LEX
Modelos de Computación

Jose Antonio Padial Molina

josepadial@correo.ugr.es

November 14, 2019

Contents

1 Calculadora de escritorio

Con lex y yacc se va a desarrollar una calculadora simple de escritorio. La cual realiza operaciones de suma, resta, multiplicación y división. Este programa de calculadora también le permite asignar valores a las variables (cada una designada con una letra minúscula) y luego usar las variables en los cálculos.

2 calc.lex

Especifica el archivo de especificación del comando lex que define las reglas de análisis léxico.

```
%{

#include <stdio.h>
#include "y.tab.h"
int c;
%}
%%
" "         ;
[a-z]       {
              c = yytext[0];
              yylval.a = c - 'a';
              return(LETTER);
            }
[0-9]       {
              c = yytext[0];
              yylval.a = c - '0';
              return(DIGIT);
            }
[^a-z0-9\b]     {
                  c = yytext[0];
                  return(c);
                }
%%
```

3 calc.yacc

Especifica el archivo de gramática del comando yacc que define las reglas de análisis y llama a la subrutina yylex creada por el comando lex para proporcionar información.

El código contiene las siguientes secciones:

- Declaraciones.
- Reglas: inicio, %union, %type, %toksn.
- Programas: Principasl, errores.

```
%{
#include<stdio.h>
```

```
int regs[26];
int base;

%}

%start list

%union { int a; }


%token DIGIT LETTER

%left '|'
%left '&'
%left '+' '-'
%left '*' '/' '%'
%left UMINUS  /*supplies precedence for unary minus */

%%                   /* beginning of rules section */

list:                        /*empty */
         |
        list stat '\n'
         |
        list error '\n'
         {
           yyerrok;
         }
         ;
stat:    expr
         {
           printf("%d\n",$1);
         }
         |
         LETTER '=' expr
         {
           regs[$1.a] = $3.a;
         }

         ;

expr:    '(' expr ')'
         {
           $$ = $2;
         }
         |
```

```
expr '*' expr
{

  $$.a = $1.a * $3.a;
}
|
expr '/' expr
{
  $$.a = $1.a / $3.a;
}
|
expr '%' expr
{
  $$.a = $1.a % $3.a;
}
|
expr '+' expr
{
  $$.a = $1.a + $3.a;
}
|
expr '-' expr
{
  $$.a = $1.a - $3.a;
}
|
expr '&' expr
{
  $$.a = $1.a & $3.a;
}
|
expr '|' expr
{
  $$.a = $1.a | $3.a;
}
|

'-' expr %prec UMINUS
{
  $$.a = -$2.a;
}
|
LETTER
{
  $$.a = regs[$1.a];
}
```

```
        |
        number
        ;

number:  DIGIT
        {
          $$ = $1;
          base = ($1.a==0) ? 8 : 10;
        }       |
        number DIGIT
        {
          $$.a = base * $1.a + $2.a;
        }
        ;

%%
main()
{
 return(yyparse());
}

yyerror(s)
char *s;
{
  fprintf(stderr, "%s\n",s);
}

yywrap()
{
  return(1);
}
```

4 Compilar el programa

1. yacc -d calc.yacc
2. lex calc.lex
3. cc y.tab.c lex.yy.c
4. ./a.out

5 Capturas de pantalla