原著論文

「情報I」を対象としたプログラミング的思考を習得するための制御構造と関数の可視化

大澤 崇琉[1)], 古川 宏[2)]
[1)] 筑波大学情報科学類, [2)] 筑波大学システム情報系

A visualization method of control structures and functions for program learning in Informatics I

Takeru OSAWA[1)], Hiroshi FURUKAWA[2)]
[1)] College of Information Science, University of Tsukuba
[2)] Institute of Engineering, Information and Systems, University of Tsukuba

概要: 高等学校の新設必修科目「情報I」において、プログラミング学習は重要な位置を占めているが、初学者の論理エラーの解決や教員の授業準備の負担が課題となっている。本研究では、問題解決において重要なプログラミング的思考の習得を支援するため、制御構造の流れを階層化や色分けで表現し、関数の値の受け渡しを画面のスライドで表現する新たな可視化手法を提案した。また、この手法を用いて「情報I」プログラミング分野の自己学習教材を開発した。高校生と大学生を対象とした比較実験を実施し、確認テストとアンケートにより、提案手法および教材の有用性を検証した。比較実験の結果、提案した手法は変数や配列、反復構造、関数の概念の理解に寄与する可能性が示唆された。特に次の変数の状態の事前可視化やアニメーションによる代入と参照の表現が変数の変化の理解を促進する可能性を示した。

Keywords: Informatics I, control structures and functions, visualization, program learning, learning support
キーワード: 情報I, 制御構造と関数, 可視化, プログラミング的思考, 学習支援

1. はじめに

1.1 研究背景

近年、高等学校における共通必修科目「情報I」の新設や、大学入学共通テストでの「情報I」の実施など、急速に進展する情報技術の活用を目的とした情報科教育の重要性が高まっている。一方で、教育現場では授業の準備時間や指導の難しさが課題として報告されており[1]、課題の解決には、多くの授業時間を要する「コンピュータとプログラミング」分野を対象としたICT教材の開発や導入が有効だと考えられる。

プログラミングの学習で初学者がつまずく要因に論理エラー（文法的には正しいプログラムが、期待された結果とは異なる挙動を示すエラー）がある。論理エラーには文法上の誤りが無く、コンパイラでの直接的な原因の検出が難しい。また、伊藤ら[2]は、プログラムの複雑化につれ、変数の値の変化や処理の流れの把握が困難になる事例を報告しており、反復構造や選択構造などの構造（以下、「制御構造」）を組み合わせたプログラムでの論理エラーは、解決がより困難になると考えられる。

2025年2月17日受理.（2025年3月1日スマートライフ学会2025年大会にて発表）

著者照会先: 〒305-8573 茨城県つくば市天王台 1-1-1
システム情報系 認知支援システム研 大澤 崇琉

論理エラーの解決には「プログラミング的思考」[3]（意図した活動を実現するために、必要な動きを分解し、対応した記号を組み合わせて論理的に改善をしていく能力）の習得が有効である。「意図する活動の実現」と「記号の組み合わせの改善」は、論理エラーの解決における「期待される動作の実現」と「プログラムの修正」に対応しており、「プログラミング的思考」の習得は、論理エラーの原因の特定と修正に対して極めて重要であると言える。

1.2 研究目的

「プログラミング的思考」の習得には、動きや記号の組み合わせを論理的に考えることが求められるため、プログラムの実行過程を追跡して流れを理解することが不可欠である。この点から、制御構造や関数の理解のために実行の流れを可視化することは、「プログラミング的思考」の習得に寄与する可能性が高いと考えられる。一方、既存の可視化手法では、条件式の評価順序や関数呼び出し時の処理の流れを追跡する機能が限定的であり、制御構造や関数の実行の流れを十分に可視化できていない。そこで、本研究では制御構造や関数の可視化を含む新たなプログラム可視化手法を提案し、論理エラーの解決や「プログラミング的思考」の習得に対する提案手法の有用性を検証することを目的とする。

2. 関連研究

本研究と同様に「情報I」の「コンピュータとプログラミング」分野における学習支援に焦点を当てた研究として大門らの研究が、また既存の可視化手法の提案として伊藤ら、TRANら、大城ら、小山らの研究がある。

2.1 XTetra の開発と授業実践による評価

大門ら[4]は、「共通テスト手順記述標準言語(DNCL)」[5]の学習環境の開発を目的に、実行環境「XTetra」を開発した。XTetra にはシンタックスハイライトや標準出力、ボタンによる入力補助などの機能がある。高校 1 年生を対象に XTetra を用いた授業を実践し、授業後のアンケートから XTetra が制御構造の理解に貢献することを示した。

2.2 プログラム動作の可視化によるプログラムの動作確認支援に関する研究

伊藤ら[2]は、初心者のプログラミング学習支援を目的に、各ステップとその一つ前のステップとの変数の値を表で並べて比較できるシステムを開発した。さらに、前後で変化した変数を着色して強調した。大学生を対象に、開発したシステムを用いて論理エラーを含むプログラムを修正する実験を実施し、システムが論理エラーを含むプログラムのデバッグに有効であることを示した。

2.3 プログラミング初学者のための可視化システムによるプログラムの動作確認の支援に関する研究

TRAN ら[6]は、伊藤ら[2]のシステムを拡張し、変数の変化や条件式の評価の理由を明確にする目的で、計算式や条件式の変数に実際の値を代入した結果を表示するシステムを開発した。伊藤らと同様の実験から、システムが論理エラーを含むプログラムのデバッグに有効であることを示した。

2.4 初学者向けプログラミング学習のための初等アルゴリズム視覚化システム

大城ら[7]は、アルゴリズムの動きや仕組みの可視化、およびプログラミングの補助を目的として、変数を「変数名の書かれた箱の中に、格納される値の書かれたカードを入れたペア」として表現し、値の変化をアニメーションで表現するシステムを提案した。条件判定や値の操作、演算についてはポップアップによる補足説明を加えている。

2.5 変数の値の変化の可視化によるプログラム理解支援

小山ら[8]は、プログラム全体を通した変数の変化を理解することの支援を目的に、全体を通しての変数の値の変化を一覧表示し、代入や参照された箇所を別の色で着色して強調するシステムを提案した。また、小山らは制御構造の働きを可視化するために、プログラム自体をブロックごとに枠で囲み、階層的に表現した。さらに繰り返し処理が行われる箇所については、対応するプログラムをその実行回数分だけ複製して並べて表示する手法を提案した。

3. 本研究の意義と方針

3.1 本研究の意義

関連研究に示した可視化手法にはそれぞれ以下の課題がある。

- 伊藤らの手法では、制御構造は可視化されていない。
- TRAN らと大城らの手法は、変数の変化や条件式の評価が、どのような順序や理由で発生するのかという、制御構造の実行の流れまでを可視化したものではない。
- 小山らの手法でのプログラムの複製は、元の構造を大きく変更することになり、利用者の理解を阻害する可能性がある。

加えて、これらの提案手法はいずれも関数の実行の流れの可視化を含んでおらず、さらに大城らと小山らの手法は検証実験が未実施で有用性が実証されていない。総じて、制御構造や関数を可視化する手法には議論の余地が残されており、本研究ではこれらの課題に対応した可視化手法を提案する。

なお、大門らの研究は「情報I」の「コンピュータとプログラミング」分野の学習支援に焦点を当てている点で本研究と共通している。しかし、大門らの研究の中心は実行環境の開発や入力支援の実装にあり、可視化手法の提案と評価を目的としたものではない。また、大門らの研究は、テストの点数などの客観的な評価を通じて実際の理解度を検証したものではない。本研究では、理解度を確認するテストを実施し、有用性をより客観的に検証する。

3.2 本研究の方針

本研究では以下の方針で研究を進める。

1. 「情報I」の指導分野の一つである「コンピュータとプログラミング」分野に着目し、論理エラーの原因の特定と修正や「プログラミング的思考」の習得を目的として、制御構造や関数の流れの可視化を含む新たなプログラム可視化手法を提案する。
2. 提案手法の評価のため、提案手法を活用した学習教材を開発する。教材は「Python」を学習対象とし、変数や型、制御構造、関数、アルゴリズムの基礎を学習できる内容とする。教材内で提示される各プログラムには提案手法による可視化を組み込む。
3. 提案手法の有効性を検証するため、高校生および大学生を対象に教材を用いた学習実験を行う。学習後の確認テストとアンケートの結果から、有用性を評価し、また、それらの課題を特定して改善策を検討する。

4. 可視化手法の提案および教材の実装

4.1 自己学習教材

自己学習教材は、実際に使用されている教科書教材[9]の内容に沿い、表 1 に示す構成(以下、「教材構成」)で設計した。また、実験の自己学習時間に合わせて構成ごとに学習の目安時間を設けた。

表 1：教材構成および目安の学習時間

教材の構成	目安時間
0. 教材の使い方と進め方	2 分
1. プログラミングの基本（変数、型、演算子）	7 分
2. 配列	5 分
3. 反復構造	10 分
4. 選択構造	8 分
5. 関数	8 分
6. アルゴリズム①：探索	10 分
7. アルゴリズム②：並び替え	15 分
合計	65 分

自己学習教材は、教材の概要を説明するホーム画面と、学習画面（図 1）からなり、利用者は表 1 の流れで学習を進める。

図 1：学習画面の画面構成

学習画面は画面左半分の「テキスト部分」と右半分の「可視化部分」から構成される。テキスト部分では教科書に基づいた説明を提供し、可視化部分で提案手法を用いてプログラムを可視化する。可視化部分下部に配置されたボタンでプログラムをステップ実行できる。実行中のステップはプログラム上に枠線で強調され、下部には出力が表示される。また、各ページには学習内容の理解度を問う確認問題を設けている。

4.2 提案する可視化手法

4.2.1 変数、型、配列

大城らは変数を箱のイメージで表現し、伊藤らや TRAN らは変数の変化を色で強調し、小山らは参照を別の色で表現した。しかし、これらの手法は独立して提案されており、同時に適用されたものではない。本研究ではこれらの既存の手法を統合し、変数を箱のイメージを用いて代入や宣言をアニメーションで表現し、加えて参照は黄色の枠、代入は緑色の枠で強調表示した（図 2）。また、伊藤らや TRAN らの提案手法は、一つ前と現在の変数との変化の比較に留まり、次の変数とは比較できない。小山らは変数の変化を一覧表で表現したが、手法の提案のみで有用性が実証されていない。本研究では次のステップの変数との比較を可能にするため、上から順に、一つ前、現在、次の 3 段階の変数の状態を示し、実行に応じて行が上下にスライドする設計とした（図 3）。加えて値の型に応じて数値型は青色、ブーリアン型はオレンジ色、テキストシーケンス型は黒色の文字を用いた。

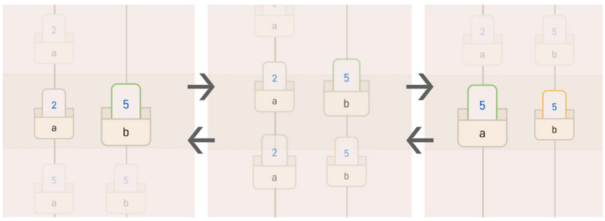

図 3：ステップの変化による上下移動

配列は変数の箱を複数個並べて表現した（図 4）。各箱が配列の要素を表し、隅には順にインデックスが振られている。

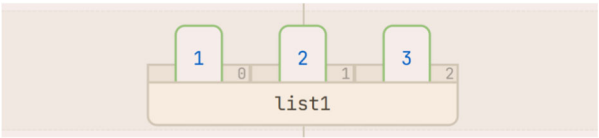

図 4：配列の可視化

4.2.2 反復構造

小山らは反復処理をプログラムの該当箇所を複製して表現したが、プログラムの構造を大きく変えるため利用者の理解を阻害する可能性がある。本研究では for 文および while 文の反復構造を、プログラム上への変更を避け、反復回数に応じた可視化領域上の層の重なりで表現した（図 5）。

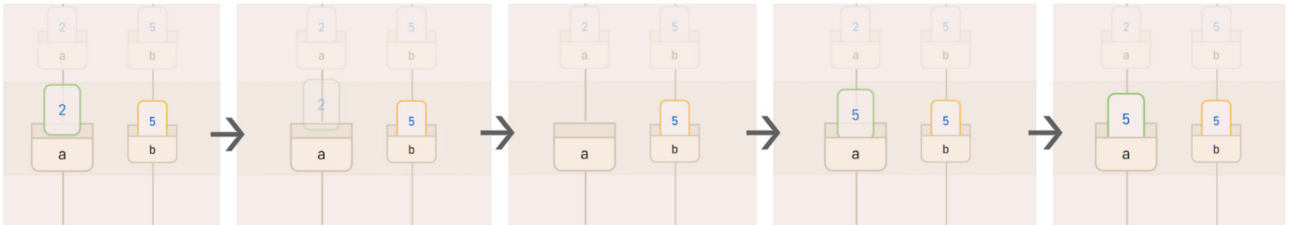

図 2：値の変化のアニメーション例（変数 a の値が 2 から 5 へと変化）

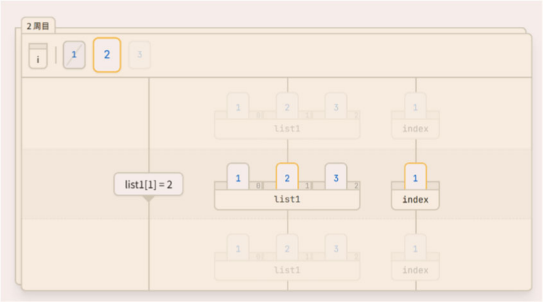

図 5：反復構造の可視化（反復回数 2 回に応じて層が 2 重に重なっている）

さらに、各反復にて参照されるイテレータ変数は、通常の変数表示から上部に独立させ、反復ごとの値の変化を横並びに一覧表示した。

while 文の反復終了の条件式の評価は、TRAN らの手法を参考に、条件式の評価過程を変数の具体的な値を代入や絵文字による結果判定（結果が真:✅、結果が偽:❌）で表示する（図 6）。選択構造の条件式の評価も同様に可視化する。

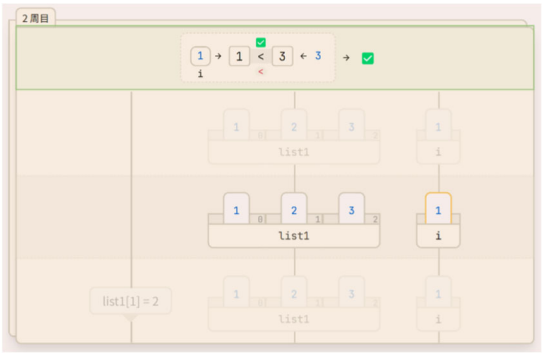

図 6：while の条件判定の評価式の全体像

4.2.3 選択構造

TRAN らや大城らは具体的な値の代入や文章説明で条件式の評価の理由を明確したが、選択構造の「処理の流れ」の可視化は不十分であった。そこで本研究では if 文について、以下の 3 つから選択構造の処理の流れを可視化した。

- if-elif-else の並列する条件式を色分けして表示し、評価結果を左隣に絵文字（結果が真:✅、結果が偽:❌、評価待ち:🤔）で表現した（図 7）。これにより、条件式の評価結果や状態を容易に把握できることを目指した。
- 条件式間を縦線でつなぎ、条件式が真となった場合、対応して実行される箇所を矢印で指し示した（図 7）。これにより、条件式の評価の流れを視覚的に表現することを目指した。
- 条件式が真となった場合、可視化部分と処理中の行とを同色で囲み、処理が実行されていることを明示した（図 8）。

なお、図 7 は並列する条件式が、順に 2 行目、4 行目、6 行目と評価され、6 行目の条件式が真となり、対応する 7 行目が実行されている段階を示している。また、7 行目の青枠の実行に対応し、7 行目の実行中は可視化部分に同色の枠組みが生成される（図 8）。

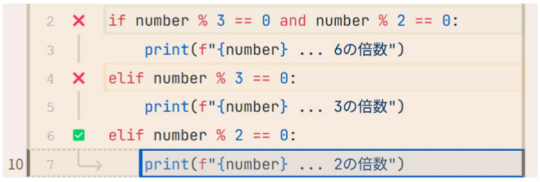

図 7：選択構造の処理の流れの表示

図 8：実行に対応した枠組み

4.2.4 関数

既存の手法は関数の定義や呼び出しの様子自体を可視化したものではない。本研究では以下の 2 つから関数の処理の流れを新たに可視化した。

- 関数の処理の流れを、通常の可視化部分の右隣に独立して表示し、関数の宣言や呼び出し時の引数の受け渡し、返り値の受け取りを画面のスライドやアニメーションで表現した（図 9）。これにより「関数の呼び出し時に呼び出し元の処理が一時的に中断され、関数内の処理が実行された後に再び呼び出し元の処理に戻る」という関数の処理の流れを把握しやすくすることを目指した。
- 関数の呼び出し、実行中、値の受け取りに応じて関数を呼び出した行の状態を図 10 のようにプログラム上に明示した。

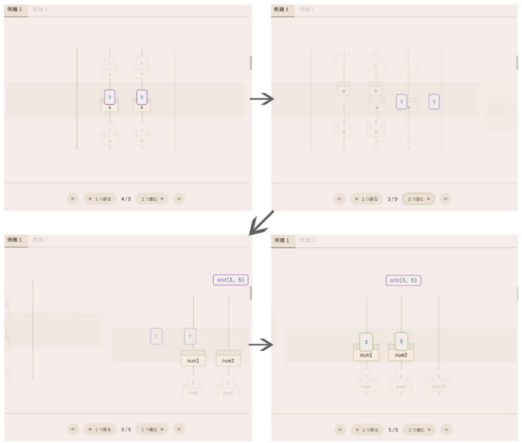

図 9：関数へ引数を渡す際の画面のスライド

```
6  b = 5
7  c = add(a, b) # add関数を呼び出 関数を呼び出す→
8  print(c) # 8が出力される
```

図 10：関数呼び出し行の表示

5. 評価実験

5.1 実験の目的

本研究では、以下の目的で評価実験を行う。

- 論理エラーの原因の特定と修正や「プログラミング的思考」の習得に対する提案手法の有用性を調査すること。
- 実験参加者から提案手法や教材に対する意見を収集し、今後の課題や改善策を検討すること。

5.2 実験の流れ

実験は、プログラミング未経験者およびプログラミングに課題意識を持つ学生 8 名（高校生 4 名、大学生 4 名。うち、男性 7 名、女性 1 名）を対象に実施した。表 2 の流れで実施し、適宜休憩を挟み約 2 時間 15 分実験に協力して頂いた。自己学習での教材は、プログラミングの理解度や経験を問う事前アンケートの回答結果が均等になるように、比較対象の教材（以下、「教材 A」）と提案手法を用いた学習教材（以下、「教材 B」）に 4 名（高校生 2 名、大学生 2 名）ずつ割り当てた。なお、本実験の計画は筑波大学システム情報系研究倫理委員会の審査にて承認されたものである（審査承認番号：「2024R921」）。

表 2：実験の流れ

順番	実施内容	所要時間
1	事前アンケート	15 分
2	実験の事前説明	10 分
3	自己学習	65 分
4	確認テスト	20 分
5	事後アンケート	15 分

自己学習では、教材を用いた個人学習を行ってもらう。教材 A では提案手法による表現を除外し、代わりに各ステップにおける変数の値を一覧表形式で提示する（図 11）。可視化部分以外の仕様は両教材で同一である。

図 11：教材 A の学習画面の画面構成

確認テストでは、「問 1:2 重の反復構造」「問 2:2 重の選択構造」「問 3:反復構造と選択構造の組み合わせ」の 3 つのプログラムに対して、以下の 3 点を提示し、論理エラーの原因を特定、修正する問題を出題した。

1. 作成するプログラムの概要と実装（論理エラーを含む）
2. 期待する出力結果
3. 実際の出力結果

事後アンケートでは提案手法や教材、実験などへの評価を収集した。アンケートは選択式の設問（「非常にそう思う」「どちらかといえばそう思う」「どちらかといえばそう思わない」「そう思わない」の 4 段階）と自由記述式の設問で構成した。

6. 実験結果と考察

6.1 確認テストの結果と考察

表 3：教材 A 利用者ごとの得点、得点平均、標準偏差

設問番号	A1	A2	A3	A4	A 平均	A 標準偏差
問 1	2	2	2	2	**2.00**	**0.000**
問 2	0	2	2	1	**1.25**	**0.829**
問 3	2	0	1	0	**0.75**	**0.829**
合計	4	4	5	3	**4.00**	**0.829**

表 4：教材 B 利用者ごとの得点、得点平均、標準偏差

設問番号	B1	B2	B3	B4	B 平均	B 標準偏差
問 1	2	1	2	2	**1.75**	**0.433**
問 2	2	2	1	1	**1.50**	**0.500**
問 3	2	2	0	0	**1.00**	**1.000**
合計	6	5	3	3	**4.25**	**1.299**

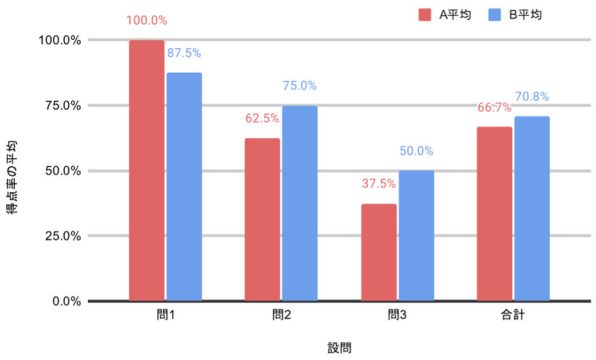

図 12：確認テストの得点率の群ごとの比較

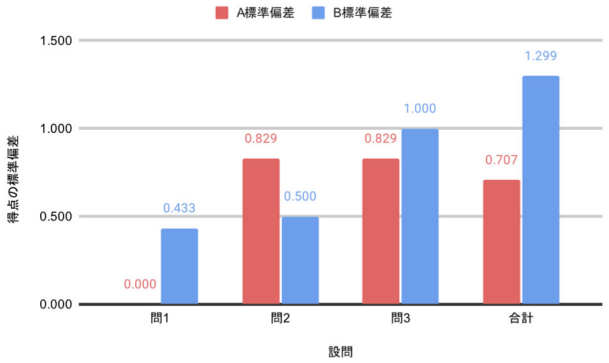

図 13：確認テストの標準偏差の群ごとの比較

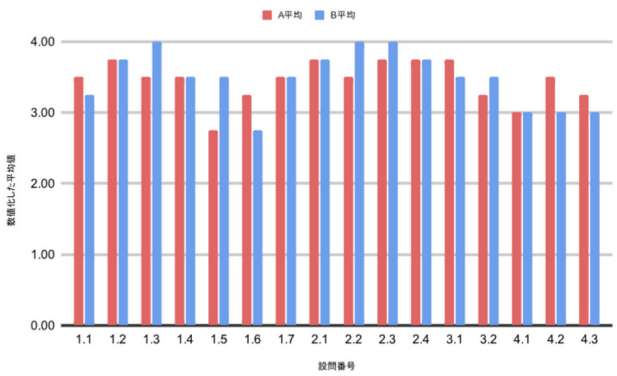

図 14：選択式設問の回答の群ごとの平均比較

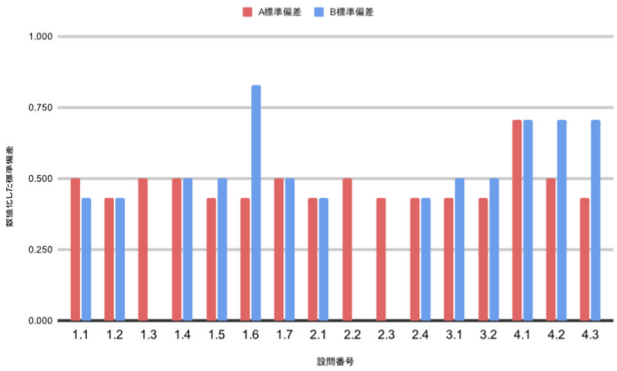

図 15：選択式設問の回答の群ごとの標準偏差比較

α =0.05 とした t 検定の結果(p 値は、問 1:0.391、問 2:0.670、問 3:0.750、合計:0.780)ではテストの得点に有意差は見られなかった。実験参加者が 8 名と少なく、統計的な有意差が確認できなかったと考えられる。しかし、回答の傾向や内容からは示唆的な考察が得られたため、以下に述べる。

問 1 に関して、教材 B 利用群の全員が反復構造の繰り返し回数の誤りを適切に修正できた。事後アンケートでも、教材 B 利用群の全員が反復構造の理解度向上に関する設問で最高評価を選択し、反復構造の可視化の有用性についても肯定的な回答を示した。これは提案手法による反復回数の可視化が、反復構造の理解促進に寄与した可能性を示唆している。一方、教材 A 利用群の全員が満点であることから、問題が易しすぎた可能性があり、難易度を上げた問題を出題して提案手法の有用性を再検証する必要がある。

問 2 では、両群でインデントの欠落や段数の不明確な回答が見られた。これは提案手法にインデントの段数などの文法構造の可視化が不足していたことが原因だと考えられる。また、教材 B 利用群にのみ「else」と「elif」の混同が見られた。提案手法では選択構造の句(if、elif、else)を明示する表現が不足していたため、句の違いへの意識が希薄化した可能性がある。問 3 でも同様の誤答が見られたことから、選択構造のインデントや句に着目した可視化表現の追加が必要である。ただし、教材 B 利用群の半数は問 2、問 3 ともに満点を獲得しており、提案した選択構造の可視化手法に効果があった可能性が示唆される。

6.2 事後アンケートの結果と考察

選択式設問に対する回答結果と平均値を表 5 に示す。平均値は「非常にそう思う」を 4 点、「どちらかといえばそう思う」を 3 点、「どちらかといえばそう思わない」を 2 点、「そう思わない」を 1 点と数値化して算出した。設問番号に「*」が付く設問は教材 B 利用群にのみ存在する項目である。両群に共通する 16 個の設問の回答平均を対象として α =0.05 の t 検定を実施したが、p 値は 0.800 であり有意差は見られなかった。

確認テストの結果と同様に、実験参加者のサンプルサイズが小さく、統計的な有意差が確認できなかったと考えられる。以下、事後アンケートの回答結果からそれぞれの群の回答内容を整理し、抽出できた点に対する考察を順に述べる。

6.2.1 プログラムの概念を理解するうえでの提案手法の有用性

プログラミングの理解度の向上を問う設問 2.1～2.4 では、教材 B 利用群の平均値が教材 A 利用群の平均値を下回ることがなかった。また、可視化が各概念の理解に役立ったかを問う設問 2.1*～2.4*においても全体的に高い評価が得られた。これらから、提案した可視化手法が「プログラミングの各概念の理解」の支援に対して有用に働いた可能性が示唆される。また、両教材には同一の確認問題を設けているが、確認問題の難易度を問う設問 1.5 の平均値には比較的大きな差が見られた。このことからも、提案手法が「プログラミングの各概念の理解」の支援において教材 A の可視化よりも有効に働いた可能性が示唆される。

6.2.2 次のステップの変数の状態を表示することの有用性

自由記述にて、教材 A 利用群にのみ、プログラムの実行過程で新しい変数が宣言されて表の行が増える際に、急な値の出現に驚きを感じるとの意見が確認された。提案手法では次のステップにおける変数の状態を表示し、その予測が可能で

表 5：選択式の設問に対する回答結果、および群ごとの数値化した平均

設問番号と設問内容		A1	A2	A3	A4	A 平均	B1	B2	B3	B4	B 平均
1. 学習教材について											
1.1	学習教材の説明は理解しやすかったか	3	3	4	4	3.50	3	3	4	3	3.25
1.2	学習の進め方は適切であったか	3	4	4	4	3.75	4	3	4	4	3.75
1.3	操作方法は理解しやすかったか	3	4	3	4	3.50	4	4	4	4	4.00
1.4	自分から取り組みたいと思える実験内容、教材内容であったか	3	3	4	4	3.50	4	3	3	4	3.50
1.5	学習教材の問題の難易度は適切であったか	3	2	3	3	2.75	3	3	4	4	3.50
1.6	学習教材の学習時間の長さは適切であったか	3	4	3	3	3.25	2	2	4	3	2.75
1.7	このような学習教材があれば、実際に利用したいと思うか	3	3	4	4	3.50	3	4	3	4	3.50
2. プログラミングの理解について											
2.1	Python を用いた変数や配列の値の変化の学習について、理解度は向上したか	3	4	4	4	3.75	4	3	4	4	3.75
2.2	Python を用いた反復構造の学習について、理解度は向上したか	3	4	3	4	3.50	4	4	4	4	4.00
2.3	Python を用いた選択構造の学習について、理解度は向上したか	3	4	4	4	3.75	4	4	4	4	4.00
2.4	Python を用いた関数の学習について、理解度は向上したか	3	4	4	4	3.75	4	3	4	4	3.75
2.1*	変数や配列の値の変化を理解する上で、プログラムの可視化は役に立ったか	-	-	-	-	-	3	4	4	4	3.75
2.2*	反復構造を理解する上で、プログラムの可視化は役に立ったか	-	-	-	-	-	3	4	4	4	3.75
2.3*	選択構造を理解する上で、プログラムの可視化は役に立ったか	-	-	-	-	-	3	3	4	4	3.50
2.4*	関数を理解する上で、プログラムの可視化は役に立ったか	-	-	-	-	-	3	4	4	3	3.50
3. 確認テストについて											
3.1	確認テストは学習の理解度を確認するうえで、役に立ったか	3	4	4	4	3.75	3	3	4	4	3.50
3.2	確認テストの難易度は適切であったか	3	3	3	4	3.25	3	3	4	4	3.50
4. プログラミングの考え方について											
4.1	プログラムの作成のために必要な処理を適切な順番で組み合わせる力が身についた（または向上した）か	2	3	3	4	3.00	3	2	4	3	3.00
4.2	実行結果から修正が必要な箇所を見つける力が身についた（または向上した）か	3	3	4	4	3.50	3	2	4	3	3.00
4.3	実行結果から必要な箇所をする力が身についた（または向上した）か	3	3	3	4	3.25	3	2	4	3	3.00

あったため、上述のような意見は見られなかった。このことから、次のステップの変数の状態を事前に可視化することが、変数の変化を理解する上で効果的に作用する可能性が推察された。

6.2.3 アニメーション表現の有用性

自由記述にて、教材 A 利用群にのみ、可視化部分の変化に気づきにくいという意見が確認された。これは教材 A の可視化はアニメーションを伴わないため、ステップ実行による変数の変化を把握することが困難であったことを示唆している。一方、提案手法ではそのような指摘は見られなかった。これは、ステップ実行に応じた行のスライドや、変数の変化をカードの出し入れで表現するアニメーションにより、変化が視覚的に捉えやすくなったためと考えられる。このことから、アニメーションを用いた可視化表現は、プログラミング初学者がステップ実行に伴う変数の変化を理解する上で有効である可能性が示唆される。

7. まとめ

本研究では、高等学校「情報I」におけるプログラミング学習に着目し、論理エラーの原因の特定と修正、および「プログラミング的思考」の習得の支援を目的として、制御構造や関数の可視化を含む新たなプログラム可視化手法を提案した。提案は以下の通りである。

- 変数：独立して提案された「箱のイメージ」「前の状態との比較」「着色による代入・参照の強調」の統合。さらに、現在の次の状態との比較を加えた 3 段階の変数変化の表現。
- 反復構造：イテレータ変数を独立表示した注目箇所の分離。反復回数に応じた層の重なりによる反復回数表現。
- 制御構造：並列する条件式の着色。評価結果を示す絵文字の併記。条件式間を縦線で連結した評価フロー表現。
- 関数：関数処理を独立領域で表示し、領域間のスライドや領域を横断する変数のアニメーションによる、関数定義・呼び出し表現。

提案手法の有効性を検証するため、提案手法を活用した学習教材を使った評価実験を行った。事後テストとアンケートの結果から、提案手法が、変数や配列、反復構造、関数といったプログラミングの基本概念の理解に寄与する可能性が示唆された。また、次のステップでの変数の状態を事前に可視化することや、アニメーションを用いた代入や参照の表現が、変数の変化の理解に有効である可能性が見られた。一方で、確認テストの結果から、提案手法は選択構造の理解の支援では不十分であることが確認された。

なお、実験参加者が少数であったため統計的有意差は得られず、これは本研究の限界である。そのため、統計的な結論は留保し、データの傾向や質的分析を中心に考察をした。

今後の課題として、インデントや選択構造の句に着目した可視化手法の提案、動的な可視化への対応、教材の説明内容、学習時間、問題内容の見直しが挙げられた。加えて、今回の実験で得られた示唆の信頼性を高め、統計的な裏付けを得るためには、より多くの参加者を対象にした実験の実施が求められる。実際の授業等における提案手法の利用を想定し、各グループ高等学校の標準的な学級規模である 40 名[10]を対象とした比較実験の実施を、今後の重要な課題とする。

参考文献

[1] 特定非営利活動法人みんなのコード: 2022 年度プログラミング教育・高校「情報 I」実態調査報告書 高校教員の意識調査 単純集計結果,
https://speakerdeck.com/codeforeveryone/puroguramingujiao-yu-shi-tai-diao-cha-bao-gao-shu-2022-gao-xiao-jiao-yuan-noyi-shi-diao-cha-dan-chun-ji-ji-jie-guo
(最終閲覧 2025 年 1 月 15 日)

[2] 伊藤福晃, 北英彦: プログラム動作の可視化によるプログラムの動作確認支援に関する研究, 2018PC カンファレンス論文集, pp.29-32 (2018).

[3] 文部科学省: 小学校段階におけるプログラミング教育の在り方について(議論の取りまとめ),
https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm
(最終閲覧 2025 年 1 月 20 日)

[4] 大門巧, 大西建輔, 青山浩: XTetra の開発と授業実践による評価, 情報処理学会論文誌 教育とコンピュータ, Vol.9, No.1, pp.23-32 (2023).

[5] 独立行政法人大学入試センター: 共通テスト手順記述標準言語(DNCL)の説明,
https://www.dnc.ac.jp/albums/abm.php?d=67&f=abm00000819.pdf
(最終閲覧 2025 年 1 月 15 日)

[6] T.TUNG, 北英彦, 高瀬治彦: プログラミング初学者のための可視化システムによるプログラムの動作確認の支援に関する研究, 2021PC カンファレンス論文集, pp.274-277 (2021).

[7] 大城正典, 永井保夫: 初学者向けプログラミング学習のための初等アルゴリズム視覚化システム, 情報教育シンポジウム論文集, Vol.2018, No.15, pp.104-111 (2018).

[8] 小山秀明, 山田俊行: 変数の値の変化の可視化によるプログラム理解支援, 情報処理学会論文誌プログラミング, Vol.10, No.4, pp.1-11 (2017).

[9] 萩谷昌己 他: 情I703 高校情報I Python, 実教出版 (2022)

[10] e-Gov 法令検索: 公立高等学校の適正配置及び教職員定数の標準等に関する法律(昭和三十六年法律第百八十八号),
https://laws.e-gov.go.jp/law/336AC0000000188
(最終閲覧 2025 年 7 月 30 日)

著者紹介

大澤 崇琉(学生会員)
2025 年筑波大学情報学群情報科学類卒業、学士(情報工学)。2025 年同大大学院修士課程在学中。プログラミング教育における学習支援に関心を持ち、制御構造や関数に着目した可視化手法の提案と評価に関する研究に従事。スマートライフ学会所属。

古川 宏(正会員)
1990 年東北大学工学部原子核工学科卒業。1995 年東北大学大学院工学研究科博士課程修了、博士(工学)。1996 年日本原子力研究所博士研究員原子炉安全工学部人的因子研究室。1998 年筑波大学電子情報工学系講師を経て、現在、システム情報系准教授に至る。ヒューマンマシンインタラクション、認知的インタフェース、空間認知とナビゲーション支援、メンタルモデルの獲得に関する研究に従事。2003 年アメリカカソリック大学認知科学研究室客員研究員。2016～2022 特定非営利活動法人モバイル学会理事長および学会長。