numbers have L at the end of the literal Example : 23454353L
Hexidecimal	Start with 0x Example : 0x45
Real number	can be in any form among $.d^+$, $d^+.$ or $d^+.d^+$ they can be either 32bit (float) which finish with F, or 64bit (double) Example : 34.23F .3445
Scientific notation	A string containing a real number , an e , a minus or plus sign (optional) and another decimal number. Example : 3.4e + 2 4.5e 1 6343e -3

- **Strings & Characters:**
  Like C language, Strings start and end with “ while Characters start and end with ‘.
  Special chars are the ones starting with \ like \t.

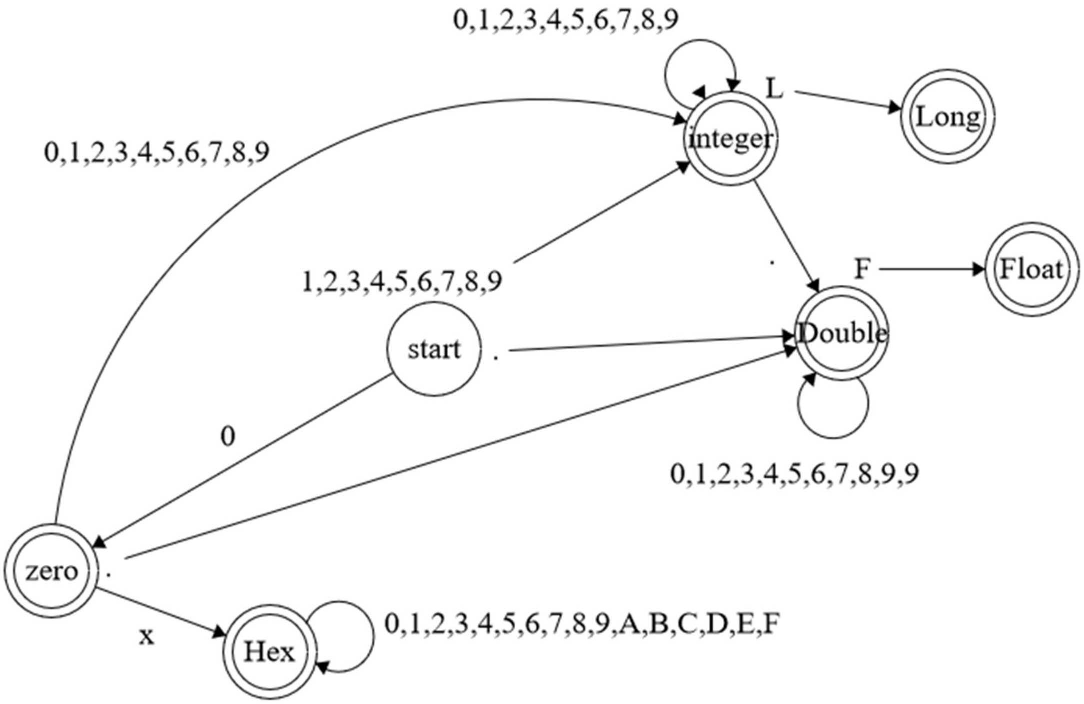
- **Comments**:
  Comments have two types : ones that start with // and ones that start with /* and end with */ and can span in multiple lines.

- The language also contains following symbols which must have black Color.

Equal	==	Dot	.
Not equal	!=	Comma	,
Less equal	<=	Colon	:
Less	<	Semicolon	;
Greater	>	Opening and Closing Brace	[ ]
Greater equal	>=	Prefix and postifx increament	++

Assignment	=	Prefix and postifx decreament	--
Bitwise negation	~	Unary minus	-
Arithmetic and	&	Subtraction assignment	-=
Logical and	and	Multiplication assignment	*=
Logical or	or	Division assignment	/=
Logical not	not	Division	/
Arithmetic or	|	Mod	%
Arithmetic XOR	^	Opening and Closing Blocks	begin end
Production	*	Opening and Closing Parenthesis	( )
Add	+	String literal	"
Addition assignment	+=	Character literal	'

***Part of Scanner Graph*:**



Notes:

- Your program must output an HTML file that highlights text based on rules described above.
- What you must upload is a zip file containing your program and also a pdf file containing complete DFA for Scanner.
- This Phase of the Project must be done individually.

Good Luck