[1]

Principles of programming languages
Assignment 1

[Internal Document]

20203361 장민석

[1] 표지이미지는 Dall-E 3에 의해 생성됨

1-1. 구현 세부 사항 결정: WARNING 처리

이 프로그램은 코드의 일부분이 누락되거나 부적절하게 작성되었을 때 경고 메시지를 통해 사용자에게 알리고, 가능한 경우 자동으로 코드를 수정하여 의미 있는 문장으로 변환할 수 있다. 다음은 10가지 유형의 경고와 그에 따른 수정 사항이다.

1. 중복 연산자 (경고_0): 하나 이상의 연산자가 연속적으로 사용된 경우이다. 예를 들어, "operand2 := operand1 + + - 2;"에서 '+' 연산자가 중복된다. 이 때 첫 번째 연산자를 제외한 나머지를 제거하여 문장을 정정하고 정상화한다.

```
operand1 := 3;
operand2 := operand1 + + -/ 2;
target := operand1 + operand2 * 3
```

```
operand1 := 3;
ID: 1; CONST: 1; OP: 0;
(OK)
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Warning) 연산자(+) 중복
(Warning) 연산자(-) 중복
(Warning) 연산자(/) 중복
target := operand1 + operand2 * 3;
ID: 3; CONST: 1; OP: 2;
(OK)
Result ⟹ target: 18; operand2: 5; operand1: 3;
```

2. 닫는 괄호 누락 (경고_1): 열린 괄호에 비해 닫는 괄호가 부족한 경우이다. 해당 문장의 마지막에 도달했을 때, 부족한 닫는 괄호의 수에 맞게 적절하게 닫는 괄호를 추가하여 괄호의 쌍을 완성하여 정상화한다.

```
operand1 := (3 * (3 - 1) * (5 - 3;
target := operand1 * ((1) - (((2
```

```
operand1 := ( 3 * ( 3 - 1 ) * ( 5 - 3 ) );
ID: 1; CONST: 5; OP: 4;
(Warning) 닫는 괄호')' 누락
(Warning) 닫는 괄호')' 누락
target := operand1 * ( ( 1 ) - ( ( ( 2 ) ) ) );
ID: 2; CONST: 2; OP: 2;
(Warning) 닫는 괄호')' 누락
(Warning) 닫는 괄호')' 누락
(Warning) 닫는 괄호')' 누락
(Warning) 닫는 괄호')' 누락
Result ⟹ target: -12; operand1: 12;
```

3. 불필요한 <factor> 요소 (경고_2): <factor> 구문 뒤에 예상치 못한 요소가 나타난 경우이다. 이 불필요한 요소들을 모두 제거하여 문장을 정상화한다.

```
operand1 := 3 21 3 + (2 321 * 2) 132 operand1;
operand2 := operand1 operand1 operand2+ operand1 123 operand2 operand1;
target := operand1 1 2
```

```
operand1 := 3 + ( 2 * 2 );
ID: 1; CONST: 3; OP: 2;
(Warning) <factor> 뒤 불필요한 요소(21) 등장
(Warning) <factor> 뒤 불필요한 요소(3) 등장
(Warning) <factor> 뒤 불필요한 요소(321) 등장
(Warning) <factor> 뒤 불필요한 요소(132) 등장
(Warning) <factor> 뒤 불필요한 요소(operand1) 등장
operand2 := operand1 + operand1;
ID: 3; CONST: 0; OP: 1;
(Warning) <factor> 뒤 불필요한 요소(operand1) 등장
(Warning) <factor> 뒤 불필요한 요소(operand2) 등장
(Warning) <factor> 뒤 불필요한 요소(123) 등장
(Warning) <factor> 뒤 불필요한 요소(operand2) 등장
(Warning) <factor> 뒤 불필요한 요소(operand1) 등장
target := operand1;
ID: 2; CONST: 0; OP: 0;
(Warning) <factor> 뒤 불필요한 요소(1) 등장
(Warning) <factor> 뒤 불필요한 요소(2) 등장
Result ⟹ target: 7; operand2: 14; operand1: 7;
```

4. 대입 연산자 누락 (경고_3): 대입 연산자가 없는 경우에는 필요한 위치에 추가하여 문장을 정상화한다.

```
operand1   3;
operand2 := operand1 + 2;
target   operand1 + operand2 * 3
```

```
operand1 := 3;
ID: 1; CONST: 1; OP: 0;
(Warning) 대입 연산자(:=) 누락
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(OK)
target := operand1 + operand2 * 3;
ID: 3; CONST: 1; OP: 2;
(Warning) 대입 연산자(:=) 누락
Result ⟹ target: 18; operand2: 5; operand1: 3;
```

5. 정의되지 않은 토큰 (경고_4): 문장 내에 정의될 수 없는 문자열이 발견되어 의미 있는 토큰으로 변환 할 수 없는 상황이 발견되면, 해당 토큰을 제거하고 문장을 정상화한다.

```
operand1 := &%3 ^^;$#
operand2 := operand1 &$ + &$2
```

```
operand1 := 3;
ID: 1; CONST: 1; OP: 0;
(Warning) 인식할 수 없는 UNDEFINED 토큰(&) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(%) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(^) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(^) 등장
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Warning) 인식할 수 없는 UNDEFINED 토큰($) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(#) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(&) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰($) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰(&) 등장
(Warning) 인식할 수 없는 UNDEFINED 토큰($) 등장
Result ⟹ operand2: 5; operand1: 3;
```

6. 피연산자 부족 (경고_5): 연산자에 짝이 맞도록 필요한 피연산자가 없는 경우, 누락된 위치에 정의되지 않은 가상의 값을 추가하여 문장을 정상화한다.

```
operand1 := 3 - ;
operand2 := * 2 + (1 - 2);
target :=
```

```
operand1 := 3 - [Unknown];
ID: 1; CONST: 1; OP: 1;
(Warning) 피연산자(상수 or 변수) 누락
operand2 := [Unknown] * 2 + ( 1 - 2 );
ID: 1; CONST: 3; OP: 3;
(Warning) 피연산자(상수 or 변수) 누락
target := [Unknown];
ID: 1; CONST: 0; OP: 0;
(Warning) 피연산자(상수 or 변수) 누락
Result ⟹ target: Unknown; operand2: Unknown; operand1: Unknown;
```

7. 숫자가 아닌 문자 포함 (경고_6): 숫자로만 이루어져야 하는 상수에 숫자 외의 다른 문자가 포함된 경우, 해당 문자를 제거하여 정상적인 숫자가 될 수 있도록 복구한다.

```
operand1 := 1%1^1@1#;
operand2 := operand1 + 2!!@#
```

```
operand1 := 1111;
ID: 1; CONST: 1; OP: 0;
(Warning) CONST 내부에 숫자 외의 문자 포함(1%1^1@1#)
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Warning) CONST 내부에 숫자 외의 문자 포함(2!!@#)
Result ⟹ operand2: 1113; operand1: 1111;
```

8. 변수명에 부적절한 문자 포함 (경고_7): 변수명에 허용되지 않는 문자가 포함된 경우, 이러한 문자들을 모두 제거하여 규칙에 맞는 변수명이 될 수 있도록 복구한다.

```
op%^er$@and1 := 3;
op@era#n%d2 := op^^er^^and1 + 2;
tar^^$#%get := o^pe^rand^1 + operand2^
```

```
operand1 := 3;
ID: 1; CONST: 1; OP: 0;
(Warning) IDENT 내부에 규칙 외의 문자 포함(op%^er$@and1)
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Warning) IDENT 내부에 규칙 외의 문자 포함(op@era#n%d2)
(Warning) IDENT 내부에 규칙 외의 문자 포함(op^^er^^and1)
target := operand1 + operand2;
ID: 3; CONST: 0; OP: 1;
(Warning) IDENT 내부에 규칙 외의 문자 포함(tar^^$#%get)
(Warning) IDENT 내부에 규칙 외의 문자 포함(o^pe^rand^1)
(Warning) IDENT 내부에 규칙 외의 문자 포함(operand2^)
Result ⟹ target: 8; operand2: 5; operand1: 3;
```

9. 여는 괄호 누락 (경고_8): 여는 괄호가 닫는 괄호보다 적을 경우, 여는 괄호를 적절히 넣을 수 없으므로 현재 등장한 짝이 맞지 않는 닫는 괄호를 모두 제거하여 문장을 정상화한다.

```
operand1 := 2 + 3) * 2;
operand2 := operand1) ))+ ((2))))
```

```
operand1 := 2 + 3 * 2;
ID: 1; CONST: 3; OP: 2;
(Warning) 여는 괄호'(' 누락
operand2 := operand1 + ( ( 2 ) );
ID: 2; CONST: 1; OP: 1;
(Warning) 여는 괄호'(' 누락
(Warning) 여는 괄호'(' 누락
(Warning) 여는 괄호'(' 누락
(Warning) 여는 괄호'(' 누락
(Warning) 여는 괄호'(' 누락
Result ⟹ operand2: 10; operand1: 8;
```

10. 대입 연산자 중복 (경고_9): 한 문장 내에서 대입 연산자가 여러 번 사용된 경우, BNF 상으로 올바른 위치에 있는 하나의 대입 연산자만을 남기고 나머지는 모두 제거한다.

```
operand1 := 3 := := := + 1 := := - 3;
operand2 := := := operand1 := + 2;
target := operand1 + operand2 * 3 :=
```

```
operand1 := 3 + 1 - 3;
ID: 1; CONST: 3; OP: 2;
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
(Warning) 대입연산자 중복(:=)
target := operand1 + operand2 * 3;
ID: 3; CONST: 1; OP: 2;
(Warning) 대입연산자 중복(:=)
Result ⟹ target: 10; operand2: 3; operand1: 1;
```

1-2. 구현 세부 사항 결정: ERROR 처리

이 프로그램은 특정 문장을 최선으로 복구해도 유의미한 문장을 구성할 수 없는 경우 ERROR로 판정하였다. 다음은 3가지 유형의 경고와 그에 따른 수정 사항이다.

1. 프로그램 내에서 선언되지 않은 변수가 사용될 경우 (오류 유형 0): 실행 중인 프로그램에서 미리 선언하지 않은 변수가 사용되면, 이는 오류로 간주된다.

```
operand2 := operand1 + 2;
target := operand3 + operand2 * 3
```

```
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Error) 정의되지 않은 변수(operand1)가 참조됨
target := operand3 + operand2 * 3;
ID: 3; CONST: 1; OP: 2;
(Error) 정의되지 않은 변수(operand3)가 참조됨
(Error) 정의되지 않은 변수(operand2)가 참조됨
Result ⟹ operand3: Unknown; target: Unknown; operand2: Unknown; operand1: Unknown;
```

2. 대입 연산의 왼쪽에 변수가 없는 경우 (오류 유형 1): 대입 연산자의 왼쪽 부분에 변수가 존재하지 않으면 이는 문법적으로 잘못된 것으로 오류로 분류된다. 예를 들어, ":= 1 + 2;" 또는 "2 := 1 * 3;" 같은 구문은 왼쪽 항이 빠져 있다. 이 경우 LHS에 임시 값을 지정해 놓고 무의미한 문장을 실행한다.

```
:= 1 + 2;
2 := 1 * 3
```

```
/Users/minseok/Desktop/CPP_Parser/cmake-build-debug/CPP_Parser test_case/test0.txt
[Unknown] := 1 + 2;
ID: 0; CONST: 2; OP: 1;
(Error) 대입 연산의 Left-hand side 명시되지 않음
[Unknown] := 1 * 3;
ID: 0; CONST: 2; OP: 1;
(Error) 대입 연산의 Left-hand side 명시되지 않음
Result ⟹
```

3. 0으로 나누기 시도 (오류 유형 2): 프로그래밍에서 0으로 나누는 것은 정의되지 않은 연산으로, 문장 내에서 이를 시도할 경우 즉시 오류로 분류한다. "operand1 := 21 / 0 + 1;" 같은 예에서 볼 수 있듯, 0으로 나누기 시도는 오류로 간주된다.

```
operand1 := 21 / 0 + 1;
operand2 := 2 / operand1
```

```
operand1 := 21 / 0 + 1;
ID: 1; CONST: 3; OP: 2;
(Error) 0으로 나누려는 시도
operand2 := operand1 + 2;
ID: 2; CONST: 1; OP: 1;
(Error) 정의되지 않은 변수(operand1)가 참조됨
Result ⟹ operand2: Unknown; operand1: Unknown;
```

1-2. 구현 세부 사항 결정: -v 옵션

-v 옵션이 argv[1]에 제공된 경우, 기존의 토큰의 문자열을 출력하는 부분에서 문자열이 아닌 토큰의 타입을 출력하는 방식으로 구현하였다.

```
operand1 := 1;
operand2 := (operand1 * 3) + 2;
target := operand1 + operand2 * 3
```

```
MINSEOK ~/Desktop/CPP_Parser master ±
./CPP_Parser -v ./test_case/test0.txt
IDENT
ASSIGN_OP
CONST
SEMI_COLON
IDENT
ASSIGN_OP
LEFT_PAREN
IDENT
MULT_OP
CONST
RIGHT_PAREN
ADD_OP
CONST
SEMI_COLON
IDENT
ASSIGN_OP
IDENT
ADD_OP
IDENT
MULT_OP
CONST
END_OF_FILE
```