Team 3 Tiny-Ada Syntax Analyzer 설명서

강병헌, 이성원, 황선영

# Team 3 Tiny-Ada Syntax Analyzer 설명서

What is the purpose and function of this analyzer?

* 지난 과제에서 구현한 ada Syntax analyzer의 일부를 기초로, Semantic Analyzer를 구현하였다.

How to Compile? How to Use?

* 터미널에서, python TerminalApp.py로 실행

What does it consist of?

* Token class, CharIO class, Scanner class, Parser class, SymbolEntry, SymbolTable

What is the role of each class?

* Token, CharIO, Scanner class는 지난 과제와 동일하므로 기재하지 않는다. Parser, SymbolEntry, SymbolTable은 변경 및 추가 되었으므로 하단에 기재하였다.

## Parser Class

Semantic Analysis가 추가되며 Role을 설정하도록 바뀌었다. 따라서 actualParameterPart가 더 이상 쓰이지 않는다. name을 호출할 때, entry라는 변수에 저장하고 acceptRole을 통해 entry가 role과 일치하는지 검사하는 구조이다.

아래는 Semantic Analyzer부분으로 추가된 함수들이다.

setRole(s, role): s의 role을 설정한다.

acceptRole(s, expected, errorMessage): role error가 발생하는지 검사하는 함수다. mode와 Parser.ROLE을 비교하고, s의 role이 NONE이거나 expected와 일치하지 않을 경우 (또는 expected 집합에 속해 있지 않을 경우), 에러를 발생시키고 에러메세지를 출력한다.

appendEntry(head, tail): role error가 발생하지 않은 경우 새로운 entry tail을 head에 추가한다.

initTable(): SymbolTable에 여러 type과 role을 설정한다.

enterScope(): role error가 없을 경우 경우 table.enterscope()를 재귀적으로 호출한다. 새로운 테이블을 스택에 push한다.

exitScope(): role error가 없을 경우 self.mode의 table.exitcope()를 재귀적으로 호출한다. 스택에서 테이블 하나를 pop한다.

enterId(), findId(): token의 code가 identifier가 맞는지 확인하고 그렇지 않다면 fatalError를 발생시킨다.

typeDeclaration(): type 선언에 대한 syntax error를 검사한다.

typeDefinition(): 입력된 token.code에 따라 enumerationTypeDefinition(), arrayTypeDefinition(), range()을 호출해 syntax를 검사하고 role을 설정한다. 입력된 코드가 Token.ID일 경우 entry가 SymbolEntry의 type과 일치하는지 확인한다.

subprogramSpecification(): procedure와 identifier에 대한 검사를 실시하고 ‘(’ 이 입력된 경우 parameter에 대한 syntax 검사를 한다.

## SymbolTable Class

오류 메시지 출력을 위한 Chario 개체를 생성한다. 각 scope에 해당하는 symbol table을 관리하는 class이다.

아래는 SymbolTable을 구현하기 위한 함수들이다.

reset() : scope 구분에 이용되는 stack을 초기화 시켜준다.

enterScope() : stack에 새로운 scope를 생성시키고 level을 1 올려준다.

exitScope() : stack의 가장 위에 있는 table을 제거하고 테이블을 출력한다. Level을 1 감소 시킨다.

enterSymbol(id): stack의 가장 위에 있는 table을 꺼내와서 특정 identifier가 존재하는지 확인하고 있으면 chario를 통해 오류메시지를 출력한다. 없다면 symbol table에 추가한다.

findSymbol(id) : stack에 있는 table들을 순서대로 확인하며 특정 identifier의 entry를 확인합니다. 없다면 chario 를 이용해 오류를 출력한다.

printable(table) : table을 level에 따라 모두 출력한다.

## SymbolEntry class

입력된 Symbol이 None, constant, Parameter, Procedure, Type, Variable 중에서 어떤 타입이며, 이름은 무엇이고, 다음 Symbol은 무엇인지 저장하는 Class이다.

ToString(self, mode): parameter와 parser의 role이 동일할 경우 name과 role을 문자열로 return한다. 동일하지 않을 경우 이름만 리턴한다.

setRole(self, r): parameter로 들어온 숫자에 따라 해당 객체에 role을 부여한다.

Append(self, entry): 마지막 symbolentry 객체의 다음에 parameter로 들어온 symbolentry 객체를 추가한다.

roleToString(self): 해당 객체의 role을 리턴한다. (none, constant 등)