UACM
Universidad Autónoma
de la Ciudad de México
Nada humano me es ajeno

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MEXICO

Proyecto: creación de un compilador
Parte 1: creación del analizador léxico (lexer)

TOMÁS ASCENCIO HERNANDEZ

Profesor: M. en C. Orlando Muñoz Texzocotetla

16/SEPTIEMBRE/2019

UACM
Universidad Autónoma
de la Ciudad de México
Nada humano me es ajeno

INTRODUCCIÓN

El objetivo de este trabajo es crear un analizador léxico con flex,que nos ayudará en el proceso de traducción en la siguiente etapa el parser.

Este programa recibe como entrada un código fuente y genera un tabla de de tokens que tienen

token palabra_clave código_numérico

y la almacena en un archivo externo para posteriormente sea utilizada en el parser

Por medio de expresiones regulares y declaraciones crearemos nuestro lexer para que pueda identificar lo que son:

Signos de puntuación **: . : ; , ( ) [ ] { }**

Palabras reservadas (keywords) **si, entonces, otro,**
mientras, para, hasta, haz, hasta, leer, imprimir, programa, constante, entero, flotante, carácter,cadena, arreglo, función, potencia, sqrt, seno, coseno, tangente, log, exp.

Operadores aritméticos, lógicos y relacionales: **+ / * - mod < > <= >= == != && || !**

Otros operadores: **<- (asignación) += -= *= =/ mod= ++ --**

Cadenas de caracteres

Comentarios

Números enteros

Números flotantes

Identificadores de variable y de función

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

DESARROLLO

Para el desarrollo de la práctica primero enumeramos los signos de puntuación,palabras reservadas,etc para que puedan ser utilizadas posteriormente

```
enum  {
  PUNTO=248,
  DOSPUNTOS=249,
  PUNTOCOMA=250,
  COMA=251,
  PARENTESISABRE=252,
  PARENTESISCIERRA=253,
  CORCHETEABRE=254,
  CORCHETECIERRA=255,
  LLAVEABRE=256,
  LLAVECIERRA=257,
  SI = 258, // KEYWORD O PALABRA RESERVADA SI
  ENTONCES = 259, // KEYWORD O PALABRA RESERVADA ENTONCE
.
.
.
.
}
```

Despues hicimos las expresiones regulares y las accione spara el conjunto de palabras

```
"."             { strcpy(cadena,yytext);  return PUNTO;}
":"             { strcpy(cadena,yytext);  return DOSPUNTOS;}
";"             { strcpy(cadena,yytext);  return PUNTOCOMA; }
","             { strcpy(cadena,yytext);  return COMA; }
"("             { strcpy(cadena,yytext);  return PARENTESISABRE;}
[a-z][a-zA-Z0-9_]*  { strcpy(cadena,yytext);  return IDVARIABLE;
}
[A-Z][a-zA-Z0-9_]*  { strcpy(cadena,yytext);  return IDFUNCION;  }
```

posteriormente en e main desplegamos la información y creamos el método para almacenar los tokens en el archivo

```
FILE *fp;
    fp= fopen("tabla_simbolos.txt","w");
    int tok;
    printf("\nToken\t\tpalabra_clave\t\tcodigo_numerico\n");
    fprintf(fp,"\nToken\t\t palabra_clave\t\tcodigo_numerico\n");
    while(tok = yylex()) {
        /*
        Si el lexema está dentro de este rango de
        valores, dados en "enum", entonces es un
        identificador de variable. Así que por ejemplo
        si el lexema es "if", entonces el renglón
        correspondiente en la tabla de símbolos
        deberá tener la información siguiente:
        if     id_var 263
        */



        if(tok >= PUNTO && tok <= LLAVECIERRA){
               printf("\n%s\t\tSIGNO PUNTUACION\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tSIGNO PUNTUACION\t\t%d", cadena, tok);
        }

        if(tok >= SI && tok <= EXPONENTE){
               printf("\n%s\t\tPALABRA RESERVADA\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tPALABRA RESERVADA\t\t%d", cadena, tok);
        }

        if(tok >= SUMA && tok <= MODULO){
               printf("\n%s\t\tOPERADOR ARITMETICO\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tPALABRA RESERVADA\t\t%d", cadena, tok);
        }
        if(tok >= MENORQUE && tok <= NOTEQUALS){
               printf("\n%s\t\tOPERADOR RELACIONAL\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tOPERADOR RELACIONAL\t\t%d", cadena, tok);
        }

        if(tok >= AND && tok <= NEGACIONLOGICA){
               printf("\n%s\t\tOPERADOR LOGICO\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tOPERADOR LOGICO\t\t%d", cadena, tok);
        }

        if(tok >=  ASIG && tok <= DECREMENTO){
               printf("\n%s\t\tOPERADOR ASIGNACION\t\t%d", cadena, tok);
               fprintf(fp,"\n%s\t\tOPERADOR ASIGNACION\t\t%d", cadena, tok);
        }


        if(tok==NUMEROENTERO){
        printf("\n%d\t\tint\t\t%d", yylval, tok);
```

```
        fprintf(fp,"\n%d\t\tint\t\t%d", yylval, tok);
        }

        if(tok==NUMEROFLOTANTE)    {
        printf("\n%f\t\tfloat\t\t%d",yylval, tok);
        fprintf(fp,"\n%f\t\tfloat\t\t%d",yylval, tok);
        }
        if(tok == IDVARIABLE){
               printf("\n%s IDENTIFICADOR VARIABLE",cadena);
               fprintf(fp,"\n%s IDENTIFICADOR VARIABLE",cadena);
        }

        if(tok == IDFUNCION){
               printf("\n%s IDENTIFICADOR FUNCION",cadena);
               fprintf(fp,"\n%s IDENTIFICADOR FUNCION",cadena);
        }
        printf("\n");
        fprintf(fp,"\n");


    }
    fclose(fp);
}
```