

Programming Languages Project 1: Lexical Analyzer for Ceng++

Lexical rules for the programming language Ceng++ are as follows:

1- Identifiers:

- Maximum identifier size is 20 characters. If you use an identifier larger than that, the lexical analyzer issues an error message.
- Ceng++ language is not case sensitive
- Identifiers start with an alphabetic character (a letter) and are composed of one or more letters, digits or (underscore)
- Example Token: Identifier(my var 1)

2- Integer constants

- Maximum integer size is 10 digits. If you use an integer value longer than that, the lexical analyzer issues an error message.
- Negative values are not supported.
- Example Token: IntConst(352)

3- Operators

- Valid operators of the language are +,-,*,/,++,--,:=
- Example Token: Operator(++)

4- Brackets

- LeftPar: (RightPar:)
- LeftSquareBracket: [RightSquareBracket:]
- LeftCurlyBracket: { RightCurlyBracket: }
- Example Token: LeftCurlyBracket

5- String constants

- String constants of Ceng++ are delimited by double quotes (ASCII code 34)as in "this is a string"
- String constants have unlimited size
- String constants cannot contain the double quote character. when you reach one, the string terminates.
- If a string constant cannot terminate before the file end, there should be a lexical error issued.

6- Keywords:

- Keywords are:
 - break, case, char, const, continue, do, else, enum, float, for, goto, if, int, long, record, return, static, while
- Ceng++ language is not case sensitive and all the keywords are standardized as lower case. You can write the same word as "while" OR "WHILE" and they all generate the
- Example Token: Keyword(while)

```
7- End of line: ;
```

- Example Token: EndOfLine
- 8- Comments: Anything between (* and *) is a comment.
 - If a comment cannot terminate before the file end, there should be a lexical error issued.
 - Comments are just like blank space and they provide no tokens.

<u>Project Definition</u>: The Program should accept a source file called code.Ceng and produce a text file named as code.lex that contains all the tokens of the code.lex listed one after the other.

Example:

```
if code. Ceng contains:
```

```
var:=var_1+18; (*additonn*)
var_1++; (*increment*)
code.lex would be:
```

```
Identifier(var)
```

Operator(:=)

Identifier(var_1)

Operator(+)

IntConst(18)

EndOfLine

Identifier(var_1)

Operator(++)

EndOfLine
