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

伊藤隼人*1・北英彦*1
Email: hitou@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 e ラーニング，プログラム作成問題，自動採点

1. はじめに

近年，多くの大学で，動画を含む教材の提供，学生と講師や学生間でのコミュニケーション，自己学習機会の提供などを目的として，e ラーニングシステムが導入されている。その中の有用なもののひとつにオンラインテストがある。

従来のオンラインテストでは，一般に，多肢選択か文字列として完全に一致するものしか自動採点することはできず，解答の表現の自由が高い記述式問題における学生の解答を自動採点することは困難であった。したがって，講師は学生の記述式解答を一つずつ見て手動で採点することしかできなかった。

また，プログラミング科目にも e ラーニングシステムが導入されている。しかし，学生の解答となるソースコードの扱われ方は，回収を除いては，紙に解答を記述して提出する場合とそれほど大きな違いはない。

本研究では，プログラミング科目におけるソースコードを解答とする問題を対象として，学生の解答となるソースコードを自動採点する方法を検討する。具体的には，プログラミングスタイルのうち字下げの適切さ，構文エラーの有無，動作の正しさのチェックを行う。動作の正しさに関しては，ソフトウェア開発で用いるテストではなく，本研究で提案するプログラミング初心者向きに出力に関する条件を緩めたテストを用いる。

2. 研究背景

プログラミング科目において，e ラーニングシステムを導入することによって，学習の効率を上げるという効果が期待できる。実際に，三重大学工学部電気電子工学科のプログラミング科目では，e ラーニングシステムの 1 つである Moodle[1]が利用されている。しかし，プログラミング科目に e ラーニングシステムを導入する時にいくつかの問題がある。

2.1. 現状

プログラミング科目における e ラーニングシステムの使われ方は，演習としてプログラム作成問題を出して学生にプログラムを書かせ，図 1 に示すように，その解答をファイル形式で提出してもらい，それを講師が一つ一つ手動で採点するというものである。これでは，学生はインターネットが繋がる環境であればいつでも提出できる，解答であるソースコードを講師がコンパイラにかけやすいなどといった利点はあるものの，紙にソースコードを記述して提出する場合とそれほど大きな違いはない。

また，このような演習形式では，学生がプログラム作成問題を数多くこなすことができないという欠点がある。プログラミング技術の向上には，様々なプログラムを数多く作成させることが重要である。

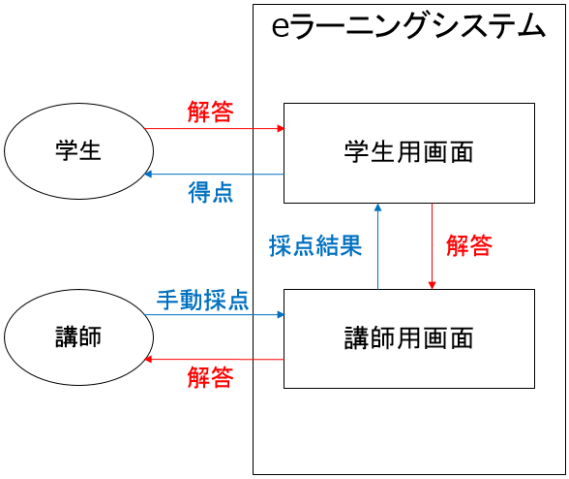


図 1　ソースコードの採点（現状）

2.2. 小テスト機能の活用

三重大学で導入されている Moodle をはじめ，Blackboard Learn[2], Maple T.A.[3]などの e ラーニングシステム（または，LMS: Learning Management System）と呼ばれるものには，小テスト機能が存在する。その中で多肢選択問題，短答記述式問題，作文問題（エッセイ問題）など，様々な形式の問題を作成することができる。この小テスト機能を活用すれば，学生にプログラム作成問題を数多くこなしてもらうことができる。プログラム作成問題においては，学生にソースコード全体，またはソースコードのコアとなる部分のみを解答させるために，作文形式が有用である。

しかし，表現の自由度が高い作文形式の問題における解答は，多肢選択形式，短答記述形式の問題における解答とは違って，自動で採点することができない。これでは，図 1 と変わらず，なおかつ採点しなければならない学生の解答であるソースコードの数も増加してしまい，回収を除いては，講師の負担が非常に大きくなってしまう。

2.3. 数式解答評価システム STACK の利用

STACK（System for Teaching and Assessment using Computer algebra Kernel）[4]とは，Moodle と連携した数式解答評価システムであり，学生の数式による解答を受け付け，正誤評価・自動採点をするシステムである。ソースコードと数式には，形が一致している部分や似ている部分があることから，STACK をプログラミング演習に利用することについて研究した[5]。しかし，数式と同じ形である条件式や代入文などしか扱うことができず，図 2 のように，非常に限られた範囲で，短答記述形式の問題しか作成することができなかった。また，条件式や代入文に関しても，自然な形では扱えない場合がある。例えば，「tmp = a」という代入文は，STACK の内部では，代入文ではなく等式として扱われる。そのため，講師が慣れるまでに時間がかかると考えた。

次のソースコードの穴を埋め、配列arrayに1~10の数値を格納し、表示するプログラムを完成させよ。

```
#include <stdio.h>
#define MAX 10

int main(void)
{
   int i = 0;
   int array[MAX];

   while(                ) {
      array[i] = i + 1;
      printf("array[%d] = %d\n", i, array[i]);
      i++;
   }

   return (0);
}
```

図 2　STACK で作成できる問題の例（C 言語）

2.4. ソースコードの自動採点

本研究では，採点における講師の負担を軽減することを目的として，ソースコードの採点をどこまで自動化できるかについて検討する。また，実際にそれらの自動採点を行うシステムを考案する。

以下のプログラムを埋めて，2つの抵抗R1=6.0[Ω]，R2=4.0[Ω]が，直列に繋がれている場合，並列に繋がれている場合それぞれの合成抵抗値（rSerial，rParallel）[Ω]を表示するプログラムを作成しなさい．ただし，解答スペースは既に2レベルの字下げがされているものとする．

```
public class Main {
    public static void main(String[] args) {
        double r1 = 6.0;
        double r2 = 4.0;
        double rSerial, rParallel;

        ここを埋めてください

    }
}
```

図 3　コアとなる部分のみを解答させる問題の例

対象となる言語は Java，C 言語とし，問題はプログラム全体を解答させる問題，図 3 に示すような，ソースコードのコアとなる部分のみを解答させる問題を想定する。

3. 採点基準

本研究を進めるにあたって，プログラミング科目を受け持つ講師が，学生の解答を採点する際に，どのような点を見て採点するのかを検討する必要がある。検討の結果，以下のような項目を見れば，適切な採点ができると考えた。

- 構文エラーの有無
  コンパイラにかけてもエラーが出ないかどうか。
- 問題の目的通りの動作をするか
  期待される出力がされるかどうか。
- プログラミングスタイル
  読みやすいソースコードになっているかどうか。
  - 字下げ
    適切なレベル・量で字下げがされているかどうか。
  - 名前の書き方・付け方
    - 変数名，関数名などが適切な書き方（キャメルケース，スネークケースなど）がされているかどうか。
    - その変数・関数などが何を表すかがわかる単語が使われているかどうか。
  - コメントの書き方
    - コメントを付けている場所が適切かどうか。
    - コメントの内容が適切かどうか。
- 効率
  効率のいいプログラムになっているかどうか。
- その他
  プログラムの動作に不要なコードが入っていないかどうか。

4. 自動採点

3 章で紹介した採点基準の中で，自動で採点できそうな項目を検討して，以下の 3 つの項目に着目し，自動採する方法を考えた。効率に関しては自動採点が困難であるため，今後の課題とする。

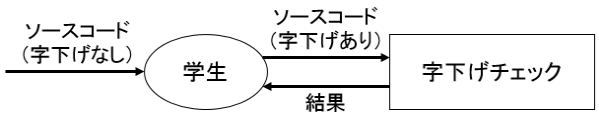
4.1. 字下げの適切さ

字下げのレベル・量が適切にされているかをチェックするプログラムを使う。また，変数と演算子の間の空白についてもチェックする。ただし，字下げ・空白のルールは，株式会社永和システムマネジメントの「Java コーディング標準」[5]に従って書けているかをチェックする。C 言語も字下げ・空白に関しては，基本的に同じルールであるため，これを基準に評価する。

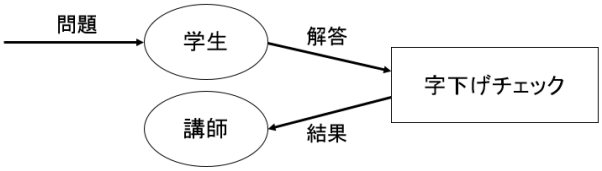
この字下げチェックプログラムは，私の所属する研究室の先行研究[6]を参考にして，新しく作成する。この先行研究では，学生にあらかじめ全く字下げがされていないソースコードを与え，そのソースコードを字下

げしてもらい，間違っている箇所があればその行と，どう間違っているのかを色を使って示すプログラムが提案されている。これは，学生が適切に字下げをできるようになるための演習用に作られたものであり，図 4(a)に示すように，結果は学生に返される。本研究では，演習用ではなく小テスト用として，図 4(b)に示すように，学生の解答ソースコードを対象に字下げ・空白のチェックを行い，その結果を講師に返すプログラムを作成する。

プログラミングスタイルのうち，名前の書き方・付け方，コメントの書き方に関しては自動採点することは困難であるため，今後の課題とする。



(a)　先行研究



(b)　本研究

図 4　字下げチェックプログラム

4.2. 構文エラーの有無

構文エラーの有無は，コンパイラを呼び出して学生の解答ソースコードをコンパイルすることによってチェックする。コンパイラは，ソースコードに構文エラーがあれば，エラーメッセージを出力する。

4.3. 動作の正しさ

各問題にいくつかのテストケースを講師が用意し，そのテストケースを満たすかどうかをテストするブラックボックステストを用いる。

ただし，ソフトウェア開発で用いるテストではなく，5 章で説明するプログラミング初心者向きに出力に関する条件を緩めたテスト[8]を用いる。

5.　自動テスト方法

テストを自動化するにあたって，プログラミング初心者の場合には，テストケースの出力値と一字一句一致するプログラムを作成することは難しいという問題点がある。例えば，図 5 のような，標準体重表示プログラムを作成させる問題では，出力の「kg」を全角文字にしたり，「：」記号の前や後に余計な空白や改行を入れたりする。従来の自動テストでは，このような細かい違いによってエラーと判定されてしまう。この問題点を解決する手段としては，問題の中で出力の形式を厳密に指定するか，出力の形式の条件を緩めるかという方法が考えられる。

望月は，文献(8)の中で，プログラミングの初心者にも課題の目的を達成するプログラムを作成できるように，出力の形式の指定を緩めたテストを提案している。具体的には，出力全体ではなく，出力に必ず含まれるべき文字列のみを指定している。本研究では，このテスト方法を利用する。

一般には，テストケースとは，テストのための入力と，プログラムが正しく動作した時の出力の組のことである。しかし，ここでは，テストのための入力と，出力に含まれるべき文字列の組をテストケースと定義する。また，問題の目的以外の細かい点は，間違いの検出の対象としないこととする。間違い検出の対象としない項目は，以下のものとする。

- 全角・半角の違い
- 空白の有無
- 句読点（またはコンマ，ピリオド）の有無
- 改行の有無

図 5 の問題で考えると，身長が 175cm の場合，標準体重は 67.5kg である。すなわち，175 と入力したとき，出力に含まれるべき文字列は「67.5kg」である。よって，入力値 175 と文字列「67.5kg」がこの問題のテストケースの一つとして設定できる。

以下に示すように，身長を整数値として読み込み，標準体重を実数で表示するプログラムを作成しなさい。標準体重は，（身長 － 100）× 0.9 によって求めること。小数点以下は 1 桁のみ表示すること。

身長を入力してください。：175
標準体重は 67.5kg です。

図 5　標準体重表示プログラム作成問題

6.　配点

「プログラミング演習Ⅰ」を受講している学生に，Moodle 上で教科書[9]を参考にして作成した，図 6 に示すようなプログラム作成問題を解いてもらい，その採点を通して自動採点を行う項目の配点比率についても検討した。以下にその検討結果を示す。

- 字下げ・空白が適切である（10%）
  一箇所でも不適切な字下げ・空白があれば 0%
- 構文エラーがない（30%）
  コンパイルできなければ 0%
- 正しい動作をしている（60%）
  60%－（動作エラー数／テストケース数）×60%

以降それぞれを字下げ点，構文点，動作点と呼ぶ。

以下のプログラムを埋めて，キーボードから製品の名前と税抜き価格，税率（小数）を読み込み，その製品の税込価格（小数点以下切捨て）を表示するプログラムを作成しなさい．ただし，解答スペースは既に2レベルの字下げがされているものとする．

```
public class Main {
    public static void main(String[] args) {
        ここを埋めてください
        priceAfterTax = (int)(priceBeforeTax * (1 + taxRate));
        System.out.println(productName + "の税込価格は" + priceAfterTax + "円です。");
    }
}
```

図 6　プログラム作成問題の一例

この配点比率の問題点は，構文エラーがあり，コンパイル不可能な場合，テストもできないので動作点も0%となってしまうことである。例えば，プログラムの文末につける“;（セミコロン）”をどこか1箇所つけ忘れただけで，90%の減点になってしまう。プログラミングの世界はシビアであるためそれでもいいかもしれないが，厳しすぎるかもしれない。

7. 自動採点システム

実際に 4 章で述べたような項目について自動採点を行うシステムの構想について述べる。フィルターのような形で，e ラーニングシステムの外部モジュールとして作成し，導入することにより，図 7 に示すように，従来（図 1）と比べて，講師の採点における負担を大きく軽減させることが期待できる。

この自動採点システムが受け取るものと渡すものを以下と図 8 に示す。

- 受け取るもの
  - ➢ 学生の解答部分を含むソースコード
  - ➢ テストケース
  - ➢ オプション
    字下げ，構文，動作全てのチェックを行わなくていいときに，このオプションでチェック機能の取り外しを可能にする。
- 渡すもの
  （構文エラーの有無に関わらず）
  - ➢ 学生の解答部分を含むソースコード
  - ➢ 字下げチェックの結果

  （構文エラーがあった場合）
  - ➢ エラーメッセージ

  （構文エラーがなかった場合）
  - ➢ テストケース
  - ➢ 自動テストの結果

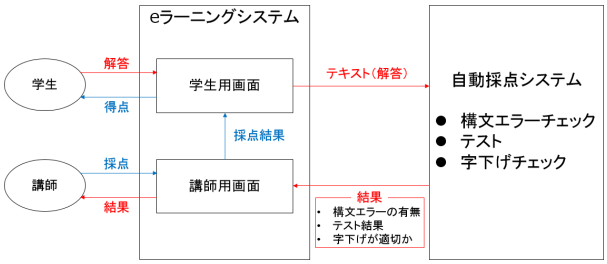


図 7　自動採点システムを使った採点

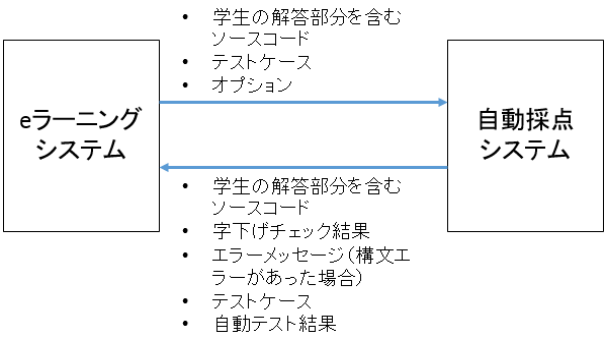


図 8　自動採点システム概念図

8. 今後の課題

今後の課題として，本研究のシステムの有用性の検証を進めていきたい。また，プログラミングスタイルのうち，名前の書き方・形式の自動採点についての研究を進めていく予定である。

名前の書き方に関しては，Java の場合はキャメルケース（例 : camelCase），C 言語の場合はスネークケース（例 : snake_case）が一般的であるため，その法則に従って自動採点することが可能であると考えられる。名前の付け方に関しては，課題の意図を汲み取る，自然言語処理が必要となる。

9. まとめ

e ラーニングシステムのオンラインテスト上で，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非常に困難である。それゆえ，プログラミング科目を受け持つ講師は，学生の解答ソースコードを採点する際に，大きな負担がかかっていた。実際に配点について検討するために，採点作業を行ったため，そのことがよくわかった。

本研究で述べたような自動採点システムを完成させることができれば，採点における講師の負担を大きく軽減させることが期待できる。

参考文献

(1) Moodle.org, https://moodle.org/（2015 年 6 月参照）
(2) Blackboard：Blackboard Learn, http://www.blackboard.jp/platforms/learn/（2015 年 6 月参照）
(3) Maplesoft，サイバネットシステム株式会社 : Maple T.A., http://www.cybernet.co.jp/maple/product/maple_ta/（2015 年 6 月参照）
(4) STACK, http://stack.bham.ac.uk/（2015 年 6 月参照）
(5) 伊藤隼人，北英彦 : “数式解答評価システム STACK を利用したプログラミング演習用問題”，コンピュータ利用教育議会，2014PC カンファレンス（2014）
(6) （株）永和システムマネジメント，平鍋健児 : “Java コーディング標準”, http://www.objectclub.jp/community/codingstandard/CodingStd.doc.pdf
(7) 伊藤雅人 : “プログラミングスタイルの習得のための自己学習環境に関する研究”，三重大学工学研究科修士論文（2014）
(8) 望月将行 : “プログラミング初心者のための自動テスト機能を備えた演習支援システムに関する研究”，三重大学工学研究科修士論文（2004）
(9) 中山清喬，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（2014）