Práctica Nro 03

Código:

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

#define MAYOR '>'
#define MENOR '<'
#define PUNTOYCOMA ';'
#define PARI '('
#define PARD ')'
#define MAS '+'
#define MENOS '-'
#define MULTIPLICACION '*'
#define DIVISION '/'
#define LLAVEI '{'
#define LLAVED '}'
#define CORCI '['
#define CORCD '}'
#define COMA ','
#define ASIGNACION '='
#define ID 256
#define NUM 257
#define MAYORIGUAL 258
#define WHILE 259
#define IF 260
#define MENORIGUAL 261
#define IGUAL 262
#define DIFERENTE 263
#define INT 264
#define VOID 265
#define FLOAT 266
#define DOUBLE 267
#define BOOL 268
#define LONG 269
#define UNSIGNED 270
#define DO 271
#define FOR 272
#define COMENLINE 273
#define COMENBLOQUE 274

int scaner();
void mostrar(int);
```

```c
int espalres();

FILE *f;
char lexema[80];

int main(int n, char *pal[]){
    int token;
    f = stdin;
    if(n == 2){
        f = fopen(pal[1], "rt");
        if(f == NULL) f = stdin;
    }
    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.

    return 0;
}

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) or 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 PUNTO y PARI

if((c==';') or (c=='(') or (c=='(') or (c==')')
    or (c=='{') or (c=='}') or (c=='[') or (c==']')
    or (c=='+') or (c=='-'))
    return c; //regla del ";" y "(" y mas

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
}
return 0;

if(c == '<'){
    c = fgetc(f);
    if(c == '='){
        lexema[0] = '<';
        lexema[1] = '=';
        lexema[2] = 0;
        return MENORIGUAL;
    }
    ungetc(c, f);
    return MENOR;
}

if(c == '='){
    c = fgetc(f);
    if(c == '='){
        lexema[0] = '=';
```

```
            lexema[1] = '=';
            lexema[2] = 0;
            return IGUAL;
        }
        ungetc(c, f);
        return ASIGNACION;
    }

    if(c == '!'){
        c = fgetc(f);
        if(c == '='){
            lexema[0] = '!';
            lexema[1] = '=';
            lexema[2] = 0;
            return DIFERENTE;
        }
    }

    if(c == '/'){
        c = fgetc(f);
        if(c == '/'){
            lexema[0] = '/';
            lexema[1] = '/';
            lexema[2] = 0;
            while(c != '\n'){
                c = fgetc(f);
            }
            return COMENLINE;
        }

        else if(c == '*'){
            int d = fgetc(f);
            lexema[0] = '/';
            lexema[1] = '*';
            lexema[2] = 0;
            while(d != '/'){
                c = d;
                d = fgetc(f);
                if(c == '*' and c == '/') break;
            }
            return COMENBLOQUE;
        }
        ungetc(c, f);
        return DIVISION;
    }
}
```

```
int espalres(){
    if(strcmp(lexema,"while")==0) return WHILE;
    if(strcmp(lexema, "if")==0) return IF;
    if(strcmp(lexema, "int")==0) return INT;
    if(strcmp(lexema, "void")==0) return VOID;
    if(strcmp(lexema, "float")==0) return FLOAT;
    if(strcmp(lexema, "double")==0) return DOUBLE;
    if(strcmp(lexema, "bool")==0) return BOOL;
    if(strcmp(lexema, "long")==0) return LONG;
    if(strcmp(lexema, "unsigned")==0) return UNSIGNED;
    if(strcmp(lexema, "do")==0) return DO;
    if(strcmp(lexema, "for")==0) return FOR;

    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 MENORIGUAL: printf("token = MENORIGUAL [%s]\n", lexema);
break;
        case IGUAL: printf("token = IGUAL [%s]\n", lexema); break;
        case DIFERENTE: printf("token = DIFERENTE [%s]\n", lexema);
break;
        case WHILE: printf("token = WHILE [%s]\n", lexema); break;
        case IF: printf("token = IF [%s]\n", lexema); break;
        case INT: printf("token = INT [%s]\n", lexema); break;
        case VOID: printf("token = VOID [%s]\n", lexema); break;
        case FLOAT: printf("token = FLOAT [%s]\n", lexema); break;
        case DOUBLE: printf("token = DOUBLE [%s]\n", lexema); break;
        case BOOL: printf("token = BOOL [%s]\n", lexema); break;
        case LONG: printf("token = LONG [%s]\n", lexema); break;
        case UNSIGNED: printf("token = UNSIGNED [%s]\n", lexema); break;
        case DO: printf("token = DO [%s]\n", lexema); break;
        case FOR: printf("token = FOR [%s]\n", lexema); break;
        case PARI: printf("token = PARI [%s]\n", lexema); break;
        case PARD: printf("token = PARD [%s]\n", lexema); break;
        case LLAVEI: printf("token = LLAVEI [%s]\n", lexema); break;
        case CORCI: printf("token = CORCI [%s]\n", lexema); break;
        case CORCD: printf("token = CORCD [%s]\n", lexema); break;
        case MAYOR: printf("token = MAYOR [%s]\n", lexema); break;
        case MENOR: printf("token = MENOR [%s]\n", lexema); break;
```

```
        case PUNTOYCOMA: printf("token = PUNTOYCOMA [%s]\n", lexema);
break;
        case COMA: printf("token = COMA [%s]\n", lexema); break;
        case ASIGNACION: printf("token = ASIGNACION [%s]\n", lexema);
break;
        case MAS: printf("token = MAS [%s]\n", lexema); break;
        case MENOS: printf("token = MENOS [%s]\n", lexema); break;
        case MULTIPLICACION: printf("token = MULTIPLICACION [%s]\n",
lexema); break;
        case DIVISION: printf("token = DIVISION [%s]\n", lexema); break;
    }
}
```

Capturas de pantalla: