2ZC-08

プログラミング演習を対象とした教師支援の検討

石嶋　慧†　　平川　豊‡　　大関　和夫‡

†芝浦工業大学大学院理工学研究科　　‡芝浦工業大学工学部

1. はじめに

近年、IT 社会になるに連れて、必要となるシステムの規模が大きくなっている。特に日本では IT 技術者不足と言われており、教育機関でのプログラミング教育がますます重要となっている。しかし、大学などの教育機関では１つのプログラミングの授業で受講生が 100 人を超えることもある。また、プログラミングの課題の採点は、ソースコードの確認、コンパイル、実行の確認といった、普通の課題の採点と違い多くの手順が必要となる。そのため、教員の負担が大きい。

そこで本研究では、プログラミング言語 C の穴埋め問題を対象とし、ソースコードの自動抽出、自動コンパイル、構文解析を利用した自動評価を行うことで、課題評価における教員へ掛かる負担を減らすシステムを提案する。

2. 関連研究

藤原ら[1]は、ファイル表示やコマンド実行の作業を１つのウィンドウで行い、自動コンポーネントを組み込むことで教員の負担を減らす手法を提案している。

しかし、ファイルを開く作業やウィンドウの切り替えを減らしているだけであり、ソースコード採点やコンパイル・実行も手動で行わなければならない。

柳田ら[2]は、穴埋め形式の問題を作成して、問題を出題・自動採点を行い、学習履歴や回答履歴のログを収集するシステムを提案している。

だが、このシステムは教員ではなく受講生の自習を補助することが目的であり、自動採点機能はこのシステムを用いて作成された問題しか採点ができない。

[1],[2]いずれも教員の負担の軽減が不十分である。

3. 対象と事例

3.1 前提条件

本研究では、本学の情報工学科の 2 年前期で実施している、半年間のプログラミング演習の問題を扱う。

Study of teacher support for programming exercise
† Kei Ishijima, ‡Yutaka Hirakawa, ‡Kazuo Ohzeki
†Graduate School of Engineering, Shibaura Institute of Technology
‡Shibaura Institute of Technology

演習は C 言語を全 15 週の授業で学ぶものである。演習の一つ前にアルゴリズムなどの講義の授業が有り、講義で学んだことを題材に、演習で実際にプログラミングを行う。演習では穴埋め問題、指定された動作をするプログラムを自力で作成する問題などがある。本提案手法では、その中の穴埋め問題のみを対象問題とする。該当する問題数は 43 問である。

また、受講生は課題を提出する際、1 回の授業で出題された問題のソースコードと実行結果を、指定されたフォーマットと名前で１つのテキストファイルにする。そして、指定されたフォルダへ提出することとする。

3.2 事前調査

演習で提出された課題において、提出された出力結果と解答を比較した際、どれくらい完全一致するか、また完全一致しない原因について調べた。結果は表 1、表 2 の通りである。また調査件数は、表１はコンパイルができるソースコードと出力結果の両方がある 199 件、表 2 は出力結果のみを提出してあるものも含めた 215 件である。

表１　受講生と解答の出力結果の完全一致

事例　（199 件）	件数
完全一致	13
一致しなかった	186

表２　出力結果の事例

事例　（215 件）	件数
スペースの有無	105
改行の有無	13
文字の過不足	29
出力結果の途中がない	3
出力の誤り	25
別の文字へ変わっている	5
メモリ番地の値が違う	80
手入力が指示よりも多い	1

この結果から、提出された出力結果と解答の出力結果の完全一致率は、6.5%であった。そして。一致しない原因は print 文に揺れが多いあるためだとわかった。そのため、print 文の揺れを吸収するなどの対処が必要である。

4. 提案手法

上で述べた print 文の揺れは、多種のバリエーションがあり、個別対処には困難が予想される。そこで、提出ファイルの print 文を教員が準備した正しいファイルの print 文で置き換える手法を提案する。また、教員の負担を少しでも減らすため、提出されたソースコードの自動抽出、自動コンパイルも行う。

構文解析は関数名、print 文、{}などを元に、図 1 のようにブロック構造を意識して解析をする。print 文の出現パターンは 3 つあり、①のときは無条件削除、②のときは無条件上書き、③のときは print 文同士の類似度判定を行い、類似度の低い方を削除し、高い方に解答を上書きする。

なお、C 言語の if 文は if 文内 1 行しかない場合は{}を入れなくてもよいという規則がある。これにより、受講生と解答のソースコードによって{}があるものと無いものがあるため、正確に解析できないことがある。そこで、if 文は全て{}が入るよう前処理をした。

処理は、ソースコードの自動抽出、自動コンパイル、構文解析・置換、置換されたソースコードのコンパイル・実行、出力結果のマッチングの順番で行っている。構文解析以降は、コンパイルができたもののみを対象としている。

これらより、教員の評価における負担を減らす。

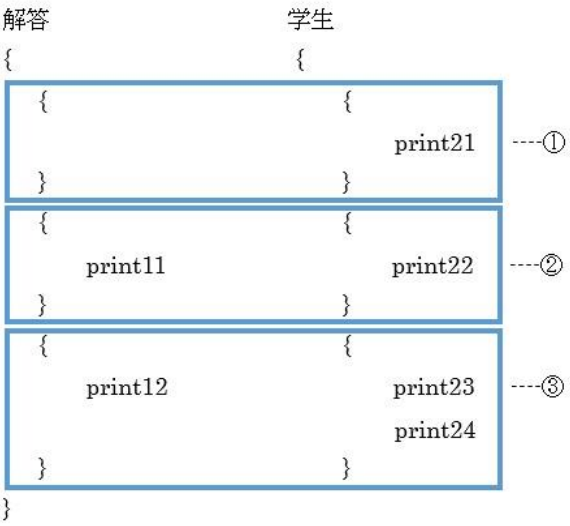

図 1　print 文のパターン

5. 評価実験

評価対象は事前実験と同様、穴埋め問題 43 問、実験件数は事前調査と同じ 215 件をシステムで採点した。結果は以下の表の通りである。

表 3　コンパイルの評価

事例	件数
コンパイルできない (ソース無　出力結果有)	13
コンパイルできない (ソースコードの途中部分がない)	3
コンパイルができる	199

表 4　置換後のマッチング結果

事例 (199 件)	件数
完全一致	184
一致しなかった	15

一致しなかった原因	件数
提出されたソースコードのミス	9
printf の類似度判定のミス	8

表 3 は、ソースファイルをそのままコンパイルした結果である。ソースコードがないものだけでなく、ソースコードの途中部分がないものも自動判定ができている。

置換後の出力結果のマッチングは、コンパイルが成功したもののみを対象とした。結果は約 92.5%が完全一致し、正確に判定できた割合は約 96.0%であった。置換前よりも約 89.5%一致率が上昇した。類似度判定のミスは、print 文の中身が例えば“アルゴリズム 4.3”となるはずのものが、“4.3”と大幅に省略されているものや全く別の文字列になっているものがあり、これらを類似度判定では置換できなかったためである。

6. まとめと今後の課題

本研究では、構文解析を用いてソースコードの print 文の置換を行い、テキストの自動抽出、自動コンパイル・実行、受講生の出力結果と解答出力結果の完全一致をして自動評価を行った。結果、穴埋め問題の約 96.0%を正確に判定でき、置換前よりも約 89.5%判定率が向上した。

今後は、穴埋め以外の問題にも対応させる、コンパイルにかかる時間の短縮などが課題に挙げられる。

参考文献

[1]藤原巧，松浦佐江子，“評価方法に従った自動採点を可能にするプログラム採点支援ツールの開発(情報教育と授業支援システム/一般)”，電子情報通信学会技術研究報告．ET，vol. 105(298)，pp. 23-28，2014 年 09 月 10 日

[2] 柳田崚，太田康介，大月美佳，掛下哲郎，“穴埋め問題を用いたプログラミング教育支援ツール pgtracer の運用実験”，情報教育シンポジウム 2014 論文集，vol. 2014，no. 02，pp. 135-142，2014 年 08 月 17 日