Apêndice C

Calculador Infixo

De seguida apresentam-se as especificações *Flex* e *Bison* usadas para criar um calculador de expressões aritméticas em notação infixa.

Reconhecedor Léxico: a especificação *Flex* trata do reconhecimento dos algarismos e dos símbolos de operação. Foi dado o nome de `incalc.l` ao ficheiro contendo a especificação *Flex*.

```
%{
#include "incalc.tab.h"
%}

%%
[ ]            /* Salta espaços em branco */
[0-9]+             sscanf(yytext,"%d",&yylval); return(NUM);
.|\n               return yytext[0];
```

É de notar a não existência da rotina `main`, isto deve-se ao facto de o reconhecedor léxico que se pretende criar não ser autónomo mas sim ir-se integrar no reconhecedor sintáctico produzido pelo *Bison*.

Reconhecedor Sintáctico: A especificação *Bison* contém as regras necessárias para o reconhecimento e posterior manipulação das expressões aritméticas em notação infixa.

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

[1] (Versão 1.1)

```
%token NUM
%left '-' '+'
%left '*' '/'
%left '~'
%right '^'

%%
input:          /* linha vazia */
      |         input line
;

line:           '\n'
     |          exp '\n' {printf("\t%d\n",$1);}
     |          error '\n' {yyerrok;}
;

exp:            NUM         {$$ = $1;}
    |           exp '+' exp {$$ = $1 + $3;}
    |           exp '-' exp {$$ = $1 - $3;}
    |           exp '*' exp {$$ = $1 * $3;}
    |           exp '/' exp {$$ = $1 / $3;}
/* potência */
    |           exp '^' exp {$$ = pow($1,$3);}
/* menos unário */
    |           '~' exp     {$$ = -$2;}
    |           '(' exp ')' {$$ = $2;}
;

%%

main() {
  yyparse();
}

yyerror(s)
 char *s; {
 printf("%s\n",s);
}
```

Como compilar:

```
> bison -d incalc.y
> flex incalc.l
> gcc lex.yy.c incalc.tab.c -lfl -lm -o incalc
```