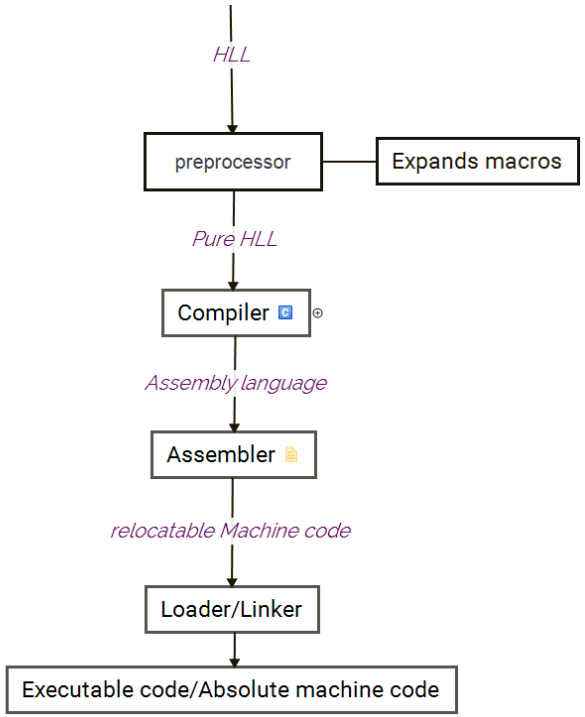
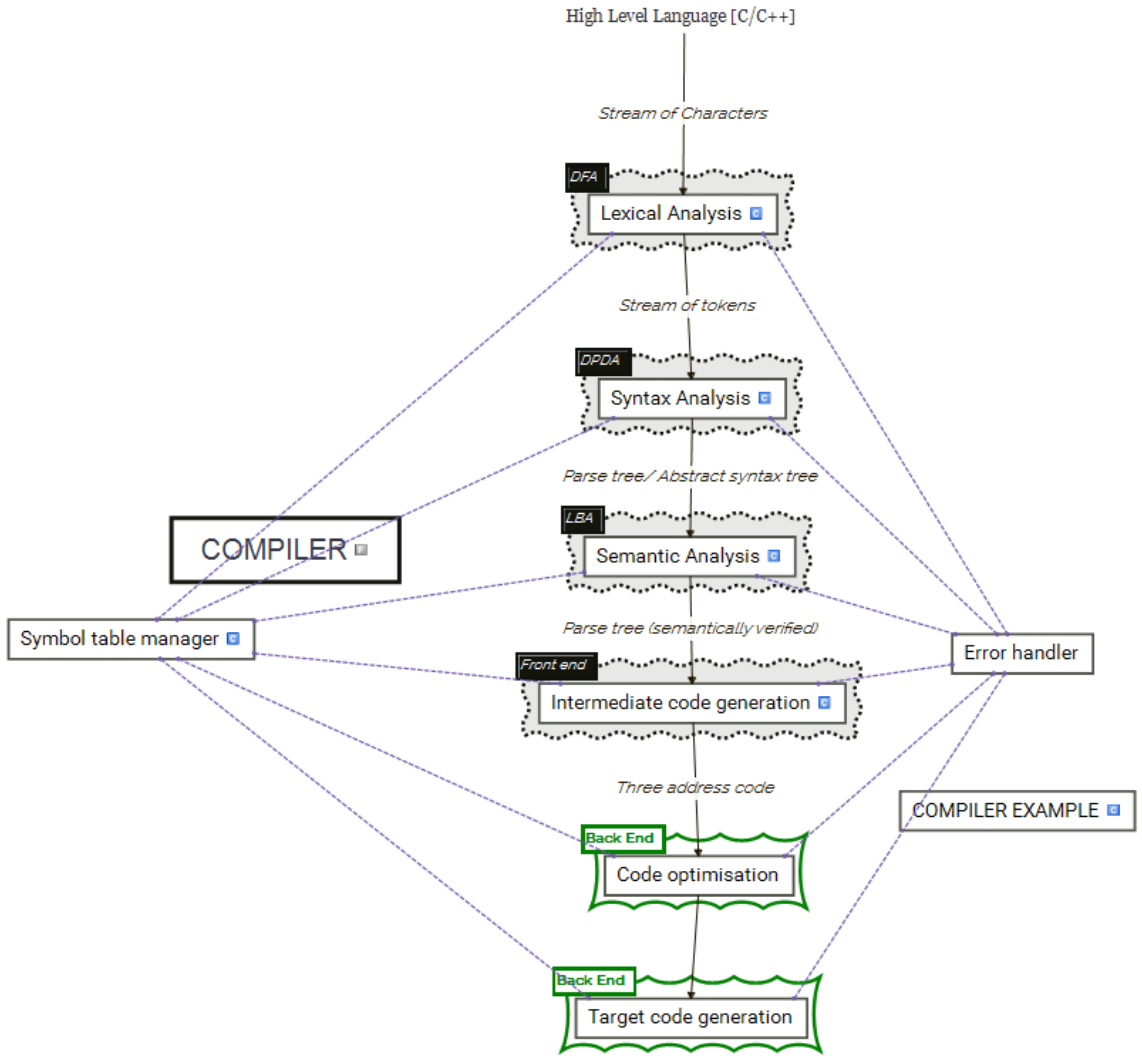
Yacc/Bison

Compiler overview





Introduction to Bison

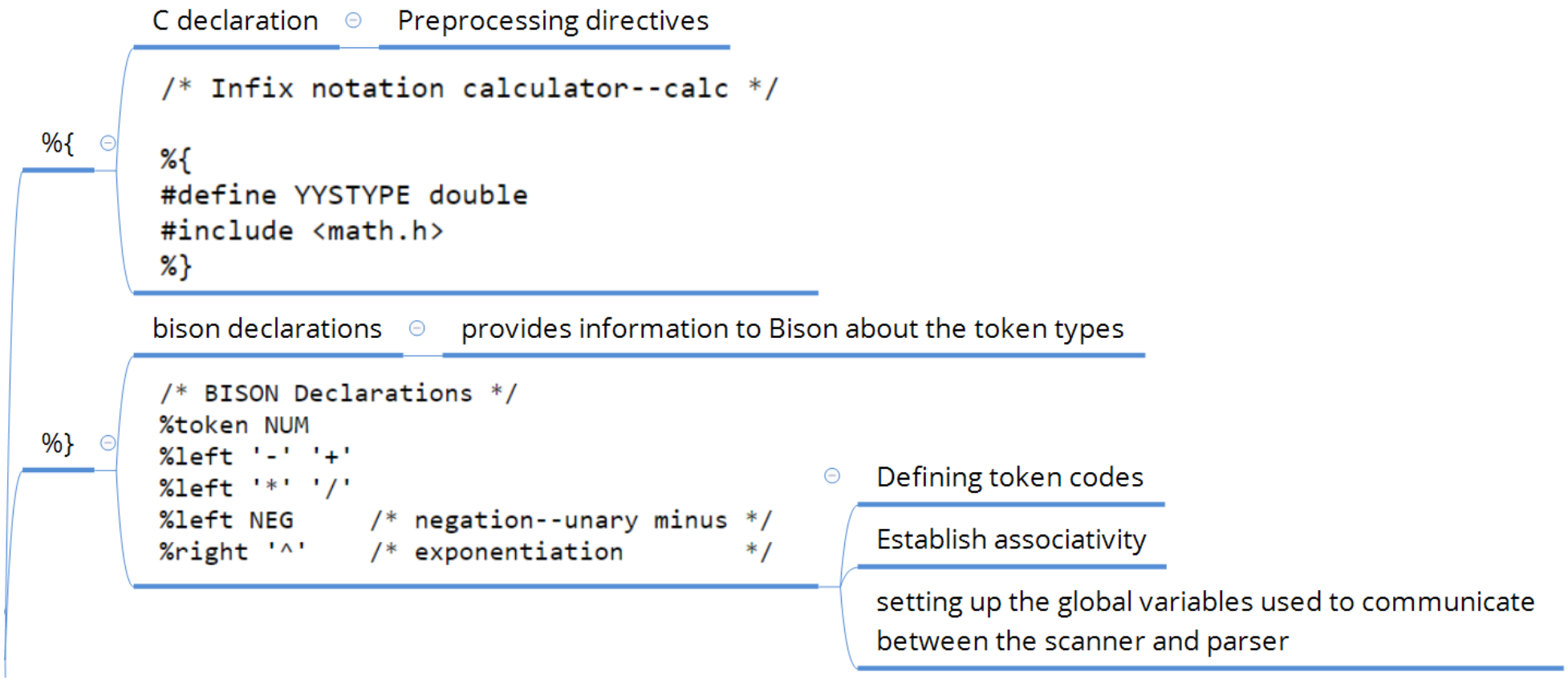
- What is parser?
- Two basic parsers:
  - Top down
  - Bottom –up
- What are flex and bison?
- Why should you learn Flex/Bison pattern syntax when you could just write your own parser?
- What is the difference between lex/flex and yacc/bison?

Bison file structure

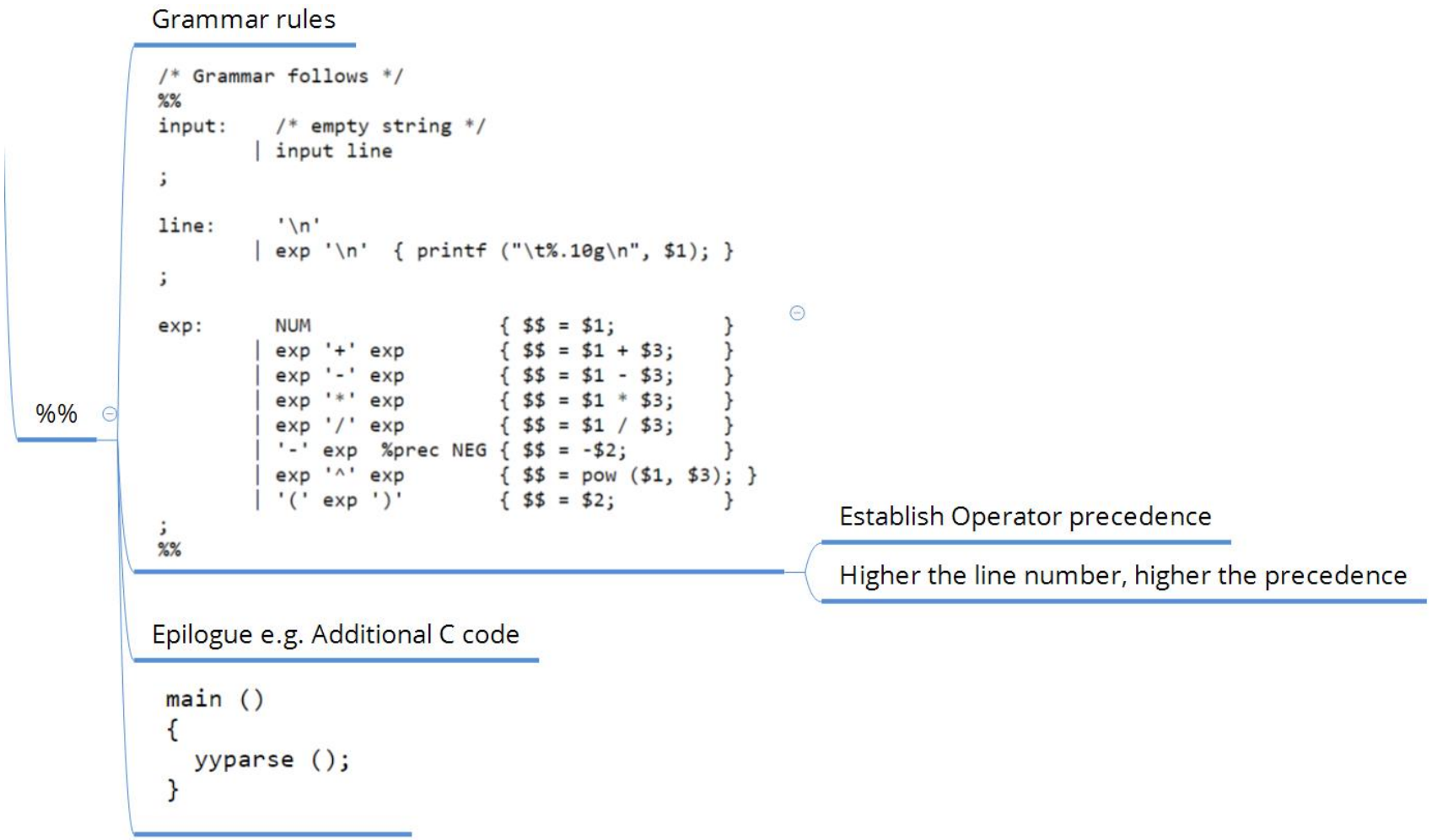
- A bison input file (bison grammar file) has the following structure with special punctuation symbols %%, %{, %}

```
%{
Prologue e.g. C declaration
%}
bison declarations
%%
Grammar rules
%%
Epilogue e.g. Additional C code
```

Bison file structure -continued



Bison file structure - continued



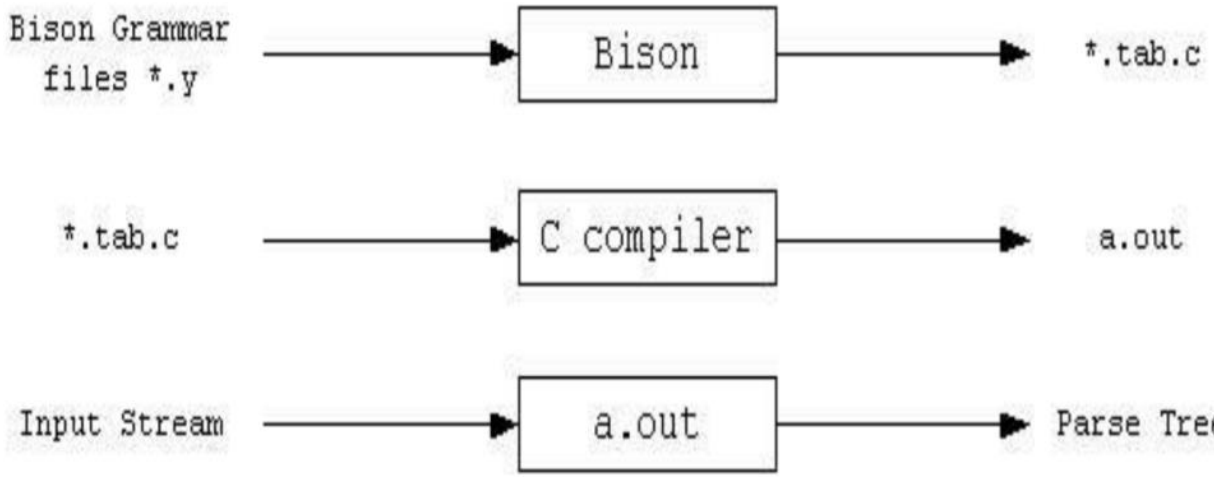
Flex File

```
/* Mini Calculator */
/* calc.lex */

%{
#include "heading.h"
#include "tok.h"
int yyerror(char *s);
int yylineno = 1;
%}

digit           [0-9]
int_const       {digit}+

%%

{int_const}     { yylval.int_val = atoi(yytext); return INTEGER_LITERAL; }
"+"             { yylval.op_val = new std::string(yytext); return PLUS; }
"*"             { yylval.op_val = new std::string(yytext); return MULT; }

[ \t]*          {}
[\n]            { yylineno++;   }

.               { std::cerr << "SCANNER "; yyerror(""); exit(1);        }
```

Definitions ⊖ include declarations of constant, variable and regular definitions.

Rules ⊖ define the statement of form p1 {action1} p2 {action2}....pn {action}.

How to execute bison file ?

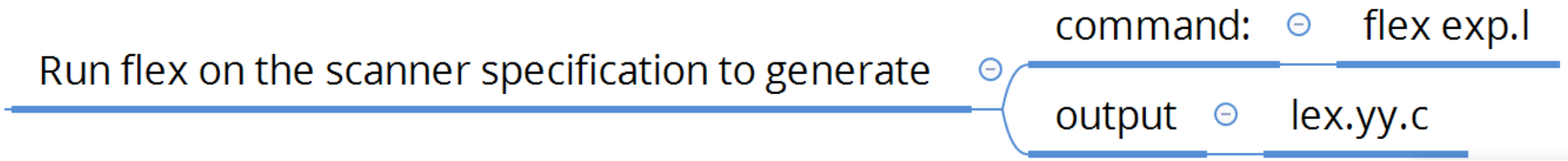
- 

.Compile the example.y file using bison.

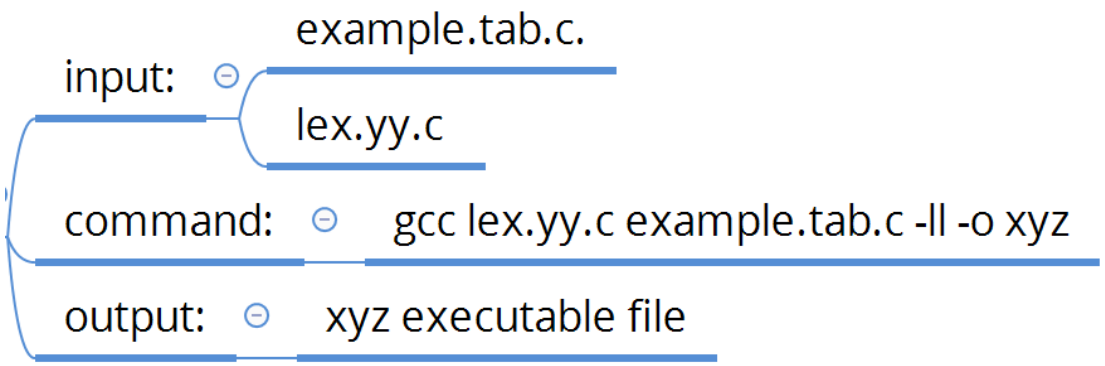
- command: bison -d example.y
- output
  - example.tab.c.
  - example.tab.h
- Write the grammar specification for bison (example.y), including grammar rules, yyparse() and yyerror().

Execution steps

- Write a lexical analyzer to process input and pass tokens to the parser



- Compile the two .c files and link them together



- Run