プログラミングスタイル習得のための自己学習環境

北英彦*1
Email: kita@ce.elec.mie-u.ac.jp

*1: 三重大学大学院 工学研究科 電気電子工学専攻

◎Key Words プログラミング教育，プログラミングスタイル，自己学習，演習システム

1. はじめに

情報化社会の進展に伴い，多くの IT 技術者が必要とされている。これに対応すべく，多くの大学ではプログラミング教育を行っている。しかし，十分な指導を行えているかというと，多くの課題が残されている。その一つとしてプログラムの読みやすさがあげられる[1-5]。将来，情報系の企業でプログラミングを行う際には他の人が理解しやすいよう読み易いプログラムを書く必要がある。読みやすいプログラムを書くための基本的な方法として「字下げ」「変数名を適切に付ける」「適切なコメントを付ける」などがある。これらの方法は，プログラムの見た目を改善することにより，読み易さを向上させるというものである。

本研究の目的は，手始めとして「字下げ」を対象として，学生が読みやすいプログラムを書けるようになるための自己学習環境を提供することである。

2. 学生の書いたプログラムの現状とその原因

20 行程度と短くても学生の書いたプログラムは，読みやすいプログラムを書くための基本的な方法を使用していないため，理解しにくい。プログラミング演習において学生が書いたプログラムを調査した結果を表 1 に示す。また，表 2 からプログラミング演習の科目としての成績が優秀なものでも字下げを適切に行えていないことがわかる。

表 1 学生が書いたプログラム(324 件)

字下げ不適切	ブロック記述不適切	コメントなし	変数名不適切
173	144	265	324

表 2 学生の成績(10 段階評価, 28 名)

	成績上位 (10,9)	成績中間 (8,7,6)	成績下位 (6 以下)
字下げ不適切	8 名	8 名	9 名
字下げ適切	1 名	1 名	1 名

原因としては，演習において，講師は学生のプログラムが動作するように指導することで精一杯で，読みやすさについてまで指導できていないことがあげられる。また，表 3 に示すようにプログラミング言語の教科書にもプログラムの読みやすさについて説明されているものは少ない。

表 3 C 言語の教科書における説明(12 件)

	十分な説明あり	多少説明あり	説明なし
字下げの仕方	4	4	4
変数名の付け方	0	5	7
コメントの付け方	1	2	9
波括弧{ }の付け方	1	3	8

3. 提案

著者らは，プログラミング学習の支援を目的としたシステムを開発しており，その一つに PROPEL[6]がある。このシステムはブラウザ上で稼働し，学生は C 言語を用いたプログラムの作成およびコンパイル，実行，提出などが行える。一方，講師は学生のプログラム作成経過を講師用画面から確認することができ，学習が進んでいない学生への円滑な指導を行うことができる。

本研究では，プログラミング演習システム PROPEL に，字下げを学習するための自己学習環境と，その後の普段の演習において学生が自分のプログラムの字下げをチェックできる機能を新規に実装する。

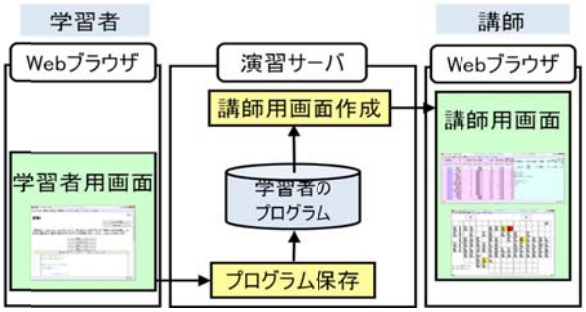

図 1 PROPEL のシステム構成図

3.1 プログラミングスタイル

読みやすいプログラムが書けるようになるには，以下にあげるような基本的なプログラミングスタイルを適切に使えるようになることが重要となる。

- 字下げを適切に行う。
- ブロックを適切に付ける。
- コメントを付ける。
- 適切な変数名を付ける。

本研究では，上記のうち最も基本的である字下げに関する学習を研究の対象とする。

なお，最近のエディタは字下げの補助を行ってくれるので，字下げの学習は意味がないように思われる。しかし実際には，そのようなエディタを使っても，表1からわかるように過半数の学生が字下げを適切に行うことができていない。

3.2 字下げの自己学習環境

自己学習を提案する理由を以下にあげる。

- 字下げの知識自体は難しいものではないため，プログラミング初心者であっても自分で理解できると考えられること。
- 字下げの習得に必要なのは演習を繰り返すことであり，演習環境を用意することで学生が字下げを習得できるようになると予想されること。
- 字下げのチェックは講師でなくてもシステムが自動的に行えること。
- 演習中，講師はプログラミングスタイルまで指導の手がまわらないこと。

字下げの自己学習環境は，4 つの段階に分けた。

(1) 字下げに関する事前知識の確認
(2) 教材を用いた字下げの知識の修得
(3) 字下げに関する実践演習
(4) 字下げに関する確認テスト

段階 1 では，学生に十数行程度の簡単なプログラムを作成してもらう。これは字下げの学習を行う前に，どの程度学生が普段字下げを行っているかを知るために用意した。

段階 2 では，学生が字下げについて詳細に説明した教材を用いて学習を行う。

段階 3 では，学生は字下げに特化した演習を行う。演習において字下げが完璧に行えるようになるまで，段階 4 には進めないよう工夫した。

段階4では，学生に確認テストとして段階1で用いた問題と全く同じ問題を解いてもらう。この段階は，学習前と学習後で字下げが適切に行えるようになったかどうか自己評価するために用意した。

3.3 段階 1:字下げに関する事前知識の確認

字下げの学習を行う前に，学生がどの程度普段から字下げを行っているかどうかを調べるために，図 2 に示す問題を課した。字下げを無意識に行うかどうかを見る課題なので，ごく簡単なプログラムで，かつ，プログラムの構成部品はあらかじめ与える。

以下のパーツを用いて、次のプログラムを完成させてください.

２つの整数値を比較し、小さい方の値を表示するプログラムを作成せよ.
※ パーツはコピーができません.

[パーツ]----------

```
int m = 2;
int n = 5;

if ( m > n )
} else {
}
printf("小さい方は m で、値は%d です。", m);
printf("小さい方は n で、値は%d です。", n);
return(0);
```

図 2 字下げに関する事前知識の確認の問題

3.4 段階 2:教材を用いた字下げの知識の修得

この段階では，教材により，字下げを行うことによりプログラムの読みやすさが向上するということを学生に感じてもらう。字下げを行う目的は，プログラムの制御構造を一目で分かりやすくすることである。このことを学生に理解してもらうため，図 3 のように字下げされていないプログラムと適切に字下げされたプログラムを対比した図などを用意した。教材は全体として 3 ページからなり丁寧に読んでも 10 分程度で読める内容となっている。

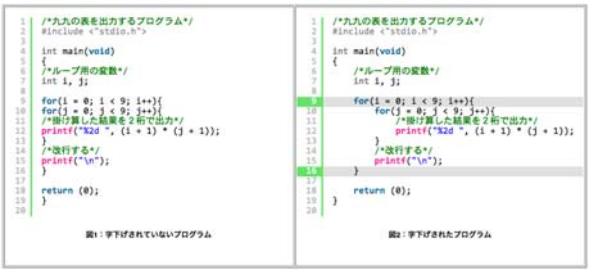

図 3 教材:字下げの有無の比較

3.5 段階 3:字下げに関する実践演習

字下げを実践できるようになるためには，字下げが行えるようになるまで演習を繰り返すことが理想である。そこで，段階 3 では段階 2 で学んだ字下げの知識をもとに字下げの演習に取り組んでもらうだけでなく，字下げが適切に行えるようになるまで何度でも字下げについて修正してもらう。

この段階では，図 4 に示す問題を学生に取り組んでもらう。図に示すようにプログラムは予め用意されているが，字下げがされていないため読みにくい。このコードを段階 2 で学んだ知識をもとに適切に字下げを行うという問題である。「字下げチェック」ボタンを押すとプログラム中の字下げをシステムがチェックし，もし字下げが不適切な箇所があれば図 5 のようなチェック結果を学生に返す。学生はこの結果を見て字下げの修正を行い，再度「字下げチェック」ボタンを押し字下げのチェックを行う。字下げが適切になるまで，学生は繰り返しこの作業を行う。字下げが適切に行えていれば，「提出」ボタンが有効になり学習過程 4 へ進めるようなる。

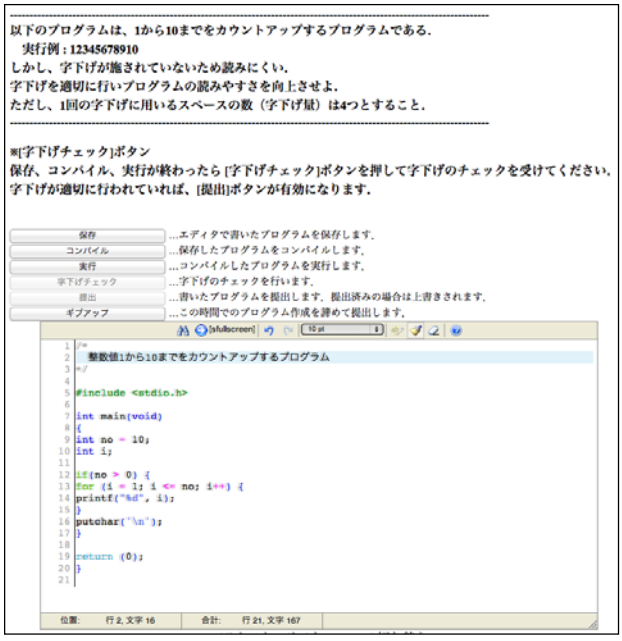

図4 字下げの実践演習

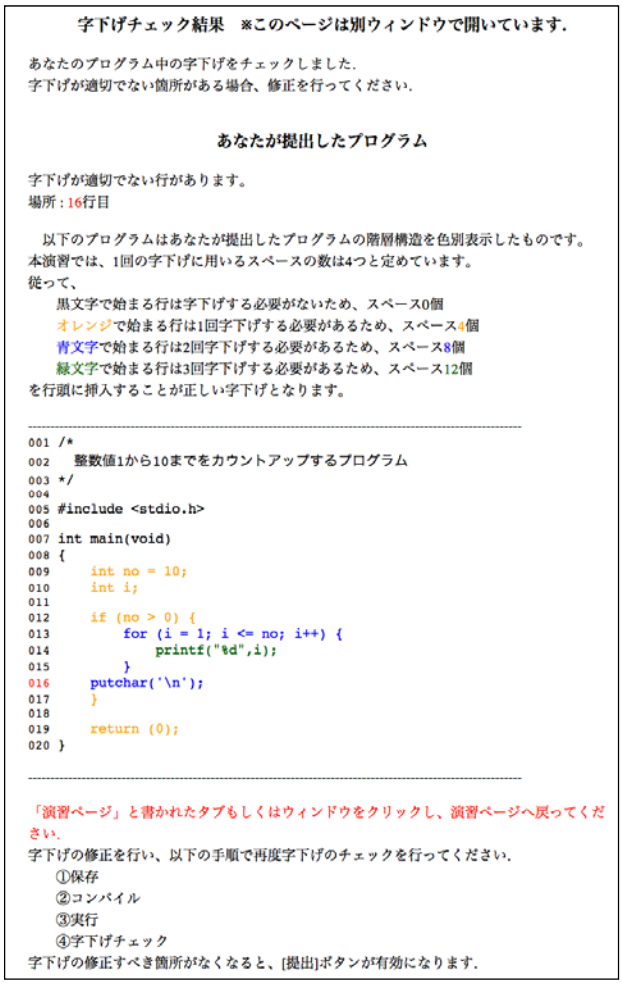

図5 字下げのチェック結果

3.6 段階4：字下げに関する確認テスト

段階4では，段階1で用いた問題と全く同じ問題を学生に課した。学生が確認テストを終わらせ，提出ボタンを押すと図6に示すような図を学生にフィードバックするようにした。この図は，事前の知識の確認（字下げ学習前）で作成したプログラムと確認テスト（字下げ学習後）で作成したプログラムを横に並べたものであり，学生に学習効果を実感してもらうためのものである。

図6 確認テストの結果

4. 運用結果と考察

自己学習により学生が字下げを適切に行えるようになるかを調査するために提案した字下げの学習のための自己学習環境の運用を行った。

- 実施期間：2013年11月12日（日）～2014年1月22日（水）
- 対象：三重大学電気電工学科2年生および3年生
- 実施者：2年生 7名　3年生 20名　計27名

4.1 学習前の字下げの状況

字下げについて学習を行う前に，学生が行った事前テストの結果を表4に示す。字下げをすべき箇所が8箇所しかない短いプログラムであっても，字下げを完璧に行った学生は27名中3名しかいなかった。

また，前述の表2のように，プログラミング演習の科目としての成績が優秀なものでも字下げを適切に行えていないことがわかった。

表 4 学習前の字下げ正答率
(字下げすべき箇所は全 8 箇所)

全て正解	間違い 2 箇所以下	全て間違い
3 名	7 名	17 名

4.2 学習後の字下げの状況

字下げについて学習を行った後に，学生が行った確認テストの結果を以下の表 5 に示す。適切に字下げを行った学生が 22 名であった。また，字下げについて学習を行った後でも字下げを適切に行えていなかった学生 5 名は，字下げを浅くする場所を理解できていなかった。字下げを浅くする場所について，教材を補充する必要がわかった。

表 5 学習後の字下げ正答率
(字下げすべき箇所は全 8 箇所)

全て正解	ほぼ正解
22 名(81%)	5 名(19%)

4.3 学習に要した時間

表 6 に，学生が自己学習を始めてから終わるまで，どの程度の時間がかかったかをまとめた。多くの学生が 20 分前後で自己学習を完了させていることがわかる。学習所要時間が長過ぎると，学習意欲が減る。運用の結果，学生のほとんどが 20 分前後で学習を終えていたため，適度な学習時間であると思われる。

表 6 学習所要時間

6 分	10-19 分	20-29 分	30 分以上
1 名	16 名	7 名	3 名

4.4 字下げチェック機能を用いた回数

段階 2 では教材を用いて学生は字下げの正しい知識について学び，段階 3 では実践的な演習により字下げを習得することを前章で説明した。段階 3 では，字下げチェック機能により学習システムが学生の字下げをチェックするが，学生がこの機能により何度字下げの訂正を行ったかを以下の表 7 にまとめる。表 7 を見ると，2 回以上字下げの訂正を行った学生がほとんどである。このことから，字下げの演習は字下げを習得するために必要であるといえる。

表 7 字下げチェック機能を使用した回数

1 回	2 回	3 回	3 回以上
1 人	10 人	10 人	6 人

4.5 演習後のアンケート結果

以下は，自己学習後に学生に回答してもらったアンケートの一部である。

設問 2:プログラムの作成中に，字下げについて，意識していましたか？

- いつも，または，ときどきしていた。(12 名，52 名)
- あまりしていなかった。(11 名 48%)

字下げを意識しているとの回答は 12 名であるが，適切に字下げが行えていたのは 3 名のみであった。この結果からも実際に字下げについて演習させた方がよいことがわかる。

自由記述では，「今回行った演習は今までの授業等で扱われていなかった内容で，良い機会になったと思う」「字下げは，プログラミング文が見やすくなり，確認が容易で，人が見て理解しやすくなる重要なことだと思う」など，字下げという基本的なことをしっかり学べたことが良かったと回答した学生が多かった。

5. まとめ

本研究では，読みやすいプログラムを書く上で最も基本的な「字下げ」について扱い，字下げを習得するための教材と演習環境を用意した。自己学習環境を運用した結果，27 名の学生が学習を行い，約 20 分前後の学習でほとんどの学生が適切に字下げを行えるようになった。このことから，自己学習であっても教材を充実させ，演習環境を整えることで学生が読みやすいプログラムの書き方を学ぶことができるということがわかった。

字下げ以外のプログラミングスタイルについては，ブロックの書式については，K&R 方式などスタイルを決めれば，システムによって自動的にチェックできるので，今回と同様なやり方で，自己学習環境を提供することが可能である。

コメントの付け方と変数名の付け方については，言葉の意味に関わる部分があるので，システムが確定的な指導を行うのは難しいが，チェック結果の判断を学習者自身にまかせることで，アドバイスはできるのではないかと考えている。

参考文献

(1) Steve McConnell : CODE COMPLETE 上 pp.317-354, 下 pp323-428, 日経 BP 社 (2012)
(2) Dustin Boswell, Trevor Foucher : The Art of Readable Code, O'REILLY (2011)
(3) Brian Kernighan, Jon Bentley, まつもと ゆきひろ他 : ビューティフルコード, O'REILLY (2008)
(4) 縣俊貴 : 良いコードを書く技術, 技術評論社 (2011)
(5) 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター : コーディング作法 C 言語版, 翔泳社 (2008)
(6) 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)