A-016

変数宣言機能を付与した Ruby 処理系の作成

Making of Ruby processing system that gives variable declaration function

大垣　聡†　五百蔵　重典†　野木　兼六†
Satosi Ogaki　Shigenori Ioroi　Kenroku Nogi

1.目的

プログラムの作成時には，変数名の誤入力によって生じるエラーが多い.そこで，多くのプログラミング言語では，変数の宣言を行うことで，誤った名前を持つ変数の使用に対して警告を発している．それに対し，プログラマの手間を考え，変数の宣言を行わない言語も存在する．

変数の宣言を行う言語の利点としては，変数名の誤りを検出しやすい点がある．欠点としては変数を宣言しなければ使えない点がある．

変数の宣言を行わない言語の利点としては，プログラミング中使いたいときに好きな変数名を持つ変数が使える点がある．欠点としては変数名が誤っていても分からない点がある．

スクリプト言語の一つである Ruby[1]には，変数の宣言が存在しない．そこで，変数の宣言を使用する事によって，本当に誤った名前を持つ変数の使用を修正しやすくなるのかどうかを検討することにした．

そのために，Ruby に変数宣言を導入することにした．

2.具体的な解決方法

2.1.解決方法の選択肢

Ruby に変数宣言を導入する方法は，大きく分けて以下の 2 つの方法がある．

1)Ruby 本体を拡張する方法

2)プリプロセッサを作成する方法

まず，Ruby 本体を拡張する方法．この方法の利点としては，通常の Ruby で生じる，文法のエラーと同じ見た目で出力できるという点がある．欠点としては，拡張が容易ではないという点がある．

次に，プリプロセッサを作成する方法．この方法の利点としては，作成が容易である点が挙げられる．欠点としては，構文のエラーと一緒には出力できないという欠点がある．

今回は，エラーを修正しやすくすることに重点を置く．

その為，通常の Ruby 処理系で生じる構文のエラーと同じ見た目で出力でき，なおかつ一度に出力出来る，Ruby を拡張する方法をとった．

2.2.Ruby を拡張することによる解決方法の詳細

2.2.1.文法的な拡張

新たな予約語として，"var"を追加した．これは，各ブロックの先頭でのみ使用できる予約語で，この次に変数名を記述する必要がある．例えば"var abc;"と記述すると変数 abc の宣言とする．

†神奈川工科大学工学，Kanagawa Institute of Technology

そして，変数の参照時に，その変数名が宣言されているかどうかを確かめる．

2.2.2.意味的な拡張

予約語"var"は，この次に記述された変数名を持つ変数が宣言されたものとして動作する．宣言された変数名は，変数名のリストに登録する．

また，メソッド定義に現れる引数も，引数名を持つ変数が宣言されたものとして動作する．こちらも，変数名のリストに登録する．

そして，変数の参照時にその変数名をリストから探索し，宣言されたものかを判断する．そして，リストに無い変数の場合，宣言されていない変数の使用として，その参照された変数の使用に対して，警告文を出力する．

なお今回の追加では，複数の種類の型宣言は導入していないので，C 言語などのように，引数の型を記述することは出来無いものとした．

2.2.3.実現方法

まず，予約語のリストに新たな予約語"var"を追加した．次に，Ruby のパーサに予約語"var"の構文と，変数名のリスト，リストに対する処理を追加した[2].

3.実験と結果

3.1.実験方法

実験として，意図的に変数名のエラーと文法上のエラーを含むプログラムを，本来の Ruby 処理系と拡張した Ruby 処理系のそれぞれで実行し，その出力結果を比較した．その際，変数名のエラーに関するメッセージが出力されることを期待する．

3.2.実験結果

3.2.1.本来の Ruby 処理系

本来の Ruby 処理系でプログラムを実行した結果を図 1 に示す．本来の Ruby 処理系は変数名のエラーを出力できないため，構文上のエラーのみの出力となった．

この場合，変数名に存在するエラーは分からない為，出力された構文上のエラーを訂正した後で，再度実行する．その結果，期待する結果を得られないため，プログラムのどこかに問題があることが分かる．

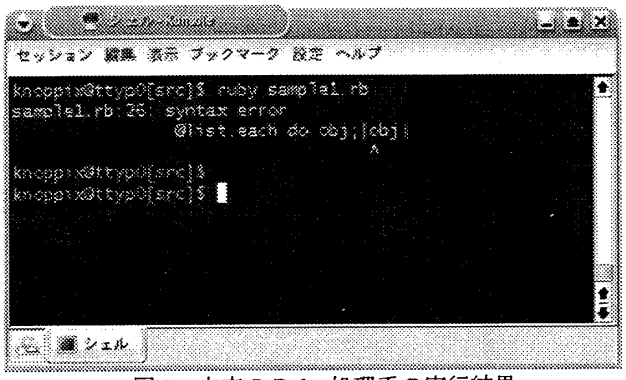

図 1　本来の Ruby 処理系の実行結果

3.2.2.拡張した Ruby 処理系

今回拡張した Ruby 処理系でプログラムを実行した結果を図 2 に示す．本来の Ruby 処理系では出力されなかった，未宣言の変数名に関して警告が出力された．

この場合，変数名に存在するエラーも同時に出力される為，出力された構文上のエラーと未宣言の変数名を訂正した後で，再度実行する．その結果，期待する結果を得ることができる．

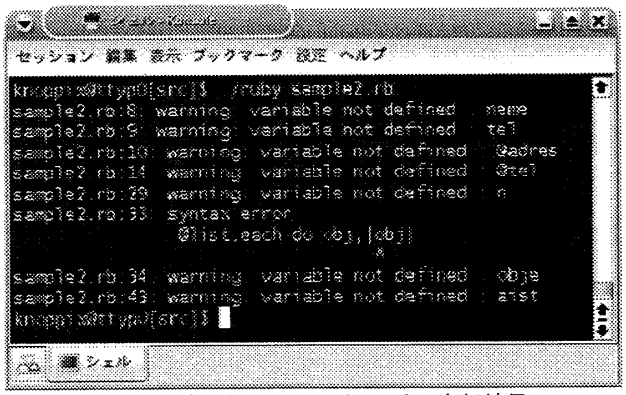

図 2　拡張した Ruby 処理系の実行結果

4.考察

3.2.2 の出力から，変数名の入力を誤ったと思われる行数がわかる．その行を見直すことによって，変数名の誤入力に関するエラーを修正することができる．よって，本来の Ruby 処理系より，エラー訂正をしやすくすることができたといえる．

しかし，今回の実装には，いくつかの問題点が挙げられる．

まず，ツールなどによってプログラムを作成した場合など，警告が必要でない場合において，警告を出力しない選択が出来ない点が挙げられる．そこで，コマンドラインからのオプションの入力によって，警告を出力しないように改善する必要がある．

次に，宣言された変数名のテーブルを独自に作成し，それを保持している点が挙げられる．これを本来の Ruby 処理系の持つ変数テーブルを拡張し，変数宣言の機能を持たせることで，実行に必要なリソースを削減可能であると推測される[3].

最後に，今回は誤入力による変数名のエラー訂正を，容易にすることに重点を置いたため，型宣言などの方法を用いて，変数の使用を誤ったエラーは訂正できない．そこで，型宣言を導入することで，誤ったメソッド呼び出しなどに対して，警告を出せるようにする．

これら 3 点を今後考慮する必要がある．

参考文献

[1]Ruby webページ，http://www.ruby-lang.org/ja/

[2]R.Levine，D.Brown，T.Mason (共著) 村上 列 (翻訳)：lex&yacc プログラミング，アスキー出版局 （1994）．

[3]青木 峰郎：Ruby ソースコード完全解説，株式会社 インプレス （2002）．