プログラミングの授業における課題の自動採点

関　勝寿*

Automatic scoring system for programming assignment

Katsutoshi Seki*

Abstract

Automatic scoring system was developed for the use in scoring the assignments of a programming course. Common mistakes that students make are categorized by output of the program with test input, where feedback comments for each type of mistakes can be written in configuration file. The score and comments are assigned automatically by running this program. The teacher should still read the submitted code carefully and write feedback comments individually. As the numbers of the programs which need special attention are small compared to the whole numbers of submitted programs, the teacher can concentrate on writing such individual comments.

Keywords：programming, scoring

要旨：プログラミングの課題に対する採点を省力化するために、課題を自動採点するプログラムを作成した。学生のミスをテスト入力に対する出力によってパターン化して、ミスに応じたコメントを設定ファイルに記述することができる。このようにして、自動的に採点と学生へのフィードバックコメントを返すことができる。パターン化されていないミスに対しては、個々のプログラムを確認してエラーの原因を特定してコメントを書く必要があるが、その作業が省力化でき、大人数の授業であっても学生に効率的に詳しく個別のフィードバックをすることが可能となった。

1. 序論

2020年度からすべての小学校においてプログラミング教育が必修化された。これからの

*）東洋大学自然科学研究室 〒112-8606 東京都文京区白山 5-28-20
Natural Science Laboratory, Toyo University, 5-28-20 Hakusan, Bunkyo-ku, Tokyo 112-8606, Japan

時代を生きていくためには、情報技術を効果的に活用する力をつけることが必要であり、プログラミング的思考を身につけることの重要性が認識されているためである。著者が所属する東洋大学経営学部では、必ずしも情報の専門教育を施すことを目的としている学部ではないものの、教育・研究においてプログラミングを活用する場面は増えており、学生もプログラミングに対する関心が高まっている。そのことを著者が改めて認識したのは、2009年度から担当しているJavaのプログラミングの授業において、例年は10名程度の履修者であったものが、2019年度に授業の名称を「情報処理実習D」から「プログラミング実習講義」へと変えてプログラミングを表に出したことで、履修希望者数が299名へと急激に増えたときであった。

この授業は、90分の授業の前半でその回のテーマを説明し、課題を提示して、後半では学生が課題に取り組む、という形式としているが、2018年度までは毎年10名程度の学生を相手にしていたため、後半で学生一人ひとりのパソコンの画面を見ながら、つまづいている箇所があれば個別指導をして、授業内に課題を完成させることも可能であった。そして、課題の採点については、提出されたプログラムを１つ１つ教員のPCで実行して、動かない場合には何が原因で動かないのかを確認して、コメントとともに採点結果を返すことが可能であった。ところが、2019年度には299名の履修希望者があり、教室の制限から118名の履修者を受け入れることとなったものの、それまで同様の指導方法を継続することが困難となった。とりわけ、課題の採点については提出された課題のプログラムを１つ１つコンパイル、実行してその結果を確認しながら採点とコメントを返すことは非効率的であり、その作業をある程度自動化できないかと着想した。

プログラミングの課題を自動的に採点することは、競技プログラミングにおいて広く実施されている。競技プログラミングとは、プログラミングの能力や技術を競うプログラミングコンテストの競技の１つであり、参加者全員に出題された課題に対して、それぞれの参加者はその要求を満たすようなプログラムを作成して提出し、そのプログラムの完成度（実行速度など）を競うものである。国内ではAtCoderというサービスによって頻繁に開催されているプログラミングコンテストが有名である。そこでは、コンテストで出題された問題に対して参加者がプログラムを提出すると、自動的にそのプログラムでいくつかのテスト（ある入力に対して、期待通りの出力が得られるかどうかを実際にプログラムを実行して確認すること）が実行されて、プログラムが課題で要求された要件を満たしているかどうかが判定され、さらにプログラミングの実行速度が計測されて、採点される。このようなテスト入力と出力による自動採点システムを構築することで、大人数の学生の課題の採点を効率的に実施できるのではないかと考えた。

そこで本論文では、2019年秋学期の「プログラミング実習講義」の授業において、この授業のために構築した自動採点システムを使って採点を効率化した事例を紹介する。

2. 自動採点システム

2.1. 課題の提出と採点

学生は、授業で指示された通りに、Java言語でプログラムを作成して提出する。たとえばBMI計算プログラムを作成することが課題だったとする。ここで、BMIとは身長と体重から計算される肥満度を示す指数であり、体重（kg）÷身長（m）^{2}と計算される。身長（cm）と体重（kg）を入力すると、BMIが計算されて表示されるプログラムを作成する課題に対しては、計算式は10000×体重÷身長^{2}となり（身長の単位がcmのため10000をかける）、図１のようなプログラム（ファイル名はBMI.java）がこの課題の正答例となる。

```
import java.util.Scanner;
class BMI {
    public static void main(final String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        float a = scanner.nextFloat();
        float b = scanner.nextFloat();
        scanner.close();
        float c = 10000 * b / (a * a);
        System.out.println(c);
    }
}
```

図１．BMI計算プログラム BMI.java

学生はこのプログラムを自ら作成し、コンパイルして実行する。Java開発環境がインストールされているPC教室のパソコンで、PowerShellのようなターミナルでBMI.javaを保存したフォルダ（ディレクトリ）に入って、

javac BMI.java

とすることでコンパイルされてBMI.classという名前のクラスファイルが作成され、

java BMI

と実行することで、プログラムが実行される。このプログラムは標準入力から身長と体重の値を受け取るため、身長（cm）と体重（kg）を入力すると、BMIが計算されて表示される。たとえば、身長が170 cmで体重が60 kgの場合は、「170 60」と入力することで「20.761246」のようにBMIが表示される。このように正しく実行結果が得られたことを学生が自ら確認してから、ソースファイルBMI.javaを課題ファイルとして提出するように指示をしている。

課題ファイルは、ASAHIネットが開発して東洋大学が導入している講義支援システムmanabaの「ファイル送信レポート」から提出される。複数のファイルを提出することも可能であり、複数の課題がある場合にはその機能を利用する。

課題の提出期限が過ぎてから、教員は課題をmanabaからZip形式の圧縮ファイルとして

ダウンロードする。圧縮ファイルを展開すると、学生ごとに学籍番号に基づいたIDの名前のフォルダが割り当てられ、そのフォルダ内に提出されたファイルが保存されている。また、その圧縮ファイル内には成績登録用のMicrosoft Excel形式のファイルがあり、その成績登録ファイルに学生ごとの点数と講評を書き入れ、manabaに登録することで学生に成績評価（点数）とコメントをフィードバックできる仕組みとなっている。

2.2. 自動採点プログラムの処理の流れ

自動採点プログラムはPythonで作成した。教員は、自動採点プログラムを実行するディレクトリの直下に、レポートの課題ごとに、manabaからダウンロードした圧縮ファイルを１つのディレクトリとして展開して配置する。ディレクトリにはその課題の採点方法に関する設定ファイルを配置し、設定を記述する。プログラムを実行すると、カレントディレクトリの直下で設定ファイルが存在するディレクトリすべてに対して、図２のような処理の流れで採点を実行する。ここで、設定ファイル内には採点を終了したかどうかを書き込むことができるため、常に未採点の課題のみ採点処理を実行することとなる。

提出されたファイルは、まずはファイル名が検査される。ファイル名が誤っていれば不合格の点数がつけられ、ファイル名が誤っている旨のコメントが付与される。典型的なファイル名のミスは、ファイルに.txtという拡張子が付与されているもの、ソースファイルを提出するように指示しているところ、クラスファイルを提出しているものであり、それぞれそのミスに応じたコメントが付与される。ファイル名が正しければ、コンパイルされる。この時点で、コンパイルエラーが出れば不合格の点数がつけられ、コンパイルエラーが出ている旨のコメントが付与される。たとえば、エラーメッセージの中に「この文字は、エンコーディングUTF8にマップできません」という文字列が含まれていれば、「エラーメッセージに書かれているように、プログラムの中に不正な文字が含まれています。これは、たとえば全角のスペースのような非ASCII文字です。授業で説明したように、エラーメッセージからエラーが出ている場所と内容を確認して、プログラムを修正して、プログラムが動くことを確認してから提出するようにしてください。」といったようなメッセージをあらかじめ作成して、そのようなコメントを一律に付与することが可能である。

コンパイルが通ると、その次は設定ファイルに記述されているテスト入力を与えてプログラムを実行し、その出力を得る。その際に実行時エラーが出た場合には、実行時エラーのメッセージに応じて、設定ファイルに記述されている点数とコメントが付与される。エラーが出なかった場合には、テスト入力に対する期待される出力（正しいプログラムが提出された場合に得られる出力）と一致するかどうかを判定し、一致する場合には完全回答として満点が付与され、一致しない場合には、設定ファイルに記述されている「予想される不正解出力」と一致するかどうかを調べ、一致する場合にはその出力に対応する点数とコメントが付与される。一致しない場合には、不正解である旨が仮のコメントとして出力され、後に教員が個別に検証することとなる。

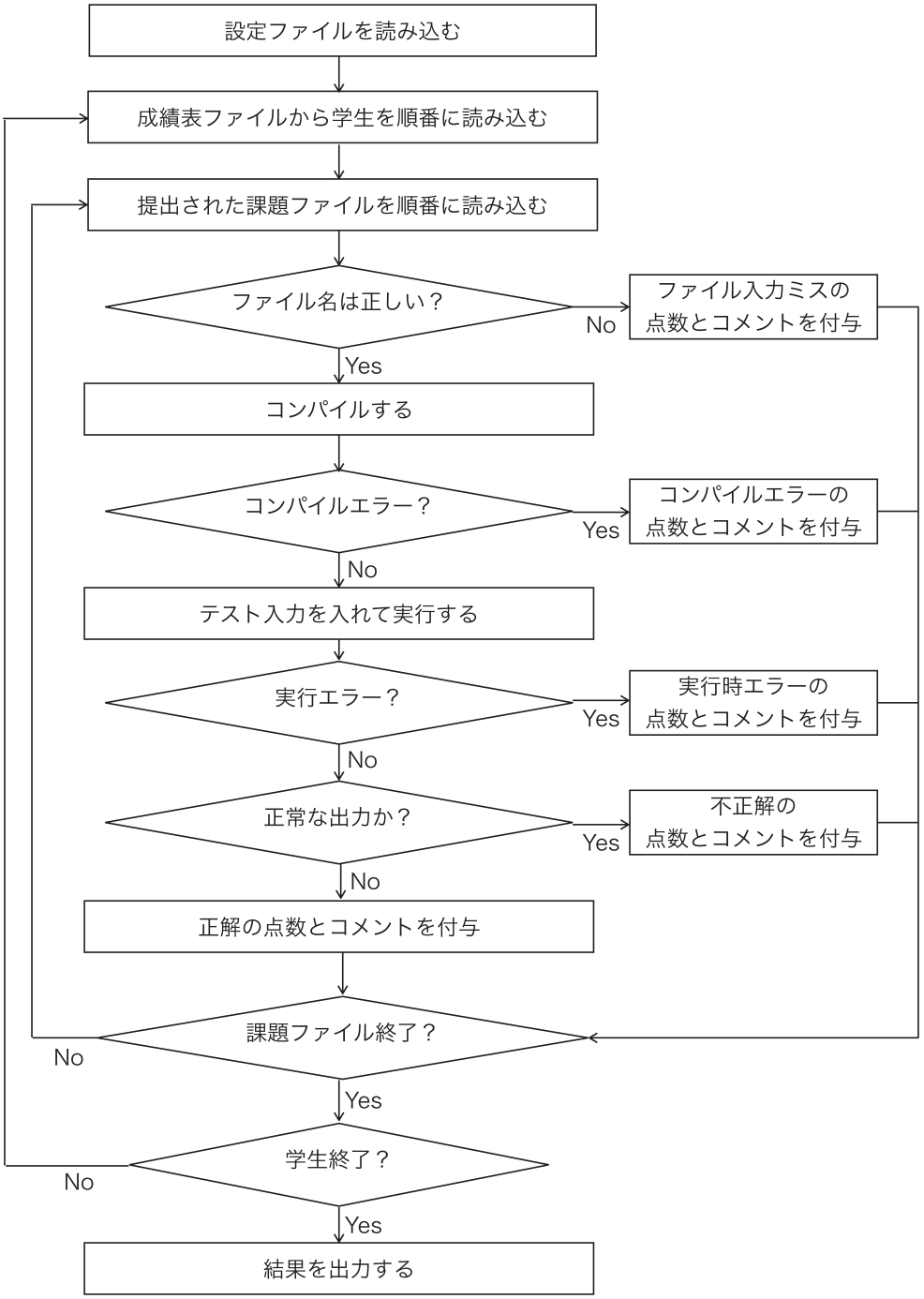

図２．課題ごとの処理の流れの概略

このようにして、学生が提出したすべてのファイルに対して同様の処理をすることで、点数とコメントが付与される。複数のファイルが課題となっている場合には、その合計点とコメントの文字連結が付与される。

なお、図２には記さなかった細かい処理がいくつかある。たとえば、異なる文字コードで提出された場合の処理、無限ループなどでタイムアウトした場合の処理、などである。

このようにしてすべての学生の採点が終了すると、その結果が標準出力とファイルに出力される。設定ファイルに記述されている予想されたエラーあるいは出力の通りであれば、あらかじめ用意されたコメントが返される。同様の間違いをしている学生が複数いる場合には、設定ファイルを書き換えてプログラムを再度実行することで、学生が間違えるパターンを分類することができる。なお、授業では提出期間中に「仮の採点」をしてmanabaの掲示板にコメントをアップし、同じような間違いがある場合にはその旨を警告して、期限までに修正して提出することができるようにしている。

最終的に生成された学生の成績は、点数とコメントがcsvファイルとして出力され、それを成績表ファイルにExcelでコピーペーストすることで採点ファイルを作成できる。その際に、Excel上で成績表ファイルの点数とコメントを直接編集することで、プログラムがうまく動いていない学生に対しては、プログラムのどこを直せば良いのかを個別にコメントをする。個別コメントを書く際には、このプログラムが出力した、不正解の学生が提出したプログラムと実行結果、エラーメッセージを一覧にしたファイルを見ながら、各学生のミスの原因を特定している。また、ミスの内容に応じ点数を加減する。このような個別の検証が必要となる学生の数は、履修者数全体の中の一部であるため、このような学生に対する個別指導に注力できる。成績表ファイルをmanabaに登録して公開することで、学生はその課題の点数とコメントを読むことができる。学生にとって見ると、各回の課題の採点結果が速やかにそれぞれの学生に応じたコメントとともに返却されるため、大人数の授業ではありながら毎回の授業でしっかりとした個別指導がされることとなる。

3. 結果

3.1. 採点結果

課題提出者数と採点結果の推移を図３に示す。授業は15回あり、最初の２回はプログラミングの基礎知識について学習し、第３回からプログラムを提出する課題を課している。図３では課題の提出ファイルが１つの回のみを集計し、課題の提出ファイルが２つあった第10回、11回、13回、14回を除外した。

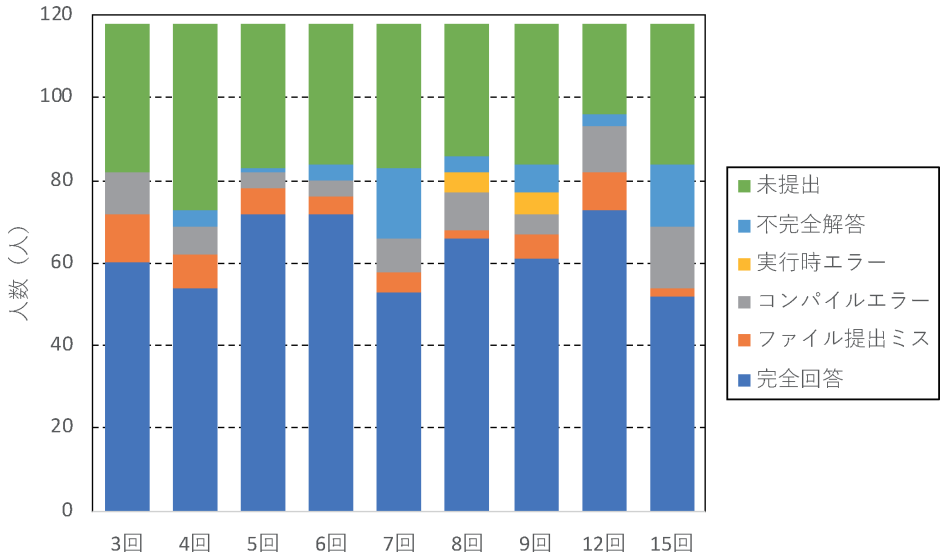

図３．課題採点結果の推移

履修者数は118名であり、その中で20名は課題提出回数が１回以下であり、授業の出席もなく早々に単位の修得をあきらめている。課題未提出の回数が４回以上の者が33名であり、平均では毎回の課題で34名の課題未提出者がいる。最終的な成績判定は、34名の学生が不合格となり、それ以外の84名の学生が合格となった。合格者の成績の内訳は、Sが14名、Aが20名、Bが29名、Cが23名となった。当初の課題提出率が授業最終回にまで維持され、それがそのまま合格率となっていることから、学生が学習を継続するモチベーションを保つことができたと言える。

第３回でははじめてのプログラム提出であり、シェルの操作に慣れていない学生がテキストに書かれている通りにHello!という文字列を出力するプログラムをテキストファイルとして作成し、Javaコンパイラによってクラスファイルを生成して、得られたプログラムを実行してそれを提出する、という基本操作を学ぶ課題である。その課題において、ファイル提出ミスが12名、コンパイルエラーが10名あった。この授業では、学生がプログラムを作成し、コンパイルと実行をして期待通りの出力が得られることを確認するまでの一連の処理を、自分の手でできるようになることを重視している。すぐに理解して実行できる学生も多いが、パソコンでターミナルやシェルを実行した経験のない学生が多い中で、たとえばシェルを起動してプログラムが格納されているディレクトリに移動する、という操作ができるようになるまでのハードルが比較的高い。また、エラーが生じた時に、そのエラーの原因を特定して修正する作業が、初学者は戸惑うところである。そのため、授業ではエラー表示の読み方（どこにエラーがあるか、エラーメッセージの意味など）、よくあるエラーについては詳しく解説している。たとえば、ASCII文字を入力するべきところで非ASCII文字（たとえばいわゆる全角アルファベットや「全角スペース」のような全角の記号など）を使うことによるエラーは頻出である。このようなミスは、授業時間中にも何度も確認をしているが、最初はしっかりと確認をせずに提出をする学生も多いため、当初

は22名の不完全なファイルを提出する学生があった。それらの学生に対しては個別コメントでその旨を詳細に説明し、manabaの掲示板にもその旨を記し、次の回の授業ではそのことを確認する、といったことを繰り返すことで、第５回の授業ではファイル提出ミスとコンパイルエラーが合計10名にまで減っている。

第７回の授業で不完全解答が増えているのは、それまでの授業と比べて課題の難易度が上がっているためである。最初は簡単な課題として、授業の進行とともに課題の難易度を上げている。第８回と第９回に「実行時エラー」があるのは、無限ループによるスタックオーバーフローエラーやタイムアウトが生じるようなプログラムを提出する学生がいたことにより、課題の内容がそのようなミスを引き起こしやすいものであったためである。また、第15回はニュートン法による数値計算の課題であり、丁寧に解説をしたものの、数学を苦手とする学生にとっては難易度が高いため、不正解率が上がっている。

3.2. 学生の感想

授業の最終回に実施した授業アンケート（回答者数51名）の結果を図４に示す。授業の難易度については難しいと感じる学生が多く（問３）、プログラミングの初学者にとっては挑戦的な内容ではあったにも関わらず、多くの学生はその挑戦的な内容に取り組み、86％の学生がプログラミングの理解が（わりに）深まったと答えている（問１）。

感想の自由記述欄には「毎回課される課題のフィードバック、採点どちらも掲示がとても早くて良かったです」という反応があった。このような速やかなフィードバックが可能となったのは、自動採点システムを構築したことによる利点である。

なお、学生の自由記述の中には「できればPythonでの提出が可能だったらよかったです。Pythonは経営学部の統計学の授業とも相性が良く、インタープリンタ言語であるため初心者が学びやすいので検討していただけたらと思います。」というものがあった。その学生の意見も参考として、2020年度からは、教えるプログラミング言語をJavaからPythonに変えて、それにともなって採点プログラムも作り直している。さらには、エラーに対する対処方法などをテキストに詳しく解説することが重要であるという知見が得られたため、テキストを大幅に改訂して、著者のホームページ（http://www2.toyo.ac.jp/~seki_k/python/）に公開している。

問1. プログラミングの理解が深まりましたか？
　　深まった　41%
　　わりに深まった　45%
　　あまり深まらなかった　12%
　　まったく深まらなかった　2%

問2. この授業はわかりやすかったですか？
　　わかりやすかった　29%
　　わりにわかりやすかった　37%
　　ややわかりにくかった　25%
　　わかりにくかった　8%

問3. 授業の難易度は、あなたにとってどうでしたか？
　　難しすぎた　57%
　　ちょうど良かった　43%
　　簡単すぎた　0%

問4. 授業の進行は、あなたにとってどうでしたか？
　　進行が速すぎた　31%
　　ちょうど良かった　65%
　　進行が遅すぎた　4%

図4．授業アンケート結果（回答者数 51名）

4. おわりに

本報では、プログラミングの授業で提出されたプログラムを自動的に採点するシステムについて報告した。学生が間違える共通のポイントについてあらかじめチェックしておくことで、多くの課題を採点する場合でも、一人ひとりに対してコメントを返すことに集中することが可能となった。

2020年度からは、新型コロナウイルスの影響で自宅からのオンライン受講に対応する必要が出たため、パソコンにプログラミング言語をインストールして実行する形式ではなく、ウェブブラウザにプログラムのコードを入力してサーバー上でプログラムを実行させるオンライン実行環境のサービスPaizaを利用する形式とした。Javaプログラミングの授業で最初の壁となったシェルを起動してコンパイル、実行する手順がなくなったことは、プログラミングの重要な手順を修得する機会が失われてしまったという面もあるが、一方で、すぐにプログラミングにかかれることは、初学者がより速やかにプログラミングを経験できるというメリットがある。

このようにオンライン実行環境を使うのであれば、課題の提出と採点、コメントを返すところまでをすべて自動化してオンライン環境の中で完結させることも可能であろう。即時に課題の正誤判定とヒントを出す練習問題を設定し、課題の進捗によって成績判定をすることが考えられる。大学としては適切なクラスサイズを保つように人的資源を含む教育

資源を配分することが優先的に考えるべきところではあるが、限られた資源の中で効率的かつ質が担保された教育をするためには、このように可能な範囲で自動化を取り入れることも有効である。