# CS 311 Language Translation and Automata
# Project #2
**(Due: 2 PM, Monday, 5/14/2012)**

There are many applications of finite state automata, such as lexical analysis and pattern recognition. Lexical analysis is the process of converting a stream of input characters into tokens, the smallest meaningful entity in a language. For example, the characters in the assignment statement

```
position = initial + rate * 60
```

would be grouped into the following tokens:

1. The identifier `position`
2. The assignment symbol =
3. The identifier `initial`
4. The plus sign
5. The identifier `rate`
6. The multiplication sign
7. The number `60`

The blanks separating the characters of these tokens would normally be eliminated during lexical analysis.

Your job for this assignment is to implement a lexical analyzer for a simple programming language called C minus minus, which is a subset of C. Your program is responsible to translate any C-- program into a sequence of tokens as defined below.

List of Tokens:

| | | | | |
|---|---|---|---|---|
| t_double, | t_else, | t_if, | t_int, | t_return, |
| t_void, | t_while, | t_plus, | t_minus, | t_multiplication, |
| t_division, | t_less, | t_lessequal, | t_greater, | t_greaterequal, |
| t_equal, | t_notequal, | t_assignop, | t_semicolon, | t_comma, |
| t_period, | t_leftparen, | t_rightparen, | t_leftbracket, | t_rightbracket, |
| t_leftbrace, | t_rightbrace, | t_id, | t_num | |

The tokens should be internally represented as integers, which are easier to manipulate than strings. You may define the above list of tokens using either enumerations or constants. However, ordinal numbers are not readable. Therefore, your project should print both a sequence of integers and the corresponding string names of those ordinal numbers for verification.

Lexical Conventions of C--:

1. Keywords, which means they cannot be used as identifiers or redefined:

   double else if int return void while

2. Special Symbols:

   + - * / < <= > >= == != = ; , . ( ) [ ] { } /* */ //

3. Identifiers start with a letter (uppercase or lowercase) or an underscore followed by any combination of letters, digits, and underscores. Lower- and uppercase letters are distinct e.g., if is a keyword, but IF is an identifier; hello and Hello are two distinct identifiers.

4. White space consists of blanks, newlines, and tabs. White space serves to separate tokens, but is otherwise ignored. Keywords and identifiers must be separated by white space or a token that is neither a keyword nor an identifier. ifintvoid is a single identifier, not three keywords. if(23void scans as four tokens.

5. A number could be either an unsigned integer or an unsigned real of decimal notation. Leading and trailing zeros are allowed. In the decimal notation, we represent the number with a whole number part, a decimal point, and a fractional part. There must be at least one digit on each side of the decimal point.

6. Both line comments and block comments are allowed. A line comment is started by // and extends to the end of the line. Block comments start with /* and end with the first subsequent */. Block comments do not nest.

For instance, given a sample C-- program as follows:

```
int fact(int x) {
// recursive factorial function
   if (x>1)
      return x * fact(x-1);
   else return 1;
}

void main(void) {
/* CS 311 project 2
A lexical analyzer */
   int x, y, z;
   double _funny;
   x = get_integer();
   _Funny = get_double();
   if (x>0)
      print_line(fact(x));
   else if (_funny != 3.14)
      print_line(x*_funny);
}
```

The output of your project should be:

```
3 27 21 3 27 22 25
2 21 27 13 28 22
4 27 9 27 21 27 8 28 22 18
1 4 28 18
26
5 27 21 5 22 25
3 27 19 27 19 27 18
0 27 18
27 17 27 21 22 18
27 17 27 21 22 18
2 21 27 13 28 22
27 21 27 21 27 22 22 18
1 2 21 27 12 28 22
27 21 27 9 27 22 18
26
```

When you print string names, use uppercase letters for reserved words:

INT id leftparen INT id rightparen leftbrace
IF leftparen id greater num rightparen
RETURN id multiplication id leftparen id minus num rightparen semicolon
ELSE RETURN num semicolon
rightbrace
VOID id leftparen VOID rightparen leftbrace
INT id comma id comma id semicolon

DOUBLE id semicolon
id assignop id leftparen rightparen semicolon
id assignop id leftparen rightparen semicolon
IF leftparen id greater num rightparen
id leftparen id leftparen id rightparen rightparen semicolon
ELSE IF leftparen id notequal num rightparen
id leftparen id multiplication id rightparen semicolon
rightbrace

Also test your program with the following input:

```
int gcd(int u, int v) {
   if (v==0) return u;
   else return gcd(v, u-u/v*v);
   /* u-u/v*v == u mod v */
}

void sort(int a[], int low, int high) {
   int i; int k;
   i = low;
   while (i<high-1) {
      int t;
      k = minloc(a,i,high);
      t = a[k];
      a[k] = a[i];
      a[i] = t;
      i = i + 1;
   }
}

/* this is
   a multi
   line comment */

void main() {
   boolean flag = true;
   if(23void<>14)
      System.exit(0);
   print_line(1+.23*1.2e+3);
   int sum = 0;
   for (int i=0; i<=10; i++)
      sum += i;
   // comment here /* and there
}
```

Submit both your program and input files to the instructor on Bb,
and turn in the printout of (1) program code with proper comments including your name and clear instructions to compile, link, and run your program (2) input files (test your project with at least three C-- programs including the above sample examples, and make sure to cover all lexical features in your test programs), and (3) output of executions.

**Grading**:
75% - correctness including turn-in materials and output format
15% - test case design: give 5 new test cases not covered in given examples
10% - program structure and comments

Your project must be your own work. Copying programs or parts of programs will receive a zero grade and will be reported to the CS department and the University. No late project will be accepted. If you cannot complete the project by the due date, then submit whatever you have completed. Partial credit will be given for reasonable partial solutions. Note that your project will not be graded if we cannot run your program based on your instructions or if the turn-in material is not complete. The highest grade you can get is 70% of the earned score if your project requires being graded second time.