polish 2

D. Michelucci

Dans cette variante, c'est l'ana ;yseur syntaxique qui découpe la chaîne d'entrée en lexèmes. Les règles yacc doivent donc gérer les espaces.

1 Fichier polish.y

```
/* Reverse polish notation calculator. */
%{
#define YYSTYPE int
#include <math.h>
#include <stdio.h>
int yylex();
int yyerror( char *);
%}
%token DIGIT SP
%% /* Grammar rules and actions follow */
input : /* empty */
   | input line
;
line : spacefac '\n'
   | spacefac exp spacefac '\n'  { printf ("\t%d\n", $2);
      } ;
chiffre : DIGIT { $$ = $1; } ;
nb : chiffre  { $$=$1; }
   | nb chiffre { $$= 10 * $1 + $2 ; } ;
/* space : SP | space SP {}; */
spacefac: | spacefac SP {} ;
exp : nb { $$ = $1;          }
   | exp spacefac exp spacefac '+'     { $$ = $1 + $3;
         }
   | exp spacefac exp spacefac '*'     { $$ = $1 * $3;
         }
   | exp spacefac exp spacefac '^'     { $$ = pow ($1, $3
      ); }
   | exp spacefac 'n'                  { $$ = -$1;
            } ;
%%

/*
Dans cette variante de la calculatrice avec notation
   postfixe (polonaise),
```

```
les noombres sont des entiers, et c'est l'analyse
   syntaxique qui reconnait les entiers
(et pas l'analyse lexicale).
C'est donc possible.
Mais il faut gerer dans la grammaire le decoupage en mots
   : si l'entree est "1 2", il
faut lire 2 entiers, et pas l'entier 12 ! C'est possible
   de gerer les espaces
au miveau syntaxique (et non lexical), mais c'est quand
   meme un peu penible.

C'est pourquoi le niveau lexical reste pertinent.
*/
#include <ctype.h>
int yylex ()
{ int c;
  if ((c = getchar ()) == ' ' || c == '\t') return SP;
  if (isdigit (c)) { yylval= c - '0'; return DIGIT; }
  if (c == EOF)                              return 0;
  /* return single chars */
  return c;
}
main ()      /* The ``Main'' function to make this stand-
   alone  */
{
  yyparse ();
}
#include <stdio.h>
int yyerror (char *s)  /* Called by yyparse on error */
{
  printf ("%s\n", s);
}
```

2 Fichier makefile

```
ok : polish.y
        #lex calcul.lex
        yacc polish.y -o polish.cpp
        g++ -o polish polish.cpp   -lfl -lc
```