研究ノート

実習補助者のいないクラス形態向け プログラミング教育支援システムの構想

上地　宏一

実習補助者のいない初等プログラミング教育授業では、入力したプログラムの文法ミスを学習者自身で修正するスキルが低く、満足する教育効果が得にくい。そこでファイル共有サービスを活用して学習者の入力しているプログラムをサーバに逐次ミラーリングし、完成プログラムとリアルタイムに比較することで、学習者への入力ミスの自動提示や教室全体の進捗確認を可能とするプログラミング教育支援システムの設計を行った。

キーワード：授業支援、プログラミング教育、実習補助者なし、リアルタイム、文字列比較

1．はじめに

文系学部生がプログラミング教育を受けるメリットはいろいろ考えられる。たとえばプログラミング教育の背後にある論理的思考の訓練は、物事の処理手順を組み立てる能力として社会人全般に必要であるし、あるいは事務処理において例えば Microsoft Office のワードやエクセルに対して VBA(Visual Basic for Applications)が書けるようになると、面倒な仕事をバッチ処理で解決できるといった具合である。また、プログラムを自分で入力してそれが動いたという達成感も得られる。

しかし文系学部生にプログラミングを教えることは難しい。その一つにはカリキュラムにおける科目数が足りないということが挙げられる。たいてい同時期(1 学期中)に 1 コマが限度であり、週 1 回 90 分の授業および教室外学習だけでは理系学部や情報系学部のカリキュラムの濃さには到底及ばない。もう一つは文系学部の場合、授業に SA や TA が付いて進度の遅い学生をフォローするという理系学部や情報系学部のような枠組みが期待できないことである。初級プログラミング科目で学生がつまずきやすいのはとにかくプログラム入力時の文法ミスである。この文法ミスは言語が同じであればどの学生でも同じ傾向となる。しかし、その傾向をあらかじめ教師が教えても定着せず、何度もそのミスに遭遇することで、初めて自分で対応できるもののようである。多くの場合 SA や TA はその授業を過去に受講して自分が通ってきた道でもあるため受講者の身になった指導がしやすい。またプログラム内のミスを見つけることは SA や TA 自身にとっても大切な学習活動になっている。さて、このミスを完全に除去できないとたいていの場合プログラムは実行できないことになり、消化不良となるだけ

でなくプログラミングの達成感も得られない。教師も全員の完了を待っているわけにはいかないため先に進んでしまうが、一部の学生は何としてもミスを解消しようと作業を続けてしまい、後の講義を聞き逃してしまう。そうするとつまずいた後のすべての内容を消化不良で終えてしまうという状況も起こりうる。これは授業時のプログラム入力だけでなく、事前学習としてのプログラム入力も同じである。そして、この「文法ミスを見つけて完全なものに仕上げる」スキルが身につかないと、自分で新たに作る「答えのない」プログラムを完成させることはできない。このように、SA や TA の配置が期待できない文系学部のプログラミング科目は「消化不良」になる危険性が多分にある。筆者は大東文化大学社会学部でプログラミング科目を担当するが、実利としてこの SA や TA といった実習補助者を補うプログラミング教育支援システムが作れないものかと考えた。単純にプログラム入力時のミスを指摘する機能が要件である。

2．システムの設計

2-1　プログラム入力時のミスの指摘

そもそもプログラム内にミスがある場合、コンパイラないしインタープリタがエラーを出力し、解釈できなくなった部位の提示がある（図 1）。記号のミスやスペルミスといった問題であれば、提示されたエラー部分を見れば気が付くものであるが、一部の学生はまず英語のエラーメッセージに拒絶反応を示す。またこのケースの間違いに気が付きやすい学生となかなか気が付かない学生に分かれ、不得手な学生は膨大な時間を費やすことになる。

```
Traceback (most recent call last):
  File "C:¥Users¥kamichi¥ownCloud¥授業¥23_社会学演習Ⅰ¥python¥bmi.py", line 1, i
n <module>
    h = float(imput("身長何cmですか？")) / 100.0
NameError: name 'imput' is not defined
>>> |
```

図 1　Python 言語におけるエラーメッセージ（input を imput と記述するミス）

また、繰り返しや条件分岐といった順列ではない構造のプログラムでは、構造記述のミスによってエラーを出したり、あるいはエラーを出さないが意図する実行結果にならないケースがあったりする。コンパイラやインタープリタはあくまでも解釈ができなくなった時点でエラーを出すため、そのエラー部位が構造上のミスと直結しないことが多く、学習者にとって修正の難易度が高い。

大東文化大学社会学部のプログラミング科目で扱う予定の Python 言語は繰り返しや条件分岐の構造を｛ や ｝の記号で記述するのではなく、インデント（字下げ）で記述する特徴を持つ。このことは初学者向けのメリットとして説明されることが多いが、実際は目に見えない情報で記述するため、かえってミスに気が付きにくいのではないかと考える。

本ノートで構想している支援システムの場合、授業時や事前学習としてテキストに掲載されているプログラムの入力を想定していて、一意のゴールがある作業である。そこで、ゴールとなる完成したプログラムと学習者が入力中のプログラムとを比較して、その差異を指摘すればよいと考えた。文字

列比較としてレーベンシュタイン距離のアルゴリズムを用い、完成プログラムと入力プログラムの「足りないもの」「余計なもの」を指摘することとした（図2）。

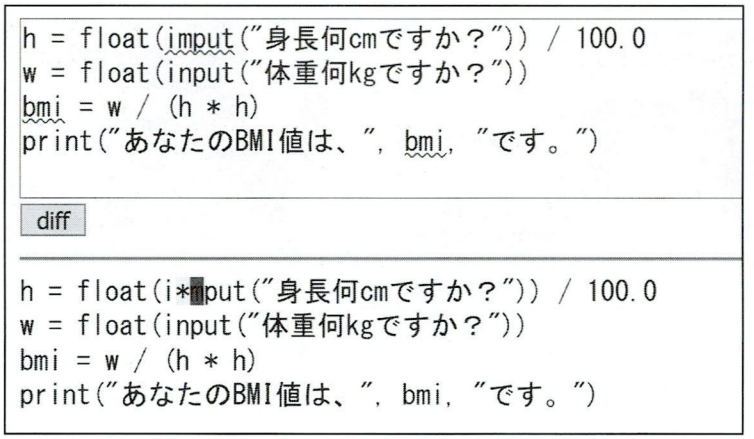

図 2　プログラム入力ミスの指摘（イメージ）

図は上部が学習者の入力したプログラムであり、下部が完成プログラムとの比較結果である。1 行目の「input」のスペルを間違えた状態で、比較結果では「m」が余計なものであり、「*」部分が 1 文字足りないと指摘している。コンピュータは足りないものが「n」であることを認識しているが、正解は教えずに自分で考えさせる、またはテキストを見直すように促すこととする。

2-2　ズルの防止

この支援システムは、あくまでも自分が入力したものに対して実習補助者がミスを指摘するイメージで文字列比較を行うものであるが、この種の比較システムを用意すると、内部でデータとして用意されている完成プログラムを探してそれをコピーアンドペーストしようとしたり、初めからコツコツと入力する気が起きない学生が発生したりする。そこで設計として内部に完成データを持たせず、あくまでも比較結果のみを出力するシステム構造とする。また、両者に一定以上の差（＝ミス）がある場合は比較結果を表示せず、もっと完成させてから比較するように注意を促すこととする。

2-3　ファイル共有システムの活用

前項で述べたように、あくまでも学習としてのプログラム入力が主であり、比較やミスの指摘の作業が主になってはならない。そのためには、支援システムを使う手順がシンプルであることが望ましい。そこで、Dropbox のようなリアルタイムのファイル共有システムを活用する設計とする（図3）。具体的には、学習者は指定したフォルダ内に指定したファイル名でプログラムを保存することとし、そのフォルダを教師（サーバ）と共有する。プログラムを保存すると自動的にファイル共有クライアントによってリアルタイムで教師のサーバにファイルがコピーされ、すぐに比較結果が計算される。比較結果は Web ページの形で表示することとし、学習者のパソコン上ではプログラムを入力するウィンドウ、実行するウィンドウと、比較結果を表示するブラウザの 3 つを切り替えて入力することとなる。

一方、教師は入力する対象のプログラムをあらかじめ用意しておくか、あるいはその場で直接プログラムを入力し、そのファイルを格納するフォルダをサーバと共有しておく。

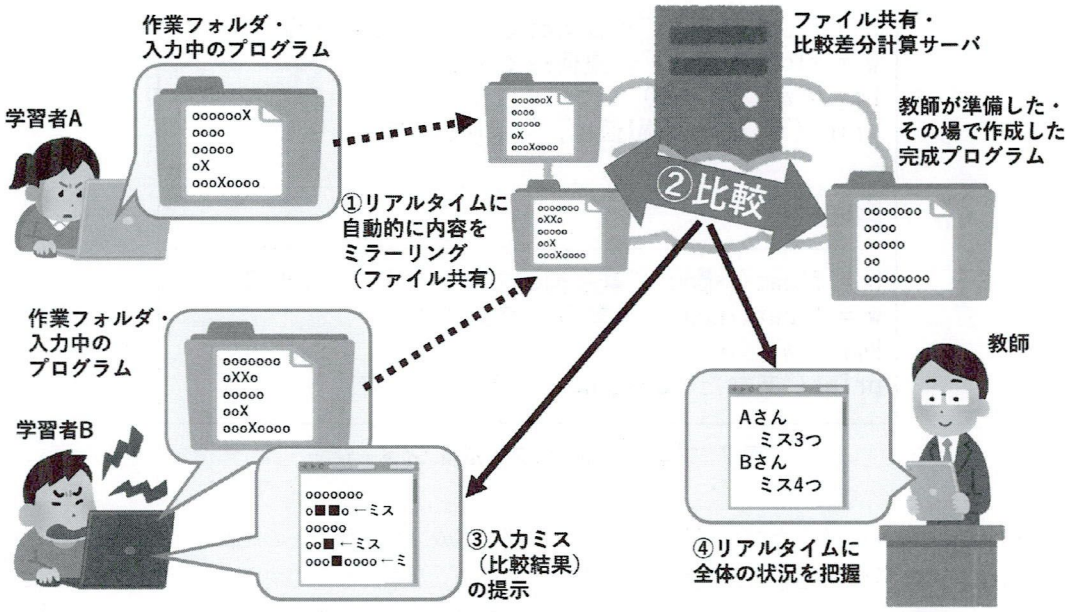

図 3　ファイル共有を援用したシステムの全体像

さらにこの方法を採ることで副次的な効果が期待できる。それは全学習者の状況を教師が一元把握できることである。各学習者について、実行できる段階までプログラムを入力したかどうか、あるいは学習者が比較を希望しなくても、現時点での保存されたプログラムを比較してミスが何か所残っているかを抽出できる。その結果、教室全体の状況を把握できることになる。比較結果の差分情報を逐次集約することで、どのようなミスの頻度が高いかをその場で指摘することや、プログラム解説時のフォロー事項として活用することができる。従来は授業終了後の教師と実習補助者との情報交換によって次回以降の授業に反映されてきたエッセンス（学習者の傾向とそれに対する指導内容）を、この支援システムでは当該授業時に即座に活用できることが従来にない利点となる。

3．想定される問題点

前章で設計した支援システムは、従来の実習補助者による入力中のプログラム内のミスの指摘を代替するものであるが一つ問題点がある。それは学習者がひたすら比較結果を見てプログラムミスを盲目的に直していくことで「内省する」ステップを省いてしまうことである。それを防ぐためには、たとえば授業時のプログラム入力であれば、差分表示機能の使用をスイッチで切り替えられるようにして、はじめの一定時間は教師のための状況把握のみに制限し、時間をかけてもミスの除去数が減らないと判断したら機能をオンにして具体的なミスを見ることができる、などの機能追加が考えられる。

また、文字の間違いが複数回あるといった同じ形態のミスは具体的な位置ではなくメッセージとして指摘するなど、全員に共通するミスを集積して「ヒント」に相当する機能を追加することも教育効果があると思われる。

この支援システムはファイル共有サービスの利用を条件とし、リアルタイムに作業フォルダを共有するためのクライアントプログラムの導入が必要であるため、大学の設置パソコンではなく大東文化大学社会学部のようなBYOD（必携PC）の制度と、教室内無線LANなどのネットワーク利用設備が前提となる。その意味では汎用性の低いシステムとなるが、社会学部科目での利用に特化していると考えれば問題にはならないだろう。

4．おわりに

以上に述べてきた支援システムの実装をこれから行うこととする。そして2020年度のプログラミング科目およびゼミ演習で試用と考察を行う計画である。結果については改めて報告の機会を持ちたい。

〈参考文献〉

山下寛人, 2009, レコメンデーションとエディットグラフ（2/4),
https://www.atmarkit.co.jp/ait/articles/0905/01/news097_2.html, 2020年1月1日閲覧

Designing a Programming Education Support System for Lessons without Practical Assistants

KAMICHI, Koichi

With respect to elementary programming education, it is difficult to obtain satisfactory educational results without the support of practical assistants because the learners possess low skills to rectify the grammatical errors present in the input program. In this study, a programming education support system has been designed using a file sharing service. The input program provided by the learner is sequentially mirrored on the server and compared with the completed program in real time, enabling the learners to observe their mistakes and the verification of the progress of the entire classroom.

Key words : Education support system, Programming education,
Lessons without practical assistants, File sharing, String comparison.