Introducción a los generadores de analizadores léxicos y sintácticos FLEX y BISON:

Implementación de una pequeña calculadora

Objetivo

Con la ayuda de FLEX, el generador automático de analizadores léxicos, y BISON, el generador automático de analizadores sintácticos, implementar una pequeña calculadora que permita realizar las operaciones aritméticas básicas y las trigonométricas como el seno y el coseno.

Especificación Léxica

Se han de reconocer los siguientes componentes léxicos definidos como:

TKN_NUM	[0-9]+("."[0-9]+)?
TKN_ASIGN	"="
TKN_PTOCOMA	";"
TKN_ID	[a-zA-z][a-zA-Z0-9]*
TKN_MULT	"*"
TKN_DIV	"/"
TKN_MAS	"+"
TKN_MENOS	"-"
TKN_PAA	"("
TKN_PAC	")"
TKN_SEN	"sen"
TKN_COS	"cos"

Especificación Sintáctica

El lenguaje a reconocer viene definido por la siguiente gramática:

Calculadora → **id** = Expresion **;**

Expresion → **num** |

Expresion **+** Expresion | Expresion **-** Expresion |

Expresion ***** Expresion | Expresion **/** Expresion |

(Expresion **)** | **sen (** Expresion **)** | **cos (** Expresion **)**

No olvidar establecer las precedencias y asociatividad de los operadores. Los operadores suma + y resta – tienen igual prioridad y son asociativos por la izquierda. Los operadores multiplicación * y división / son asociativos por la izquierda, tienen igual prioridad entre sí y mayor que la de los anteriores.

Se pide:

- Implementar el analizador léxico con Flex
- Implementar el analizador sintáctico con Bison.
- Compilar bison con la opción `--verbose` y editar el fichero para ver el autómata LR. ¿Cuántos estados tiene?
- Insertar las acciones semánticas necesarias para calcular el valor numérico de la expresión.
- Modificar la gramática para que se puedan reconocer una lista de asignaciones separadas por el punto y coma ;
- Modificar la gramática para incluir el operador potencia (^) y la función raíz cuadrada (sqrt).

Ejemplo

Ante la siguiente entrada

a =12+2*cos(3.14) ;

La salida debe ser:

El valor del identificador a es 10.

Fichero léxico_solo.l

```
%{
/* Ejemplo para una pequeña calculadora que permite
trabajar con numeros enteros y reales con las operaciones
 básicas de suma, resta, producto, division y trigonometricas como el seno y el coseno */

#include <stdio.h>
#include <stdlib.h>

int nlines=0;
%}

DIGITO [0-9]
ID [a-zA-Z][a-zA-Z0-9_]*

%%

{DIGITO}+              {printf("Encontrado TKN_NUM_ENTERO: %d",atoi(yytext));}
{DIGITO}+"."{DIGITO}+  {printf("Encontrado TKN_NUM_REAL: %f",atof(yytext));}
"="                    {printf("Encontrado TKN_ASIGN: %s",yytext);}
";"                    {printf("Encontrado TKN_PTOCOMA: %s",yytext);}
"*"                    {printf("Encontrado TKN_MULT: %s",yytext);}
"/"                    {printf("Encontrado TKN_DIV: %s",yytext);}
"+"                    {printf("Encontrado TKN_MAS: %s",yytext);}
"-"                    {printf("Encontrado TKN_MENOS: %s",yytext);}
"("                    {printf("Encontrado TKN_PAA: %s",yytext);}
")"                    {printf("Encontrado TKN_PAC: %s",yytext);}
"cos"                  {printf("Encontrado TKN_COS: %s",yytext);}
"sen"                  {printf("Encontrado TKN_SEN: %s",yytext);}
{ID}                   {printf("Encontrado TKN_ID: %s",yytext);}
"\n"                   {nlines++;}
.
```

```
%%

void main(int argc,char **argv)
{
if (argc>1)
   yyin=fopen(argv[1],"rt");
else
   yyin=stdin;
yylex();
printf("\nNumero lineas analizadas: %d\n", nlines);
}

/* para compilar
flex lexico.l
cc lex.yy.c -o milex -lfl -lm
*/
```

Fichero léxico.l (versión a enlazar con Bison)

```
%{
/* Ejemplo para una pequeña calculadora que permite trabajar
con las operaciones básicas de suma, resta, producto, division y
trigonometricas como el seno y el coseno */

#include <stdio.h>
#include <stdlib.h>

#include "sintactico.tab.h"
int nlines=0;
%}

DIGITO [0-9]
ID [a-zA-Z][a-zA-Z0-9_]*

%%

{DIGITO}+("."{DIGITO}+)? {//printf("Encontrado TKN_NUM: %f\n",atof(yytext));
                          yylval.real=atof(yytext);
                          return(TKN_NUM);}
"="                      {//printf("Encontrado TKN_ASIGN: %s\n",yytext);
                          return(TKN_ASIGN);}
";"                      {//printf("Encontrado TKN_PTOCOMA: %s\n",yytext);
                          return(TKN_PTOCOMA);}
"*"                      {//printf("Encontrado TKN_MULT: %s\n",yytext);
                          return(TKN_MULT);}
"/"                      {//printf("Encontrado TKN_DIV: %s\n",yytext);
                          return(TKN_DIV);}
"+"                      {//printf("Encontrado TKN_MAS: %s\n",yytext);
                          return(TKN_MAS);}
"-"                      {//printf("Encontrado TKN_MENOS: %s\n",yytext);
```

```
                        return(TKN_MENOS);}
"("                     {//printf("Encontrado TKN_PAA: %s\n",yytext);
                        return(TKN_PAA);}
")"                     {//printf("Encontrado TKN_PAC: %s\n",yytext);
                        return(TKN_PAC);}
"cos"                   {//printf("Encontrado TKN_COS: %s\n",yytext);
                        return(TKN_COS);}
"sen"                   {//printf("Encontrado TKN_SEN: %s\n",yytext);
                        return(TKN_SEN);}
{ID}                    {//printf("Encontrado TKN_ID: %s\n",yytext);
                        return(TKN_ID);}
"\n"                    {nlines++;}
.

%%

/********
Para el lexico solo
void main(int argc,char **argv)
{
if (argc>1)
   yyin=fopen(argv[1],"rt");
else
   yyin=stdin;
yylex();
printf("\nNumero lineas analizadas: %d\n", nlines);
}
*******/

/* para compilar
flex lexico.l
cc lex.yy.c -o milex -lfl -lm
*/
```

Fichero sintactico.y (Bison)

```
%{
/* Ejemplo para una pequeña calculadora que permite trabajar
con numeros enteros y reales con las operaciones básicas de
suma, resta, producto, division y trigonometricas como el seno y el coseno */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
extern int yylex(void);
extern char *yytext;
extern int nlines;
extern FILE *yyin;
void yyerror(char *s);
%}

%union
{
    float real;
}

%start Calculadora

%token <real> TKN_NUM
%token TKN_ASIGN
%token TKN_PTOCOMA
%token TKN_MULT
%token TKN_DIV
%token TKN_MAS
%token TKN_MENOS
%token TKN_PAA
%token TKN_PAC
```

```
%token TKN_COS
%token TKN_SEN
%token <real> TKN_ID

%type Calculadora
%type <real> Expresion

%left TKN_MAS TKN_MENOS
%left TKN_MULT TKN_DIV

%%

Calculadora : TKN_ID { printf("El valor de %s es: ", yytext);}
              TKN_ASIGN Expresion TKN_PTOCOMA { printf("%5.2f\n", $4); } ;

Expresion : TKN_NUM {$$=$1;}|
            Expresion TKN_MAS Expresion {$$=$1+$3;}|
            Expresion TKN_MENOS Expresion {$$=$1-$3;}|
            Expresion TKN_MULT Expresion {$$=$1*$3;}|
            Expresion TKN_DIV Expresion {$$=$1/$3;} |
           TKN_PAA Expresion TKN_PAC  {$$=$2;}|
            TKN_COS TKN_PAA Expresion TKN_PAC {$$=cos($3);}|
            TKN_SEN TKN_PAA Expresion TKN_PAC {$$=sin($3);};
%%

void yyerror(char *s)
{
     printf("Error %s",s);
}

int  main(int argc,char **argv)
{
if (argc>1)
   yyin=fopen(argv[1],"rt");
```

```
else
   yyin=stdin;

yyparse();

printf("FIN del Analisis. Entrada CORRECTA\n");
printf("Numero lineas analizadas: %d\n", nlines);

 return 0;
}

/* para compilar
bison -d sintactico.y
flex lexico.l
cc lex.yy.c  sintactico.tab.c -o analizador -lfl -lm

*/
```