MiniC 컴파일러 만들기(2): Syntax Analysis

컴파일러, 8번째 시간

경민기, 2025-11-04

컴파일러 구조

- 소스코드 → 어휘 분석 → 구문 분석 → 의미 분석 → 코드 생성
- 이번 주: 구문 분석(Syntax Analysis)

C Lexer와 Parser 통합

- C 언어의 간소화된 문법을 해석하는 Parser(Bison)과 Lexer(Flex) 사용
-

예제 1: MiniC 파서 구성

구성 파일

- lexer.l - 어휘 분석기 (Flex)
- parser.y - 구문 분석기 (Bison)
- ast.h, ast.c - AST 구조체 및 출력 함수
- main.c - 파서 실행 및 AST 출력
- Makefile - 전체 빌드 자동화 스크립트

lexer.l

- 역할: 코드 → 토큰 분해 후 yylval에 AST 노드 저장
- 주요 토큰: 키워드, 식별자, 상수, 연산자
- 주석 처리: //, /* ... */ COMMENT 상태로 처리

parser.y

- Lexer로부터 토큰 스트림을 받아 문법 구조 해석
- 주요 규칙:
  - translation_unit, function_definition, declaration
  - statement (if, while, return 등)
  - expression (연산자 우선순위 포함)

예제 코드

test.mc

8wk > minic > test.mc

```
1   int sum(int a, int b) {
2     int s = a + b;
3     return s;
4   }
5
6   int main() {
7     int x = 10, y = 20;
8     while (x < y) {
9       x = x + 1;
10    }
11    if (x == y) return sum(x, y);
12    return 0;
13  }
```

```
mingi_kyung@mg-tpx240:~/compiler_class/8wk/minic$ ./cparser < test.mc
=== AST ===
(TRANSLATION_UNIT) @13
  (FUNC_DEF : sum) @4
    (TYPE : int) @1
    (PARAM_LIST) @1
      (PARAM : a) @1
        (TYPE : int) @1
      (PARAM : b) @1
        (TYPE : int) @1
    (COMPOUND) @4
      (STMT_LIST) @3
        (DECL) @2
          (TYPE : int) @2
          (INIT_DECL_LIST) @2
            (INIT_DECL : s) @2
              (BINOP : +) @2
                (VAR : a) @2
                (VAR : b) @2
        (RETURN) @3
          (VAR : s) @3
  (FUNC_DEF : main) @13
    (TYPE : int) @6
    (PARAM_LIST) @6
    (COMPOUND) @13
      (STMT_LIST) @12
        (DECL) @7
          (TYPE : int) @7
          (INIT_DECL_LIST) @7
            (INIT_DECL : x) @7
              (INT : 10) @7
            (INIT_DECL : y) @7
              (INT : 20) @7
        (WHILE) @10
          (BINOP : <) @8
            (VAR : x) @8
            (VAR : y) @8
          (COMPOUND) @10
            (STMT_LIST) @9
              (EXPR_STMT) @9
                (ASSIGN : x) @9
                  (BINOP : +) @9
                    (VAR : x) @9
                    (INT : 1) @9
        (IF) @12
          (BINOP : ==) @11
            (VAR : x) @11
            (VAR : y) @11
          (RETURN) @11
            (CALL : sum) @11
              (ARG_LIST) @11
                (VAR : x) @11
                (VAR : y) @11
        (RETURN) @12
          (INT : 0) @12
```

예제 2: MiniC 파서에 for문 추가

lexer.l

```
mingi_kyung@mg-tpx240:~/compiler_class/8wk$ diff minic/lexer.l  minic_for/lexer.l
34a35
> "for"                           { return_FOR; }
```

parser.y

```
mingi_kyung@mg-tpx240:~/compiler_class/8wk$ diff minic/parser.y  minic_for/parser.y
33a34
> %token FOR
215a217
>   | for_stmt
316a319,350
>   ;
>
> /* 세미콜론 없는 간단 선언: for-init에서만 사용 */
> simple_decl
>   : type_spec init_declarator_list
>     { $$ = node(NK_DECL, yylineno, NULL, 2, $1, $2); }
>   ;
>
> /* 선택적 표현식: 비어있으면 NULL */
> expr_opt
>   : /* empty */ { $$ = NULL; }
>   | expr        { $$ = $1; }
>   ;
>
> /* for 의 세 파트 */
> for_init_opt
>   : simple_decl         /* 선언형 초기식 */
>   | expr_opt            /* 표현식 또는 비어있음 */
>   ;
>
> for_cond_opt
>   : expr_opt
>   ;
>
> for_step_opt
>   : expr_opt
>   ;
>
> /* for 문 본체 */
> for_stmt
>   : FOR '(' for_init_opt ';' for_cond_opt ';' for_step_opt ')' statement
>     { $$ = node(NK_FOR, yylineno, NULL, 4, $3, $5, $7, $9); }
```

ast.c

```
mingi_kyung@mg-tpx240:~/compiler_class/8wk$ diff minic/ast.c  minic_for/ast.c
106a107
>     case NK_FOR: return "FOR";
```

test2.mc

test2.mc U

8wk > minic_for > test2.mc

```
1   int main() {
2     int i = 0, sum = 0;
3     for (int k = 0; k < 3; k = k + 1)
4       sum = sum + k;
5
6     for (i = 0; i < 5; i = i + 1) {
7       sum = sum + i;
8     }
9
10    for (;;)
11      break;
12    return sum;
13  }
```

실행

```
mingi_kyung@mg-tpx240:~/compiler_class/8wk/minic_for$ ./cparser < test2.mc
=== AST ===
(TRANSLATION_UNIT) @13
  (FUNC_DEF : main) @13
    (TYPE : int) @1
    (PARAM_LIST) @1
    (COMPOUND) @13
      (STMT_LIST) @12
        (DECL) @2
          (TYPE : int) @2
          (INIT_DECL_LIST) @2
            (INIT_DECL : i) @2
              (INT : 0) @2
            (INIT_DECL : sum) @2
              (INT : 0) @2
        (FOR) @4
          (DECL) @3
            (TYPE : int) @3
            (INIT_DECL_LIST) @3
              (INIT_DECL : k) @3
                (INT : 0) @3
          (BINOP : <) @3
            (VAR : k) @3
            (INT : 3) @3
          (ASSIGN : k) @3
            (BINOP : +) @3
              (VAR : k) @3
              (INT : 1) @3
          (EXPR_STMT) @4
            (ASSIGN : sum) @4
              (BINOP : +) @4
                (VAR : sum) @4
                (VAR : k) @4
```

다음 단계:
심볼 테이블, 타입 검사, IR 생성