# プログラムのソースコードを解答とする問題の自動採点

四方 雅晴[*1]・北 英彦[*1]
Email: 416m242@m.mie-u.ac.jp

*1: 三重大学大学院工学研究科電気電子専攻

◎Key Words e ラーニング、プログラム作成問題、自動採点

## 1. はじめに

e ラーニングシステムは多くの大学で導入されており、プログラミング科目にも導入されている。有用な機能のひとつに小テスト機能がある。従来の小テストでは、一般に多肢選択か文字列として一致するものしか自動採点できず、ソースコードのように解答の表現の自由度の高い記述式問題の解答を自動採点することは困難である。

著者らの所属する研究室の伊藤はソースコードを解答する問題に対して以下のような自動採点システムを考案した[1]。問題はソースコードの一部を記述する穴埋め問題とする。採点項目は、講師が採点のときに確認していると思われる、構文エラーの有無、動作の正しさ、などとする。この自動採点システムを実装し授業で使用した結果、学生に数の多くのプログラム作成の問題に取り組ませることができ、その一方、講師の負担は増えないということを確認した。解決すべき課題としては、構文エラーのあるものは、動作の正しさを判定できないため 0 点になってしまうことがあげられる。人間である講師が手動で採点する場合、軽微な構文エラーに関してはその部分が正しいものとして採点できるのに対して、初心者に対しては厳しすぎる採点になっている可能性がある。

本研究は、軽微な構文エラーはシステムが自動的に修正を行い、コンパイル可能で実行できるものに対しては動作の正しさをチェックし点数を与えることができるようにすることを目的とする。これによって、講師が手動で行う場合に近い採点が自動採点でできるようになる。

## 2. 従来の自動採点システム

伊藤の作成した自動採点システムについて説明する。三重大学電気電子工学科では Java を用いたプログラミングを学んでいるので対象は Java で書かれたソースコードとする。ソースコードを解答とする小テストの目的のひとつは学習者に数多くのコードを書かせることにあるので、ソースコード全体を解答させるのではなく、小テストの問題の出題意図であるコアの部分のみを穴埋め問題形式で解答させる。

### 2.1 採点項目

以下の 4 つの項目に関して自動採点を行う。括弧内は配点の標準の割合である。

- 構文エラーの有無（30%）
- 動作の正しさ（60%）
- 課題で指定された機能の使用（上記 60%中 10%）
- プログラミングスタイルの適切さ（10%）

### 2.2 運用結果

伊藤の自動採点システムを用いた小テストを 2015 年度の三重大学工学部電気電子工学科のプログラミング演習の受講者 88 名を対象に実施した。得点分布を図 1 に示す。結果は二極化した。原因は、コンパイル不可能な場合は動作の正しさを判定できないためである。例えば、現状の採点では文末につける「;」をある 1 行で付け忘れただけで 90%減点される。

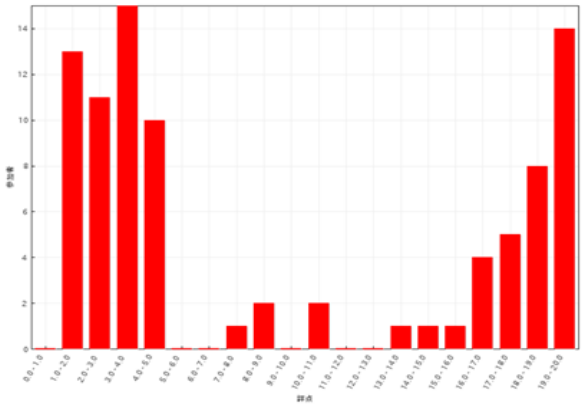

図 1 得点分布

## 3. 解答の分析

2016 年度のプログラミング演習の受講者 78 名の解答を構文エラーに関して分析した結果を表 1 と表 2 に示す。ソースコードを解答とする小テストの解答を分析した結果、表 1 のように構文エラーを含む解答が約 4 割あり、表 2 に示すように文末の「;」を 1 文字忘れただけのような軽微なものはそのうちの約 3 割であった。構文エラーの重要度は人によって意見が分かれるところであると思うが、本研究では何を書こうとして間違えたのかがその部分のみで分かるようなものを軽微なものとして扱うことにした。これは、講師が手動で採点するときに、コードを丁寧に読まなくても判断できる場合に誤りを許容して採点を継続するであろうと考えたからである。

## 4. 自動採点の改良

本研究では、以下のような方法で軽微な構文エラーはシステムがそれらを自動的に修正する。なお、自動的に修正を行った場合は、適当な減点をすることを考えている。修正の結果、解答であるソースコードがコンパイル可能で実行できれば動作の正しさの採点、および、プログラミングスタイルの適切さの採点を行う。

表 1　2016 年度小テスト結果

| 解答数 | 構文エラーを含むもの |
|---|---|
| 498 | 212 |

表 2　構文エラーの内訳

| エラーの種類 | 件数 | 重要度 |
|---|---|---|
| 文末の「;」忘れ | 31 | 軽 |
| 変数名の綴り間違い | 21 | 軽 |
| print 文の書式の間違い | 16 | |
| 全角文字の混入 | 11 | 軽 |
| その他 | 133 | |

- 全角文字の混入

  ソースコードの文字列中以外に 1 箇所でも全角文字、特に、全角空白が含まれていると構文エラーになる。小テスト中は、コンパイラを構文チェッカーとして利用できないため全角文字、特に、全角空白を見落とす者がいると思われる。文字列中以外の全角文字は対応する半角文字がある場合には半角文字にシステムが自動的に変換することで、全角文字の混入を除去する。
- 文末の「;」忘れ

  Java などのいくつかのプログラミング言語では、文の最後に「;」をつけることで文の終わりを示す。「;」をコード中でひとつでも忘れれば構文エラーとなり実行できなくなる。コード中のある行が「;」を付けるべき文であるかどうかは文法規則から比較的に簡単に分かるので、文末の「;」を付け忘れていた場合は、システムで自動的に補完し、文末の「;」忘れをなくす。
- 変数名などの綴り間違い

  変数は宣言した段階での綴りが変数名となり、それ以降で使用する際は同じ綴りでなければ構文エラーが発生する。以前の解答を調べた結果では、綴り間違いの種類としては「文字の置き換わり」「文字が多い」「文字の抜け落ち」があった。

  綴り間違いの基準は文字列の類似度を調べる Levenshtein 距離[3]を採用して、1 文字までの間違いを許容することにする。Levenshtein 距離における操作は「挿入」「削除」「置換」「転置」の 4 種類で、これらの操作を 1 度のみ行い修正することができれば綴り間違い、すわなち、軽度の構文エラーとして扱うことにする。

  綴り間違いと判断した場合いは、システムが自動でそれを正しいものに修正し、綴り間違いをなくす。

以上の変更により、軽微な構文エラーで解答全体の点数が 0 点となることを避けることができる。これによって，講師が手動で採点したときの評価に近づけることができると考えられる。

## 5.　まとめ

e ラーニングシステムの小テストにおいて、ソースコードのような表現の自由度の高いものを自動採点することは困難である。伊藤は、講師の手動採点に近い自動採点システムの構築を試み、プログラミング科目で実際に使用することのできる可能性を示した。課題としては、軽微ではあっても構文エラーを含む解答は 0 点になることであった。

本研究は、軽微な構文エラーに関してはシステムが自動的に修正し、その後の動作の正しさなどの採点ができるようにした。これによって、講師の手動による採点により近づき、小テストの問題で意図している部分の理解度を測ることができるようになると思われる。

## 参考文献


(1) 伊藤隼人、北英彦、: e ラーニングシステムの小テストにおけるソースコードを解答とする問題の自動採点、コンピュータ利用教育協議会、CIEC 春季研究会 2016（2016）

(2) 中山清橋、国本大悟:“スッキリわかる Java 入門”pp.60-96、株式会社インプレス（2016）.

(3) Guy Rutenberg: https://www.guyrutenberg.com/2008/12/15/damerau-levenshtein-distance-in-python/（参照 2017 年 5 月）