Vol.2021-CE-158 No.3
2021/2/13

プログラミング初学者を支援するソースコードと実行結果を対応付けた可視化手法の提案と評価

林 恭平[1,a)]　西田 誠幸[2,b)]

概要：プログラミング初学者は，個々のステートメントのプログラムの動作を説明できていても，プログラム全体，あるいはコードのまとまりごとにどのような動作をするか正しく把握できないことがある．一方で，プログラム理解という観点では，プログラム全体あるいはコードのまとまりごとにその動作を説明する力が必要だとされる．そこで本稿では，プログラミング初学者に対し，プログラム全体，あるいはコードのまとまりごとのコードリーディングを促すことで，プログラムの理解支援を行うことを目的として，ソースコードと実行結果を対応付けた色付けによる可視化手法を提案する．また，プログラミング導入科目において本手法を受講生に提供し，学習ログの分析とアンケート調査と行った．この結果を本稿で報告する．

キーワード：プログラム理解，プログラム可視化，プログラミング初学者支援

1. はじめに

プログラミング初学者は，個々のステートメントごとにプログラムの動作を説明できていても，プログラム全体，あるいはコードのまとまりごとにどのような動作をするか正しく把握できないことがある．例えば，図 1 のプログラムは，初期値 j=0 から i までアスタリスク記号（*）の印字を繰り返す for 文である．このようなプログラムの動作を理解できても，「アスタリスク記号を 1 行に i 個分印字する」ことを把握することはプログラミング初学者には比較的難しい．一方で，プログラム理解という観点では，プログラム全体あるいはコードのまとまりごとにその動作を把握する力が必要だとされる [3], [5], [9].

そこで本稿では，プログラミング初学者に対してコードのまとまりごとにその動作を把握する能力を促進するため，コードと実行結果に焦点を当てた可視化によるプログラム理解の支援手法を提案する．本稿の提案手法は，実行対象のプログラムとその実行の結果，印字した文字列との対応をハイライト表示によって可視化することで，プログラム理解を支援する．

```
1  for(int j=0; j<i; j++) {
2    System.out.print("*");
3  }
4  System.out.println();
```

図 1　プログラム例

以下 2 章では，可視化によるプログラム理解の支援に関する研究について述べる．3 章では，プログラム理解の支援手法を提案する．4 章では，提案手法の授業での利用とアンケート調査の結果について報告する．5 章では，本稿のまとめと今後の課題について述べる．

2. 関連研究

本章では，可視化手法を用いたプログラミング教育に関連する研究について紹介する．

コードと実行結果を対応付けて提示するツールとして，柿島のタートルグラフィックスプログラムの可視化ツール [5] がある．このツールは，タートルグラフィックスにおいて再帰呼び出しを含むプログラムの理解支援を目的としており，指定したコードと実行の結果描画する線分の対応を強調表示する．コードと実行結果の対応という点では本稿の提案手法と同様であるが，柿島の手法は再帰呼び出しを対象とするため初学者向けではないことや，コードと対応付ける要素が出力した文字列である点が本稿の手法とは異なる．

[1] 拓殖大学大学院工学研究科情報・デザイン工学専攻，193-0985 八王子市館町 815-1
Information andDesign Science Course, GraduateSchool of Engineering, TakushokuUniversity
[2] 拓殖大学工学部情報工学科，193-0985 八王子市館町 815-1
Department ofComputer Science, Faculty ofEngineering, Takushoku University
[a)] 20m307@st.takushoku-u.ac.jp
[b)] snishita@cs.takushoku-u.ac.jp

可視化によるプログラム読解支援の研究として，喜多らが開発したプログラム読解を支援するプログラム自動可視化ツール Avis[6] がある．このツールは，ソースコードからフローチャート図や逐次型実行経路図，モジュール遷移型実行経路図を生成し提示する．各図はプログラム全体の流れ，プログラムの振る舞いやモジュール間のつながりを示す．しかし，ツールの使用には事前にプログラムに対し一定の知識を持っている必要があり，初学者を対象とする本稿の目的とは異なる．

小宮らはプログラムのブロック構造の可視化による学習支援ツール AZUR[7] を開発した．このツールは，ブロック構造の明示的な可視化に加え，ステップ実行によるプログラムの挙動をアニメーションで提示することで理解支援を行う．また，ステップ実行時の状態を可視化する手法として Python Tutor[1][2] などがある．これらのツールは，オブジェクトや変数が持つ値の状態の逐次的に可視化することでプログラムの理解を支援する．しかし，これらの支援手法は，プログラムの 1 文ごとの動作を可視化する手法であり，本稿で目的としているプログラム全体，あるいはコードのまとまりごとの理解を支援するものではない．

小山らは，プログラム実行時の変数の値の変化を可視化する支援手法を提案している [8]．この手法は，プログラム実行時の変数の値が変化する過程を逐次的，あるいはプログラム全体で示すことで，学習者に対し理解支援を行う．しかし，この手法もまたプログラム全体，あるいはコードのまとまりごとの動作を可視化するものではない．

Victor はコードと実行結果を対応付けて提示する教育手法を提案している [4]．提示方法の一例として，図形を描画するプログラムが紹介されている．ここでは，例えば円を描画する関数呼び出しの引数を選択すると，円の半径が追加描画され，この引数が半径を指定するものであることを示す方法を提案している．さらに，for 文の繰り返し処理の各文がどの描画図形に対応するかを図示する方法を例示している．ここで紹介されている方法は本稿の提案手法に類似しているが，Victor の手法では for 文の繰り返し処理の一文が出力のどの部分に対応するかを可視化するのに対して，提案手法は for 文の一回の繰り返し処理がどの出力に対応するか示すという点で異なる．また，Victor の手法は図形の描画を行うプログラムを対象とするのに対して，提案手法は文字列を印字するプログラムを対象とする点でも異なる．

3. ソースコードと実行結果を対応を示すことによるプログラム理解支援

3.1 対象とするプログラミング言語と構文

本稿では，プログラミング初学者がテキストに記載されたプログラムの読むときや，自分で書いたプログラムの動作を確認するときに，そのプログラムの理解を支援する方法を提案する．対象とするプログラムは，標準入出力，変数，if 文，for 文，配列の構文で書かれた プログラムとする．本稿では，Java プログラムを例に提案手法を説明する．

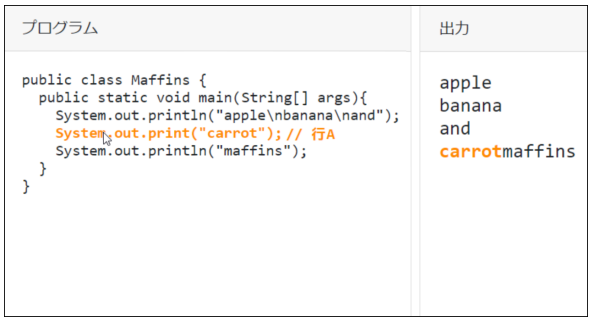

図 2　出力文字列における改行コードの働きの可視化例

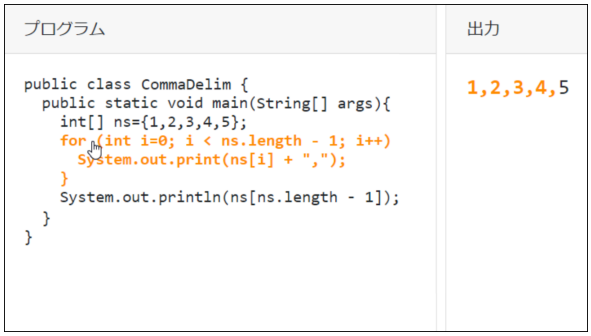

図 3　ループ処理の可視化例 1

3.2 ソースコード-実行結果の対応の可視化手法

コードリーティングにおいて，初学者のプログラムへの理解を促進するため，文やコードのまとまりごとの働きを可視化する方法をとる．特に，その働きを「標準出力への文字列印字」に絞って，文字列の印字を工夫することが求められるようなプログラムの理解支援の手法を提案する．また，本提案手法はプログラムの作成を支援するものではなく，プログラムを完成させた学生に対して，プログラムの構造を視覚的に提示することでプログラムの理解支援を行う．

本稿の提案手法は，実行対象のプログラムとその実行の結果，印字した文字列との対応をハイライト表示によって可視化する方法である．提案手法による可視化の例を示す．

図 2 のプログラムは改行コードの印字方法の学習を想定している．提案手法では，学習者が左ペーンのプログラムの一文にマウスカーソルを重ねたときに，文とその文が印字した文字列の両方をハイライト表示する．図 2 では，プログラムの行 A にマウスカーソルを重ねたときに，行 A が印字した 「carrot」をハイライト表示している．

図 3 のプログラムは，配列要素をカンマ区切りで印字するプログラムである[*1]．提案手法では，学習者が for 文に

[*1] Java の String#join() メソッドによって同じ処理を簡潔に記述できるが，ここでは 繰り返し処理の一部を例外的に処理する例

図 4　ループ処理の可視化例 2

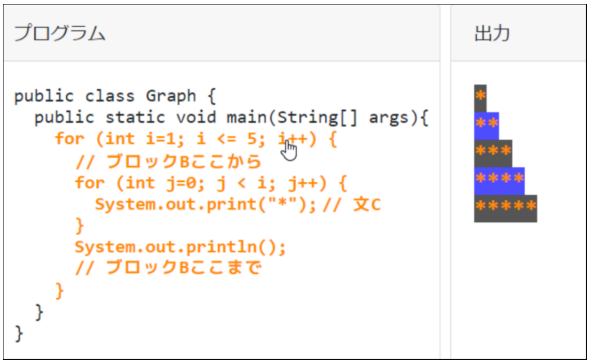

図 5　二重ループの可視化例

マウスカーソルを合わせてクリックすると，印字した文字列の背景色を繰り返し回ごとに切り替えて表示する．この様子を図 4 に示す．この状態で背景色が色付けられた文字列にマウスカーソルを重ねると，ツールチップを表示して繰り返しの何回目の実行結果なのかを示す．

図 5 のプログラムはアスタリスク文字（*）を三角形の形状で印字するプログラムである．特に，二重ループの内側の for 文は一行分の記号列を印字する．図 5 右の出力ペーンは，提案手法において外側の for 文をマウスクリックしたときの表示の様子を示している．

ソースコードと実行結果の印字文字列との対応の提示方法は次のルールに従う．

M　プログラム表示ペーンにおいて，main() メソッドに記述された文 s にマウスカーソルが重なるとき，s と s が印字した文字列の両方をハイライト表示する

L1　main() メソッドに記述された for 文がクリックされたときに，出力ペーンにおいて印字した文字列の背景色を繰り返し回ごとに色分け表示する

L2　出力ペーンにおいて，背景色が色付けられた文字列にマウスカーソルが重なるとき，ツールチップを表示して，この文字列が何回目の繰り返し処理による印字かを示す

提案手法は，main() メソッドに記述された文とその文が印字した文字列の対応をハイライト表示する．このような文には，if 文や for 文などの内部にブロックを持つ制御構文が含まれる．特に for 文に限っては，for 文のブロックとそのブロックで印字した文字列との対応を詳細表示する．for 文に関して詳細な可視化を行う理由は，プログラミング言語の学習対象の構文のうち，プログラム作成に学生が最もつまずきやすいのは for 文であり，コードリーディングに関する支援を行うことで，学生による for 文のプログラム理解を促進したいという意図がある．

提案手法では，入れ子の内側のブロックを可視化の対象から除外している．例えば，図 5 のプログラムでは，ブロック B を可視化の対象とするが，その中の単独の文 C は可視化の対象外とする．これは次の理由による．

- 入れ子の内側の文のみを可視化する方法は，コードのまとまりごとの理解という観点からはずれる
- 入れ子の外側から内側へ選択できる可視化対象を広げる場合，UI の操作が複雑になり初学者には扱いづらくなる恐れがある
- プログラミング導入教育において学生に提示する，あるいは書かせるプログラムでは，ブロックの入れ子が浅いものが多い

4.　提案手法の実装と評価について

本稿の提案手法を実装し，プログラミング演習支援システムの一機能として組み込んだ．プログラミング演習支援システムは Web アプリケーションであり，Java プログラムの作成，実行結果の確認，完成したプログラムの提出など拓殖大学工学部情報工学科の初年次科目「プログラミング I」の受講に必要な学習環境をブラウザ上で提供するものである．提案手法の機能は，本システムのページ上のボタンを入り口して，学生がボタンをクリックすると，作成中のプログラムと実行結果を提示し，図 2〜5 の形式のモーダルウィンドウを提示する．

提案手法の機能を学生に提供したのは，「プログラミング I」全 15 回のうち，for 文を初めて学ぶ第 10 回以降である．10 回目の授業では，講師から学生に対して，本機能の概要と使用方法についての説明があった．

提案手法によるプログラム理解支援を評価するためには，対照実験を行うのがよいと考えるが，現時点では実験を実施できていない．

本節では，プログラミング演習支援システムで収集した学習ログと提案手法の利用に関するアンケート調査の結果から，提案手法の使用状況について報告し，本手法を学生に提供する方法の改善について考察する．

4.1　学習ログの分析

プログラミング演習支援システムで収集した学習ログをもとに，学生による提案手法の利用頻度について調査した．「プログラミング I」の受講生 117 名のうち，提案手法の機

としてこのプログラムを例示している．

表 1　提案手法の利用状況調査

＼　利用タイミング　＼ 問題の種類　＼	(1) 完成後（件）	(2) 作成途中（件）	合計（件）
(a) 提案手法が有効	33	101	134
(b) 提案手法が有効ではない	4	18	22
合計	37	119	156

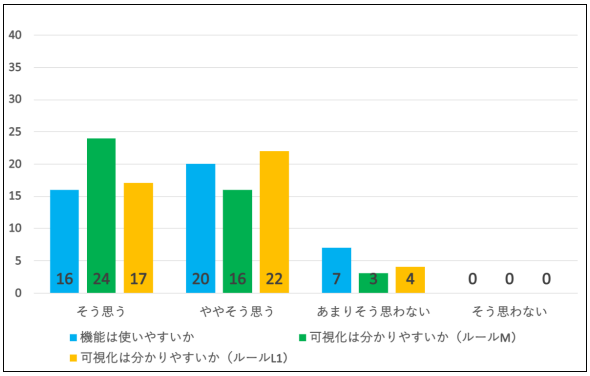

図 6　提案手法のアンケート結果

能を一度以上利用した学生は約半数の 62 名であった．また，学生の問題への取り組みを学生 1 名問題 1 問あたり 1 件とカウントすると，提案手法の機能の利用件数は 156 件であった．第 10 回以降に出題した問題は 38 問であり，最大で $38 \times 117 = 4446$ 件中の 3.5%に留まった．利用頻度が低かった理由としては，出題された問題には，整数配列の要素の総和を計算するなど印字処理を工夫する必要がなく提案手法の可視化が有効ではない問題が多いこと，類似問題が多いため提案手法の機能を使わなくても可視化の結果が類推しやすいことなどが考えられる．

さらに，学習ログから，提案手法の利用するタイミングについて調査した．学生が提案手法を利用するタイミングには (1) プログラム完成後，(2) プログラム作成途中の 2 通りに分けられる．このうち，(2) については，作成過程のプログラムが意図した通りの動作になっているかどうかを確認する場合や，プログラム作成に行き詰まってヒントとして提案手法を利用する場合などが考えられる．また，出題した問題には，(a) 提案手法の利用がプログラム理解の支援になる可能性がある問題と (b) 提案手法がプログラム理解の支援につながらない問題がある．特に (b) の問題には，前述の総和を計算する問題のように計算結果だけを印字する問題がある．提案手法の利用タイミングと問題の種類の 2 つの観点から学習ログを分類した結果を表 1 に示す．

表 1 から，提案手法の 151 件の利用のうち，76.3%に当たる 119 件ではプログラム作成途中に提案手法を利用していた．この 119 件についてさらにプログラム作成の過程を目視で確認したところ，119 件の多くはプログラム作成中に行き詰まって提案手法を利用しており，利用後も試行錯誤しながら課題を進めていた．

また，151 件のうち 14.6%に当たる 22 件は提案手法が有効に働かない問題への取り組みの過程で使用された．このうち，4 件についてはプログラム完成後であるため，この種のプログラムに対する可視化結果を確認するために提案手法の機能を使用したことが考えられる．その他の 18 件については，本機能利用後も試行錯誤しながらプログラムを作成していた．6

4.2　アンケート調査とその結果

プログラミング演習支援システムに組み込んだ提案手法の機能の分かりやすさ，使いやすさを知るため，アンケート調査を行った．回答者は「プログラミング I」受講生のうち 109 名で，ルール M とルール L1 による可視化の分かりやすさと，組み込んだ機能の分かりやすさについて 4 件法で回答を求めた．

まず，回答者 109 名中，本機能を使った経験があると回答したのは 43 名にとどまった．本機能の学習ログ上の利用人数は 62 名であり，アンケートの結果の人数と一致していない．この理由の考察として，一部の学生が「講義内で紹介されたので，試しに使用してみた」「誤って機能を開いてしまった」という理由により「機能を利用していない」と回答した可能性が考えられる．なお，本機能を一度のみ使用した学生は 13 名であり，「プログラミング I」受講生のうち 117 名の中でアンケートに回答していない学生を含めると，学習ログ上の人数と概ね一致する．

「本ツールの機能が使いやすいか」との設問に「そう思う」「ややそう思う」との回答をしたのは，全体の約 80%にあたる 36 名であった．また，「ルール M（とルール L1）の可視化が分かりやすいか」との設問に「そう思う」「ややそう思う」と回答したのは，全体の約 90%に当たる 40 名（と 39 名）であった．

自由記述欄の回答を以下に示す．提案手法の可視化に関して肯定的な意見に加えて，二重ループの内側の for 文についても同様の可視化を求める意見があった．

- 間違った箇所がわかりやすく使いやすかった
- 出力結果を色分けで表示してあったので見やすかった
- とても分かりやすくていいと思う。どの部分が実行結果と関係してるのかがみれて良い。
- 2 重の for 文の場合、それぞれの for 文がなにを表示しているかもわかるようにしてほしい

4.3　考察

まず，学習ログの分析結果について考察する．プログラム作成の過程で行き詰まっていると思われる学生が，ヒントを得る目的で提案手法を利用しており，かつ提案手法の機能が問題解決の参考となっていない．また，少数ではあるが印字文字列を工夫する必要がなく提案手法が有効では

ない問題に対して，本機能を使用している学生がいた．以上のことから，的外れな問題での本機能の使用や間違ったタイミングでの使用を抑制するために，提案手法の利用方法に関して，次のことが必要である．

(1) 本機能の利用場面や用途を含めた利用方法の説明を充実させる
(2) 本機能が利用可能な問題を限定する
(3) 作成過程のプログラムの誤りの種類によって本機能の利用を制限する

特に (2) については，テキスト掲載のプログラムを書き写して動作確認する種類の問題のみ，あるいは，印字方法を工夫する必要がある問題に利用を絞る方法が考えられる．また，(3) についてはプログラムにおける印字処理の誤りを解析する方法がすでにあることが前提となる．

次に，アンケート調査結果について考察する．アンケート回答者のうち，提案手法の機能を使用したことがあると回答した者は全体の 39.4%にとどまった．この理由としては，「プログラミング I」の課題がすべてプログラム作成問題であったのに対して，提案手法はプログラム理解を支援するものであり，課題を取り組むにあたって提案手法の機能が有効ではないと受講生が考えた可能性がある．

提案手法の機能を利用した経験がある学生の約 80%と 90%が，それぞれ本機能が比較的使いやすい，可視化の手法は比較的分かりやすいと感じていることがわかった．一方で，自由記述欄の回答においては，「プログラム作成につまずいたときに解決のヒントを与えてくれる機能である」とみなしている学生がいることから，学習ログの考察同様で本機能の説明を充実させる必要がある．また，自由記述欄では多重ループの内側でも同様の可視化を求める意見があった．この点については今後検討を要する．

5. おわりに

本稿では，プログラミング初学者に対してコードブロックなど複数の文のまとまりに対するプログラム理解を支援することを目的として，コードと実行結果の対応を可視化する手法を提案した．プログラミング導入科目において本手法を提供してアンケート調査を行ったところ，本手法の機能を利用した受講者からはコードと実行結果の対応が比較的分かりやすいとの回答を多く得た．一方で，学習ログを用いて本手法の利用状況を分析したところ，プログラム理解という観点から外れた利用をしていると思われる受講生が多数見つかった．今後の課題として，本手法の利用方法の分かりやすい説明，利用場面の限定などの方法を検討する必要がある．また，本手法によるプログラム理解の支援についての評価実験を実施する必要がある．

参考文献

[1] Guo, P. J.: Python Tutor - Visualize Python, Java, C, C++, JavaScript, TypeScript, and Ruby code execution, (online), available from ⟨http://pythontutor.com/⟩ (accessed 2021-01-10).
[2] Guo, P. J., White, J. and Zanelatto, R.: Codechella: Multi-user program visualizations for real-time tutoring and collaborative learning, *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, pp. 79–87 (2015).
[3] Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E. and Whalley, J. L.: Going SOLO to assess novice programmers, *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pp. 209–213 (2008).
[4] Victor, B.: Learnable Programming, (online), available from ⟨http://worrydream.com/LearnableProgramming/⟩ (accessed 2021-01-10).
[5] 柿島遥，西田誠幸：タートルグラフィックスにおけるプログラム理解支援手法の提案，研究報告コンピュータと教育 (CE)， Vol. 2020, No. 1, pp. 1–7 (2020).
[6] 喜多義弘，片山徹郎，冨田重幸：Java プログラム読解支援のためのプログラム自動可視化ツール Avis の実装と評価，電子情報通信学会論文誌 D， Vol. 95, No. 4, pp. 855–869 (2012).
[7] 古宮誠一，今泉俊幸，橋浦弘明，松浦佐江子：プログラミング学習支援環境 AZUR—ブロック構造と関数動作の可視化による支援—，研究報告ソフトウェア工学 (SE)，Vol. 2014, No. 5, pp. 1–8 (2014).
[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 (PRO)， Vol. 10, No. 4, pp. 1–11 (2017).
[9] 松田憲幸，柏原昭博， 平嶋宗，豊田順一：プログラムの振舞いに基づく再帰プログラミングの教育支援，電子情報通信学会論文誌 D， Vol. 80, No. 1, pp. 326–335 (1997).