プログラミングスタイル習得のための自己学習環境

上村 拓磨[*1]・北 英彦[*1]
Email:416n207@m.mie-u.ac.jp

*1：三重大学工学研究科電気電子専攻

◎Key Words プログラミング演習，プログラミングスタイル，自己学習

1. はじめに

現在多くの大学がプログラミングの講義を行っている．しかし，プログラミング初心者に対するプログラミング演習において，プログラミングスタイルについて十分な指導を行う時間がないというのが現状である．最近では自動で字下げを調整するエディタ[1]も増えてきているが，それに頼っていてはプログラミングスキルのひとつであるプログラミングスタイルを整える能力が身につかない．

本研究では，学習者にプログラミングスタイルを学習する機会を提供するために，学習者が提出したプログラムに対してどのようなスタイルにすればよかったのかを提示する機能を我々が開発および運用を行っているプログラミング演習システム PROPEL[2][3]に付け加えた．またこの機能を実際の授業の学習者に使用してもらい，プログラミングスタイルに関する能力の改善に有効であることを確認した．

2. 演習時のプログラミングスタイルの現状

学習機能を使う前に字下げのことを知っているかというアンケートを行った[4].

- 電子工学科 2 年生および 3 年生 2013 年度
- 実施者：2 年生 7 名　3 年生 20 名　計 27 名

設問：「字下げ」という言葉・概念を知っていましたか？

- 完全に，おおよそ知っていた．(15 名，65%)
- ほとんど，まったく知らなかった．(9 名，39%)

このように字下げの認知率は 65%と高くはなかった．対象者は少なくとも 1 度はプログラミング演習の授業を受けているはずであるのにも関わらず，このような結果がでたということは学習者がプログラミングスタイルをしっかりと学習できていなかったということである．

3. プログラミング演習支援システム

本研究室では，プログラミング学習の支援を目的とした演習システムの開発を行っており，プログラミングスタイルの学習機能を既存の演習システム PROPEL 上に構築した．

PROPEL のシステム構成図を図 1 に示す. PROPEL では，学習者はプログラミング演習において必要な作業のうち「デバッグ」以外の「コーディング」「コンパイル」「プログラムの実行」「保存」「提出」を Web 上のひとつの画面上で行うことができる．

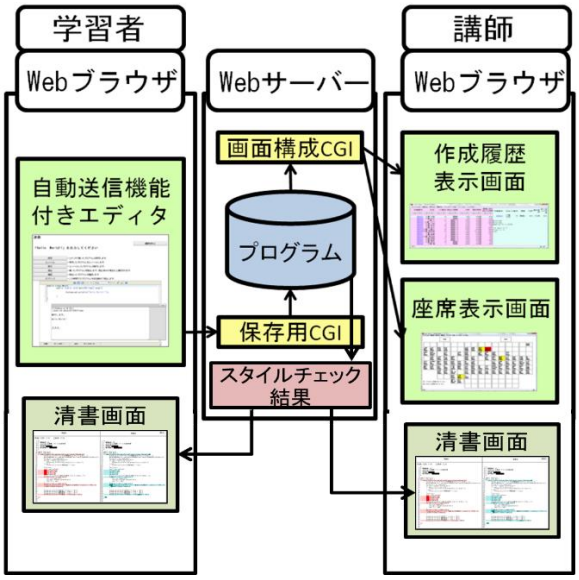

図 1　PROPEL のシステム構成図

4. プログラミングスタイル

プログラミングスタイルとはプログラムを書くときの規則のことであり，特定の規則に沿ってプログラムを書くことで，他の人が読んでも理解しやすいプログラムを作成することが可能である．プログラミングスタイルとして以下のものがある．

- 字下げ
- 空白の入れ方
- 適切な変数名

他にもループ文の制御構造やリストの書き方などがあるが上記の 3 つが基本である．これらを用いることによる利点は以下の通りである．

字下げ（インデント）：プログラムの構造をわかりやすくするために行うものであり，基本的には波括弧{}で囲まれた範囲を 1 段字下げする．

空白の入れ方：演算子の後に空白を入れることにより演算子の関係などが明瞭になるという利点がある．

適切な変数名：どのような意味を持った変数か明確に理解することができる．Java では Camel 形式や Pascal 形式などの形式がある．

表 1　Pascal 形式，Camel 形式の例

Pascal 形式	StuedentNumber
Camel 形式	studentNumber

本研究で対象とするプログラミング演習では Java によるプログラミングを教えている．Java の特徴として C 言語と比べてプログラミングスタイルの流派が少ない点があげられる．この理由として Java をリリースした Sun Microsystems[5]がコーディング規約を提示しているというのがある．

Java でも細かい点では様々なプログラミングスタイルの流儀が存在するが，本研究では，字下げ量を 4 つ，演算子間には空白をあけるものとする．一部の例を図 2，図 3 にのせる．

変数名については今後の課題とし，今回は取り上げない．その理由としては変数名を命名規則に従って付けているかを判定する際に，短縮語を使用していた時の判定が難しいということがあり，また変数名を付ける上でどのような変数名が適切かを判定するためには課題の文章の意味を自然言語理解する必要があるからである．コメントについても上記と同じ理由のため今後の課題とする．

```
if (condition) {
    statements;    // ブロック内は字下げ
} else {           //if 文に else 文が続く場合
    statements;
}
```

図 2　if-else 文の字下げ

```
switch (condition) {
    case ABC:
        statements;
        break;
    case DEF:
        statements;
        break;
}
```

図 3　switch 文の字下げ

5.　プログラミングスタイル学習機能の構築

学習者が作ったプログラムをチェックすると字下げや空白の付け方が不適切であることが多い．上記でも述べたが，プログラミング演習でプログラミングスタイルを学習しているはずであるにも関わらず，このような結果が出たということは座学だけではなく，実習においてプログラミングスタイルを学ぶ学習環境が必要である．前述通り今回の研究では字下げ，空白の付け方に注目する．

先行研究として同じようなものはないかと探したところ正しいプログラミングスタイルに整形するツール[6][7]は見つかったが，それは学習に使用するものではなく，プログラムを読みやすくするためのツールだった．そこで本研究ではプログラミングスタイルを学習する機会を提供する環境を構築する．

5.1 プログラミングスタイルのチェック方法

学習者のプログラムがプログラミングスタイルとして正しいかどうかをコード整形ツールである Artistic Style[7]を用いて判定する．選択した理由として当ツールはコマンドライン上で実行でき，引数によって細かくプログラミングスタイルの形を指定できるからである．Artistic Style とは GNU Lesser General Public ライセンスで書かれたプログラムで C, C++, Objective-C, C#, Java などのプログラミング言語のソースコードの字下げや整形を行えるツールである．ただし最新のバージョンでも，以下のことについては適切な整形されない．

(1) 配列として宣言された変数の後につく四角括弧「[」の間に空白があっても削除されない．
(2) フィールドやインスタンス，パッケージ，メッソッド間につくピリオドの前後に空白が含まれていても削除されない．

表 2　上手く整形されない例

適切な例	System.out.println(“点数:” + score[i]);
不適切な例	System.out. println(“点数:” + score [i]);

※網掛けの部分が不適切（余分な空白）

5.2 清書モジュール

今回はプログラミング演習システム PROPEL を利用するため Web 上から Artistic Style を起動し清書を行う．上記のうまく整形できない例の 2 つについては Artistic Style で清書する前にその部分の空白を取り除いておく．Artistic Style はあくまでもコード整形ツールのため一目でどこのプログラミングスタイルが良くなかったかを理解することは難しい．そこで清書前のソースコードと清書後のソースコードを比較し，さらにどこが違っているのかをよりわかりやすくするために，色を使って表示することにした．システムの構成図を図 4 に示す．

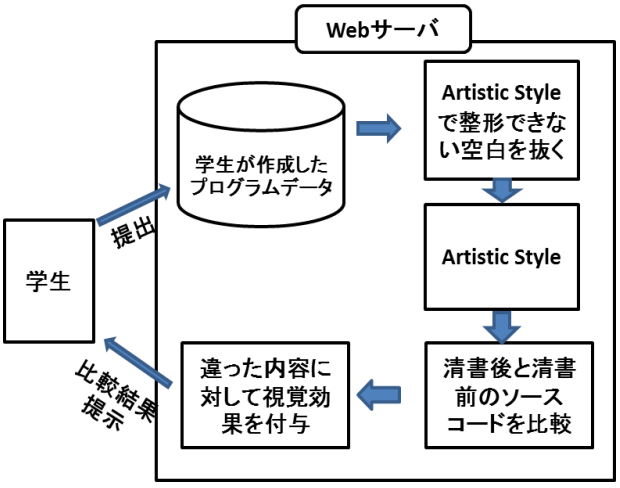

図 4　清書システムの構成

次に学習者に返す画面を図 5 に示す．図のように間違えた行に対しては清書前であったら桃色，清書後であったら水色として表示している．そしてその間違えた行の空白に余分なものがあった場合は余分な部分が赤色で表示され，空白が少なかった場合は青色で表示される．このように色分けすることによってどこの箇所を間違っているか一目でわかるようになった．後述にある正解率を表示することによって全体的にどの程度自分のプログラミングスタイルが正しいかわかるようになっている．

6. 運用方法

運用にあたって，学習者自身にプログラミングスタイルを身につけてもらう事がこの機能を作成した目的の一つであるため，プログラムを提出した後でしか清書結果を見ることができないようにした．つまり次回の演習時に以前に提出したプログラムの清書結果を見たことによる効果がでてくるということである．

また学習システムを提供するにあたってこのシステムの使用を促すアナウンスの仕方を変えてみた．アナウンスの流れとしては図 6 の通りである。実装したというアナウンスのみで使用の強制をしなかった理由としては強制的に学習させるのではなく，どこまで自主的にプログラミングスタイルを学び，それを活かすことができるかを調べるためである．次の段階としてプログラミングスタイルを採点するとアナウンスした理由としては学習者に直接利点があったとき，プログラミングスタイルが向上するかを調べるためである．それによる結果の違いについても述べる．

この研究の調査には 2015 年度の三重大学工学部電気電子工学科 2 年次開講のプログラミング演習

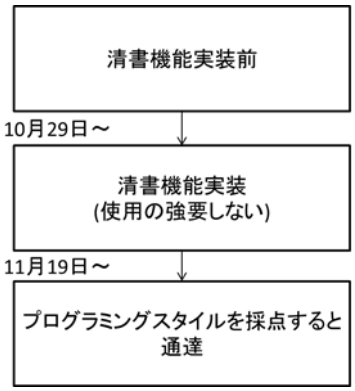

図 6 正解率測定の流れ

Ⅰ・Ⅱで作成したプログラムを用いた．Artistic Style を用いて学習者が提出したプログラムを清書し，その清書後と清書前を比べ不適切な行がないかを検出した．相違があった行を間違えた行とし，学習者のプログラミングスタイルの正解率を以下のように計算する．ただし空白行は含めてしまうと余分な空白行があった時に正解率が上がってしまうため行数に空白行は含めない．

正解率 =((行数 － 間違えた行数) / 行数) * 100

本来プログラミングスタイルは間違ってはならないものである．そのため正解率の目標値は 100%とするのが当然であるが，学習者のケアレスミスの可能性を考慮し，本研究の目標値は 90～100%とする．図 5 のプログラムは正解率が 73.2%であるが，プログラミングスタイルとして必ずしも適切でないことがわかる．そこで恣意的ではあるが，正解率の 1 桁目を四捨五入し，その値が 70%以下すなわち元の値が 75%未満のプログラムはプログラミングスタイルが不適切なものと考える．

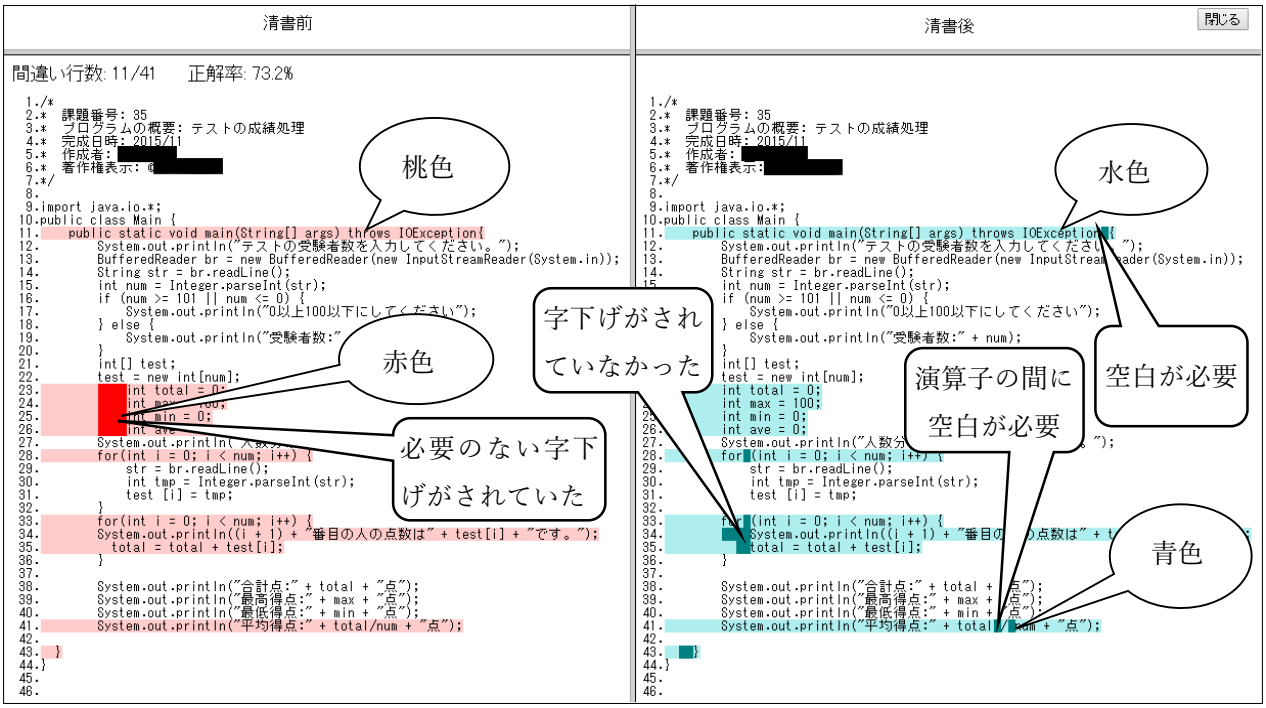

図 5　清書画面

7. 運用結果

まず結果の比較のために，システムの提供前に正解率を測った．正解率の 1 桁目を四捨五入し，その値の 10%毎の人数の割合を以下の表 3 に示す．ただし学習者によって提出していないプログラムがあるのでそれらは数から除いている．

表 3　正解率に対する度数分布表（実装前）

正解率	10%台	20%台	30%台	40%台	50%台
人数(%)	2.1%	6.0%	8.0%	6.0%	2.9%
正解率	60%台	70%台	80%台	90%台	100%台
人数(%)	7.1%	9.7%	16.7%	31.5%	9.9%

※　70%以下合計：41.8%　90%以上合計：41.4%
※　課題数：34 個，受講者：86 人
※　期間：2015/4/23～2015/10/28

表 3 の結果からプログラミングスタイルの正答率が 70%以下の人が半数弱いるということがわかる．プログラミングスタイルを意識してプログラミングをした際にプログラミングスタイルが不適切となるとは考えにくいので，この結果から半数の学習者達はプログラミングスタイルを意識していないことがわかる．

次の段階として清書システムを実装する．しかし図 6 に書いてある通りに，実装直後はこのような機能があるということだけをアナウンスするのみにした．その結果を表 4 に示す．

表 4　正解率に対する度数分布表（非強要）

正解率	10%台	20%台	30%台	40%台	50%台
人数(%)	0.0%	6.1%	4.9%	3.7%	2.4%
正解率	60%台	70%台	80%台	90%台	100%台
人数(%)	1.2%	11.0%	11.0%	30.5%	29.3%

※　70%以下合計：29.3%　90%以上合計：59.8%
※　課題数：3 個，受講者数：76 人
※　期間：2015/10/29～2015/11/12

システム提供前の表 3 と比べると正解率が 100%の割合が約 3 倍程度上昇していることがわかる．この結果から自主的だったとしてもプログラミングスタイルを学ぶ機能さえあればある程度プログラミングスタイルが向上することがわかった．

次に図 6 の通りにシステムの存在があるというアナウンスだけでなく，プログラミングスタイルの正解率が高いと加点するということをアナウンスした．この結果も同様に表 5 に示す．また正解率の度数分布の変化がわかるように棒グラフを図 7 に示す．図 7 から加点すると伝えた方がより正解率 100%の人数が上昇している事がわかり，100%の人数の割合は半数程度まで増加している．また正解率が 70%以下の人数が 14.3%と減少していることがわかる．

表 5　正解率に対する度数分布表（加点を通達）

正解率	10%台	20%台	30%台	40%台	50%台
人数(%)	0.4%	1.4%	1.4%	1.9%	1.8%
正解率	60%台	70%台	80%台	90%台	100%台
人数(%)	2.5%	4.9%	12.7%	23.9%	49.2%

※　70%以下合計：14.3%　90%以上合計：73.1%
※　課題数：15 個，受講者数：76 人
※　期間：2015/11/19/～2015/12/10

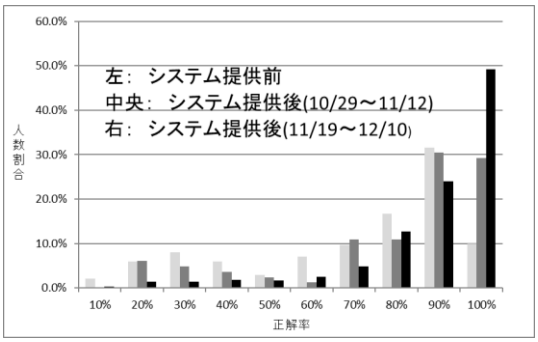

図 7　正解率の変化

8. まとめ

プログラミング演習においてプログラミングスタイルを教える時間は少ない．その中で学習者たちにプログラミングスタイルを身に着けさせるためには学習者たちに自主的に学んでもらう必要がある．

今回の学習機能を使用させることにより学習者たちのプログラミングスタイルが向上するという結果が得られた．しかしある程度こちら側から強制，もしくはメリットを与えないとプログラミングスタイルは大きく向上しないことも分かった．

今後の課題として清書の提供後もプログラミングスタイルが良くならない学習者の対応をしていく必要がある．具体的には学習者たちにプログラミングスタイルが不適切なものと適切なものを比較させ，理解のしやすさを比べさすことによって，学習者たちに適切なプログラミングスタイルでプログラミングする意味を改めて理解させるなどの対応を検討している．

参考文献

[1] Eclipse: https://eclipse.org
[2] 伊冨昌幸，北英彦，高瀬治彦，林照峯：コーディング状況に応じたアドバイスを可能にするプログラミング演習システムに関する研究, コンピュータ利用教育議会，2010PC カンファレンス（2010）
[3] 戸上稔崇，北英彦：プログラミング演習システム PROPEL の Java 対応とエラーメッセージ改善，コンピュータ利用教育協議会，2015PC カンファレンス（2015）
[4] 北英彦:プログラミングスタイルの収得のための自己学習機能，コンピュータ利用教育議会，2014PC カンファレンス（2014）
[5] Sun Microsystems: https://www.oracle.com/sun/
[6] Checkstyle: http://checkstyle.sourceforge.net/
[7] Artistic Style: http://astyle. sourceforge.net/