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，関数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た．この表により変数の二重定義，未定義の変数及び手続きの参照を防止できる．それぞれの表の構成要素は以下の表の通りである．

変数表	変数名	変数の型	変数のサイズ
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．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．

- 変数の宣言において，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 global_variable_table を，ローカル変数を宣言する処理の際は local_variable_table を参照することによって要件を満足することができた．
- 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　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 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．

3.2　表への格納

変数表，関数表への格納は以下の図 1 のフローチャートに従って実行される．

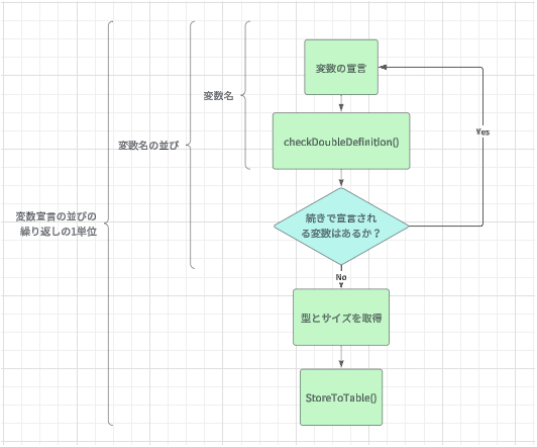

図 1: 変数宣言の並びの繰り返しの 1 単位

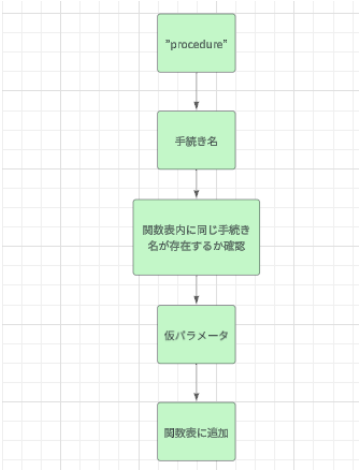

図 2: 副プログラム頭部

図 1 は 1 回以上繰り返される変数宣言の並びの繰り返しの 1 単位の処理を表している．checkDoubleDefinition() と StoreToTable() はそれぞれ二重定義の処理と変数表への格納の処理を行う．また，ここではグローバル変数として list_of_variablename という String 型のリストを使用し，checkDoubleDefinition() の結果二重定義でなければ add() を使用し追加している．ここでいきなり変数表に格納しない理由は，この段階では変数名がわかっただけでありその変数の型やサイズが不明であるためである．したがって，型とサイズが判明してから StoreToTable() を呼び出している．以下では checkDoubleDefinition() と StoreToTable() の詳細について述べる．

- checkDoubleDefinition() では変数名が二重定義されていないかを確認している．引数には宣言された変数名が渡されており，それが各表の名前と一致していないかを確認する．一致している場合即ち二重定義されている場合，文字列”NO”を返し，一致していない場合文字列”YES”を返す．まず関数名，

list_of_variable_name と一致していないかを確認する．繰り返し文と 3.1 で紹介したメソッドを用いて表に格納された関数名を取り出して比較する．そして，プログラムの宣言の際はグローバル変数用の表と，手続きの宣言の際はローカル変数用の表と一致していないかを確認する．

- StoreToTable() では表への格納を行っている．引数の list は二つの要素があり，一つ目は格納する変数の型，二つ目は格納する変数のサイズである．global_flag によってグローバル変数用の表に格納するか，ローカル変数用の表に格納するかを分岐させている．このフラグはプログラムの宣言の際に立ち上げ，手続きの宣言の際には下げられている．また，list_of_variablename はこのメソッドが呼び出されるたびに初期化される．これは，新たな変数宣言の繰り返し単位が存在した場合に再度表に追加されてしまうことを防ぐためである．

関数表への格納は副プログラム頭部のメソッドに処理を追加して実装した．確認と追加の方法は変数表の際と同じである．

3.3 未定義の変数の参照

未定義の変数の参照を防ぐために checkVariable() というメソッドを定義した．これは変数の定義の後に呼び出される．引数として識別子が渡され，その文字列が変数表内で定義済みかどうかを判定する．定義されていない場合は文字列”NO”を，定義されている場合はその変数の標準型を文字列として返す．

3.4 型の制約

文の処理を行う際に，定義済みの変数を扱う場合や代入文を扱う場合などは正しい型が使われているか確認する必要がある．したがって，式，変数，単純式，項，因子，定数を表現するメソッドが文字列として型を返すように変更した．これにより，

4 考察

今回のテストケースは最低限

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]