Podstawy Kompilatorów

Laboratorium 11

Generator YACC: gramatyki niejednoznaczne.

Zadanie 1:
Zaimplementować kalkulator operujący na liczbach zmiennoprzecinkowych i udostępniający operacje: dodawania (+), odejmowania (-), mnożenia (*), dzielenia (/), potęgowania (^), unarny minus i plus oraz funkcję sinus (sin). Operatory +, -, * i / mają mieć lewostronną łączność i standardowe priorytety, operator ^ - prawostronną łączność i priorytet wyższy od operatorów binarnych.

Konstruując analizator składniowy nie wolno posługiwać się zmiennymi globalnymi

Przykłady:
Dla pliku o postaci: **1.1+0.1*2**
powinniśmy otrzymać wynik: **1.3**

Dla pliku o postaci: **2^3^2**
powinniśmy otrzymać wynik: **512**

Dla pliku o postaci: **2*sin(0)**
powinniśmy otrzymać wynik: **0**

Analizator leksykalny ma następującą postać:

```
%{
  int yywrap(void);
  int yylex(void);
  #include <stdio.h>
  #include "y.tab.h"
%}
%%
\+                  { return '+'; }
\-                  { return '-'; }
\*                  { return '*'; }
\/                  { return '/'; }
\^                  { return '^'; }
\(                  { return '('; }
\)                  { return ')'; }
[0-9]+("."[0-9]*)?  { yylval.val = atof(yytext);
                      return NUM;
                    }
sin                 { return FUN_SIN; }
%%
int yywrap(void) { return 1;}
```

Zadanie 2:
Dana jest niejednoznaczna gramatyka dla uproszczonego kalkulatora zapisana w YACCu (w komentarzach ponumerowano produkcje):

```
E : E '+' E    /* 1 */
  | E '-' E    /* 2 */
  |   '-' E    /* 3 */
  | NUM        /* 4 */
  ;
```

Tablice A/G analizatora LR obliczono metodą SLR (obliczenie przedstawiono poniżej), a konflikty rozwiązano w sposób przedstawiony w tabeli.
Proszę w równoważny sposób ujednoznacznić przedstawioną gramatykę w YACCu.

s_0={E'→•E, E→•E'+'E, E→•E'-'E, E→•'-'E, E→•NUM}
s_1=goto(s_0,E)={E'→E•, E→E•'+'E, E→E•'-'E}
s_2=goto(s_0,'-')={E→'-'•E, E→•E'+'E, E→•E'-'E, E→•'-'E, E→•NUM}
s_3=goto(s_0,NUM)={E→NUM• }
s_4=goto(s_1,'+')={E→E'+'•E, E→•E'+'E, E→•E'-'E, E→•'-'E, E→•NUM}
s_5=goto(s_1,'-')={E→E'-'•E, E→•E'+'E, E→•E'-'E, E→•'-'E, E→•NUM}
s_6=goto(s_2,E) = {E→'-'E•, E→E•'+'E, E→E•'-'E}
goto(s_2,'-') = s_2
goto(s_2,NUM) = s_3
s_7=goto(s_4,E) = {E→E'+'E•, E→E•'+'E, E→E•'-'E}
goto(s_4,'-') = s_2
goto(s_4,NUM) = s_3
s_8=goto(s_5,E) = {E→E'-'E•, E→E•'+'E, E→E•'-'E}
goto(s_5,'-') = s_2
goto(s_5,NUM) = s_3
goto(s_6,'+') = s_4
goto(s_6,'-') = s_5
goto(s_7,'+') = s_4
goto(s_7,'-') = s_5
goto(s_8,'+') = s_4
goto(s_8,'-') = s_5

FIRST(E) = {'-', NUM}
FOLLOW(E) = {'+', '-', $}

	'+'	'-'	NUM	$	E
s_0		s2	s3		1
s_1	s4	s5		ACC	
s_2		s2	s3		6
s_3	r4	r4		r4	
s_4		s2	s3		7
s_5		s2	s3		8
s_6	s4/r3	s5/r3		r3	
s_7	s4/r1	s5/r1		r1	
s_8	s4/r2	s5/r2		r2	

Zadanie 3:
Dla języka $a^n b^n c^n$ (a, b, c $\geq$ 0) skonstruowano i zapisano w YACCu niejednoznaczną gramatykę:

```
P : A B C
  ;
A : 'a' A { printf("1"); }
  | 'a'   { printf("2"); }
  |       { printf("3"); }
  ;
B : 'b' B { printf("4"); }
  | 'b'   { printf("5"); }
  |       { printf("6"); }
  ;
C : 'c' C { printf("7"); }
  | 'c'   { printf("8"); }
  |       { printf("9"); }
  ;
```

Dla pliku o postaci:
aabbcc
analizator daje odpowiedź:
215487
proszę zreorganizować produkcje (zmienić sposób rozstrzygania konfliktów redukcja/redukcja) tak, aby uzyskać odpowiedź:
311644977

Odpowiedzi do zadań

Zadanie 1:

```
%union
{
  double val;
}

%{
  int yylex(void);
  void yyerror(const char *,...);
  int yyparse(void);
  #include <stdio.h>
  #include <math.h>
  extern int yylineno;
%}

%token <val> NUM
%token FUN_SIN
%left '+' '-'
%left '*' '/'
%right '^'
%left UMINUS
%type <val> L E

%%
L : E                       { printf("%f",$1); }
  ;
E : E '+' E                 { $$ = $1 + $3; }
  |   '+' E  %prec UMINUS   { $$ = $2;      }
  | E '-' E                 { $$ = $1 - $3; }
  |   '-' E  %prec UMINUS   { $$ = -$2;     }
  | E '*' E                 { $$ = $1 * $3; }
  | E '/' E                 { $$ = $1 / $3; }
  | '(' E ')'               { $$ = $2;      }
  | E '^' E                 { $$ = pow($1,$3); }
  | FUN_SIN '(' E ')'       { $$ = sin($3); }
  | NUM                     { $$ = $1;      }
  ;
%%
void yyerror(const char *fmt,...) {
   printf("%s in line %d\n", fmt, yylineno); }
int main() { return yyparse(); }
```

Zadanie 2:

```
%right '-'
%right '+'
%left UMINUS
%%
E : E '+' E                  /* 1 */
  | E '-' E                  /* 2 */
  |   '-' E  %prec UMINUS    /* 3 */
  | NUM                      /* 4 */
  ;
```

Zadanie 3:

```
P : A B C
  ;
A : 'a' A { printf("1"); }
  |       { printf("3"); }
  | 'a'   { printf("2"); }
  ;
B : 'b' B { printf("4"); }
  |       { printf("6"); }
  | 'b'   { printf("5"); }
  ;
C : 'c' C { printf("7"); }
  |       { printf("9"); }
  | 'c'   { printf("8"); }
  ;
```