

UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE
AREQUIPA



Ciencia de la Computacion

COMPILADORES GRUPO A

YUBER ELMER VELAZCO PAREDES

PRACTICA 03

INTEGRANTES:

PUCHO ZEVALLOS KELVIN PAUL

2020

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

```
#include <stdio.h>
#include <ctype.h>
#include <iostream>
#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 scanner();
void mostrar(int);
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;
    }
    if(f==stdin)
        printf("Ingrese texto a ..... termine con Ctrl_Z\n");

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

```

        if (f != stdin)
            fclose(f);

        return 0;
    }

    int scanner()
    {
        int c;
        int i;
        do {
            c=fgetc(f);
        } while (isspace(c));

        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);
            i=espalres();

            if (i>=0)
                return i;
            return ID;
        }

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

            lexema[i]=0;

            ungetc(c,f);

            return NUM;
        }

        if ((c==';' || c=='(')) return c;

        if (c=='>')
        {
            c=fgetc(f);
            if (c=='=')
            {

```

```

        lexema[0]='>'; lexema[1]='='; lexema[2]=0;
        return MAYORIGUAL;
    }
    ungetc(c,f);

    return MAYOR;
}

}

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;
    }
}

```

```

judal@judal-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/COMPUTACION/2020/Compiladores/semana_3/practica_
Ingrese texto ..... termine con Ctrl z
if(contador == 0){cont++;} if(contador >= 2){contador = 5;} while (numero < 5){ numero = numero + 2;}
token = IF [if]
token = PARI [(]
token = ID [contador]
token = NUM [0]
token = ID [cont]
token = PUNTOYCOMA [;]
token = IF [if]
token = PARI [(]
token = ID [contador]
token = MAYORIGUAL [>=]
token = NUM [2]
token = ID [contador]
token = NUM [5]
token = PUNTOYCOMA [;]
token = WHILE [while]
token = PARI [(]
token = ID [numero]
token = NUM [5]
token = ID [numero]
token = ID [numero]
token = NUM [2]
token = PUNTOYCOMA [;]
^Z
[4]+ Detenido ./lex

```

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

- ID
- NUM
- +, -, *, /
- <, <=, >, >=, =, !=
- (,), [,], ', ' , ,
- Palabras reservadas
- Ignore comentario en línea y en bloque.

```

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

using namespace std;

#define MAYOR '>'
#define MENOR '<'
#define ASIG '='
#define ADMI '!'
#define COMA ','
#define PUNTOYCOMA ';'
#define PARI '('
#define PARO ')'
#define CORCHI '['
#define CORCHO ']'
#define LLAVEI '{'
#define LLAVEO '}'
#define SUMA '+'
#define RESTA '-'
#define MULTI '*'
#define DIVI '/'
#define ID 256
#define NUM 257
#define MAYORIGUAL 258
#define MENORIGUAL 259
#define IGUAL 260
#define WHILE 261
#define IF 262
#define FOR 263
#define DO 264
#define ELSE 265
#define PRINTF 266
#define DIFERENTE 267

#define COMENLINEA 268
#define COMENBLOQUE 269
#define ERRORCOMEN 270

static int cont=0;
static int cont2=0;

int reservadas();
FILE *f;
char lexema[80];

int scanner() {
    int c;
    int i;
    do{
        c=fgetc(f);
        if(cont > 0)
            return '\n';

        if( c == '/') {
            c = fgetc(f);

```

```

    if (c == '/') {
        do
            c=fgetc(f);
            while(c != '\n');
        } else if (c == '*') {
            do
                c=fgetc(f);
                while(c != '*');
                c=fgetc(f);
                if (c == '/') {
                    c=fgetc(f);
                }
            }
        }
    } while (isspace(c));

    if (c == EOF) return EOF;

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

        lexema[i] = 0;
        ungetc(c,f); //regresa al principio de la lectura
        i = reservadas();

        if (i >= 0)
            return i;
        return ID;
    }

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

        lexema[i]=0;

        ungetc(c,f);

        return NUM;
    }

    if ((c == ';' ) || (c == '(' ) || (c == ')' ) || (c == ',' ) || (c == '[' ) || (c == ']' ) || (c == '{' ) || (c == '}' ) || (c == '+' ) || (c == '-' ))
        return c;
}

```

```

if(c == '>')
{
    c=fgetc(f);
    if(c == '=') // MAYORIGUAL
    {
        lexema[0]='>'; lexema[1]='='; lexema[2]=0;
        return MAYORIGUAL;
    }
    ungetc(c,f);

    return MAYOR; //rMAYOR
}
if(c == '<')
{
    c=fgetc(f);
    if(c == '=') // MENORIGUAL
    {
        lexema[0]='<'; lexema[1]='='; lexema[2]=0;
        return MENORIGUAL;
    }
    ungetc(c,f);
    return MENOR; //MAYOR
}
if(c == '!')
{
    c=fgetc(f);
    if(c == '=') // DIFERENTE DE
    {
        lexema[0]='!'; lexema[1]='='; lexema[2]=0;
        return DIFERENTE;
    }
    ungetc(c,f);
    return ADMI; //DIFERENTE
}
if(c == '=')
{
    c=fgetc(f);
    if(c == '=') // IGUAL
    {
        lexema[0]='='; lexema[1]='='; lexema[2]=0;
        return IGUAL;
    }
    ungetc(c,f);
    return ASIG; //ASIGNACION
}
if(c == '/' || c == '*'){
    int d = c;
    c=fgetc(f);
    if(d == '/' && c == '/'){
        cont++;
        return COMENLINEA;
    }
    if( d == '/' && c == '*'){

        do
            c=fgetc(f);
        while(c != '*');
    }
}

```



```

        c=fgetc(f);
        if(c == '/') {
            c=fgetc(f);
            ungetc(c,f);
            return COMENBLOQUE;
        }
        return ERRORCOMEN;

    }
    ungetc(c,f);

    if( d == '/')
        return DIVI;

    return MULTI;
}

}

int reservadas()
{
    if(strcmp(lexema,"while")==0) return WHILE;
    if(strcmp(lexema,"if")==0) return IF;
    if(strcmp(lexema,"for")==0) return FOR;
    if(strcmp(lexema,"printf")==0) return PRINTF;
    if(strcmp(lexema,"do")==0) return DO;
    if(strcmp(lexema,"else")==0) return ELSE;
    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 DIFERENTE: printf("token=_DIFERENTE_[%s]_\\n",lexema);
            break;
        case IGUAL: printf("token=_IGUAL_[%s]_\\n",lexema); break;

        case WHILE: printf("token=_WHILE_[%s]_\\n",lexema); break;
        case IF: printf("token=_IF_[%s]_\\n",lexema); break;
        case FOR: printf("token=_FOR_[%s]_\\n",lexema); break;
        case ELSE: printf("token=_ELSE_[%s]_\\n",lexema); break;
        case PRINTF: printf("token=_PRINTF_[%s]_\\n",lexema); break;
        case DO: printf("token=_DO_[%s]_\\n",lexema); break;

        case PARI: printf("token=_PARI_[%c]_\\n",token); break;
        case PARO: printf("token=_PARO_[%c]_\\n",token); break;
        case CORCHI: printf("token=_CORCHI_[%c]_\\n",token); break;
        case CORCHO: printf("token=_CORCHO_[%c]_\\n",token); break;
        case LLAVEI: printf("token=_LLAVEI_[%c]_\\n",token); break;
        case LLAVEO: printf("token=_LLAVEO_[%c]_\\n",token); break;
    }
}

```

```

        case SUMA: printf("token==SUMA[%c]\n",token); break;
        case RESTA: printf("token==RESTA[%c]\n",token); break;
        case MULTI: printf("token==MULTI[%c]\n",token); break;
        case DIVI: printf("token==DIVI[%c]\n",token); break;

        case MAYOR: printf("token==MAYOR[%c]\n",token); break;
        case MENOR: printf("token==MENOR[%c]\n",token); break;
        case ASIG: printf("token==ASIG[%c]\n",token); break;
        case PUNTOYCOMA: printf("token==PUNTOYCOMA[%c]\n",token);
            break;
        case COMA: printf("token==COMA[%c]\n",token); break;
        case ADMI: printf("token==ADMI[%c]\n",token); break;
    }
}
int main(int n, char *pal[]){

    int token;
    int elejir;
    printf("Escoja 1 entrada standar del teclado:\n");
    printf("Escoja 2 entrada archivo.txt\n");
    printf("Elija que entrada leer:\n");
    scanf("%d", &elejir);
    if( elejir == 1){
        f=stdin;
    }else if( elejir == 2){
        f =fopen("archivo.txt","r");
    }

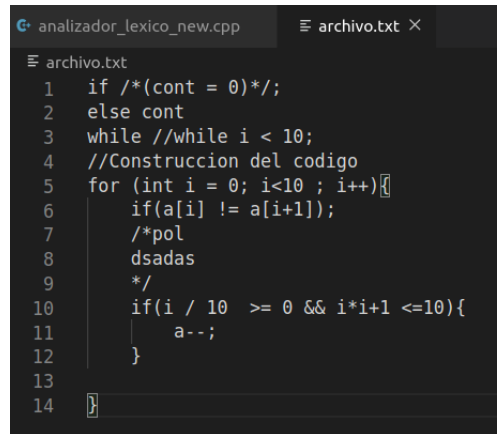
    if(n==2)
    {
        f=fopen(pal[1],"rt");
        if(f==NULL)
            f=stdin;
    }
    if(f==stdin)
        printf("Ingrese texto..... termine con Ctrl_Z\n");

    while(1)
    {
        token=scanner();
        if(token==EOF) break;
        mostrar(token);
    }
    if(f !=stdin){
        printf("Lectura Completa\n");

        while(1)
        {
            token=scanner();
            if(token==EOF) break;
            mostrar(token);
        }
    }
    return 0;
}

```

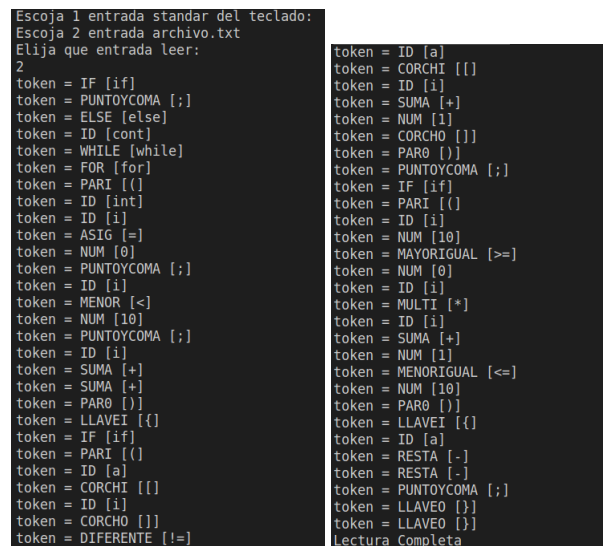
2.1. Captura de resultados por entradas de Texto plano



The screenshot shows a code editor with a tab labeled 'archivo.txt'. The code is as follows:

```
1  if /*(cont = 0)*/;
2  else cont
3  while //while i < 10;
4  //Construccion del codigo
5  for (int i = 0; i<10 ; i++) {
6      if(a[i] != a[i+1]);
7      /*pol
8      dsadas
9      */
10     if(i / 10  >= 0 && i*i+1 <=10){
11         a--;
12     }
13
14 }
```

Figura 1: Texto plano : Archivo.txt



The screenshot shows a terminal window with two columns of text. The left column lists tokens identified in the code, and the right column shows the corresponding token values.

Token	Value
token = IF [if]	token = ID [a]
token = PUNTOYCOMA [;]	token = CORCHI [[]]
token = ELSE [else]	token = ID [i]
token = ID [cont]	token = SUMA [+]
token = WHILE [while]	token = NUM [1]
token = FOR [for]	token = CORCHO [[]]
token = PARI [[]]	token = PARO []]
token = ID [int]	token = PUNTOYCOMA [;]
token = ID [i]	token = IF [if]
token = ASIG [=]	token = PARI [[]]
token = NUM [0]	token = ID [i]
token = PUNTOYCOMA [;]	token = NUM [10]
token = ID [i]	token = MAYORIGUAL [>=]
token = MENOR [<]	token = NUM [0]
token = NUM [10]	token = ID [i]
token = PUNTOYCOMA [;]	token = MULTI [*]
token = ID [i]	token = ID [i]
token = SUMA [+]	token = SUMA [+]
token = SUMA [+]	token = NUM [1]
token = PARO []]	token = MENORIGUAL [<=]
token = LLAVEI [[]]	token = NUM [10]
token = IF [if]	token = PARO []]
token = PARI [[]]	token = LLAVEI [[]]
token = ID [a]	token = ID [a]
token = CORCHI [[]]	token = RESTA [-]
token = ID [i]	token = RESTA [-]
token = CORCHO [[]]	token = PUNTOYCOMA [;]
token = DIFERENTE [!=]	token = LLAVEO {}]
	token = LLAVEO {}]

Lectura Completa

2.2. Captura de resultados por entrada estandar del teclado

```
Escoja 1 entrada standar del teclado:
Escoja 2 entrada archivo.txt
Elija que entrada leer:
1
Ingrese texto ..... termine con Ctrl z
if /*(cont = 0)*/; else cont while for (int i = 0; i<10 ; i++){
token = IF [if]
token = PUNTOYCOMA [;]
token = ELSE [else]
token = ID [cont]
token = WHILE [while]
token = FOR [for]
token = PARI [(]
token = ID [int]
token = ID [i]
token = ASIG [=]
token = NUM [0]
token = PUNTOYCOMA [;]
token = ID [i]
token = MENOR [<]
token = NUM [10]
token = PUNTOYCOMA [;]
token = ID [i]
token = SUMA [+]
token = SUMA [+]
token = PARO [)]
token = LLAVEI [{]
if(a[i] != a[i+1]); if(i / 10 >= 0 && i*i+1 <=10){ a--; }}
token = IF [if]
token = PARI [(]
token = ID [a]
token = CORCHI [[]]
token = ID [i]
token = CORCHO []]
token = DIFERENTE [!=]
token = ID [a]
token = CORCHI [[]]
token = ID [i]
token = SUMA [+]
token = NUM [1]
token = CORCHO []]
token = PARO [)]
token = PUNTOYCOMA [;]
token = IF [if]
token = PARI [(]
token = ID [i]
token = NUM [10]
token = MAYORIGUAL [>=]
token = NUM [0]
token = ID [i]
token = MULTI [*]
token = ID [i]
token = SUMA [+]
token = NUM [1]
token = MENORIGUAL [<=]
token = NUM [10]
token = PARO [)]
token = LLAVEI [{]
token = ID [a]
token = RESTA [-]
token = RESTA [-]
token = PUNTOYCOMA [;]
token = LLAVEO [}]
token = LLAVEO [}]
^Z
[2]+ Detenido ./lex
```