Lex et Yacc, exemple 2 de somme ou produit de liste d'entiers

D. Michelucci

1 fichier makefile

```
ok: exemple_liste2.pdf liste2
liste2: liste2.lex liste2.yacc makefile
	lex liste2.lex
	yacc -o liste2.cpp liste2.yacc
	g++ -o liste2 liste2.cpp -lfl -ly -lc

exemple_liste2.pdf : exemple_liste2.tex  liste2.lex liste2.yacc makefile
	latex exemple_liste2
	dvipdf exemple_liste2

clean:
	rm -fr liste2.cpp  liste2
	rm -fr lex.yy.c
	rm -fr *.aux *.log  *.dvi
```

2 fichier liste2.lex

```
%{
#include<stdlib.h>
%}
%%
[ \t\n] { ;}
[0-9]+  { yylval= atoi( yytext); return ENTIER; }
produit { return PRODUIT  ; }
somme { return SOMME  ; }
","     {return  ','; }
"."     {return FIN; }
"("     { return LP; }
")"     { return RP; }
<<EOF>> {return FIN; }
.       {printf("caractere inconnu: [%c]\n", yytext[0]); }
%%
```

3 fichier liste2.yacc

```
%{
#include<stdio.h>
#include<stdlib.h>
int res; /* partout, on pourrait remplacer res par $$, ou une autre valeur $ */
extern "C"
{
        int yydebug;
        int yyparse();
        void yyerror(char*);
        int yylex();
        int yywrap() { return 1; }
};
%}
%token SOMME PRODUIT
%token ENTIER
%token FIN
%token LP RP
%%

programme: FIN { printf("Adieu, monde cruel\n"); return 0; }
| expr FIN { printf("resultat=%d\n", $1); } programme ;
expr: ENTIER { $$=$1; }
|     SOMME ls { $$=$2;}
|     PRODUIT lp { $$=$2;}
|     LP expr RP { $$=$2; } ;
ls: expr { $$=$1; }
|   expr ',' ls {$$=$1+$3;} ;
lp: expr { $$=$1;}
|   expr ',' lp {$$=$1*$3;} ;


%%
#include "lex.yy.c"
extern int yydebug;
/* int main() { yyparse(); } */
main()
{
        yydebug=1;
        yyparse();
}

/* autres regles possibles pour ls et lp :
ls: { $$=0;} |  ls ',' ENTIER { $$= $1 + $3; };
lp: { $$=1;} |  lp ',' ENTIER { $$= $1 * $3; };
elles admettent:
somme .
produit .
*/
```

4 Session

```
produit 1,2,3,4.
somme 1,2,3,4.
produit 2, (somme 1, 2, 3), 4.
.
```