Universidade do Minho

Transcrição de aula

Disciplina	Programação Imperativa – 1º ano - LEI	
Secretário	Número: 51807	Nome: Ana Costa
Data: 2011-05-02		Nº Página 4

TP1

30 Alunos

SUMÁRIO

Desenvolvimento de parsers com o flex+yacc com acções semânticas.

1

+ - / * ( ) Z → Exp '$'

Recursividade à direita

Exp → Termo $$ = $1;
| Exp Add_Op Termo
$1 $2 $3

Termo → factor
| Termo Mul_op factor

factor → var
| Num
| '(' Exp ')'

Var → par

Num → num

Mul_Op → '*'
| '/'

Add_Op → '+'
| '-'

Transcrição (folha de continuação) 2

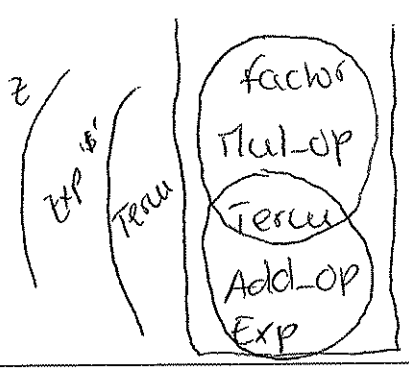
exp.l

```
%{
#include "y.tab.h"
%}
%%
[0-9]+ { yylval.inteiro = atoi(yytext); return NUM; }
[a-z] return PAL;
\( return '(';
\) return ')';
\+ return '+';
\- return '-';
\* return '*';
\/ return '/';
\n return '$';
[ \t] ;
.  ;
```

makefile

```
exp: y.tab.o lex.yy.o
	gcc -o exp y.tab.o lex.yy.o -lfl

y.tab.o: y.tab.c
	gcc -c y.tab.c

lex.yy.o: lex.yy.c
	gcc -c lex.yy.c

y.tab.c y.tab.h: exp.y
	yacc -d exp.y

lex.yy.c: exp.l y.tab.h
	flex exp.l
```

2 - 3 + 2 '$'



Transcrição (folha de continuação) 3

exp.y

```
%{
#include <stdio.h>
extern char* yytext;
%}

%union
{ int inteiro;
  char operador;
}

%token NUM PAL

%type <inteiro> NUM Num factor Term Exp
%type <operador> Mul-Op Add-Op

%%
Z: Exp '$' { printf("Z -> Exp $ :: %d\n", $1); };

Exp: Term {$$ = $1; printf("Exp -> Term :: %d\n", $$); }
   | Exp Add-Op Term {
        switch($2) { case '+': $$ = $1 + $3; break;
                     case '-'; $$ = $1 - $3; break;
                   }
        printf("Exp -> Exp Add-O Term :: %d\n", $$); }
   ;

Term: factor { $$ = $1; printf("Term -> factor :: %d\n", $$); }
    | Term Mul-Op factor {
        switch($2) { case '*': $$ = $1 * $3; break;
                     case '/': $$ = $1 / $3; break;
                   }
        printf("Term -> Term Mul-Op factor :: %d\n", $$); }
    ;

factor: var { $$ = 0; printf("factor -> var\n"); }
      | Num { $$ = $1; printf("factor -> NUM :: %d\n", $$); }
      | '(' Exp ')' { $$ = $2; printf("factor -> ( Exp ) :: %d\n", $$); }
      ;

var: PAL { printf("var -> PAL\n"); };
```

Transcrição (folha de continuação)

```
Num: NUM { $$ = $1; printf("Num → NUM :: %d\n", $$); };

Mul-OP: '*' { $$ = '*'; printf("Mul-op → *\n"); }
      | '/' { $$ = '/'; printf("Mul_OP → /\n"); }
      ;

Add_op: '+' { $$ = '+'; printf("Add_op → +\n"); }
      | '-' { $$ = '-'; printf("Add_OP → -\n"); }
      ;

%%
int yyerror(char *s)
{ fprintf(stderr, "ERRO: %s; TOKEN: %s\n", s, yytext);
}

int main()
{ yyparse();
}
```

```
?a
!a+z
b = a+5+z
!b
```

```
Z → Prog '$'
Prog → Com
     | Prog '\n' Com

Com → Input
    | Output
    | Atrib
```

```
Input → '?' Var
Output → '!' Exp

Atrib → Var = Exp
```