情報科学演習Ｄ　課題3レポート

氏名　山久保孝亮
所属　大阪大学基礎工学部情報科学科ソフトウェア科学コース
メールアドレス　u327468b@ecs.osaka-u.ac.jp
学籍番号　09B22084

提出日　2024 年 12 月 5 日
担当教員　桝井晃基　松本真佑

1　システムの仕様

課題 3 の外部仕様は以下のようになる．

- 第一引数で指定された ts ファイルを読み込んでい未解析を行い，意味的に正しい場合は文字列”OK”を，正しくない場合は文字列”Semantic Error”に続いて対応する行番号を返す．
- 複数の意味的誤りが含まれる場合では，最初に見つけた誤りのみを出力する．
- 意味解析よりも先に構文解析を適用し，構文エラーを見つけると ”Syntaxerror: line ”という文字列に続いて対応する行番号を返す．
- 入力ファイルが見つからない場合は文字列 ”Filenot Found ”を返す．

2　課題達成の方針と設計

課題 3 のテストケースをパスするために実装した機能は以下の通りである．

1. 変数が重複して定義されない．
2. 未定義の変数及び関数を参照しない．
3. 型における制約を守る．
4. 代入文の左辺に配列型の変数名を使用しない．

課題 3 では課題 2 の parser.java のプログラムの一部を変更して実装した．上記の機能を満足するために変更及び追加した方針は以下の通りである．

- 変数名，関数名等を記憶するための表を作成する．今回作成した表は言語処理工学 A の授業スライドを参考にして作成しており，記号表，関数表，変数表の三つである．記号表は宣言の重複を，変数表は未定義の変数の参照を，関数表は未定義の手続きの呼び出しを防ぐために作成した．それぞれの表の構成要素は以下の表の通りである．

記号表	記号名	記号の種類	記号の型
変数表	変数名	変数の型	記号の型
関数表	関数名	引数の型	null

表 1: それぞれの表の構成要素

null は要素が存在しないことを，つまり関数表には二つの要素しか存在しないことを表す．〇〇名及び〇〇の型はそれぞれの表に格納される識別子及び標準型のいずれかを格納する．記号の種類にはプログラム，変数，手続きの内いずれであるかを格納する．変数のサイズには配列の際は要素数を，配列でなければ 1 を格納する．
表を活用するためにはまず格納する必要があるが，そのタイミングについては 3 の実装プログラムで詳細を述べる．宣言の重複について，指導書よりプログラムの宣言と手続きの宣言で処理を分ける必要がある．したがって，変数表はグローバル変数用のものとローカル変数用のものをそれぞれ用意した．これにより，グローバル変数を宣言する処理の際はグローバル変数用の変数表と関数表を，ローカル変数を宣言する処理の際はローカル変数用の変数表と関数表を参照することによって要件を満足することができた．また，未定義の変数の参照について，ローカル変数用の表，グローバル変数用の表の順番で探索するようにした．これにより，グローバル変数でもローカル変数でも定義されている変数名に対して，ローカル変数として正しく参照できるようにした．

- 式，単純式，項，因子を判定するメソッドが型を返すようにする．課題 2 では構文定義を判定するメソッドはすべて void 型であったが，前述のメソッドが ”integer ”, ”char ”, ”boolean ”のいずれかを String 型で返すようにした．これにより，各被演算子の型を把握することができるので，型の整合性がとれているのか，特定の演算子に適した型が使用されているのかを判定する．

3 実装プログラム

今回実装したプログラムは以下の 3 つである．

3.1 意味エラーの検出

3.2 表の作成

2 で記述したように，変数の情報を記録しておくために表を作成した．それぞれの表に対応する symbol,variable,function というクラスを作成し，それぞれグローバル変数のリストとしてどのメソッドからでも参照できるようにした．それぞれのクラス内の変数は表 1 に対応しており，それぞれのクラス内で以下のようなメソッドを作成した．

```
1  public Type get_A_B(){
2    return B;
3  }
```

ここでは A がどの表であるか，B が表のどの変数であるかに対応している．これにより，例えば variable というリストの i 番目の要素に対して上記のメソッドを実行すれば i 番目の B の値を参照できるようになる．

3.3 表への格納

表への格納は StoreToTable() を定義して実現した．StoreToTable() のプログラムは以下のようになる．

```
1  private void StoreToTable(ArrayList<Object> list) {
2                 String type = (String) list.get(0);
3                 int size = (int) list.get(1);
4                 if(global_flag == true) {
5                         for(int i = 0;i<list_of_variablename.size();i++) {
6                                 global_variable_table.add(new variable(
                                      list_of_variablename.get(i),type,size));
7                                 symbol_table.add(new symbol(list_of_variablename.get(i),"
                                      variable",type));
8                         }
9                 }else {
10                        for(int i = 0;i<list_of_variablename.size();i++) {
11                                local_variable_table.add(new variable(
                                      list_of_variablename.get(i),type,size));
12                                symbol_table.add(new symbol(list_of_variablename.get(i),"
                                      variable",type));
13                        }
14                }
15                list_of_variablename.clear();
16        }
```

3.4　型の制約

4　考察

5　感想

課題 3 を通して学んだ感想としては，課題 2 においてウォータフォール型の開発をしたことの反省からテストファーストな開発を試してみて，かなり効率的に実装を進めることができたと感じている．

参考文献

[1]