Compiler Project#2 Parser

Software Engineering
2015004120 Park Seonha

1. Goal of Project#2

- Implementation of C minus Parser using Yacc (bison)

2. Environment

OS : Ubuntu 16.04
Environment : GCC(5.4.0), flex(2.6.0), GNU Make(4.1), bison(3.0.4)
Executable : cminus (./loucomp/cminus)

3. Implementation

* cminus.l
To solve undefined reference to yywrap() problem, add '%option noyywrap' in this file.

* cminus.y (added)
Modify tokens and rules in yacc/tiny.y to meet cminus syntax rules in BNF.

* globals.h
Copy whole file in yacc/globals.h and modify nodekinds for cminus syntax tree. Include y.tab.h to get token values generated by bison.

* main.c
Set options to trace Parser not Scaneer. NO_PARSE, NO_ANALYZE, TraceScan, TraceParse is changed.

* util.h & util.c
Change tiny compiler's node generation to meet cminus syntax tree's nodekind, printTree is also modified to print cminus syntax tree.

* Makefile
Remove cminus_flex and tm. In generate cminus execution, remove analysis.o, code.o, cgen.o since they are not modified for c minus. Add y.tab.o and it is generated by bison and cminus.h. Since lex.yy.o is in OBJS not OBJS_FLEX, cminus_flex and tm removed, 'make clean' is modified that do not remove non exist executions.

4. Compilation

Makefile is modified for Parser.

To compile, use '**make**' or '**make cminus**' to generate executable. **bison** must be installed in machine since **y.tab.o** and **y.tab.h** is generated automatically by bison(yacc).

5. Result Screenshot

```
cmla@cmla-VirtualBox:~/2017_ITE4029_2015004120/loucomp$ cat test.cm
/* 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));
}
```

test data (test.cm)

```
cmla@cmla-VirtualBox:~/2017_ITE4029_2015004120/loucomp$ ./cminus test.cm

C-MINUS COMPILATION: test.cm

Syntax tree:
  Function Decleration : gcd
    Type: int
    Non Array Parameter: u
      Type: int
    Non Array Parameter: v
      Type: int
    Compound Statement
      If Statement
        Op: ==
          Id: v
          Const: 0
        Return
          Id: u
        Return
          Call: gcd
            Id: v
            Op: -
              Id: u
              Op: *
                Op: /
                  Id: u
                  Id: v
                Id: v
  Function Declaration : main
```

```
  Function Decleration : main
    Type: void
    Type: void
    Compound Statement
      Var Decleration: x
        Type: int
      Var Decleration: y
        Type: int
      Assign to: (null)
        Id: x
        Call: input
      Assign to: (null)
        Id: y
        Call: input
      Call: output
        Call: gcd
          Id: x
          Id: y
```

result of test.cm

```
cmla@cmla-VirtualBox:~/2017_ITE4029_2015004120/loucomp$ cat array.cm
int main(void)
{
  int arr[32];
}
cmla@cmla-VirtualBox:~/2017_ITE4029_2015004120/loucomp$ ./cminus array.cm

C-MINUS COMPILATION: array.cm

Syntax tree:
  Function Decleration : main
    Type: int
    Type: void
    Compound Statement
      Array Var Decleration: arr  [size : 32]
        Type: int
```

Since example doesn’t have array check, make new test for check array.