rusper(RUby Scheme PERl)

예제 소스

최대 공약수 구하기

```
VAR a
VAR b
VAR r

; initialize a
=(a, 64)

; initialize b
=(b, 12)

DO(-1)
::
        =(r, a)
        =(a, b)
        =(b, %(r, b))

        IF(<=(b, 0))
        ::
                BREAK
        :
:

; print gcd
["gcd is " {a} ]

; end of source
```

소수 판별 프로그램

```
VAR prime
VAR i
VAR flag

OUTPUT("prime_result.txt")

; initialize prime
=(prime, 131)

; initialize flag
=(flag, 0)

IF(&&(<(2, prime), !=(0, %(prime, 2))))
::
        ; initialize i
        =(i, 3)

        DO(-1)
        ::
                IF(<=(prime, i))
                ::
                        =(flag, 1)
                        BREAK
                :

                IF(==(0, %(prime, i)))
                ::
                        BREAK
                :

                =(i, +(i, 2))
        :
:

IF(==(1, flag))
::
        [{prime} "is prime."]
:

IF(==(0, flag))
::
        [{prime} "is not prime."]
:

; end of source
```

랜덤 리스트 출력

```
DO(10)
::
        ; print current index
        [{$}]

        ; generate random number 1 ~ 10
        [{RANDOM(1, 10)}]
:

; end of source
```

1부터 10까지 합구하기

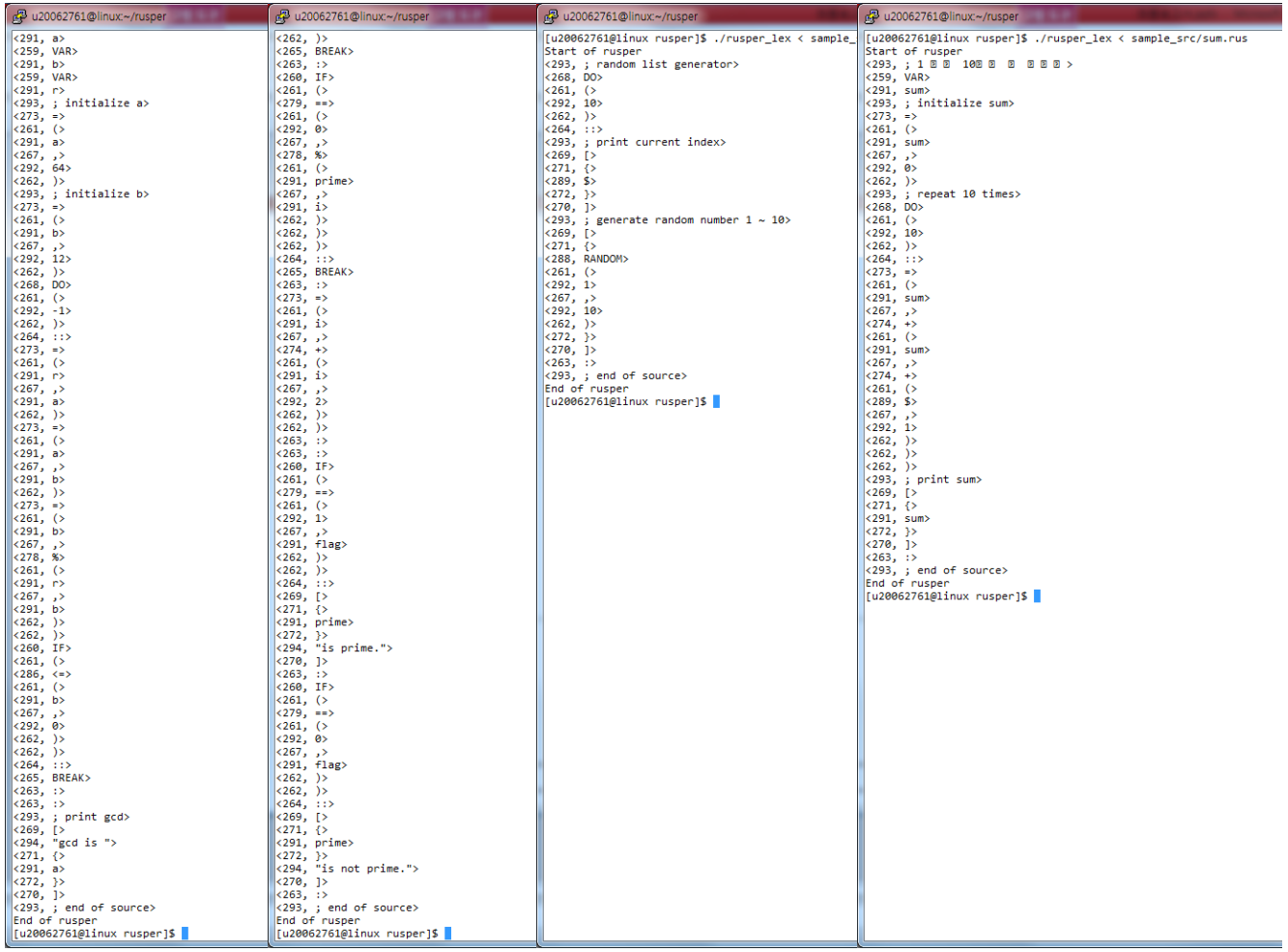
```
VAR sum

; initialize sum
=(sum, 0)

; repeat 10 times
DO(10)
::
        =(sum, +(sum, +($, 1)))

        ; print sum
        [{sum}]
:

; end of source
```

Lex 입력 파일 (주요 부분)

u20062761@linux:~/rusper

```
20
21 extern int cmp_stmt_cnt_stack[STACK_MAX];
22 extern int cmp_stmt_cnt_sp;
23 extern int cmp_stmt_cnt;
24 extern int print_stmt_cnt;
25 extern int sp;
26
27 %}
28 letter  [a-zA-Z_]
29 nonzero_digit   [1-9]
30 digit   [0-9]
31 %%
32 "VAR"           { yylval.str = yytext; return T_VAR; }
33 "IF"            { yylval.str = yytext; return T_IF; }
34 "("             { yylval.str = yytext;return T_LPAREN;}
35 ")"             { yylval.str = yytext;return T_RPAREN;}
36 "::"            { yylval.str = yytext; cmp_stmt_cnt_stack[cmp_stmt_cnt_sp++] = sp; return T_DCOLON;}
37 ":"             { yylval.str = yytext; cmp_stmt_cnt = sp - cmp_stmt_cnt_stack[--cmp_stmt_cnt_sp]; return T_COLON;}
38 "BREAK"         { yylval.str = yytext;return T_BREAK;}
39 "FOR"           { yylval.str = yytext;return T_FOR;}
40 ","             { yylval.str = yytext;return T_SEPARATOR;}
41 "DO"            { yylval.str = yytext;return T_DO;}
42 "["             { yylval.str = yytext; print_stmt_cnt = sp; return T_LBRACKET;}
43 "]"             { yylval.str = yytext; print_stmt_cnt = sp - print_stmt_cnt; return T_RBRACKET; }
44 "{"             { yylval.str = yytext;return T_LCURLYBRACE;}
45 "}"             { yylval.str = yytext;return T_RCURLYBRACE;}
46 "="             { yylval.str = yytext;return T_ASSIGNMENT;}
47 "+"             { yylval.str = yytext;return T_PLUS;}
48 "-"             { yylval.str = yytext;return T_MINUS;}
49 "*"             { yylval.str = yytext;return T_MUL;}
50 "/"             { yylval.str = yytext;return T_DIV;}
51 "%"             { yylval.str = yytext;return T_REMAINDER;}
52 "=="            { yylval.str = yytext;return T_EQUAL;}
53 "!="            { yylval.str = yytext;return T_NOT_EQUAL;}
54 "!"             { yylval.str = yytext;return T_NOT;}
55 "&&"            { yylval.str = yytext;return T_AND;}
56 "||"            { yylval.str = yytext;return T_OR;}
57 "<"             { yylval.str = yytext;return T_LT;}
58 ">"             { yylval.str = yytext;return T_GT;}
59 "<="            { yylval.str = yytext;return T_LT_EQUAL;}
60 ">="            { yylval.str = yytext;return T_GT_EQUAL;}
61 "RANDOM"        { yylval.str = yytext;return T_RANDOM;}
62 "$"             { yylval.str = yytext;return T_ITERATOR;}
63 "OUTPUT"        { yylval.str = yytext;return T_OUTPUT;}
64 ";".*           { yylval.str = yytext;return T_COMMENT;}
65 \".*\"          { yylval.str = yytext;return T_STRING_LITERAL;}
66
67 {letter}({letter}|{digit})*     { yylval.str = yytext; return T_IDENTIFIER; }
68 [-+]?{digit}+                   { yylval.str = yytext; return T_INTEGER; }
69
70 [ \t\n\r]       ;
71 .               return T_ERROR;
72
73 %%
```

어휘분석기 실행 결과



u20062761@linux:~/rusper

```
<291, a>
<259, VAR>
<291, b>
<259, VAR>
<291, r>
<293, ; initialize a>
<273, =>
<261, (>
<291, a>
<267, ,>
<292, 64>
<262, )>
<293, ; initialize b>
<273, =>
<261, (>
<291, b>
<267, ,>
<292, 12>
<262, )>
<268, DO>
<261, (>
<292, -1>
<262, )>
<264, ::>
<273, =>
<261, (>
<291, r>
<267, ,>
<291, a>
<262, )>
<273, =>
<261, (>
<291, a>
<267, ,>
<291, b>
<262, )>
<273, =>
<261, (>
<291, b>
<267, ,>
<278, %>
<261, (>
<291, r>
<267, ,>
<291, b>
<262, )>
<262, )>
<260, IF>
<261, (>
<286, <=>
<261, (>
<291, b>
<267, ,>
<292, 0>
<262, )>
<262, )>
<264, ::>
<265, BREAK>
<263, :>
<263, :>
<293, ; print gcd>
<269, [>
<294, "gcd is ">
<271, {>
<291, a>
<272, }>
<270, ]>
<293, ; end of source>
End of rusper
[u20062761@linux rusper]$
```

u20062761@linux:~/rusper

```
<262, )>
<265, BREAK>
<263, :>
<260, IF>
<261, (>
<279, ==>
<261, (>
<292, 0>
<267, ,>
<278, %>
<261, (>
<291, prime>
<267, ,>
<291, i>
<262, )>
<262, )>
<262, )>
<264, ::>
<265, BREAK>
<263, :>
<273, =>
<261, (>
<291, i>
<267, ,>
<274, +>
<261, (>
<291, i>
<267, ,>
<292, 2>
<262, )>
<262, )>
<263, :>
<263, :>
<260, IF>
<261, (>
<279, ==>
<261, (>
<292, 1>
<267, ,>
<291, flag>
<262, )>
<262, )>
<264, ::>
<269, [>
<271, {>
<291, prime>
<272, }>
<294, "is prime.">
<270, ]>
<263, :>
<260, IF>
<261, (>
<279, ==>
<261, (>
<292, 0>
<267, ,>
<291, flag>
<262, )>
<262, )>
<264, ::>
<269, [>
<271, {>
<291, prime>
<272, }>
<294, "is not prime.">
<270, ]>
<263, :>
<293, ; end of source>
End of rusper
[u20062761@linux rusper]$
```

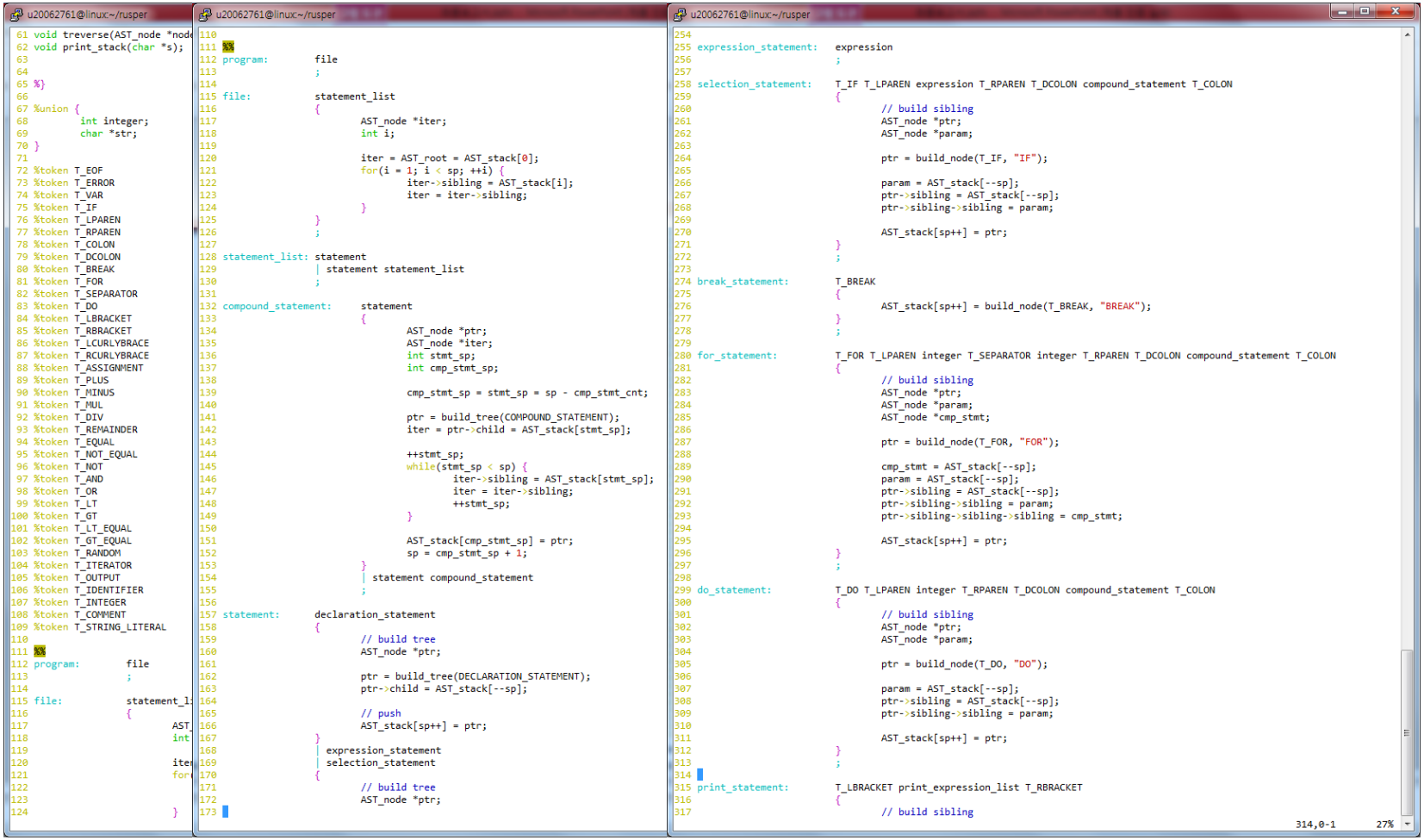
u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper_lex < sample_
Start of rusper
<293, ; random list generator>
<268, DO>
<261, (>
<292, 10>
<262, )>
<264, ::>
<293, ; print current index>
<269, [>
<271, {>
<289, $>
<272, }>
<270, ]>
<293, ; generate random number 1 ~ 10>
<269, [>
<271, {>
<288, RANDOM>
<261, (>
<292, 1>
<267, ,>
<292, 10>
<262, )>
<272, }>
<270, ]>
<263, :>
<293, ; end of source>
End of rusper
[u20062761@linux rusper]$
```

u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper_lex < sample_src/sum.rus
Start of rusper
<293, ; 1 □ □  10□ □  □  □ □ □ >
<259, VAR>
<291, sum>
<293, ; initialize sum>
<273, =>
<261, (>
<291, sum>
<267, ,>
<292, 0>
<262, )>
<293, ; repeat 10 times>
<268, DO>
<261, (>
<292, 10>
<262, )>
<264, ::>
<273, =>
<261, (>
<291, sum>
<267, ,>
<274, +>
<261, (>
<291, sum>
<267, ,>
<274, +>
<261, (>
<289, $>
<267, ,>
<292, 1>
<262, )>
<262, )>
<262, )>
<293, ; print sum>
<269, [>
<271, {>
<291, sum>
<272, }>
<270, ]>
<263, :>
<293, ; end of source>
End of rusper
[u20062761@linux rusper]$
```

Yacc 입력 파일 (주요부분)



u20062761@linux:~/rusper

```
61 void treverse(AST_node *node
62 void print_stack(char *s);
63
64
65 %}
66
67 %union {
68         int integer;
69         char *str;
70 }
71
72 %token T_EOF
73 %token T_ERROR
74 %token T_VAR
75 %token T_IF
76 %token T_LPAREN
77 %token T_RPAREN
78 %token T_COLON
79 %token T_DCOLON
80 %token T_BREAK
81 %token T_FOR
82 %token T_SEPARATOR
83 %token T_DO
84 %token T_LBRACKET
85 %token T_RBRACKET
86 %token T_LCURLYBRACE
87 %token T_RCURLYBRACE
88 %token T_ASSIGNMENT
89 %token T_PLUS
90 %token T_MINUS
91 %token T_MUL
92 %token T_DIV
93 %token T_REMAINDER
94 %token T_EQUAL
95 %token T_NOT_EQUAL
96 %token T_NOT
97 %token T_AND
98 %token T_OR
99 %token T_LT
100 %token T_GT
101 %token T_LT_EQUAL
102 %token T_GT_EQUAL
103 %token T_RANDOM
104 %token T_ITERATOR
105 %token T_OUTPUT
106 %token T_IDENTIFIER
107 %token T_INTEGER
108 %token T_COMMENT
109 %token T_STRING_LITERAL
110
111 %%
112 program:        file
113                 ;
114
115 file:           statement_l
116                 {
117                         AST
118                         int
119
120                         iter
121                         for
122
123
124                 }
```

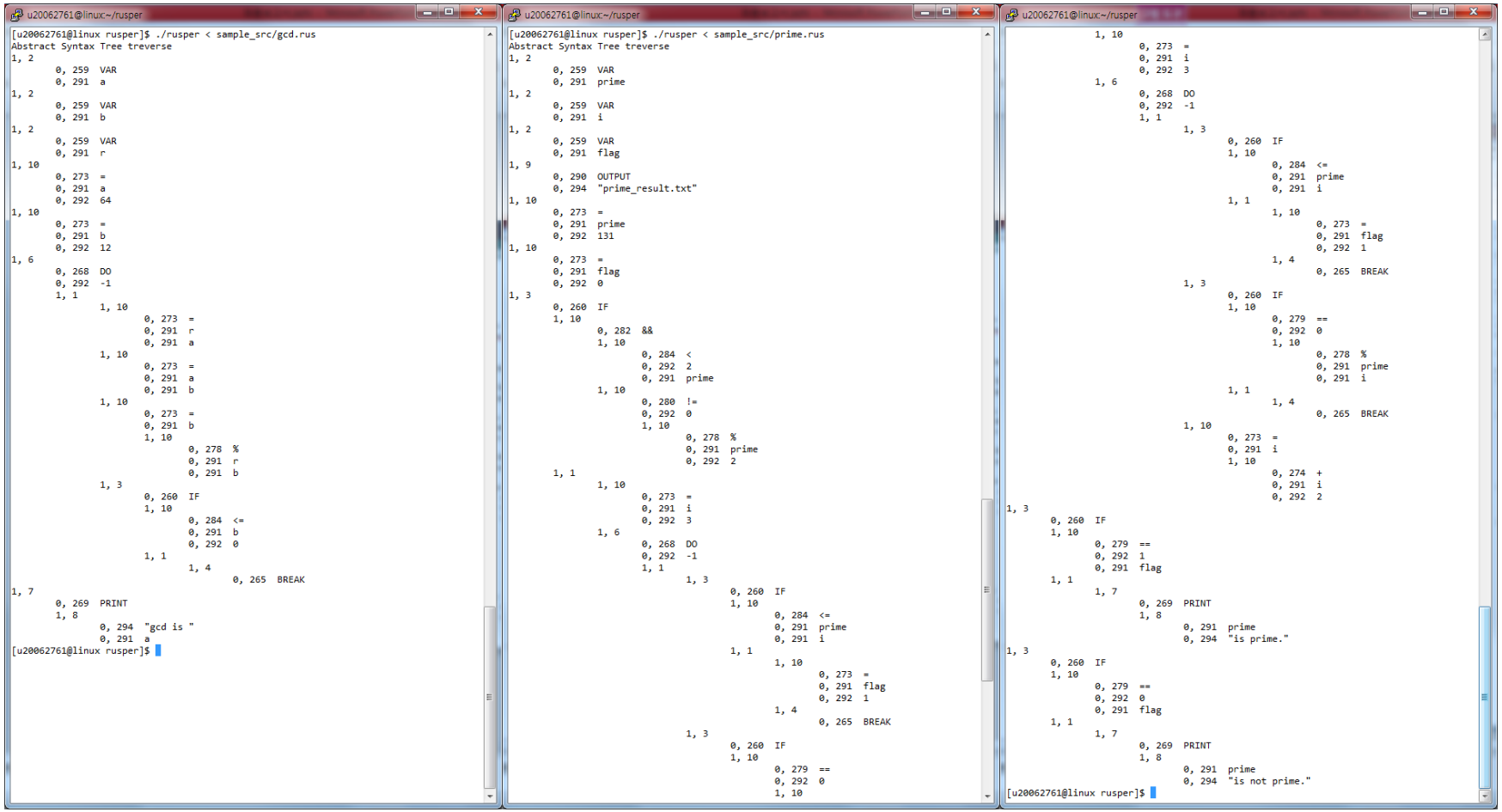
u20062761@linux:~/rusper

```
110
111 %%
112 program:        file
113                 ;
114
115 file:           statement_list
116                 {
117                         AST_node *iter;
118                         int i;
119
120                         iter = AST_root = AST_stack[0];
121                         for(i = 1; i < sp; ++i) {
122                                 iter->sibling = AST_stack[i];
123                                 iter = iter->sibling;
124                         }
125                 }
126                 ;
127
128 statement_list: statement
129                 | statement statement_list
130                 ;
131
132 compound_statement:     statement
133                         {
134                                 AST_node *ptr;
135                                 AST_node *iter;
136                                 int stmt_sp;
137                                 int cmp_stmt_sp;
138
139                                 cmp_stmt_sp = stmt_sp = sp - cmp_stmt_cnt;
140
141                                 ptr = build_tree(COMPOUND_STATEMENT);
142                                 iter = ptr->child = AST_stack[stmt_sp];
143
144                                 ++stmt_sp;
145                                 while(stmt_sp < sp) {
146                                         iter->sibling = AST_stack[stmt_sp];
147                                         iter = iter->sibling;
148                                         ++stmt_sp;
149                                 }
150
151                                 AST_stack[cmp_stmt_sp] = ptr;
152                                 sp = cmp_stmt_sp + 1;
153                         }
154                         | statement compound_statement
155                         ;
156
157 statement:      declaration_statement
158                 {
159                         // build tree
160                         AST_node *ptr;
161
162                         ptr = build_tree(DECLARATION_STATEMENT);
163                         ptr->child = AST_stack[--sp];
164
165                         // push
166                         AST_stack[sp++] = ptr;
167                 }
168                 | expression_statement
169                 | selection_statement
170                 {
171                         // build tree
172                         AST_node *ptr;
173
```

u20062761@linux:~/rusper

```
254
255 expression_statement:   expression
256                         ;
257
258 selection_statement:    T_IF T_LPAREN expression T_RPAREN T_DCOLON compound_statement T_COLON
259                         {
260                                 // build sibling
261                                 AST_node *ptr;
262                                 AST_node *param;
263
264                                 ptr = build_node(T_IF, "IF");
265
266                                 param = AST_stack[--sp];
267                                 ptr->sibling = AST_stack[--sp];
268                                 ptr->sibling->sibling = param;
269
270                                 AST_stack[sp++] = ptr;
271                         }
272                         ;
273
274 break_statement:        T_BREAK
275                         {
276                                 AST_stack[sp++] = build_node(T_BREAK, "BREAK");
277                         }
278                         ;
279
280 for_statement:          T_FOR T_LPAREN integer T_SEPARATOR integer T_RPAREN T_DCOLON compound_statement T_COLON
281                         {
282                                 // build sibling
283                                 AST_node *ptr;
284                                 AST_node *param;
285                                 AST_node *cmp_stmt;
286
287                                 ptr = build_node(T_FOR, "FOR");
288
289                                 cmp_stmt = AST_stack[--sp];
290                                 param = AST_stack[--sp];
291                                 ptr->sibling = AST_stack[--sp];
292                                 ptr->sibling->sibling = param;
293                                 ptr->sibling->sibling->sibling = cmp_stmt;
294
295                                 AST_stack[sp++] = ptr;
296                         }
297                         ;
298
299 do_statement:           T_DO T_LPAREN integer T_RPAREN T_DCOLON compound_statement T_COLON
300                         {
301                                 // build sibling
302                                 AST_node *ptr;
303                                 AST_node *param;
304
305                                 ptr = build_node(T_DO, "DO");
306
307                                 param = AST_stack[--sp];
308                                 ptr->sibling = AST_stack[--sp];
309                                 ptr->sibling->sibling = param;
310
311                                 AST_stack[sp++] = ptr;
312                         }
313                         ;
314
315 print_statement:        T_LBRACKET print_expression_list T_RBRACKET
316                         {
317                                 // build sibling
                                                        314,0-1       27%
```

파서 실행 결과 및 파스트리(1/2)



u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/gcd.rus
Abstract Syntax Tree treverse
1, 2
        0, 259  VAR
        0, 291  a
1, 2
        0, 259  VAR
        0, 291  b
1, 2
        0, 259  VAR
        0, 291  r
1, 10
        0, 273  =
        0, 291  a
        0, 292  64
1, 10
        0, 273  =
        0, 291  b
        0, 292  12
1, 6
        0, 268  DO
        0, 292  -1
        1, 1
                1, 10
                        0, 273  =
                        0, 291  r
                        0, 291  a
                1, 10
                        0, 273  =
                        0, 291  a
                        0, 291  b
                1, 10
                        0, 273  =
                        0, 291  b
                        1, 10
                                0, 278  %
                                0, 291  r
                                0, 291  b
                1, 3
                        0, 260  IF
                        1, 10
                                0, 284  <=
                                0, 291  b
                                0, 292  0
                        1, 1
                                1, 4
                                        0, 265  BREAK
1, 7
        0, 269  PRINT
        1, 8
                0, 294  "gcd is "
                0, 291  a
[u20062761@linux rusper]$
```

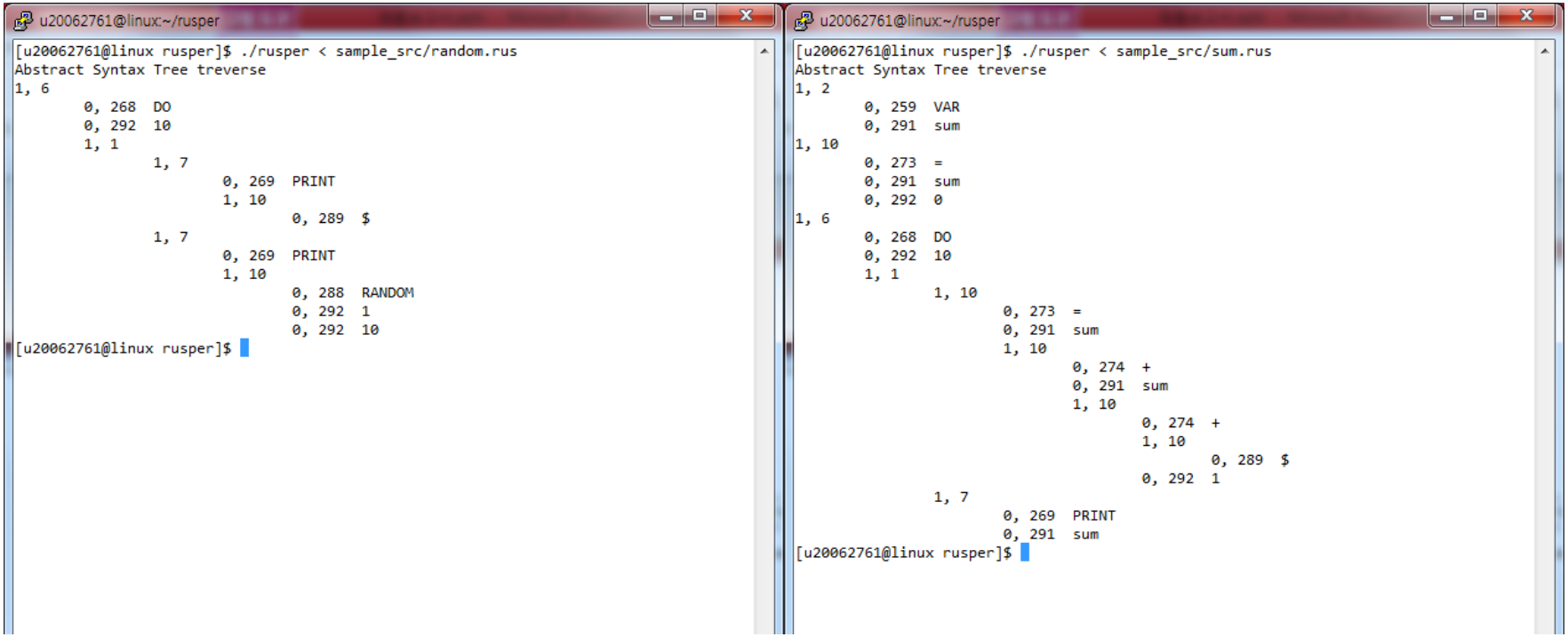
u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/prime.rus
Abstract Syntax Tree treverse
1, 2
        0, 259  VAR
        0, 291  prime
1, 2
        0, 259  VAR
        0, 291  i
1, 2
        0, 259  VAR
        0, 291  flag
1, 9
        0, 290  OUTPUT
        0, 294  "prime_result.txt"
1, 10
        0, 273  =
        0, 291  prime
        0, 292  131
1, 10
        0, 273  =
        0, 291  flag
        0, 292  0
1, 3
        0, 260  IF
        1, 10
                0, 282  &&
                1, 10
                        0, 284  <
                        0, 292  2
                        0, 291  prime
                1, 10
                        0, 280  !=
                        0, 292  0
                        1, 10
                                0, 278  %
                                0, 291  prime
                                0, 292  2
        1, 1
                1, 10
                        0, 273  =
                        0, 291  i
                        0, 292  3
                1, 6
                        0, 268  DO
                        0, 292  -1
                        1, 1
                                1, 3
                                        0, 260  IF
                                        1, 10
                                                0, 284  <=
                                                0, 291  prime
                                                0, 291  i
                                        1, 1
                                                1, 10
                                                        0, 273  =
                                                        0, 291  flag
                                                        0, 292  1
                                                1, 4
                                                        0, 265  BREAK
                                1, 3
                                        0, 260  IF
                                        1, 10
                                                0, 279  ==
                                                0, 292  0
                                                1, 10
```

u20062761@linux:~/rusper

```
                1, 10
                        0, 273  =
                        0, 291  i
                        0, 292  3
                1, 6
                        0, 268  DO
                        0, 292  -1
                        1, 1
                                1, 3
                                        0, 260  IF
                                        1, 10
                                                0, 284  <=
                                                0, 291  prime
                                                0, 291  i
                                        1, 1
                                                1, 10
                                                        0, 273  =
                                                        0, 291  flag
                                                        0, 292  1
                                                1, 4
                                                        0, 265  BREAK
                                1, 3
                                        0, 260  IF
                                        1, 10
                                                0, 279  ==
                                                0, 292  0
                                                1, 10
                                                        0, 278  %
                                                        0, 291  prime
                                                        0, 291  i
                                        1, 1
                                                1, 4
                                                        0, 265  BREAK
                                1, 10
                                        0, 273  =
                                        0, 291  i
                                        1, 10
                                                0, 274  +
                                                0, 291  i
                                                0, 292  2
1, 3
        0, 260  IF
        1, 10
                0, 279  ==
                0, 292  1
                0, 291  flag
        1, 1
                1, 7
                        0, 269  PRINT
                        1, 8
                                0, 291  prime
                                0, 294  "is prime."
1, 3
        0, 260  IF
        1, 10
                0, 279  ==
                0, 292  0
                0, 291  flag
        1, 1
                1, 7
                        0, 269  PRINT
                        1, 8
                                0, 291  prime
                                0, 294  "is not prime."
[u20062761@linux rusper]$
```

파서 실행 결과 및 파스트리(2/2)



```
u20062761@linux:~/rusper
[u20062761@linux rusper]$ ./rusper < sample_src/random.rus
Abstract Syntax Tree treverse
1, 6
        0, 268  DO
        0, 292  10
        1, 1
                1, 7
                        0, 269  PRINT
                        1, 10
                                0, 289  $
                1, 7
                        0, 269  PRINT
                        1, 10
                                0, 288  RANDOM
                                0, 292  1
                                0, 292  10
[u20062761@linux rusper]$
```

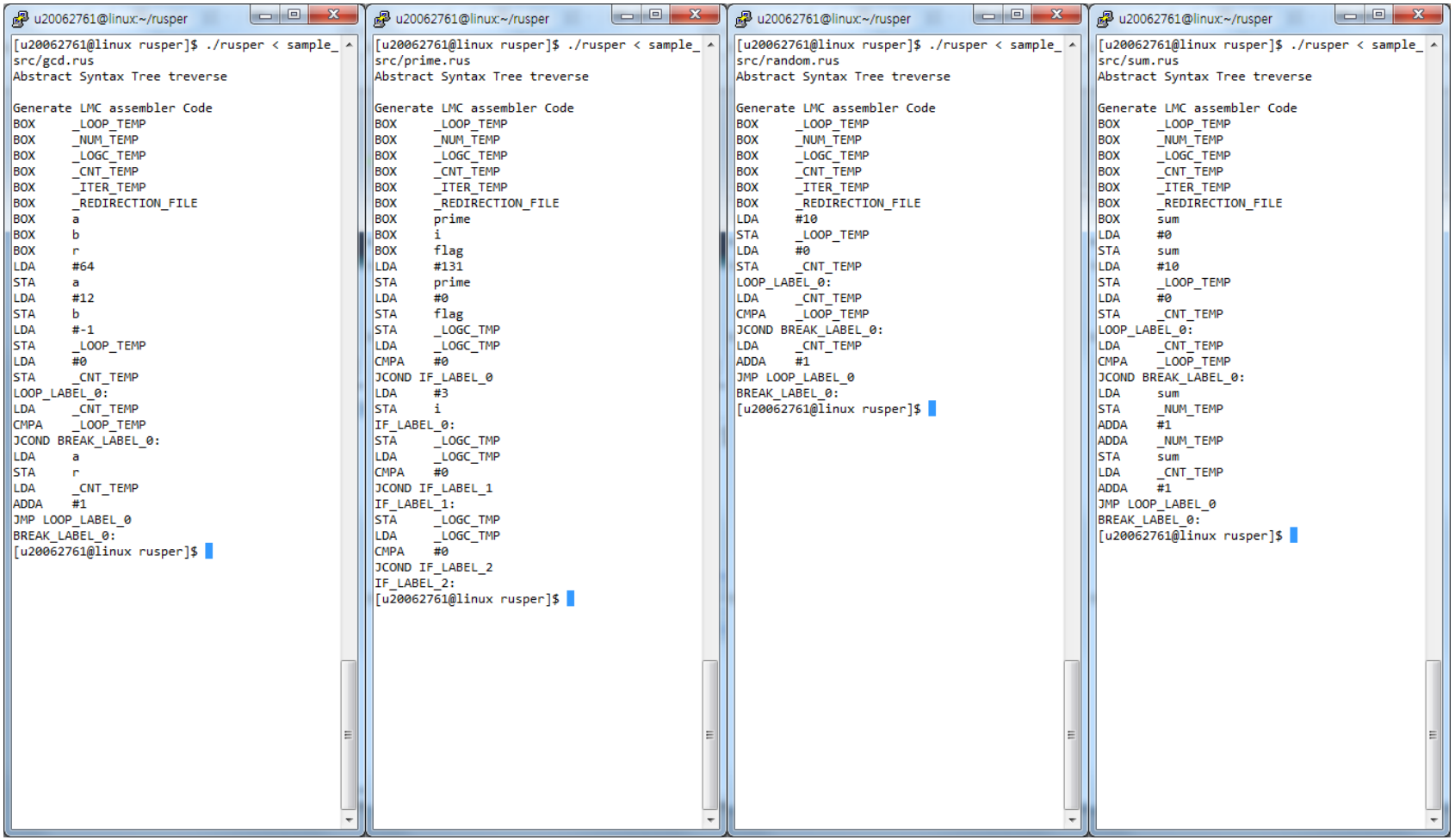
```
u20062761@linux:~/rusper
[u20062761@linux rusper]$ ./rusper < sample_src/sum.rus
Abstract Syntax Tree treverse
1, 2
        0, 259  VAR
        0, 291  sum
1, 10
        0, 273  =
        0, 291  sum
        0, 292  0
1, 6
        0, 268  DO
        0, 292  10
        1, 1
                1, 10
                        0, 273  =
                        0, 291  sum
                        1, 10
                                0, 274  +
                                0, 291  sum
                                1, 10
                                        0, 274  +
                                        1, 10
                                                0, 289  $
                                        0, 292  1
                1, 7
                        0, 269  PRINT
                        0, 291  sum
[u20062761@linux rusper]$
```

목적코드 관련 소스

```
749 void generate_lmc_asm_code(FILE *fp, AST_node *root)
750 {
751         AST_node *iter = root;
752 
753         fprintf(fp, "BOX\t_LOOP_TEMP\n");
754         fprintf(fp, "BOX\t_NUM_TEMP\n");
755         fprintf(fp, "BOX\t_LOGC_TEMP\n");
756         fprintf(fp, "BOX\t_CNT_TEMP\n");
757         fprintf(fp, "BOX\t_ITER_TEMP\n");
758         fprintf(fp, "BOX\t_REDIRECTION_FILE\n");
759 
760         while(NULL != iter) {
761 
762                 switch(iter->token_number) {
763                         case DECLARATION_STATEMENT:
764                                 fprintf(fp, "BOX\t%s\n", iter->child->sibling->token_value);
765                                 break;
766                         case SELECTION_STATEMENT:
767                                 generate_lmc_asm_code_stmt_expr(fp, iter->child);
768                                 break;
769                         case BREAK_STATEMENT:
770                                 fprintf(fp, "JMP LOOP_LABEL_%d\n", loop_label_count);
771                                 break;
772                         case FOR_STATEMENT:
773                                 break;
774                         case DO_STATEMENT:
775                                 generate_lmc_asm_code_stmt_expr(fp, iter->child);
776                                 break;
777                         case PRINT_STATEMENT:
778                                 break;
779                         case OUTPUT_STATEMENT:
780                                 break;
781                         case EXPRESSION:
782                                 generate_lmc_asm_code_stmt_expr(fp, iter->child);
783                                 break;
784                         default:
785                                 fprintf(stderr, "ERROR\n");
786                                 exit(255);
787                                 break;
788                 }
789 
790                 iter = iter->sibling;
791         }
792 }
```

```
794 void generate_lmc_asm_code_stmt_expr(FILE *fp, AST_node *node)
795 {
796         AST_node *lhs;
797         AST_node *rhs;
798 
799         if(NULL == node)
800                 return;
801         if(NULL == node->sibling)
802                 return;
803 
804         lhs = node->sibling;
805         rhs = node->sibling->sibling;
806 
807         switch(node->token_number) {
808                 case T_DO:
809                         fprintf(fp, "LDA\t%s%s\n", T_INTEGER == lhs->token_number ? "#" : "", lhs-
810                         fprintf(fp, "STA\t_LOOP_TEMP\n");
811                         fprintf(fp, "LDA\t#0\n");
812                         fprintf(fp, "STA\t_CNT_TEMP\n");
813                         fprintf(fp, "LOOP_LABEL_%d:\n", loop_label_count);
814                         fprintf(fp, "LDA\t_CNT_TEMP\n");
815                         fprintf(fp, "CMPA\t_LOOP_TEMP\n");
816                         fprintf(fp, "JCOND BREAK_LABEL_%d:\n", loop_label_count);
817 
818                         // loop for compound statement
819                         if(NON_TERMINAL == rhs->node_type) {
820                                 if(NON_TERMINAL == rhs->child->node_type)
821                                         generate_lmc_asm_code_stmt_expr(fp, rhs->child->child);
822                                 else
823                                         generate_lmc_asm_code_stmt_expr(fp, rhs->child);
824                         }
825 
826                         fprintf(fp, "LDA\t_CNT_TEMP\n");
827                         fprintf(fp, "ADDA\t#1\n");
828                         fprintf(fp, "JMP LOOP_LABEL_%d\n", loop_label_count);
829                         fprintf(fp, "BREAK_LABEL_%d:\n", loop_label_count);
830                         ++loop_label_count;
831                         break;
832                 case T_ASSIGNMENT:
833 
834                         if(NON_TERMINAL == rhs->node_type) {
835                                 generate_lmc_asm_code_stmt_expr(fp, rhs->child);
836                         } else {
837                                 fprintf(fp, "LDA\t%s%s\n", T_INTEGER == rhs->token_number ? "#" :
838                         }
839 
```

목적코드 생성 실행 결과



u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/gcd.rus
Abstract Syntax Tree treverse

Generate LMC assembler Code
BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
BOX     a
BOX     b
BOX     r
LDA     #64
STA     a
LDA     #12
STA     b
LDA     #-1
STA     _LOOP_TEMP
LDA     #0
STA     _CNT_TEMP
LOOP_LABEL_0:
LDA     _CNT_TEMP
CMPA    _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA     a
STA     r
LDA     _CNT_TEMP
ADDA    #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:
[u20062761@linux rusper]$
```

u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/prime.rus
Abstract Syntax Tree treverse

Generate LMC assembler Code
BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
BOX     prime
BOX     i
BOX     flag
LDA     #131
STA     prime
LDA     #0
STA     flag
STA     _LOGC_TMP
LDA     _LOGC_TMP
CMPA    #0
JCOND IF_LABEL_0
LDA     #3
STA     i
IF_LABEL_0:
STA     _LOGC_TMP
LDA     _LOGC_TMP
CMPA    #0
JCOND IF_LABEL_1
IF_LABEL_1:
STA     _LOGC_TMP
LDA     _LOGC_TMP
CMPA    #0
JCOND IF_LABEL_2
IF_LABEL_2:
[u20062761@linux rusper]$
```

u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/random.rus
Abstract Syntax Tree treverse

Generate LMC assembler Code
BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
LDA     #10
STA     _LOOP_TEMP
LDA     #0
STA     _CNT_TEMP
LOOP_LABEL_0:
LDA     _CNT_TEMP
CMPA    _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA     _CNT_TEMP
ADDA    #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:
[u20062761@linux rusper]$
```

u20062761@linux:~/rusper

```
[u20062761@linux rusper]$ ./rusper < sample_src/sum.rus
Abstract Syntax Tree treverse

Generate LMC assembler Code
BOX     _LOOP_TEMP
BOX     _NUM_TEMP
BOX     _LOGC_TEMP
BOX     _CNT_TEMP
BOX     _ITER_TEMP
BOX     _REDIRECTION_FILE
BOX     sum
LDA     #0
STA     sum
LDA     #10
STA     _LOOP_TEMP
LDA     #0
STA     _CNT_TEMP
LOOP_LABEL_0:
LDA     _CNT_TEMP
CMPA    _LOOP_TEMP
JCOND BREAK_LABEL_0:
LDA     sum
STA     _NUM_TEMP
ADDA    #1
ADDA    _NUM_TEMP
STA     sum
LDA     _CNT_TEMP
ADDA    #1
JMP LOOP_LABEL_0
BREAK_LABEL_0:
[u20062761@linux rusper]$
```

감사합니다