Práctica 6

Domingo Gómez y Cristina Tîrnăucă

Materiales de la práctica:

- Generador de parsers bison http://gnuwin32.sourceforge.net/ para windows
- Generador de analizadores léxicos flex http://gnuwin32.sourceforge.net/
- Compilador g++

Objetivos de la práctica

- Seguir avanzando en el uso práctico de las gramáticas libres de contexto.
- Conocer el generador de analizadores léxicos flex.

Enunciado de la práctica

1. Dado el siguiente código:

```
%option noyywrap

%%

\n                          ;

[a-d][0-3]*                 printf("Expr. 1\n");

[^a-zA-Z0-9]        printf("Expr. 2\n");

[-ab.]+             printf("Expr. 3\n");

\[99\\[abc]*99\]            printf("Expr. 4\n");

^[a-d][0-3]*$       printf("Expr. 5\n");

.                           ECHO;
%%
int main()
{
```

```
  yylex();
  return 1;
}
```

Explicar la salida dando las siguientes entradas:

```
a33
[99\acbc99\]
 a33
%
-
- --
1
```

2. Se proporciona el siguiente fichero bison

```
%code provides {

#include <iostream>

#define YY_DECL yy::parser::token_type yylex(int *yylval)

YY_DECL;

}

%defines // ahorramos el -d al llamar a bison

%token END 0 "end of file"

%token NUM PLUS TIMES LPAR RPAR

%%

eval : expr { std::cout << $1 << std::endl; }

expr : NUM { $$ = $1; }
     | expr PLUS expr { $$ = $1 + $3; }
     | expr TIMES expr { $$ = $1 * $3; }
     | LPAR expr RPAR { $$ = $2; }
     ;

%%

using namespace yy;

void
```

```
  parser::error (const location_type& loc, const std::string& msg){
  std::cerr << msg << std::endl;
  exit(1);
}

int main() {
  parser p;
  p.parse();
}
```

y el siguiente archivo flex.

```
%{

#include "expr005xx.tab.hh"
#include <math.h>

typedef yy::parser::token token;

#define yyterminate() return token::END
int numeroCaracter=0;
%}

%option noyywrap

%%

[0-9]+    {*yylval = atoi(yytext); return(token::num);}

\(        return(token::lpar);
\)        return(token::rpar);
\+        return(token::plus);
\*        return(token::times);

.         ECHO; return(token::END);
\n        ;
%%
```

Cambiar la definición de un token número para que no pueda empezar por cero a menos que sea el número cero. Añadir a este analizador léxico un token que sea ** y que se llame elevado.