Kalkulator bez akcija

```
/* Jednostavni kalkulator */

%{
   #include "calc1.tab.h"
%}

%%

[ \t]+
[0-9]+    { yylval = atoi(yytext); return NUMBER; }
"+"       { return PLUS; }
"-"       { return MINUS; }
\n        { return NEWLINE; }
.         { printf("Unknown char %c\n", *yytext); }
```

```
%{
   #include <ctype.h>
   #include <stdio.h>
   int yyparse(void);
   int yylex(void);
%}

%token NUMBER
%token PLUS
%token MINUS
%token NEWLINE

%%

lines
   :
   |  lines NEWLINE
   |  lines e NEWLINE
   ;

e
   :  e PLUS NUMBER
   |  e MINUS NUMBER
   |  NUMBER
   ;

%%

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

```
int yyerror(char *s) {
   fprintf(stderr, "%s\n", s);
   return 0;
}
```

Kalkulator sa akcijama

```
%{
   #include <ctype.h>
   #include <stdio.h>
   int yyparse(void);
   int yylex(void);
%}

%token NUMBER
%token PLUS
%token MINUS
%token NEWLINE

%%

lines
   :
   |  lines NEWLINE
   |  lines e NEWLINE   { printf("%d\n", $2); }
   ;

e
   :  e PLUS NUMBER     { $$ = $1 + $3; }
   |  e MINUS NUMBER    { $$ = $1 - $3; }
   |  NUMBER            { $$ = $1; }
   ;

%%

int main() {
   return yyparse();
}

int yyerror(char *s) {
   fprintf(stderr, "%s\n", s);
   return 0;
}
```

Rukovanje greškama

```
%{
   #include <ctype.h>
   #include <stdio.h>
   int yyparse(void);
   int yylex(void);
%}

%token NUMBER
%token PLUS
%token MINUS
%token NEWLINE

%%

lines
   :  /* prazna linija */
   |  lines NEWLINE
   |  lines e NEWLINE      { printf("%d\n", $2); }
   |  lines error NEWLINE  { yyerror("reenter last line:\n");
                             yyerrok; }
   ;

e
   :  e PLUS NUMBER        { $$ = $1 + $3; }
   |  e MINUS NUMBER       { $$ = $1 - $3; }
   |  NUMBER               { $$ = $1; }
   ;

%%

int main() {
   return yyparse();
}

int yyerror(char *s) {
   fprintf(stderr, "%s\n", s);
   return 0;
}
```

Kompletan kalkulator - prioritet operatora

```
%{
   #include <ctype.h>
   #include <stdio.h>
   int yyparse(void);
   int yylex(void);
%}
```

```
%token NUMBER
%token PLUS
%token MINUS
%token MULTIPLY
%token DIVIDE
%token NEWLINE

%left  PLUS MINUS
%left  MULTIPLY DIVIDE
%right UMINUS

%%

lines
   :  /* prazna linija */
   |  lines NEWLINE
   |  lines e NEWLINE          { printf("%d\n", $2); }
   ;

e
   :  e PLUS e                 { $$ = $1 + $3; }
   |  e MINUS e                { $$ = $1 - $3; }
   |  e MULTIPLY e             { $$ = $1 * $3; }
   |  e DIVIDE e               { $$ = $1 / $3; }
   |  MINUS e %prec UMINUS     { $$ = -$2; }
   |  NUMBER                   { $$ = $1; }
   ;

%%

int main() {
   return yyparse();
}

int yyerror(char *s) {
   fprintf(stderr, "%s\n", s);
   return 0;
}
```