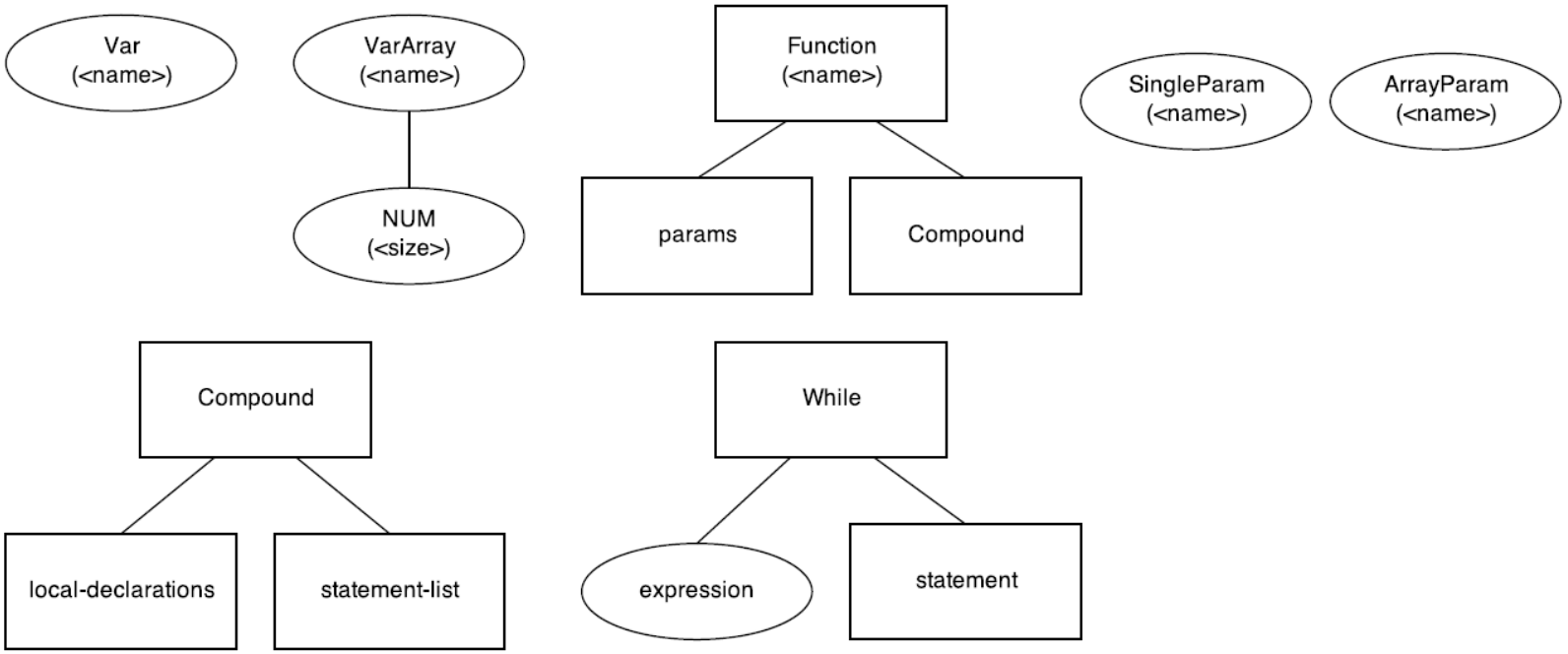
Project #2. Parser

Parser

- **C-Minus Parser Implementation**
  - **Implement the parser using Yacc (bison)**
    - C-Minus Scanner with Flex should be used.
    - Some source code should be obtained using Yacc (bison)

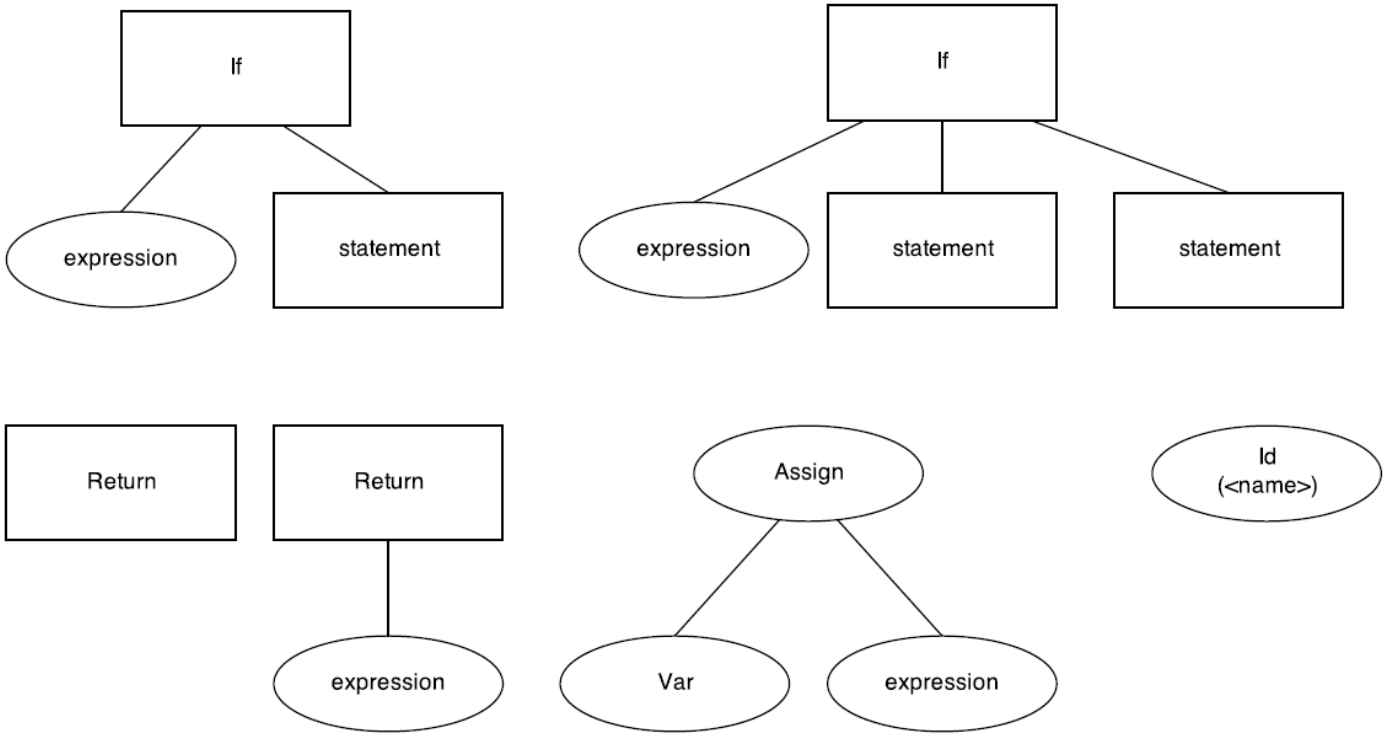
Parser Goal

- **Syntax Tree Definition**



Parser Goal

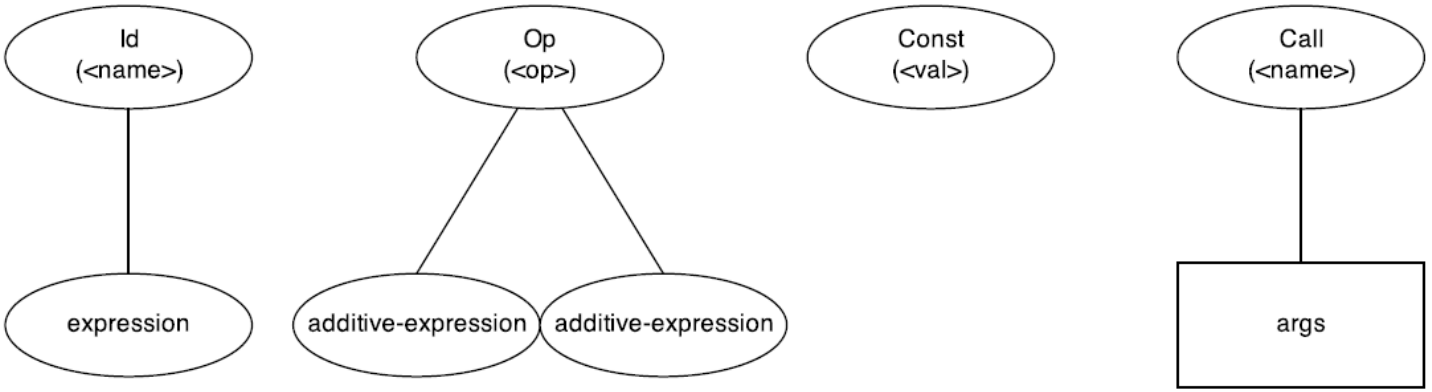
- **Syntax Tree Definition**



Hanyang University
Division of Computer Science & Engineering

Parser Goal

- **Syntax Tree Definition**



BNF Grammar for C-Minus

- **Appendix A.2**

1. *program* → *declaration-list*
2. *declaration-list* → *declaration-list declaration* | *declaration*
3. *declaration* → *var-declaration* | *fun-declaration*
4. *var-declaration* → *type-specifier* **ID ;** | *type-specifier* **ID [NUM] ;**
5. *type-specifier* → **int** | **void**
6. *fun-declaration* → *type-specifier* **ID (** *params* **)** *compound-stmt*
7. *params* → *param-list* | **void**
8. *param-list* → *param-list* **,** *param* | *param*
9. *param* → *type-specifier* **ID** | *type-specifier* **ID [ ]**
10. *compound-stmt* → **{** *local-declarations statement-list* **}**
11. *local-declarations* → *local-declarations var-declarations* | *empty*
12. *statement-list* → *statement-list statement* | *empty*
13. *statement* → *expression-stmt* | *compound-stmt* | *selection-stmt* | *iteration-stmt* | *return-stmt*
14. *expression-stmt* → *expression* **;** | **;**
15. *selection-stmt* → **if (** *expression* **)** *statement* | **if (** *expression* **)** *statement* **else** *statement*
16. *iteration-stmt* → **while (** *expression* **)** *statement*
17. *return-stmt* → **return ;** | **return** *expression* **;**
18. *expression* → *var* **=** *expression* | *simple-expression*
19. *var* → **ID** | **ID [** *expression* **]**
20. *simple-expression* → *additive-expression relop additive-expression* | *additive-expression*
21. *relop* → **<=** | **<** | **>** | **>=** | **==** | **!=**
22. *additive-expression* → *additive-expression addop term* | *term*
23. *addop* → **+** | **-**
24. *term* → *term mulop factor* | *factor*
25. *mulop* → ***** | **/**
26. *factor* → **(** *expression* **)** | *var* | *call* | **NUM**
27. *call* → **ID (** *args* **)**
28. *args* → *arg-list* | *empty*
29. *arg-list* → *arg-list* **,** *expression* | *expression*

Hint: where to see?

- **Parse.c**
  - To modify the code to meet C-Minus syntax
- **Util.c**
  - printTree function should be updated to print C-Minus Syntax Tree
- **Main.c**
  - To modify code to print only Syntax Tree
- **Globals.h**
  - "Syntax tree for parsing" should be updated to meet C-Minus Spec

Example (Syntax tree)

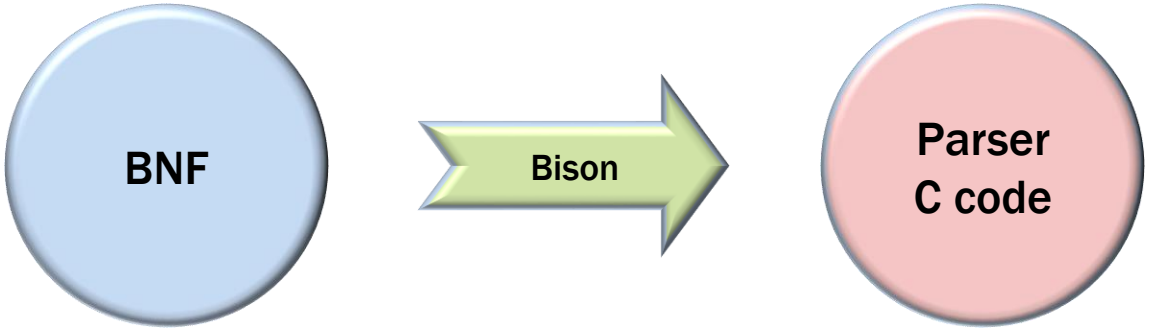
```
/* A program to perform Euclid's
   Algorithm to computer gcd */

int gcd (int u, int v)
{
    if (v == 0) return u;
    else return gcd(v,u-u/v*v);
    /* u-u/v*v == u mod v */
}

void main(void)
{
    int x; int y;
    x = input(); y = input();
    output(gcd(x,y));
}
```

```
Syntax tree:
  Function declaration, name : gcd, return type : int
    Single parameter, name : u, type : int
    Single parameter, name : v, type : int
    Compound statement :
      If (condition) (body) (else)
        Op : ==
          Id : v
          Const : 0
        Return :
          Id : u
        Return :
          Call, name : gcd, with arguments below
            Id : v
            Op : -
              Id : u
              Op : *
                Op : /
                  Id : u
                  Id : v
                Id : v
  Function declaration, name : main, return type : void
    Single parameter, name : (null), type : void
    Compound statement :
      Var declaration, name : x, type : int
      Var declaration, name : y, type : int
      Assign : (destination) (source)
        Id : x
        Call, name : input, with arguments below
      Assign : (destination) (source)
        Id : y
        Call, name : input, with arguments below
      Call, name : output, with arguments below
        Call, name : gcd, with arguments below
          Id : x
          Id : y
```

Yacc (bison)

- **Yacc: Parser generator for UNIX**
  - Yet Another Compiler Compiler
  - Bison: GNU Project parser generator (yacc replacement)

- **Input BNF**
- **Output: C-code of parser for the input BNF**



Yacc (bison) source description

Definitions

%%

Rules (BNF syntax)

%%

~~Subroutines~~

(You don't need to modify this part)

Hanyang University
Division of Computer Science & Engineering

Yacc (bison) source - tiny

- **definitions**

```
%token IF THEN ELSE END REPEAT UNTIL READ WRITE
%token ID NUM
%token ASSIGN EQ LT PLUS MINUS TIMES OVER LPAREN RPAREN SEMI
%token ERROR
```

- **rules**

```
          $$        $1 $2 $3    $4   $5
if_stmt     : IF exp THEN stmt_seq END
                { $$ = newStmtNode(IfK);
                  $$->child[0] = $2;
                  $$->child[1] = $4;
                }
            | IF exp THEN stmt_seq ELSE stmt_seq END
                { $$ = newStmtNode(IfK);
                  $$->child[0] = $2;
                  $$->child[1] = $4;
                  $$->child[2] = $6;
                }
            ;
```

Yacc (bison) Usage

Usage: yacc [options] filename

Options:

-d write definitions (y.tab.h)
-o output_file (default "y.tab.c")
-t add debugging support
-v write description (y.output)

- **[user@cminus]$ yacc -d yacc/cminus.y**
  - You need to copy the following files to the main project directory after running the yacc
    - y.tab.h, tiny.tab.c(modify this to parse.c)
    - globals.h(overwrite) in the Yacc directory

Yacc (bison) Manual

- **Manual**
  http://www.gnu.org/software/bison/manual/ (English)

Some Comments

- **You don’t need to generate exactly same output. If you generate the right result, it will be okay.**

- **You don’t need to care about Semantics, just Syntax analyzer will be okay.**

Some Comments

```
ex.) void main () {
        int a, b;
        c = a + b;
     }
```

- **For this example, this code will be parsed correctly even though the code has some semantic error.**

Report

- **Guideline**
  - Compilation method and environment
  - Explanation about how to implement and how to operate
  - Some explanation about the modified code
  - Example and Result Screenshot

- **File format**
  - MS Word, HWP, PDF, …

Submission (Important!!)

- **Submission**
  - Using Git
    - https://hconnect.hanyang.ac.kr
    - https://hconnect.hanyang.ac.kr/2017_ITE4029_10042/2017_ITE4029_Student#
  - TA
    - jht008+compiler@gmail.com
    - If you don't have the GIT account, please let him know the account information after creation.
    - You can still use my email: yj99.compiler@gmail.com
- **What to submit**
  - All the source codes and the report

Deadline

- **Deadline**
  - Push deadline: 2017/11/10 (YYYY/MM/DD) (Friday) 23:59:59
  - Clone: 2017/11/11 (YYYY/MM/DD) (Saturday) 00:00:00
  - Master branch