

コーディング教育における学習初期課題を軽減するための実践報告

江上 邦博

概要

近年、教育カリキュラムにおけるプログラミング教育が注目されるようになったが、プログラミングやWeb制作などでソースコードを扱うコーディング教育は、文系学生には難しいと認識されている。ここでは筆者が担当する授業において、学習者の学びのハードルを下げるために行った取り組みを報告する。

コーディングは難易度が徐々に上がる性質のものではなく、学習初期から正確なソースコード入力求められる。学習内容の理解が不十分な状態でのキーボード操作となるため、入力ミスが発生しやすく、加えてミスの発見も困難である。ミスを修正する作業はコーディングにおいて避けては通れないものだが、その頻度が高くなると学習者の理解を妨げ、学習実感にもつながらない。そこで、初学者が経験する入力ミスを発見し修正につなげるWebツールを作成し学習効果の改善を試みた。

さらに、授業初期の教材としてWebブラウザ上でコーディング教育を実習できるサービスを導入した。Webブラウザのように学習者が使用法に慣れたアプリケーションを用いることで、授業がより手軽に開始できるようになった。加えて学習が教室内に限定されることなく、様々なコンピュータプラットフォーム上でも実習が可能となり、リモート授業時の対応も含めた初学者への導入教育を円滑に行えることを確認した。

2020年度、新型コロナの影響により筆者らはオンライン型授業への対応が求められた。本レポートは、その当時の取り組みをきっかけとして、授業改善のために行った実践のまとめでもある。

キーワード: コーディング教育, 学習環境改善, オンライン実習, Webベースのサポートツール

A Practical Report on Reducing Initial Learning Barriers in Coding Education

Kunihiro EGAMI

Abstract

In recent years, programming education has gained attention in school curricula. Coding education, including courses in programming languages and web development, is often considered challenging for liberal arts students. This paper reports on some efforts to reduce the initial learning barriers faced by such students.

The first approach involves the development of web-based tools to improve keyboard input accuracy. Accurate coding requires precise source code input from the initial stages of learning. However, due to insufficient understanding of the course content, students often make typographical errors in their code, which are difficult to detect and correct. While error correction is an unavoidable aspect of coding, frequent error correction tasks can hinder students' understanding and negatively impact their overall learning experience. To enhance learning outcomes, web-based tools designed to detect and reduce input errors have been developed.

The second approach involves introducing web-based coding services accessible through widely used web browsers as initial teaching materials. These services allow students to begin learning more easily. Furthermore, they assist smooth learning across various computer platforms and also support remote learning classes.

In 2020, the COVID-19 pandemic necessitated a shift to online remote classes. This report also summarizes the trial-and-error efforts made to ensure class continuity and the practical improvements in teaching that were realized as a result of the pandemic.

Key-words: coding education, improve learning environment, online education, web-based support tool

1. はじめに

今世紀に入り、筋道を立てて考えることを基本とする「プログラミング的思考」が注目され、実社会で活躍するための必須とされるようになってきている。この背景には、AIやIoT、ビッグデータなどのデジタル技術が社会に浸透し、急速な技術革新が進んでいる社会情勢がある。今後の社会を担う学生に対する教育においても、こうした変化への対応が求められている。デジタル技術を基にしたIT化やDXが進む社会への適応力を高め、学生のキャリアの可能性を広げるためにも、プログラミング関連スキルの獲得は有用である。

また、プログラミングの問題や課題への挑戦は、「試行錯誤を繰り返しながら」「完成するまであきらめず継続する」ことを学ぶ機会となるため、学習成果は技術的なスキル面にとどまらない。論理的思考力と問題解決能力の向上に繋がると期待されている。加えて、学習者が実習時に向き合うコンピュータは、疲れたり、あきらめたり、学習者のミスを過剰に扱ったりすることもないため、自分のペースを乱さず学習を進めることができる優れたサポーターともいえよう。こうした学習効果への期待もあり、筆者が所属する学科ではコンピュータ系の教育を充実させてきた。

筆者が担当する授業において、プログラミング的思考の涵養につながる科目には、「C言語プログラミング I および II」、「Webサイト制作 I および II」がある。これらの科目を運営していると、履修学生が身につけている知識やスキルの格差を大きく感じる場面が散見される。

この格差を感じる原因は、対象とする授業がコンピュータに関する専門的な内容を扱っていることにあるが、さらにカリキュラムの高大接続性にもあると考えている。これまでの学習指導要領の改訂を振り返る

と、2002年度から小学校・中学校で情報教育が、さらに2003年度からは高等学校に教科「情報」が導入された経緯がある^[1]。だがこの改訂では、プログラミングを含む教科は選択となっており、必ずしも全員が学ぶものにはなっていなかった。現在本学に入学する学生のプログラミングにかかわる知識は、高等学校までの多様な学びの中で各々獲得したものであり、その結果として学生個人個人の知識やスキルには大きな格差が生じているようである。そのことが学生間での理解度や学習進度の差として表れていると思われる。

本来であれば、背景が異なる多様な履修学生との学びは、教室での一斉教育ではなく個別対応的な形態が効果的であろう。だが、現状の授業運営対応では困難も多く、代替策として電子的なツールを利用して個別指導に近づける研究も行われているが、その教育実践手法はまだ一般化されていないように思われる。こうした手法を開発し評価することに実践研究の余地がある。

さらに、プログラミング的な教育には、履修学生自身が受ける印象にも格差がある。授業を進んで履修し、課題の取り組みを楽しむ学生がいる一方で、多くは内容が「難しい」と評するようである。この原因には、コンピュータ操作時のインターフェースが改善されている一方で、プログラミングなどのハードウェアに近い低レイヤ技術との乖離が進んでいることにあるように思う。具体的には、スマートフォンやPCなどでのアプリを介した一般的な利用方法と、プログラミング関連の知識や操作手法は大きく異なっている。加えてプログラミングで学ぶ項目は、日常生活での実体験とのつながりや、他分野との関連性が乏しいこともあるため、授業開始時に新しい知識や概念を幅広く学ぶ必要がでてくる。こうしたプログラミング教育がもつ特性

があり、学習をより難しいと感じさせる要因になっていると考えられる。要因を解決する学習教材の作成や授業手法の開発も求められている。

筆者は担当する上記の科目群について、授業運営上の困難があることを感じつつも、本学で行われる他の授業と同様に、教室での一斉授業を基本として運営してきた。だが、2020年に生じた新型コロナ禍への対応のため、授業手法を大幅に見直すことが迫られた。通常時とは異なり、1か月ほど遅れて始まった学期の授業は、組織公式のWebサーバを使用したり、Microsoft Teamsを利用したりして、インターネット経由でのオンライン型として開始されたのである。履修学生が用いるクライアント側のコンピュータ環境は、当時学生が利用できるものであれば機種を問わず何でも用いることになり、例えばスマートフォンやPCをはじめとする様々なプラットフォーム上で授業が行なわれた。教員には毎週膨大な量のコンテンツ作成が求められたが、筆者はこれまでも授業資料を電子化しWebで公開してきた経験もあり、混乱のなかでもおおむね授業を開始・継続させることができた^[2]。だが、そうした状況の中で、プログラミングなどのコーディング実習を伴う授業対応は、オンライン型授業運営を想定してこなかったこともあり、授業の進め方や利用する教材の面で再検討が必要となったのである。対面型の授業では、PC画面を一つ一つ見て確かめて、一步一步進めていくことが可能であるが、筆者が経験したオンライン型の授業ではこの手法が使えなかった。さらに、授業に様々なプラットフォームが使用されるため、機器の操作方法や教材を表示する環境が統一されず、それらのすべてに対応した教材を作成することは困難でもあった。

オンライン型授業への対応から始まった教育の再構築の試みは、その後に再開された教室での対面型授業時にも有効であることが確認できた。このため履修学生の学びのハードルを下げる教育手法として、継続して取り入れることにした。現在、2020年の混乱からは数年の学期が過ぎ、新型コロナ禍の影響は薄れ、社会的には落ち着きを取り戻してきている。いささか時間が経ちすぎた感もあるが、当時の試行錯誤の記録もか

ねて本論文に教育実践としてまとめることにした。

2. コーディング教育における学習課題

プログラミングやWeb制作の際にソースコードを入力する実習作業、つまりコーディングを伴う教育を本論文では「コーディング教育」と呼んでいる。具体的には、授業で扱うC言語やWeb関連言語（HTML、CSS、JavaScript）の記述を対象とする。コーディング作業は、近年ではGUI（Graphical User Interface）を用いるものや、いわゆるノーコード的に開発を行うものもあるが、ここでは筆者の授業で行っている「キーボード操作」による文字入力を基本とした実習を想定する。

このため、履修を希望する学生は、PCの基本スキルをもつことは当然であるが、特にキーボードの基本操作を理解し確実な入力ができることが重要になる。だが現実には、短期大学の学修期間は2年間と短く、積み上げ型の学修にも学期上の制限があり、条件を満たすための学修を終えるには期間が足りない場合がある。加えて実際には様々な事情から授業科目を履修する学生がいるため、条件を満たしていない学生も履修することになる。こうしたPCの基本スキルに差がある多様な学生相手に、一斉授業を行う際の困難を改善すべく、筆者もこれまでに様々な試みを行ってきた^[2-5]。

一方、難しいと判断されがちなコーディング教育ではあるが、逆にうまくいったときには「授業についていけた」、「楽しかった」、「面白い」など、肯定的な意見や実習の成功を喜ぶ声もよせられる。コンピュータと向き合って、自らソースコードを作り出すコーディング作業は、考えていることをアウトプットする創造的な活動に類する面があり、講義型の科目では得られにくい経験になっているようである。このように、うまく実習が進めば学生が教育の効果を実感する場面につながると考えられるため、安定した学習実感や成功体験につながる工夫の必要性を感じていた。

ここでは、こうしたプログラミング教育に関する課題について、学習初期に経験したキーボード入力操作と実習プラットフォームに関する以下の2点に着目し

検討する。

2.1 キーボード入力操作の課題

コーディングを行う言語にはそれぞれに厳格な文法があり、その枠から外れるとわずかなタイピングミスでもエラーにつながって、思うようには実習が進まない。コーディングの対象はコンピュータであるため、1文字単位で正確な文字種を紡がなければならない。「だいたい同じに見えるから良いだろう」という考えでは、実習は完了せずプログラムは動作しない。これはMicrosoft Wordなどを使ってレポート課題の文書を作成し、人間相手に提出する場合とは事情が異なる。人間に提供する文書であれば、ある程度の融通が利いたり、ミスが見逃されたりすることで、このような厳密さが求められることもある。プログラミングの経験がない学生でも、Microsoft Wordなどのオフィスアプリケーションを利用したり、スマートフォンを日々使用したりしているため、入力作業には慣れているはずなのだが、「指定された文字以外はすべて不可」という事態に戸惑う学生は珍しくない。

さらに、コーディングの実習作業は、順に学修難易度が高くなるのではなく、学習者は学習初期から正確なソースコード入力が求められるという特徴もある。授業初期では授業内容の理解が不十分な状態でキーボード入力操作になるため、入力ミスが発生しやすく、その修正に追われることになる。

本来、コーディング教育では、エラーに対処する作業が発生するのは当然であり、正しい理解に繋げるためにこの作業を避けることはできない。エラーへの対処には、エラーメッセージを読み解き、その原因を特定し解決するために論理的な思考が求められる。このためにはプログラムの基本的なルールを理解できる必

要があるのだが、学習を始めたばかりの学生にはこれが難しい。多くの学生は自身の実習結果と、実習例題のコードとを比較するだけの単なる間違い箇所探しになってしまっている。

授業時はエラーがすべて発見され、課題が正常に終了するまで、この間違い探しの作業が続くことになる。中には授業の最後まで修正作業が続き、プログラムを実行できず、学習を進めることができない履修学生もいる。これは特に授業初期に見られる事例である。こうした間違い探しの作業が続くと、順調に実習を完了した学生に対して劣等感を抱きやすく、「うまくいかない」、「難しすぎる」、「自分には向いていない」と感じてしまう要因にもつながるようである。このように、入力ミスとの付き合い方が学習者の学習実感に大きな影響を与える結果になっている。

筆者は授業において、学生からの質問があればミスをチェックし修正のヒントを与えるようにしている。学生が経験するミスは、タイプミスのような単純なものに加えて、理解が不十分なことによる勘違いが原因の誤入力が目立つ。後者は例えばstdio.h (正)をstadio.h (誤) のようにしてしまうものだが、単純なタイプミスに比べると誤りに気づきにくい傾向がある。だが、こうしたミスはいずれも実習例題との違いを視認できるため、時間はかかるが初学者でも気付ける可能性が比較的高いようである。

一方でミスの種類には、単純だが学生がなかなか気づけないものも存在するようである。この種のミスの具体例として、ソースコード1にC言語で記述したプログラムを示している。これは初学者向けの入門プログラム(「Hello World」プログラム)であり、表示を確認する限り文法的に正しく問題がないように見える。コーディングには半角文字で記述する一般的ルールが

ソースコード1 ミス(全角スペース)が含まれるプログラムの例

```
1 #include <stdio.h>
2
3 main( )
4 {
5     printf("Hello, World!¥n");
6 }
```

あり、このためソースコード中の空白文字部分(" ")を全角文字のスペース(" ")で入力してしまうとエラーとなってしまう。さきほどのソースコード1の5行目行頭には全角文字のスペースが入力されているのだが、実習を行っている画面上でこれを見つけることは困難である。この場合、履修学生からは「なんど見直しても間違いが見つからないのに、エラーが出て動かない」と問い合わせがくることになる。

こうした日本語入力に係る事例は、PC教育が行き届いてきた現在では注意深く入力することで避けられるはずだが、以前に比べると発生頻度は増えてきたように感じる。この原因は、学生のキーボード操作を観察することで容易に特定できる。それは、入力がキーボードから直接ではなく、いわゆるIME (Input Method Editor) などの日本語入力システムを介して間接的に行われることにある。コーディングで使用する記号文字は物理的なキーボードがあれば、変換することなく直接入力することができるようになっている。だが、特に授業初期にはキーボードから直接入力できる文字種を、わざわざIMEを使って変換して入力する事例が散見され、入力を不確実で曖昧なものにしているようである。

スマートフォンのように物理的なキーボードを搭載しない端末にとってはIMEを使った変換入力は必須である。またPCにおいてはアプリ起動時にIMEが有効化されている設定の影響もあろう。加えて、コーディングでは普段は使用する機会が少ない記号類が使われていることも、入力時にIMEを使う動機になっていると思われる。こうしたことから、必要な文字を直接キーボードから入力するという概念が抜けている学生が見

受けられる。文字変換の結果、正しい文字が選択できればよいが、似たような別の文字が入力される場合もあり、ソースコード1に示されたような、なかなか気づけないミスを生み出すことになる。

このため、授業の初期段階では英数字や記号に慣れるための練習を重視し、半角文字や全角文字のキーボードの入力モード切り替えを徹底させるようにしているが、入力の習慣を修正させることはなかなか困難であり、定着させるまでにかなりの時間を要するのが現状である。

2.2 学習プラットフォームの課題

プログラミングの学習を進めるには、ソースコードの正しさを検証するために、対応する言語プロセッサが必要となり、その種類は学習内容によって変化する。現代のプログラミング学習は、PCなどの実行環境が手元にあれば、自らで無料の言語プロセッサを入手して手軽に始めることができるようになっている。筆者が担当する授業に関するものでは、Visual Studio Community や GNU Compiler Collection (GCC) などが無料でインターネットから入手可能であり、これらをインストールすることにより本格的な学習環境を構築することができる^[6,7]。HTML, CSS, JavaScript の学習は、コンピュータに標準的に搭載されているブラウザを使うことにより、ソースコードの動作確認を含んだプレビューが可能である。学習段階がすすめばこうした事情も理解できるようになり、所有するコンピュータ上に実習環境を構築することも可能になるのだが、初学者にそれを求めるのは難易度が高い。このため、

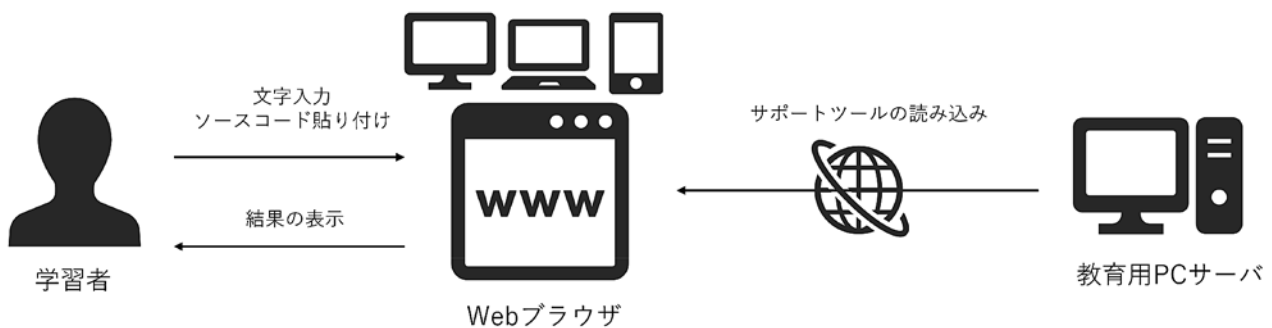


図1 システム構成図

短大の教室に設置されたPCには、あらかじめ言語プロセッサを含めた授業教材が用意されており、授業内の指示の通り実習することで、だれでもプログラミングを経験することができるように整備してある。ただし、リモート授業では教室が利用できないため、その代わりに履修学生側の準備が不要で、ただちに使用できる教育システムが必要となってくる。

また、コーディング教育のクラス内での学力差が大きく出てしまう点に関してだが、学びのサポートすべてを授業時間内で行うには限界があり、学習者の自学自習を期待することで解決できる面が大きくなっている。コーディング教育での習得の近道は、実際に実習を繰り返すことにある。実習の授業を学校のPC教室だけにとどめず、どこからでも手軽にアクセスできる手段を用意することで、習得機会を広げることが期待できる。

3. 教育改善の実践: コーディング教育における学習課題の解決

3.1 文字入力ミスの軽減に関して

2.1にまとめた入力ミスの課題への対処には、履修学生にキーボード操作スキルを高めてもらうことが一番である。そのために、ここでは文字入力を確認するためのWebツール2点を作成し授業に導入した。このツールは、(1) 履修学生自らがアクセスでき、文字が正しく入力されたかどうかを、コンピュータ内部で使われているASCIIコードのような文字コードレベルで確認できたり、(2) プログラム内に全角文字が含まれていないかを強調表示することで確認したりするものである。ブラウザ経由のWeb閲覧に慣れていれば簡単に利用できるように、いずれのツールもHTML/CSS/JavaScriptを使用して記述し、筆者が教育用に利用しているPCサーバ上に配置した(図1)^[2]。これらのツールのソースコードは、筆者がサービスを配置・公開している授業用サーバ上から確認することができる。

図2に示すように、Webツール(1)は画面内の入力領域に文字をタイプすると、即時にその内部文字コードを数値で表示する。この数値を正しい値と比べるこ

とで正確な文字入力ができているかを明確に確認することが可能である。実際の授業では、ツールを初期の教材として導入し、コーディング教育で使われる各種記号文字の確認や、半角文字と全角文字の違いを理解させるために活用している。

さらに別のWebツール(2)では、図3に示すように入力領域にソースコードを貼り付けると、全角スペースなどのコード内の不正箇所を可視化し判別が容易にできるよう強調表示する。動作原理は簡単であり、JavaScriptにより入力コードを1文字ずつ解析した後、日本語であればHTMLの太字タグ()を挿入し、これをCSSで強調表示する仕組みである。授業では、入力ミスを見つけることができずに実習に行き詰っている学生にツールの利用を促し、全角文字の入力箇所を確認させるようにしている。

入力後即座に結果を反映させるため、今回のツールはいずれもクライアント側のブラウザ上で機能させるようにしている。このため、ファイルをローカル環境にコピーすることにより、サーバレスで動作させることも可能である。こうしたWebツールの実装については、現在では同様な機能をもつサービスを見つけることができる。

図2 入力文字の文字コード確認ツール

画面上部に文字を入れると、その文字の文字コードを順に表示する。この例では、画面上部に英字"A"、半角スペース" "、全角のスペース" "の3文字を入力している。画面下部には、その3文字のコードがそれぞれ順に表示されている。



図3 全角文字を強調表示するWebツール

画面上部にソースコードを入力すると、画面下部に全角文字部分が色付きで強調されて表示される。図は全角文字が含まれたプログラム（ソースコード1）を貼り付けた例である。

3.2 コーディング教育のオンライン化に関して

2.2に示した課題については、コーディング教育をインターネット閲覧に使用するブラウザ上で行うWebベースのサービスを導入することで対応することにした。これにより、追加的な構成を授業に導入することなく、コンピュータの最小構成環境で実習教育を実現可能となる。このようなツールは、「オンラインコンパイラ」や「Webコンパイラ」、「オンラインHTMLエディタ」などと呼ばれている。これらはWebブラウザ上でソースコードをコンパイルしたり、実行や結果を表示したりすることができるサービスとして、現在では多様な選択肢から最適なものを選べるほどに充実している。リモート授業を経験した現在では、多くの組織でオンラインコンパイラをプログラミング教育に導入し利用しており、こうした授業手法は新規性がある試みとは言えない。だが2020年当時は、限られた準備期間の中で、環境を構築し動作を確認する時間的な余裕もなく、既存のサービスの中から、筆者の使用経験

があるものを中心に検討し選択するしかなかった。その際、スマートフォンなどのような限られたスクリーンサイズや限定された入力デバイス上でも実習可能なこと、実習のソースコードがクライアントではなくサーバ上に保存可能であり共有することによって教材としても提示できること、学生から提出された課題の添削指導などの修正が容易であること、さらに無料でユーザ登録などの手続き不要ですぐに利用が開始できることなどが選定の条件となった。

検討の結果、C言語プログラミングの授業については、「ideone.com」を、Webサイト制作については「JSBin」のサービスを利用することとした^[89]。こうしたサービスを用いることで、原理的には、スマートフォンなどの携帯端末を含む様々なプラットフォーム上でも授業や自学自習ができることになる。

ここで選択したideone.comはコンパイル・実行・デバッグを一括してオンラインで行うサービスであり、Webインターフェース上からソースコードを入力すると、サーバ側でコンパイルや実行ができ、統合開発環境（IDE：Integrated Development Environment）を意識したツールとなっている（図4）。ブラウザに表示されるURLアドレスを用いることで、実習したソース



図4 ideone.comの実習画面例

画面上部にある入力領域にソースコードを入力し、右下の[Run]ボタンで翻訳と実行が可能である。表示されているソースコードはC言語プログラミングI学習の最初期のものであり、初學者の学習はまずこのプログラムを動作させることから始める。

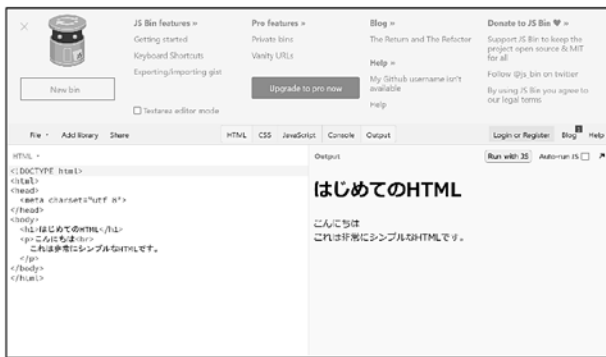


図5 JSBinの実習画面の例

画面左側にHTMLコードを入力すると、その結果の出力が即座に右側に反映される。中央部にCSSやJavaScriptの入力を可能とするボタンがあり、多彩なWeb実習に対応することができるようになっている。履修学生は学んだ知識を基に、左側のソースコード部分を書き替えながら授業をすすめていく。

コードを教員と共有し提出することができる。教員は受け取ったURLアドレスにアクセスし、ソースコードの添削を行い、結果を履修学生にフィードバックすることが可能である。だが一方で、ローカルに構築した環境のように、細かいカスタマイズには対応しておらず、またそもそもプログラミング教育用に構築されたものではないため、操作メニューやエラーメッセージは標準的である。このため、細部までこだわれば必ずしも初学者にわかりやすいシステムというわけではない。

JS Binもオンライン環境でHTML/CSS/JavaScriptのコードの結果をリアルタイムでプレビュー表示し、動作チェックができるツールである。ワードプロセッサのように表示結果をそのまま編集できるWYSIWYG形式とは異なり、Web技術に対応しソースコード入力と結果のプレビュー出力をそれぞれ別ウィンドウで管理するようになっている（図5）。ソースコードの保存方法など、それ以外の特徴は先に述べたideone.comと重なる点が多い。このサービスはWeb開発者などが、コードの簡易的な動作確認に用いる場合が一般的であり、これも本来は教育用に設置されているサービスではない。だが、いずれのサービスも使い方次第であり、こうしたサービスの機能を用いることで、初学者相手のコー

ディング教育の導入をスムーズに進める教材として活用できることが確認できた。授業では履修学生の実習利用だけにとどまらず、教員側であらかじめ用意した実習サンプルをURLとして提示することで、履修学生はキーボードからのソースコード入力を一切行うことなく授業を開始することも可能なのである。

4. 教育実践の評価

4.1 文字入力ミスの軽減に関して

今回構築したツールを利用することで、実習時に発生する入力ミスの一部を、履修学生自らが確認し修正できる環境を準備することができた。ツールの有用性を実感できた学生は、授業導入時だけでなく、履修期間を通じて実習に活用している。

本実践では日本語文字入力に関する課題をWebツールのサポートにより解決したが、この方法以外にも別の解決策はあるだろう。文字種の問題、特に全角スペースを区別するためには、空白やタブ、編集記号などを表示する多機能なエディタ（例としてVisual Studio Code や サクラエディタ）を用いたり、全角スペースを自動修正するエディタを利用すればよいという指摘はありえる^[10,11]。また、そうした機能がなくても、エディタの検索機能を積極的に活用することで、ソースコード中に不正文字が含まれていないかのチェックが可能であろう。リモート授業ではなく、教室で1つ1つ順を追って行う授業が機能すれば、高機能なツールの使いかたを扱うことは可能である。実際に担当する授業においても、学習が進んだ段階でこうしたエディタの機能を紹介し活用するようにしている。しかし、初学者にとっては、このような便利な機能をもつエディタの利用も、導入時の学習項目を増やし、新たな学びのハードルとなりうるものである。

4.2 コーディング教育のオンライン化に関して

同様に、Webブラウザ経由の実習を授業に導入した結果についても、学習導入時の学習項目を軽減し、学びのハードルを下げる効果が確認できた。学習をブラウザ上から開始するこうした方法では、アプリ操作に

困惑する学生はほとんどおらず、実習の結果に「思った以上に簡単だった」という意見も散見されるようになった。加えて、こうした外部ツールは、授業での学びとは切り離しても利用できる点や、特定の学習内容に限らずIT業界で利用される多くのプログラミング言語にも対応している点など、科目修得後の継続学習の面でも有用であるといえる。

こうしたサービスは、インターネット上で他のユーザーと共同利用することから、プログラムサイズやプログラム実行に関してCPU使用時間上の制限が設定されている。このため無限ループのような教材の実行では、処理途中でプログラムが中止させられてしまう。よって、行数が短く負荷が軽いプログラムが多い初学者向けの学習に適している。

導入時教育に適した利点を確認できる一方で、Webサービスに共通するデメリットも存在する。まず、これらはインターネット上で実行されるサービスであるため、ネットワークが接続されていないオフライン環境では使用できない点である。ネットワークが安定して接続されること、サービスが実行されているサーバがメンテナンスなどで止まることなく継続的に運用されていることが必須となる。ネットワークサービスが停止するトラブルは、ここ数年の授業中に起きた経験はないが、まれにサービスのレスポンスが大きく低下することは発生している。インターネットが発達した現在においてもサービス停止のリスクはあるので、そうした事態に備えた代替教材の準備は必要であろう。

また、無料サービスを利用していることもあって、ブラウザ内にインターネット広告が表示される点もある。これはサービスを無料で運用するために必要な手法なのかもしれないが、広告のなかには授業時にふさわしくない内容のものがあ、対策の必要性を感じている。

さらに、実習結果のソースコードはサーバ上に保存され、その情報をブラウザ上に表示されるURL形式で参照する仕組みになっているが、これもトラブルの原因となることがある。ブラウザ操作は簡便ではあるが、それゆえ簡単に他のページに遷移するため、実習URL

情報の取得に失敗して入力したソースコードや実習結果を失うリスクがある。こうした実習結果が消失してしまう事例は毎年散見される。そのうえサーバ上に保存されているソースコードや実習結果を、いつまで利用可能かとする保存期間などの条件を利用者側からコントロールすることができない。これは安定した授業運営上の懸念点である。これについては、ideone.comでは利用者登録しサインインすれば、より詳細なソースコード管理ができるようだが、利用時の簡便性とのトレードオフになる。

加えて、実習結果がインターネット上で共有されるため、他者に閲覧される可能性があり、履修者には入力に個人情報などが含まれないように注意徹底している。

デメリットへの対応や、より本格的な学習へ進めるために、現在担当する授業においては履修学生が実習に慣れてきた段階で、ローカルPC上に構築した旧来の実習方法に切り替えるようにしている。

本論文で扱った取り組みは、クライアント側で動作したり、インターネット上のサービスを利用したりしており、いずれにおいても学習者の学習記録（ログ）取得が困難である。筆者は過去に、コーディング教育を遠隔接続（telnet）経由でUnixサーバに接続し行っていた。この際、学生が入力するソースコードや翻訳時のエラーメッセージなどの実習記録を収集し、学生指導に応用してきた^[34]。今回の実践についても、独自に教育用サーバ上でサービスを構築することで、より進んだ教育研究が行えることが期待できる。

5. まとめと今後の課題

筆者が担当するコーディング教育について、入力ミス修正をサポートするWebツールを開発・構築し、加えてブラウザ上で実習が行えるWebサービスを導入することで、学びのハードルを下げる授業実践を行った。作成したツール群や、授業資料、得られた経験はその後の授業でも改善しながら活用を続けている。本レポートは、教育者側からの視点での教育改善に重きを置いており、学習者の学習成果獲得についての定性的・定

量的な評価は不十分であり、これは引き続き授業改善を目的とした研究課題である。

新型コロナ禍以前にもWebコンパイラなどのシステム構築事例や授業活用の先行研究は報告されていた。しかし当時筆者はその有効性や可能性を感じつつも、正確にその価値を認識・評価するに至らなかった。新型コロナへの対応といった教育環境の劇的な変化が、システム構築や整備の重要性を改めて確認する良い機会となった。今後も新しい技術や授業運営方法を積極的に評価・研究し、継続的な授業改善に取り組んでいきたいと考えている。

最後に、ここでは現在担当するC言語プログラミングやWebサイト制作の授業について扱っているが、学習内容は高等教育以前の教育カリキュラムが変わっていく中で、修正が求められることもあるだろう。現にプログラミング教育は小学校では2020年度から、中学校は2021年度から、高校は2022年度から必修化され、今後本学に入学するすべての学生が、プログラミングの経験者となってくる^[12-14]。ただし、使用されるプログラミング言語は統一されておらず、表記法や学習分野に大きな違いが存在するという報告もある^[15]。加えて必修化後2023年に発表された調査によると、高校でのプログラミング教育の現状には、小中高で接続された情報教育カリキュラムが未整備であることもあり、生徒のスキルなどにばらつきがあることが示されている^[16]。本学に入学する学生の傾向として、理数系の内容を苦手としている学生が目立つこともあり、今後もソースコードを扱うコーディング教育は、学生には難しい授業と認識される可能性が高い。このため、引き続きプログラミング教育に苦手意識をもつ学生への配慮が求められることが想定される。そうした現状を踏まえながら、高校のカリキュラムとの接続への対応が急がれる。

参考文献

- [1] 国立教育政策研究所 教育研究情報データベース, “学習指導要領の一覧”, <https://erid.nier.go.jp/guideline.html> (参照 2024年12月01日)。
- [2] 江上邦博: “PCを利用した教育用サーバ環境の構築と評価”, 千葉経済大学短期大学部 経営情報論集, 第18号, 1-21 (2002)。
- [3] 江上邦博: “情報機器を活用した教育法改善の試み”, 千葉経済大学短期大学部 経営情報論集, 第19号, 1-22 (2003)。
- [4] 江上邦博: “総合基礎科目としてのプログラミング教育の有効性”, 千葉経済大学短期大学部 経営情報論集, 第20号, 1-20 (2004)。
- [5] 江上邦博・内山隆: “ICTを用いた授業記録システムの構築と授業改善への応用”, 千葉経済大学短期大学部 研究紀要, 第5号, 63-73 (2009)。
- [6] “Visual Studio Community”, <https://visualstudio.microsoft.com/ja/vs/community/> (参照 2024年12月01日)。
- [7] “GCC, the GNU Compiler Collection”, <https://gcc.gnu.org/> (参照 2024年12月01日)。
- [8] “ideone.com”, <https://ideone.com/> (参照 2024年12月01日)。
- [9] “JS Bin - Collaborative JavaScript Debugging”, <https://jsbin.com/> (参照 2024年12月01日)。
- [10] “Visual Studio Code”, <https://code.visualstudio.com/> (参照 2024年12月01日)。
- [11] “サクラエディタ”, <https://sakura-editor.github.io/> (参照 2024年12月01日)。
- [12] 文部科学省: “第3章 プログラミング教育の推進”, https://www.mext.go.jp/content/20200608-mxt_jogai01-000003284_004.pdf (参照2024年12月01日)。
- [13] 文部科学省: “GIGAスクール構想について”, https://www.mext.go.jp/a_menu/other/index_0001111.htm (参照 2024年12月01日)。
- [14] 竹中章勝: “高等学校専門教科「情報科」—現状とこれからそしてわれわれができること—”, 情報処理, Vol. 61, No. 10, 1054-1057 (2020)。
- [15] 井手広康: “情報Iの教科書におけるプログラミング分野の比較と考察”, 情報処理学会論文誌 教育とコンピュータ, Vol. 8, No. 3, 8-18 (2022)。
- [16] みんなのコード: “「2022年度 プログラミング教育・高校「情報I」実体調査報告書」”, <https://speakerdeck.com/codeforeveryone/2022nian-du-hurokuraminkujiao-yu-gao-xiao-qing-bao-i-shi-tai-diao-cha-bao-gao-shu> (参照2024年12月01日)。