Trabajo Encargado Nro. 03

1.- Objetivo.

Escribir analizadores léxicos.

2.- Desarrollo.

Desarrollar un analizador léxico “scaner” que reconozca los siguientes tokens: NUM, ID, MAYOR, MAYORIGUAL, PUNTOYCOMA, PARI, WHILE, IF. La entrada de datos puede ser, o bien desde un archivo de texto plano o desde la entrada estándar. Cada vez que se reconozca un token, imprima el nombre del token y el lexema.

Solución:

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>

#define MAYOR    '>'
#define PUNTOYCOMA   ';'
#define PARI        '('
#define ID          256
#define NUM         257
#define MAYORIGUAL  258
#define WHILE       259
#define IF          260

int scaner();
void mostrar(int);
int espalres();

FILE *f;
char lexema[80];


void main(int n, char *pal[])
{
        int token;
        f=stdin;        //entrada estandar del teclado
        if(n==2)        //si se especifico un archivo de lectura
        {
                        f=fopen(pal[1],"rt");  //lectura modo texto
                        if(f==NULL)
                                        f=stdin;
        }
        if(f==stdin)     //la lectura sera desde la entrada estandar
           printf("Ingrese texto ..... termine con Ctrl z \n");

        while(1)
        {
                        token=scaner();
                        if(token==EOF) break;
                        mostrar(token);
        }

        if(f !=stdin)  // si la entrada fue de un archivo
                        fclose(f);  // entonces cerrar el archivo.
}//fin del main


int scaner()
{
        int c;
        int i;
        do   c=fgetc(f); while(isspace(c));  //ignora blancos

        if(c==EOF) return EOF;

        if(isalpha(c))   //regla del ID
        {
                        i=0;
                        do{
                                        lexema[i++]=c;
                                        c=fgetc(f);
                        } while(isalnum(c)||c=='_');

                        lexema[i]=0;
                        ungetc(c,f);   //se devuelve c al flujo de entrada
                        i=espalres();   // verifica si es palabra reservada
                                        //  WHILE , IF
                        if(i>=0)
                          return i;
                        return ID;    // se trata de un ID
        }

        if(isdigit(c))   //regla del NUM
        {
                        i=0;
                        do{
                          lexema[i++]=c;
                          c=fgetc(f);
                        }while(isdigit(c));

                        lexema[i]=0;
                        ungetc(c,f);
                        return NUM;
        }

                                //regla de PUNTOYCOMA y PARI
        if((c==';')||(c=='(')) return c;      //regla del ";" y "("

        if(c=='>')      //regla de ">" o ">="
        {
                        c=fgetc(f);
                        if(c=='=')   //return MAYORIGUAL
                        {  lexema[0]='>';  lexema[1]='=';  lexema[2]=0;
                           return MAYORIGUAL;
                        }
                        ungetc(c,f);
                        return MAYOR;  //return MAYOR
        }
}//fin de scaner

int espalres()
{
        if(strcmp(lexema,"while")==0) return WHILE;
        if(strcmp(lexema,"if")==0) return IF;

        return -1;
}

void mostrar(int token)
{

switch(token)
{
case ID: printf("token = ID [%s] \n",lexema);    break;
```

```
case NUM: printf("token = NUM [%s] \n",lexema);    break;
case MAYORIGUAL: printf("token = MAYORIGUAL [%s] \n",lexema); break;
case WHILE: printf("token = WHILE [%s] \n",lexema);     break;
case IF: printf("token = IF [%s] \n",lexema); break;
case PARI: printf("token = PARI [%c] \n",token);    break;
case MAYOR: printf("token = MAYOR [%c] \n",token);    break;
case PUNTOYCOMA: printf("token = PUNTOYCOMA [%c] \n",token);    break;
}

}
```

3. Ejercicios.

Escribir un scanner que reconozca los tokens que se menciona a continuación y realizar pruebas con varios tipos de archivos de entrada.

a. ID
b. NUM
c. +, -, *, /
d. > , >= , < , <= , = , == , !=
e. (,) , [,] , ‘,’ , ‘;’ , { , }
f. Palabras reservadas
g. Ignore comentario en línea y en bloque.

4.- Presentación.

El trabajo sebe ser presentado en un documento en formato PDF y debe contener: enunciado del problema, código fuente, archivos de texto plano para la entrada del analizador (códigos de programas), capturas de pantalla del programa en ejecución, explicación del trabajo y conclusiones. De darse el caso, también debe incluir información complementaria que sea pertinente.