# プログラミング教育における「写経型演習先行-文法学習後行方式」の実践報告

松本　幸子[1]

## １．はじめに

プログラミング学習において、初学者がテキスト型プログラミング言語を使用したコード作成技能を習得する過程を考えてみると、最初にプログラミング言語の文法を学習し、次に教材に示されたプログラムを模倣する段階を経て、自らコードを考えプログラムを作成する段階へと学習を進めていくのが一般的である。こうした過程の中でも、特に、模倣により提示されている実行可能な完全なプログラムのコードを打ち込み、プログラムの実行結果を確認する段階の学習は「写経型学習」と呼ばれている[1) 2) 3)]。

初学者にとって、この写経型学習の過程は、プログラムの動作がコードの記述を反映したものであることを認識し、プログラミング言語の命令や構文の意味を理解するために欠かせない過程である。しかし、形骸化した写経型学習を漫然と行っていても、コードの記述とプログラムの動作との対応関係を認識できず、コードの意味についての理解が進まないことも多い。そこで、筆者の担当するサイバー大学（以下「本学」と呼ぶ）におけるC言語の初学者向けのプログラミング教育科目（以下「本科目」と呼ぶ）では、2017年度より、各授業回の文法学習をまとめて行わずに、小さなステップに分けて最小限の知識のみを元にした写経型演習を繰り返し、全ての演習を終えた後に文法学習を行う「写経型演習先行・文法学習後行方式」を導入している。これは、文法などの知識学習よりも、実際の使用場面での具体的な意味を理解する経験の積み重ねにより言語の習得を目指そうという、自然言語習得に関する研究を参考にした取り組みである。そして、この方式は学生からもおおむね好評を得ている。本学は全ての授業をインターネット上で提供する通信制大学であるため、これらの学習活動は全てインターネット上で展開されている。本稿では、この完全ｅラーニングによる「写経型演習先行・文法学習後行方式」のプログラミング教育実践について報告する。

[1] サイバー大学 IT 総合学部・准教授

## ２．第二言語学習を参考にしたプログラミング言語学習方略

### 2.1. 第二言語学習における問題との類似性

従来、高等教育機関におけるプログラミング教育科目では、授業の中で、①プログラミング言語の文法解説、②その文法事項が用いられた例題プログラムを模倣しながらのコード入力と実行結果の確認を行う演習、③例題プログラムの改変を中心とした基本演習、④学習者が自ら考えてコードを作成する応用演習、という順で学習を進めていくのが一般的な学習順序であった。このうち、④の応用演習は授業の事後学習として課される場合も多かったが、文法学習と例題プログラムの模倣と改変による演習に取り組んだだけでは、自らコードを考えてプログラムを作成するところまでのプログラミング言語運用能力を発達させるのが難しいことが課題であった。

一方、第二言語（母語以外の自然言語）を学習する場合にも、文法学習、模倣による学習、改変による学習、応用学習、というのは一般的な学習順序ではあるが、長らく日本でも学校での英語教育に対して指摘されてきたように、精読（Intensive Reading）に取り組みながら語彙や構文などの文法を学習し、例文の模倣や改変による練習を行うだけでは、自ら作文や発話を行う応用段階に進んでいくのが難しく、なかなか実用的な言語運用能力が身につかないことが課題であった。こうしたプログラミング言語学習とも類似した問題をかかえる中、近年ではこれを解消する学習法として、文法や語彙などを明示的な知識として学習するよりも、その言語での表現に触れて理解する経験の積み重ねによって習得するという、言語に接する経験量を重視した「多読」（Extensive Reading）「多聴」（Extensive Listening）といった学習法が注目されている [4) 5) 6) 7)]。

### 2.2. 第二言語学習におけるインプット重視の学習法

第二言語習得研究においては、「読む」「聞く」という学習者が言語を用いた表現に接する経験、すなわち「インプット」が重要とする研究が数多く存在する。Krashen は、第二言語の能力を発達させるには、意識的な「学習」(learning)と無意識的な「習得」(acquisition)という区別された２つの方法があるとした。そして、意識的な「学習」で身につけた能力では、学習者が発する表現すなわち「アウトプット」をモニターし修正を加えることしかできず、「アウトプット」自体は母語習得過程と類似の無意識の「習得」で得た能力によってのみ産み出されるもの、「習得」は「インプット」の内容を理解する時に起こるもの、などとした仮説を唱えた [8)]。

この、文法学習やアウトプットの練習などといった意識的「学習」は、言語習得に寄与しないとする Krashen の仮説に対しては異論も多い。しかし、Krashen はアメリカの学校における第一言語指導の研究から、語彙、語法、綴りなどの技能を意識的に学習させ誤りを修正させるような直接的指導ではリテラシーを発達させられず、自由読書だけが効果

をあげていることを示した。そして、「言語を覚えること、つまり習得は、情報のインプット（入力）によるもので、情報のアウトプット（出力）によるものではありません。また、理解によるものであり、表現によるものではありません」[9] と述べるなど、言語の知識を学習するよりも言語により表現された内容の理解に主眼を置くべきとして、インプットの重要性を強く主張した。

インプット重視の学習法としては、Day & Bamford は学習者が辞書などに頼らずに自ら理解できるレベルの本を自由に選んで楽しみながら大量に読む「多読」（Intensive Reading）を提唱した[4]。日本では、酒井らが英語学習法として 100 万語という読書量の目標を掲げ、多読三原則（辞書を引かない、わからないところは飛ばす、つまらなければ途中でやめ別の本へ）に則り、英語学習者向けに語彙、文法、文長が制限された読本（Graded Readers）を読んでいく学習法を提唱した[5] [6]。さらに、読書の際に読本の朗読を聴く「多聴」（Intensive Listening）を組み合わせる手法も提案されている[7]。多読多聴学習は、学校教育へ取り入れるにあたっては、多くの読本を用意する必要があるなど導入ハードルが高いため、学校教育における広がりは限られる。しかし、多読多聴について多くの学習効果が報告されており[10] [11] [12]、インプットの重要性については広く認知されるようになっている。

## 2.3. プログラミング言語学習におけるインプット重視の学習方略

プログラミング言語は、自然言語に比べれば、圧倒的に文法は単純で例外もなく、覚えるべき語数も少ない。しかし、これらの知識を学びさえすれば、プログラミング言語が習得できコードが書けるようになるというわけではなく、具体的な用例を通した習熟が必要となる。そればかりか、文法学習、模倣による学習、改変による学習、という一般的な学習段階を踏んでもなお、自らコードを作成する応用段階に至るのは難しい。それならば、プログラミング言語学習においても、第二言語学習において効果を上げているインプット重視の学習方略が効果的なのではなかろうか。プログラミング言語も「言語」というからには、プログラムを作成するというアウトプットの練習や、それに先立つ知識としての文法学習よりも、インプットを増やすことを重視した学習方略、つまり、多くのプログラムのコードに触れ、その意味を理解する経験を増やし、経験の積み重ねによってプログラミング言語を習得しようという学習方略には効果があるはずである。

ただし、プログラミング言語によって記述されている内容は、実際にそのプログラムを実行してはじめて、プログラムの動作として認知されるものである。自然言語で表現されている内容であれば、前後の文脈や挿絵、話者の身振り手振りなど、読む・聴くというインプットと共に認知される情報から類推し、内容を理解することも可能であろう。しかし、プログラミング言語で書かれているプログラムの場合、その内容を理解するにはプログラムの動作を観察することが不可欠となる。つまり、提示されたプログラムのコードと実行時の動作とをあわせて観察する演習を通して、コードの記述の意味を理解する経験を増や

すことが重要となる。

こうした演習では、必ずしも学習者が自らタイピングしコード入力を行う必要はない。しかし、完全にタイピングによるコード入力を省いてしまうと、学習者にはコードが長い呪文のように見えてしまい、構造や意味を持った記述として認識できなくなってしまう懸念がある。また、プログラミング言語は自然言語と違い、僅かな綴りや記号の違いでも意味をなさなくなる、あるいは、意味が変わってしまうという点も、学ぶべき事項の一つである。したがって、コードの理解に影響を及ぼさない範囲でコード入力の手間を省く工夫は施しつつも、基本的には、学習者が自らタイピングすることにより提示されたプログラムのコードを丸写しして入力し、プログラムの動作を確認するという「写経型」の演習を学習の中心とすべきであろう。

結局、第二言語習得研究を参考にすれば、文法学習や学習者自身がコードを考えてプログラムを作成する演習よりも、まずは写経型の演習によってコードの意味を理解する経験を積み、経験を通してある程度まで言語が理解できるようになった段階で、その経験を振り返り整理する形で文法学習を行い、その後に自ら考えてコードを作成する応用演習へと学習段階を進める、という学習方略が有効であろうと考えられる。

## ３．プログラミング言語学習に特有の課題とその対策

プログラミング言語学習には、第二言語学習の場合と類似した問題が存在することは前述の通りであるが、さらにプログラミング言語学習に特有の事情というものも存在する。第一に、そもそもプログラミング言語で表現されている内容は、日常生活では使われていない、プログラミング特有の概念に基づいたものである。第二に、プログラミング初学者は、プログラミング特有の概念の表現が可能な既知の言語を持たず、比較対象となる言語の知識を持ち合わせていない。第三に、内容を伝える相手は人間ではなくコンピュータであるため、些細なタイプミスによってもプログラミング言語を介したコンピュータへの内容伝達に失敗するなど、プログラム実行までには困難が伴う。第四に、写経型学習においてはコードの意味理解という本来の目的が見失われやすい。本章では、これらの課題に対する方策を検討する。

### 3.1. プログラミング特有の概念に対する理解の重要性

第二言語学習の場合、学習している例文の内容は、母語である第一言語では表現することができていた内容であり、学習者にとっては既知の概念である場合がほとんどである。そのため、インプットを重視した多読学習のみならず従来の学習順序であっても、模倣や改変によるアウトプットの練習を通して、第二言語による表現とそれが示す概念との対応関係を認識することが可能であった。これに対し、プログラミング言語の初学者は、そも

そもプログラミング言語で表現されている内容そのものに対し、概念イメージを持っていない場合がある。

プログラミング言語は、変数などに対する操作として処理を記述し、さらにその処理の流れを作る制御構造を記述する言語である。そのため、プログラムのコードを理解するには変数、条件分岐、繰り返しといった抽象概念の理解が欠かせない。しかし、これらの概念は日常では接することのない概念であるため、学習者が類似の概念を学習したことがない場合には、これらの概念のイメージが形成されていないものと考えられる。こうした概念イメージを持たない初学者にとっては、例題プログラムのコードを入力しプログラムの実行結果を確認するという写経型学習に取り組むだけでは、コードとそのコードが表現している内容との対応関係がなかなか認識できない。つまり、写経型学習が、コードの意味を理解し、プログラミング言語の命令や文法を習得していくことにつながらない場合がある。

これに対する方策としては、ソースコードという具体例を題材とした学習を始める前に、あらかじめ抽象的な概念のイメージを形成しておくことが有効である。例えば、プログラミング言語の学習を始める前に、制御構造を表す図を用いて処理の流れに対する明確なイメージを形成しておくことは、プログラミングの理解度向上に効果があったと、長谷川、山住により報告されている[13) 14)]。また、近年、小学校教育へも広く導入の進んでいる、ビジュアル型プログラミング言語を用いたプログラミング教育も、概念の獲得に大きく寄与するものと考えられる。本学では、ビジュアル型プログラミング言語を用いたプログラミング教育科目も開講しており、プログラミング初学者に対しては本科目受講前の履修を推奨している。また、本科目では、毎回の授業冒頭で、動画視聴による概念イメージ形成のための学習を実施している。

### 3.2. 既知のプログラミング言語を持たない初学者の学習方略

プログラミング言語の文法学習は、初学者にとって、自然言語である第二言語学習の場合の文法学習とは全く状況が異なる。第二言語の文法学習では、獲得済みの第一言語の文法との比較が可能であるため、具体的な表現に接した経験がわずかであっても、文法学習が漠然とであれ文法について何かしらの予備知識を得る機会となり、第二言語による具体的表現に接する際の理解促進につながるだろう。しかし、プログラミング言語の学習経験を持たない初学者の場合、比較対象となりうるプログラミング特有の概念を表現可能なプログラミング言語の知識を、一切持ち合わせていない。そのため、自然言語の文法と比べ圧倒的に単純であることを考慮しても、それを理解するのは第二言語学習の場合より困難となり得る。

実際、従来の学習順序を採用していた過去の学期には、プログラミング演習に先立つ文法学習に対し、「文法を説明されてもピンとこない」という感想が寄せられていた。この感想に代表されるように、プログラムのコード作成や実行結果の確認などに取り組む前の文法学習は、既知のプログラミング言語を持たない学習者にとっては理解し難く、効果が低

いものと考えられる。初学者にとって、プログラムのコードは、括弧をはじめとした種々の記号が使用されているなど、日常の文章では見慣れない形式も相まって、意味や構造を持った記述として認識するのが難しいものである。そのような状況では、より抽象的な文法学習は効果が上がらないのも当然である。

このような既知のプログラミング言語を持たない状況、そして、学習している言語で表現される概念自体も十分には把握できていない状況を鑑みるに、初学者におけるプログラミング言語の習得過程は、第二言語の習得過程よりも、むしろ幼児が母語である第一言語を習得する過程と類似しているというべきであろう。第一言語は、その言語にさらされる状況の中で、表現が示す概念と共に獲得されていくものである。したがって、より第一言語習得過程に近い、文法などの知識学習よりもインプットの意味理解に重点をおく学習法は、第二言語習得の場合以上にプログラミング言語習得には有効であろうと考えられる。つまり、プログラミング言語学習においては、文法学習よりも写経型学習を先行させる学習方略が合理的といえよう。

### 3.3. プログラム実行までの困難軽減のための取り組み

自然言語を学習している場合、学習者が書く・話すといったその言語を用いたアウトプットによって内容を伝えたい相手は人間である。単語の綴りや発音、あるいは構文や語句の使い方などに多少の間違いがあったとしても、学習者は相手におおよその内容を伝えることができる。つまり、たとえ言語が十分に習得されていない状態であっても、例文の模倣ではなく学習者が自ら考えた多少の間違いを含む表現であっても、アウトプットを試みれば、それがただちに一定程度の成功をおさめる。従って、精読と語彙や文法などの知識学習を行った後にアウトプット練習を繰り返すということによっても、語彙や文法の定着をはかることは可能であろう。

一方、プログラミング言語の場合は、内容を伝えたい相手はコンピュータである。そのため、コンパイルや実行などの操作の誤りやコード上のわずかなスペルミスによっても、コードの内容はコンピュータに伝わらなくなる。つまり、エラーが出てプログラムが実行できない、あるいは実行はできても正しく動作しないといった状態になり、往々にして、模倣によりプログラムを入力し実行するという、写経型学習段階における試みさえも失敗してしまう。さらに、プログラミング言語の理解が不十分なまま、例示されたプログラムを改変する段階、獲得した知識を活用してプログラムを作成する段階へと進もうとすれば、構文や命令の誤用などでより多くのエラーに遭遇することになる。学習者にとって、デバッグ作業は非常に負荷の高い作業であるため、たとえ写経型学習の段階であっても、プログラムの実行結果が容易に得られるとは限らない。

こうしたプログラム実行までの困難を軽減し、学習効率を向上させるための取り組みとしては、教師の手助けや学習者同士の協力が期待できる対面での授業時間を、躓きの発生しやすい後半段階の学習へと振り向けるために、前半段階の学習を事前学習として課して

おく反転授業の取り組みが挙げられる。動画教材により文法学習を事前学習として課す手法（高井，水谷[15]）、文法学習だけでなく例題プログラムの模倣による写経型学習までを事前学習とする試み（喜多，岡本[3]）、動画教材による学習と確認問題への取り組みまでを事前学習としておき授業にはグループによる協調学習を導入する取り組み（林ら[16] [17]）などが報告されている。

いずれも演習の際の躓きを早期に解決し、多くの演習課題に取り組む機会をつくる効果があるものと考えられる。しかし、学習者のタイピングスキルなどによっても進度や理解度に大きな差が生じるという、プログラミング学習の特性による課題は依然として残る。そこで、学習者自身のペースで、周囲の助けを待たずに自立的に進められる学習方法として､躓き要因の比較的少ない写経型学習を質､量ともに充実させていくことが肝要となる。特に、対面の授業時間を持たない完全 e ラーニングで授業を進める本学においては、学生の演習状況を確認できる演習環境を利用し、学生の質問には複数の教員とティーチングアシスタント（以下「TA」と呼ぶ）により迅速に対応する体制を整えてはいるものの、さらに写経型学習を通じて最大限の学びを引き出すことが求められている。そのための方策として、一般に写経型学習と呼ばれる模倣による学習の段階だけでなく、改変による学習の段階までを、コードと実行結果を明示した写経型の演習により進めている。

### 3.4. 写経型学習が陥りやすい学習目的忘失

写経型学習の本来の目的は、ソースコードを読み、模倣してコード入力し、実行結果を確認する過程を通して、プログラムの動作がコードの記述を反映したものであることを認識し、写経対象のプログラムの持つ基本構造と個々の命令や構文の意味を理解すること、さらには、コードという具体例から文法を抽出し習得することである。しかし、写経型学習は、提示されたコードの丸写しによって演習を進めていくため、盲目的な作業として漫然と進めることも可能である。また、この単なる模倣の段階でさえ、実行までには一文字でもタイプミスがあるとプログラムが実行できなくなるなどの困難が存在するため、プログラムの実行自体が目的化してしまい、本来の学習目的を見失ってしまうことも多い。すなわち、タイプミスなどによるコンパイルエラーを無くし実行にこぎつけただけで満足してしまい、肝心のコードに着目することができず、プログラムの実行結果がコードの記述を反映したものと認識してコードの意味を理解するに至らなくなるのである。

こうした写経型学習が陥りやすい演習の形骸化を防ぎ、コードの意味理解という本来の目的に対する意識を維持するためには、躓き予防策を講じてプログラム実行までの困難を軽減するとともに、コードの内容に着目させる工夫が重要となる。これまでには岡本・喜多により、認知的負荷軽減のためプログラムの動作を視覚的に顕在化させるような写経用教材の開発、実行可能な完全なプログラムと実行例の提示、学習者がエラーに際し対処法を逐次参照できるような教材の準備、エラーの明示的経験、記号の識別力強化やコミュニケーション円滑化のためのコード音読・訳読への配慮などの取り組みが報告されている[2] [3]。

本科目では、さらなる工夫として次のような対策を講じている。

**１）模倣から改変までのスモールステップの積み重ねによる写経型演習**

本科目では、基本的なコードの模倣だけでなく、改変の過程までをコードを提示しての写経型の演習によって扱っている（以下、模倣だけでなく改変までの過程を含めて「写経型演習」と呼ぶ)。演習は、ごく短いプログラムのコード入力と実行結果の確認から始め、さらに、そのコードに対し少量ずつ改変や追記を行っては、実行結果の変化を確認するという演習を、手本を参照しながらの写経により繰り返す。実行結果の確認にあたる際の、その度毎のコード入力量を減らすことにより、コード中の着目すべき箇所を明確化して意味理解を促進するとともに、タイプミスによるエラー発生時に原因箇所の特定を容易にし、デバッグの労力軽減をはかっている。これらの工夫により、タイピングの遅い学習者であっても、できるだけ多くのコードと実行結果の対応を観察する経験を積めるよう、そしてコードの意味理解に至ることができるよう配慮している。

**２）写経型演習先行・文法学習後行という学習順序の採用**

文法学習には、演習前に取り組むのではなく、テーマとなる概念と最低限の知識のみを与えられた状態で写経型演習を行った後に取り組むこととした。この「写経型演習先行・文法学習後行」という学習順を採用したのは、前述のような、既知のプログラミング言語を持たない初学者にとって演習前の文法学習はあまり効果が見込めない、という消極的な理由からだけではない。この学習順の採用は、演習に際しコードの意味理解に対する目的意識を維持しようという積極的な理由からでもある。コードの意味をあらかじめ詳しく解説してしまうことなく、実行結果からコードの意味を類推する余地を残しておくことにより、演習の際に謎解き感覚でコードの意味を考えさせ、演習目的を見失わせないようにすることを狙っている。つまり、演習の形骸化防止効果を期待した取り組みでもある。

**３）課題でのエラー発生からデバッグまでの明示的体験**

写経型演習においても、エラー発生に伴うデバッグ作業は避けては通れない。慣れない初学者にとって、原因の究明とプログラムの修正は難しい作業であり躓き要因となる。コンパイルエラーが発生してプログラムが実行できない場合は、エラーメッセージが原因特定の助けにはなるだろうが、デバッグが負荷の高い作業であることに変わりはない。また、実行できるが正しく動作しない場合は、動作が正しくないことに気付かない場合や、気付いたとしても、コードに誤りがあるという発想にさえ至らない場合も多い。エラーの発生箇所は学生によって異なり一定しないため、原因究明やバグ修正についての一斉指導には限界があり個別指導が必要となる。しかし、どれだけ学生からの質問に対応し個別指導する体制を整えたとしても、学生自身が教員や TA に質問するなどの問題の解決に向けた行動を取らねばならず、個別指導に至るまでには一定のハードルが存在する。つまり、躓いた時に解決のための質問をすることなく、投げ出してしまう学生も多い。そこで、本科目

では、バグを含むコードを示し、そのバグの症状や修正方法についての選択式問題に答えながら、演習環境で実際にデバッグを行ってプログラムを完成させる課題を課し、エラーへの対応力強化をはかっている。

## ４.「写経型演習先行・文法学習後行方式」の実践と評価

本科目において、2022 年度春学期に実施した授業の具体的な進め方と、学期末に行ったアンケート結果について以下に報告する。

### 4.1. 本科目の構成

本科目は全 15 回で構成されており、学習内容は表 1 の通りである。本学はインターネット上で授業を提供している通信制大学であるため、学生は、動画コンテンツ視聴、プログラミング演習、テキスト主体のコンテンツ学習、小テスト、レポート課題、期末試験など全ての学習活動に e ラーニングにより取り組み、疑問点についてはメール等で教員や TA に質問する。演習には、goorm という Docker コンテナで提供されるクラウド開発環境を利用しており、学生からの問い合わせに応じて、教員・TA が各学生の演習環境内に入って演習状況を確認できるような体制を整えている。

各回は、表 2 に示すように、第 1 章～第 3 章として提供される 3 種の授業コンテンツ（解説動画、演習用テキスト、文法学習用テキスト）と課題（小テストあるいは Web レポート）で構成されている。課題としては、第 6 回、第 9 回、第 13 回では、５～６個のプログラムを、コードの穴埋めやバグ修正方法についての選択式問題に答えながら、演習環境で完成させる Web レポート課題を課し、その他の回では、比較的負荷の少ない全 8 問の小テストを課している。学生は、回ごとに決められている受講期限までに、これらの学習活動に取り組むことで各回の出席が認定される。そして、学期末には選択式問題からなる期末試験も課される。この他、第 14 回には、決められた基本仕様に各自で機能を追加するなどの工夫を加えてプログラムを完成させる、任意提出の課題も用意している。

**表１　本科目の構成（全 15 回）**

| 回 | 学習内容 | 課題 |
|---|---|---|
| 第 1 回 | ビルド・実行の方法 | 小テスト |
| 第 2 回 | コード入力・標準出力 | 小テスト |
| 第 3 回 | データ型・変数・標準入力 | 小テスト |
| 第 4 回 | 算術演算子・代入演算子 | 小テスト |
| 第 5 回 | 条件演算子・if 文・if～else 文 | 小テスト |
| 第 6 回 | 論理演算子・if 文・switch 文 | Web レポート |
| 第 7 回 | while 文 | 小テスト |
| 第 8 回 | for 文 | 小テスト |
| 第 9 回 | 配列 | Web レポート |
| 第 10 回 | 関数 | 小テスト |
| 第 11 回 | ポインタ | 小テスト |
| 第 12 回 | 文字列 | 小テスト |
| 第 13 回 | 構造体 | Web レポート |
| 第 14 回 | ファイル入出力 | 小テスト　任意提出課題 |
| 第 15 回 | 複数ファイルでの開発 | 小テスト |
| 期末試験 | ---------- | 期末試験 |

**表２　各回の授業コンテンツの構成**

| | |
|---|---|
| 第 1 章 | 動画視聴（10～15 分）<br>・前回の復習<br>・新出概念の説明とキーワードの紹介 |
| 第 2 章 | プログラミング演習<br>・テキストを参照しながらの写経型学習<br>・連載形式で示される指針に従い<br>各自のプログラム改良に取り組む応用学習 |
| 第 3 章 | テキスト学習<br>・文法のまとめ<br>・チェック問題 |
| 小テスト<br>または<br>Web レポート | 小テスト（全 8 問）<br>・文法確認と演習実施確認<br>Web レポート（全 25 問程度）<br>・第 6，9，13 回で実施<br>・穴埋め、バグ修正などについての選択式問題 |

## 4.2. 各回の構成

各回の第１章は、前回の復習と、その回で新たに登場する概念や、演習に必要な最低限の知識についての説明を行った 10～15 分程度の動画である。前回の復習を行っているのは、これから新たに学習する内容をこれまでに学習した文法事項と組み合わせて利用するという意識を持ってもらうためである。また、新たな概念については、例えば変数の宣言なら「花見の場所取り」を想起させるイラストを示すなど、図やイラストによるイメージ形成に主眼を置き、文法の詳細には立ち入らずに、キーワードとなる予約語や関数名を動画内で紹介することにより読み方を示す程度にとどめている。

続く第２章は、演習用テキストとなっている。学生はこのテキストを参照しながら、提示されているコードを模倣しタイピングして入力する「写経」を行い、まずごく短いコードのプログラムを作成する。そして、テキストの指示に従い、このプログラムのコードに少量の変更あるいは追記を加えては､実行結果の変化を確認する｡こうしたスモールステップの演習を繰り返し、このプログラムを次第に長いプログラムへと発展させていく。演習用テキストには、変更や追記を行う部分だけでなく、プログラム全体のコードと実行結果を示しており、特に変更・追記部分のコードは色分けにより明示し、紛らわしい記号などとともに注釈を付記している。通常１回の授業で完成させるプログラムは、変更や追記を繰り返して作成するプログラム２個程度とし､新たなプログラムの作成作業に移る際には、再び短いプログラムの作成から始めるか､あるいはコード入力の手間を省けるよう､コピー可能な形式で提供されたプログラムから始める。

また、写経型演習によって得た知識を活用し、自らコードを考える段階へと学習を進めるための応用演習も用意している。具体的には、テーマとなっているプログラム（｢データ管理プログラム」と「数当てゲーム」）に、複数回にわたり、新たに学習した知識を踏まえて、機能を追加するなどの改良を施していき、次第に複雑で大きなプログラムを構築していく演習である。テキストには、第３回から第 14 回までの連載形式で、改良の方針とそれを実現するためのヒントを載せ、次の回でコーディング例を示している。

第３章は、文法のまとめと文法の理解を確認するための解答付きチェック問題からなる、テキスト主体のコンテンツである。第２章に従って演習を行った後に、この第３章に取り組むことで、演習で学んだことを文法知識として整理し定着をはかっている。

これら第１章から第３章までの授業コンテンツによる学習の後に、通常（Web レポートを課している回以外）は、必須課題として小テストを課す。小テストは、文法理解を確認するための選択式問題７問の他に、第２章の演習用テキストで伏せておいた実行結果を答える記述式問題１問を加えて計８問とし、毎回の演習を実施したかどうかの確認としている。

第６回、第９回、第 13 回では、小テストは課さずに Web レポートを課す。これは、数多くのプログラムを読む経験をさせるとともに、エラー体験によりバグへの対応力を強化することを目的とした課題である。学生は、提示されている５～６個の短い不完全なプログラム（空欄やバグを含む）を、設問に回答しながら、実際に演習環境を利用して完成さ

せる。設問は、空欄に当てはまるコード、不具合症状、バグ修正方法などを選択肢から選ばせる形式を主体とし、難しくなりすぎずに多くのプログラムのデバッグ作業を経験させる課題としている。提示する不完全なプログラムはコピー可能な状態で提供し、コード入力の負担を軽減する工夫をしている。また、コンパイルエラーの発生するバグだけでなく、実行は出来るが意図と違う動作をするバグも経験させ、写経に夢中になるあまり、プログラムが実行できただけで満足しコードの意味を考えないという事態に陥らぬよう配慮している。

## 4.3. アンケート結果

以上のような 2022 年度春学期に実施した本科目の授業方式について、学期末にアンケートを実施したところ、受講した学生 408 名のうち 117 名から有効な回答が寄せられた。

### a. プログラミング学習経験について

アンケートに回答した学生の内訳は、プログラミング学習経験についての質問への回答によると、「テキスト型プログラミング言語の学習経験あり」37 名、「主にビジュアル型プログラミング言語の学習経験のみ」35 名、「プログラミング学習経験なし」45 名であり、図 1 のような構成であった。本学にもビジュアル型プログラミング言語を用いたプログラミング入門科目が用意されているが、全くプログラミング学習経験のない者も多く、プログラミング特有の概念についてイメージが形成されていない者も、相当数存在するであろうことは想像に難くない。全体としても、テキスト型プログラミング言語の学習経験の無い者が 7 割近くにのぼっていて、テキスト型プログラミング言語に関しては初学者が大半を占める状況となっている。

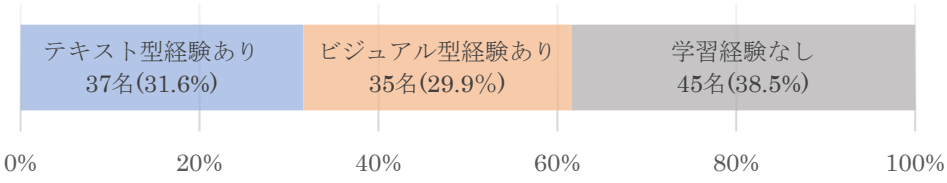


**図 1　プログラミング学習経験**

### b. 学習方式について

従来の「文法学習後に課題プログラムのコードを自分で考える演習を行う方式」と比べ、本科目で採用していた「提示されたコードを入力し実行結果を確認することを中心とした演習を行い、演習後に振り返りとして文法学習を行う方式」は効果的だったと思うかを尋ねた質問に対する回答を、学習経験別に集計した結果を表 3 に示す。それぞれの割合は図 2 の通りであった。

テキスト型プログラミング言語の学習経験を持つ者は、既知のプログラミング言語の文

法知識を有しているため、文法学習を先に行った方が効率的と考える者も多いのではないか、とも予想されるところであった。しかし、実際には、テキスト型プログラミング言語の学習経験を持つ者についても「従来方式の方が良い」と答えた者は 8.1%に過ぎず、「非常に効果あり」と「ある程度 効果あり」を合わせれば、実に 91.9%という多くの者が、文法学習に先立って写経型演習を行う学習方式を支持した（図２）。

テキスト型プログラミング言語の学習経験を持たない者については、当然、従来方式による学習経験も持っていないため、二つの方式を比較することは不可能である。しかし、「従来方式による学習経験が無い方も、どの程度、効果的だったかをお答えください」と付記したところ、「どちらとも言えない」を選んだ回答者は 14%前後にとどまり、ビジュアル型プログラミング言語の学習経験の有無にかかわらず、８割以上が本科目の学習方式について効果を認める回答を選択した（図２）。

**表１　学習方式について**

| 人数（名） | 非常に効果あり | ある程度効果あり | 従来方式の方が良い | どちらとも言えない | 計 |
|---|---|---|---|---|---|
| テキスト型 学習経験あり | 13 | 21 | 3 | 0 | 37 |
| ビジュアル型 学習経験あり | 12 | 17 | 1 | 5 | 35 |
| 学習経験なし | 12 | 26 | 1 | 6 | 45 |
| 計 | 37 | 64 | 5 | 11 | 117 |

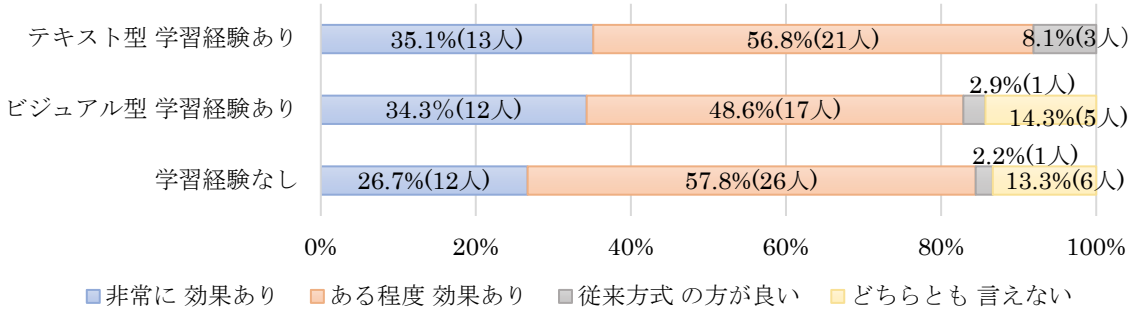


**図２　学習方式について**

## ５．まとめと今後の課題

テキスト型プログラミング言語の学習経験を持つ者については、従来の「文法学習後に課題プログラムのコードを自分で考える演習を行う方式」による学習を経験した者も多いと考えられる。したがって、こうしたテキスト型プログラミング言語の学習経験を持つ学習者の中で本科目の学習方式を支持する意見が多かったということは、テキスト型プログラミング言語の学習経験が無い者たちの意見にもまして、信頼度の高い結果だと考えられる。このように「写経型演習先行・文法学習後行方式」が、筆者の予想以上に支持を集める結果となったことから、文法などの知識についての学習よりも言語に触れる量を重視した学習は、自然言語習得だけでなくプログラミング言語習得にも有効だということが裏付けられたと考えている。

写経型演習が形骸化した作業に陥りやすいという懸念については、コードのごく一部の改変による実行結果の変化を観察させる、文法学習を後行させるなど、コードの意味理解に対する目的意識を維持させるような対策を講じたことにより、一定程度、払拭できたと考えている。一方、Web レポートとして課していたデバッグ体験課題については、演習を最後まで行わずに途中でやめてしまう学生を減らし、エラー発生に際しても投げ出さずにデバッグに取り組む姿勢を養うことに寄与できたと考えているが、エラーの修正方法を選択式問題で答えさせていることから、コードが誤っていたことやバグの修正方法については理解したものの、どのようにして誤ったコードによりおかしな動作が生じたのか、という新たな疑問をもつようになった学生も少なくなかった。今後の授業改善の取り組みとしては、このような、学生がコードについて深く考え疑問を持つという絶好のタイミングをとらえ、より充実した詳しい説明を与えていくことが、理解度を高めることにつながると期待している。

### 参考文献


1) 喜多 一, 岡本 雅子, 藤岡 健史, 吉川 直人, 「写経型学習による C プログラミングワークブック」, 共立出版, 2012.
2) 岡本 雅子, 喜多 一, 「プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察」, 滋賀大学教育学部付属教育実践総合センター紀要, 22 巻, pp. 49–53, 2014.
3) 喜多 一, 岡本 雅子, 「写経型プログラミング学習と反転授業」 第 60 回システム制御情報学会研究発表講演論文集, 2016.
4) R. Day, J. Bamford (著), 桝井 幹生(監訳), 「多読で学ぶ英語　楽しいリーディングへの招待」, 松柏社, 2006. （原著 “Extensive Reading in the Second Language Classroom”, Cambridge:

Cambridge University Press, 1998. ）

5） 酒井 邦秀, 「快読 100 万語！ペーパーバックへの道」, ちくま学芸文庫, 2002.

6） 古川 昭夫, 伊藤 晶子, 酒井 邦秀, 「100 万語多読入門―辞書を捨てれば英語が読める」 コスモピア, 2005.

7） 高瀬 敦子, 「英語多読・多聴指導マニュアル」, 大修館書店, 2010.

8） S. D. Krashen, “Principles and Practice in Second Language Acquisition”, 1982.

9） S. D. Krashen (著), 長倉 美恵子, 塚原 博, 黒沢 浩 (共訳), 「読書はパワー」（原著 ”The Power of Reading：Insights from the Research”, 1993）, 金の星社, 1996, p. 107.

10） 西澤 一, 吉岡 貴芳, 伊藤 和晃, 「英文多読による工学系学生の英語運用能力改善」 電気学会論文誌 A（基礎・材料・共通部門誌）, 126 巻, 7 号, pp. 556-562, 2006.

11） 熊田 道子, 「Extensive Reading（多読）による読み手の変化」 日本語教育方法研究会誌, 22 巻, 3 号, pp. 90-91, 2016.

12） 林 幸代, 丸尾 加奈子, 川瀬 義清, 長 加奈子, 「多読学習が英語読解に与える影響」 LET Kyushu-Okinawa BULLETIN, 20 巻, pp. 27-38, 2020.

13） 長谷川 聡, 山住 冨也, 「プログラミング教育と学習者のイメージ形成」, 名古屋文理短期大学紀要, 22 巻, pp. 9-14, 1997.

14） 長谷川 聡, 山住 冨也, 「プログラミング教育と学習者のイメージ形成（その２）」 名古屋文理短期大学紀要, 23 巻, pp. 9-14, 1998.

15） 水谷 晃三, 高井 久美子, 「プログラミング初学者を対象にした動画教材による反転授業の実践と評価」, 研究報告コンピュータと教育, 132 巻, 34 号, pp. 1-8, 2015.

16） 林 康弘, 深町 賢一, 小松川 浩, 「e ラーニング利用による反転授業を取り入れたプログラミング教育の実践」, ICT 活用教育方法研究, 16 巻, 1 号, pp. 19-23, 2013.

17） 林 康弘, 深町 賢一, 小松川 浩, 「プログラミング教育における反転授業の実践と評価」, 教育システム情報学会 第 40 回全国大会論文集, pp. 97-98, 2015.