II. C-- Language Lexical Specification

The lexical analyzer or scanner is the first phase of the translation process of a compiler for any given programming language. The main purpose of the lexical phase is to identify the tokens or valid member strings of the programming language, in the order in which they appear in the source file. Another important task of the scanner is to construct the preliminary version of the symbol table for all kind of tokens, which will later be used by syntax, semantics, and intermediate code generator phases.

For this project, the student will have to develop a scanner for the ***C--*** programming language, which is essentially a subset of ***C*** language. The lexical conventions for the language are the following:

a) The Keywords of the language are:

int	**float**	**string**	**for**
if	**else**	**while**	**return**
read	**write**	**void**	

All keywords are reserved words and they are NOT case sensitive, i.e., they can be written in lower case and/or capital letters.

b) Special symbols are the following :

+	arithmetic addition operation	!=	logic operator different	]	close square brackets
-	arithmetic subtraction operation	=	assignation	{	open curly brackets
*	arithmetic multiplication operation	;	semicolon	}	close curly brackets
/	arithmetic division operation	,	coma	/*	open comment
<	logic operator less than	"	quotation mark	*/	close comment
<=	logic operator less or equal than	.	dot		
>	logic operator greater than	(	open parenthesis		
>=	logic operator greater or equal than	)	close parenthesis		
==	logic operator equal	[	open square brackets		

c) Other tokens are ***ID***, ***STRING*** and ***NUMBER***, corresponding Regular Expressions definitions are as follows:

***letter* = [a-zA-Z]**
***digit* = [0-9]**
***STRING* = "•*"**
ID = letter* (*letter* | *digit*)
***NUMBER = digit*+ (. *digit*+)?**

- Identifiers are NOT case sensitive, i.e., they can be written in lower case and/or capital letters.
- Strings constants are enclosed by quotation marks, and as comments, may include any character.

d) White space consists of *blanks*, *newlines*, and *tabs*. White space is ignored, but it MUST be recognized. White space together with ***ID***'s, ***STRING***'s, ***NUMBER***'s, and **keywords**, are considered as delimiters.

e) Comments are ***C*** language style, i.e., they are enclosed by /* … */. Comments can be placed anywhere white space can appear, i.e., comments cannot be placed within tokens. Comments may include any character and may include more than one line.

f) Sample Program in C Minus:

The following program inputs a list of 10 integers, multiplies each element by a float number, sorts them by selection sort and outputs the resulting sorted float numbers array.

```
/* Program that reads a 10 element array of
integers, and then multiply each element of
the array by a float, stores the result into an
array of floats. Subsequently, the array of
floats is sorted and display it into standard
output.*/

int x[10];
string s;
float f1;
float f2[10];

int miniloc(float a[], int low, int high){
  int i; float y; int k;

   k = low;
   y = a[low];
   i = low + 1;
   while (i < high){
        if (a[i] < x){
            y = a[i];
            k = i;
         }
         i = i + 1;
    }
    return k;
}/* END of miniloc() */
```

```
void sort(float a[], int low, int high){
  int i; int k;

   i = low;
   while (i < high - 1){
         float t;
         k = miniloc(a,i,high);
         t = a[k];
         a[k] = a[i];
         a[i] = t;
         i = i +1;
   }
   return;
}/* END of sort() */


void readArray(void){
  int i;

   s = "Enter a float number: ";
  write(s);
   read(f1);
   while (i < 10){
         s = "Enter an integer number: ";
        write(s);
        read x[i];
        f2[i] = x[i]*f1;
        i = i + 1;
    }
    return;
}/* END of readArray() */
```

```
void writeArray(void){
   int i;
    i = 0;
    while (i < 10){
         write f2[i];
         i = i + 1;
    }
    return;
}/* END of writeArray() */


void main(void){
    s = "Reading Information…..";
    write(s);
    readArray();
    s = "Sorting…..";
    write(s);
    sort(f2,0,10);
    s = "Sorted Array:";
    write(s);
    writeArray();
    return;
}/* END of main() */
```