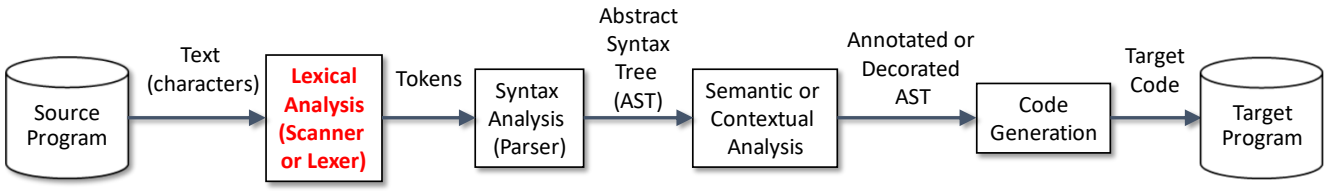
Programming Language Design
Department of Computer Science
Dr. Francisco Ortin
francisco.ortin@mtu.ie

Laboratory 3 – Lexical Analysis

The objective of this laboratory is to implement a lexical analyzer (lexer / scanner) for C-- using ANTLR.

The lexer must read a sequence of characters from an input file, and returns a sequence of tokens to the syntax analyzer (parser).



Remember:

- A token is the minimum meaningful unit to be used by the parser.
- A lexeme is a group of characters that form a token.
- Patterns specify the characters in lexemes a token may have.

Therefore, we will recognize the tokens in a C-- file by specifying the lexical patterns in ANTLR.

Lexical specification of C--

The lexical analyzer is case sensitive.

There are two types of comments to be discarded by the lexer:

- One-line comments starting with `//`
- Multiple-line comments starting with `/*` and ending with `*/`

Additionally, the lexer must recognize the following tokens:

- Identifiers starting with a letter or _, followed by a possible empty repetition of letters, digits or _.
- Integer constants without sign (only base 10 constants are supported; neither octal nor hexadecimal numbers are allowed).
- Real constants with floating point (without sign). Examples:
  `12.3  2.  .34`
- Real constants with mantissa and exponent (without sign). Examples:
  `34.12E-3  3e3  .3E+3  2.e23`
- Char constants between `'` and `'`. Examples:
  `'a'  '.'  '~'`
- Char constants specifying the ASCII code after the \ escape character; e.g., `'\126'` (i.e., the `'~'` character). Any integer constant without sign can be used.
- The two special char constants `'\n'` and `'\t'`.

Example input

The following file (provided as input.txt) could be used to test the correct recognition of tokens in C--:

```
// * One line comment!

/* Integer
   constants */

 0    123
      012

/* Real
   constants */

      12.3  2.          .34
      34.12E-3    3e3   3E+3

/* Identifiers */

  var1       _var_1            VAR_1_AB_2

/* Char
   constants */

      'a'   'b'   '.'   '-'   '~'
      '\n'  '\t'
      '\126'

// Comment at the end of the file
```

Create your first lexer

Create a new lab03 Java project.

Configure ANTLR following the instructions in the intelliJ-and-ANTLR.pdf file.

Run the Main Java class provided, passing small-input.txt as a parameter (just one integer constant) and check that it works.

Go ahead

1. Extend the Cmm.g4 file to recognize all the lexical patterns described in this document. Use input.txt to test your lexer.
2. Complete the LexerHelper class to obtain the semantic values of real and char constants. That is, to convert the lexemes associated with each token into the expected type (int for INT_CONSTANT, double for REAL_CONSTANT, and char for CHAR_CONSTANT).
3. To test the implementation, run the test.LexerTest class enabling assertions (pass –ea to the VM upon execution). A correct lexer implementation must run pass all the assertions (i.e., they must be true).

Remember:

- The lexer must not only recognize valid tokens, but also discard other characters such as comments, white spaces, etc.
- The lexer must not recognize invalid tokens, not specified in the language.
- Those tokens with a fixed lexeme (e.g., “+”, “read”, “int”, “>=”, etc.) do not need to be recognized by the lexer implemented in this lab, since ANTLR provides their recognition in the parser by just writing their lexemes between ' and '. Therefore, your lexer for lab 03 only needs to recognize INT_CONSTANT, REAL_CONSTANT, CHAR_CONSTANT and ID tokens.
- IntelliJ is the recommended IDE due to its powerful ANTLR plugin. Its Community version suffices for this module and it is free.