5ZF-7

プログラミング教育における コーディング手順矯正システムの開発

石嶋　博行[†]　山田　優里[†]　粂谷　歩[†]　高野　辰之[††]　宮川　治[†††]

東京電機大学情報環境学部情報環境学科[†]　東京電機大学大学院先端科学技術研究科[††]

東京電機大学情報環境学部[†††]

1. はじめに

初学者がコーディングを行う場合，様々なエラー（構文エラー，意味エラー，論理エラー）を発生する．エラーには学習者が気付かないもの，原因を特定することが難しいものがあり，その修正を行うのは容易ではない．適切な手順を学習者に統一的に指導することで，エラーの発生を抑え，修正も容易にすることができる．しかし，すべての学習者のコーディング手順を把握し，指導を行うことは困難である．

本研究ではコーディング手順矯正システムを開発した．これにより，指標となる手順を体験させることで，適切なコーディング手順の習得を支援する．本システムは Java 言語を対象としている．

2. コーディング手順

エラーの発生を抑え，修正を容易にするために定義した適切な手順について以下に述べる．

2.1 括弧，引用符

学習者がよく起こすコンパイルエラーに，括弧（中括弧，大括弧を含む）の対応関係が崩れているものがある[1]．その要因の一つとして，括弧の始まりを入力し，括弧内の処理を入力した後，終わりを入力することを忘れることが考えられる．その結果，括弧の対応関係が崩れてしまい，エラーが発生する．この時，発生するエラーでは，原因となる間違えた場所と実際にエラーの発生する場所が異なるため，修正を行うのが困難である．

括弧，引用符の入力の際，始まりの入力の直後に終わりを入力し，その後に内側や以降の入力を行う手順を踏む．こうすることで終わりを入力することを忘れないようになり，修正が困難であるエラーの発生を抑えることができる．

Development of Coding Procedure Restraint System in Programming Courses

Hiroyuki ISHIJIMA[†], Yuri YAMADA[†], Ayumi KUMEGAI[†]
Tatsuyuki TAKANO[††] and Osamu MIYAKAWA[†††]
[†] Department of Information Environment, School of Information Environment, Tokyo Denki University
[††] Graduate School of Advanced Science and Technology, Tokyo Denki University
[†††] School of Information Environment, Tokyo Denki University

2.2 インデント

インデントはプログラムの構造を明確にする手法として用いられている．従来からインデントとプログラミング能力の関係が研究され，構造化プログラミングにおいて，インデントがプログラムの読解力の向上につながる要素であることが示されている[2]．

適切なインデントを行うことで，意味エラーの原因を発見しやすくなり，修正が容易になる．

2.3 エラーの細かな確認と修正

細かくエラー確認，修正を行わずにコーディングを行った場合，エラーを発見した時には，ソースコードの変更箇所が多くなる傾向にある．その変更箇所の中からどこにエラーの原因があるのかを特定することは難しい．

エラーの細かな確認と修正を行うことで，ソースコードの変更箇所が少なくなり，エラーの発生原因の特定が容易になる．

3. システム概要

本システムはクライアントアプリケーションとして学習者に提供される．実装は Java を用いて行った．このシステムは見本の表示部とソースコードの入力部で構成される．見本には予め教授者が作成したソースコードを使用し，システムに組み込んでおく．学習者は見本表示部に表示されているソースコードと同じものを入力部に入力していく．

入力部には一般的なテキスト入力機能のほかに，手順を矯正するための機能がある．その機能について以下に述べる．

3.1 入力キーとソースコード情報の取得

入力部にて学習者がキーを入力する度に，入力キーと，ソースコードの情報を取得する．これにより，括弧，引用符の手順，適切なインデントの入力の判定を行う．

3.2 ソースコードのエラー確認

エラーの細かな確認を行うため，取得したソースコードにエラーが無いかの確認を自動的に行う．確認を行うタイミングは「’}’，’;’，改行」が入力された時で，宣言や文などの入力終わりと判断し定義した．

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.

エラー確認には構文エラーのためのコンパイルと，意味エラー，論理エラーのための見本との比較が行われる．

3.2.1 コンパイル

入力されたソースコードをコンパイルすることで，構文エラーの検出を行う．

ソースコードのコンパイルには Java SE 6 から利用可能な Java Compiler API を利用した．

3.2.2 見本との比較

入力されたソースコードから構文エラーが検出されなかった場合，構文解析を用いて見本との比較を行うことで，意味エラー，論理エラーの検出を行う．

構文解析には，Eclipse の API である JDT（Java Development Tools）を利用した．

3.3 誤りに応じた処理

誤った手順を検出した場合には，その誤りを学習者へ提示し，入力をキャンセルする．これにより，学習者は誤りの修正を行わない限り，次の入力へ移れなくなる．

誤りが検出されるのは，括弧，引用符の手順が守られていない場合，適切なインデントが入力されていない場合，エラーが検出された場合である．

4. 試験運用

4.1 対象と方法

```
01: public class TestMain{
02:     public static void main(String[] args) {
03:         int[] scores = {70, 90, 60, 80};
04:         int max = scores[0];
05:         for(int i = 0; i < scores.length; i++) {
06:             if(max < scores[i]) {
07:                 max = scores[i];
08:             }
09:         }
10:         System.out.println(max);
11:     }
12: }
```

図 1　見本ソースコード

実習授業において試験運用を行った．対象となった学生は 19 名である．手続き型プログラミング，オブジェクト指向プログラミングの基礎を習得している．また，インデントにはタブ文字の使用を指導している．試験運用での見本ソースコードを図 1 に示す．この見本を見て学生は入力し，ソースコードが完成する過程で検出した誤りについて集計を行った．

4.2 結果

表 1 に検出した誤りの種類，検出した回数を示す．集計した誤りについて以下に述べる．なお，誤りの検出には優先順位があり，以下の項目の並び順に検出され，他の誤りは検出されない．

表 1　誤りの検出回数

誤り	検出した回数
括弧，引用符	26 回
インデント	92 回
コンパイル	56 回
見本との比較	28 回

- 括弧，引用符：括弧，引用符の始まりを入力した後，終わりと異なる入力をした際に検出される．
- インデント：適切なインデントが入力されていない行でタブ文字と異なる入力をした際と，適切なインデントが入力されている行でタブ文字を入力した際に検出される．
- コンパイル：ソースコードのエラー確認を行った際に，コンパイルエラーがあったとき検出される．
- 見本との比較：コンパイルエラーが検出されなかった際に，スペルミスなど見本と異なる部分があったとき検出される．

集計の結果，一番多く検出した誤りはインデントの92回．次いでコンパイルの56回，見本と比較の28回があり，一番少なかったのは括弧，引用符の26回であった．

5. まとめ

エラーの発生を抑え，修正も容易にするための適切な手順を定義し，学習者の手順を矯正するシステムを開発した．これにより，学習者が適切な手順を体験することが可能となった．また，試験運用を行い，実際にどのような誤りが検出されるかを確認した．その結果，どの誤りにおいても，連続して同じ誤りを起こす学習者が少なからずいたことから，どう修正すべきかわからないことが考えられる．

今後は，誤りに応じた修正を促すメッセージの作成することで，システムの利便性の向上を目指す．また，試験運用を繰り返し行うことで誤りの検出回数などに変化が起こるか確認し，システムの有用性について検証を行う．

参考文献

[1] Hristova, M., Misra, A., Rutter, M. and Mercuri, R.: Identifying and correcting Java programming errors for introductory computer science students. *Proc.the 34th SIGCSE technical symposium on Computer science education, Comm. ACM,* Vol.35, No.1, pp.153–156, (2003).

[2] Oman, P.W. and Cook, C.R.: A paradigm for programming style research, *SIGPLAN Not.*, Vol.23, No.12, pp.69–78, (1988).

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.