プログラミング演習における動作確認を支援するための プログラム動作の可視化

伊藤 福晃*1・北 英彦*1
Email: 417m204@m.mie-u.ac.jp

*1：三重大学大学院工学研究科電気電子工学専攻

◎Key Words プログラミング，可視化，デバッグ，トレーサ

1. はじめに

プログラミング学習は近年，小学校の授業に組み込まれるなど，更なる注目を集めている．プログラミング学習において，簡単なプログラムを書くことはできるが，動作の複雑なプログラムになると書けなくなってしまう初学者が多い．その原因として，ソースコードとプログラム動作の関連付けができていないことがあげられる．プログラミング演習の際，ソースコードを書くことはできるが正しい結果が得られない場合，ソースコードからプログラム実行途中の動作を目で正しく追えていない．

そこで，プログラムの動作を可視化することにより，プログラムの動作が作成者の意図通りであるかを確認するための動作確認システムを提案する．本システムでは，プログラムの変数や計算を可視化して比較表示することにより，プログラム動作確認の支援を行う．ソースコード1行単位の実行前後での変数や配列の変化の比較や，繰り返しや条件分岐のブロック単位での実行前後での変数や配列の変化の比較表示を行う．加えて，評価式も表示する．

2. 先行研究

現在，プログラムの理解を助けるための様々なプログラム可視化システムがある．可視化システムはプログラムの理解を深めるという目標は大きく同じであるが，それぞれは機能面で異なる特徴がある．そのうちいくつかの先行研究を紹介する．

2.1 Jeliot 3

学習者が書いたプログラムの動作を動的に可視化することでプログラム学習の支援を行うシステムである[1]．Jeliot 3 を図 1 に示し，可視化部分を図 2 に示す．プログラム学習の初学者を対象としており，デバッガのように動作し，実行，ステップ実行ができるが，1 つ前の実行に戻ることが出来ない．また，Web アプリケーションではないため，ダウンロードして使用する必要がある．

2.2 Online Python Tutor

学習者が書いたプログラムを静的に可視化することでプログラム学習支援を行うシステムである[2]．Online Python Tutor を図 3 に示し，可視化部分を図 4 に示す．プログラム 1 行ずつの実行状態を静的に可視化し，1 行ずつ進んだり戻ったりして状態を確認することができる．しかし，1 つの実行状態しか可視化できず，比較することが難しい．Web ブラウザ上で動作し，ダウンロードせずとも使用できる．

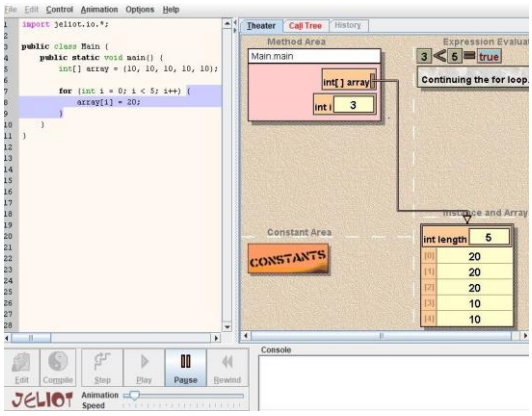

図 1 Jeliot 3

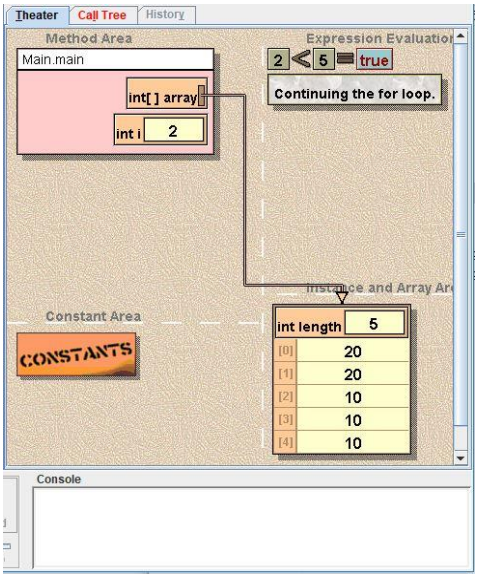

図 2 Jeliot 3 の可視化部分

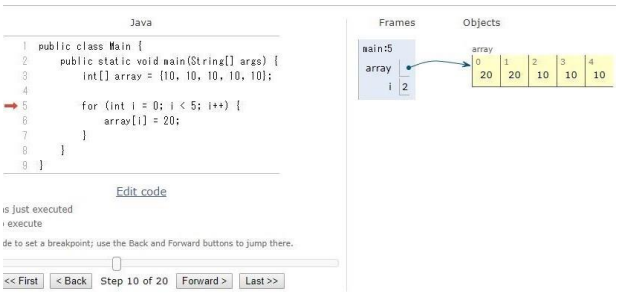

図 3 Online Python Tutor

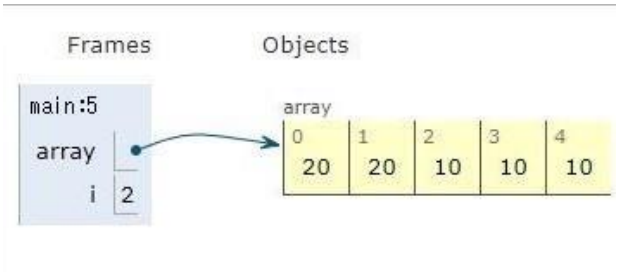

図 4　Online Python Tutor の可視化部分

2.3　静岡大学のシステム

SCBCR（Scenes Comparison Based Code Reading）と呼ばれる比較学習プロセスを使用し，TEDViT（Teacher Explaining Design Visualization Tool）を拡張して SCBCR 用のシステムを開発した[3]．指導者が用意したプログラムについて，以下の比較表示を行う．

① 実行前後の変化
② 同じプログラムでことなるデータを用いた際の挙動の変化
③ 異なるプログラムで同じデータを用いた際の挙動の変化

これらの表示を行いアルゴリズムの理解を支援するシステムである．①の画面を図 5 に，②の画面を図 6 に示す．このシステムでは，学習者が書いたプログラムを可視化するのではなく，指導者があらかじめ説明したいプログラムの描画を用意しておく必要がある．また，Web アプリケーションではない．

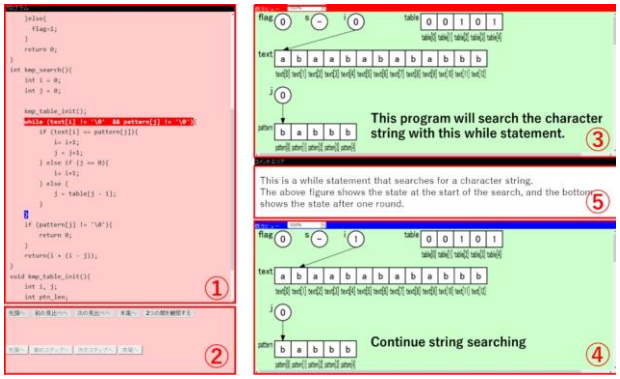

図 5　静岡大学のシステム①

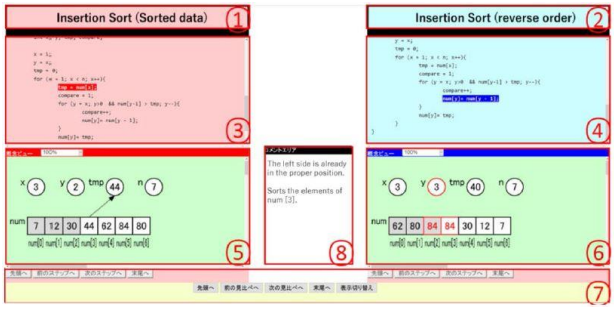

図 6　静岡大学のシステム②

3.　本システムの概要

可視化システムは既に様々なものが開発されているが，学習者本人が書いたプログラムを可視化して比較表示することによって学習支援を行う可視化システムは存在しない．

エディタ画面で入力されたプログラムは，可視化ボタンを押されることで可視化画面へと切り替わる．可視化画面では，左側には学習者の書いたソースコードが表示されるソースコード部分，右側には変数や配列，評価式が表示される可視化部分に分かれている．ソースコード部分ではソースコード 1 行ごとに左側にチェックボックスが表示され，選択されることでその行の実行前後のプログラムの状態が右側の可視化部分に表示されるようになっている．以下で詳細を述べるが，条件分岐の if 文，繰り返しの for 文，while 文はブロックとして扱われ，ブロックとして扱われる部分はソースコード部分の背景色で強調表示される．また，可視化部分の表示は各メソッド単位での表示を行う．クラス単位での表示を行わない理由は 2 つあり，本システムは初学者を対象としており，複数のクラスを扱う複雑なプログラムを組むことを想定していないためという点と，初学者がプログラムの挙動を知るのにできるだけ最小限の単位での可視化を行うことで簡単に理解できるようにするためという点が挙げられる．可視化画面全体を図 7 に，可視化画面左側のソースコード部分を図 8 に，右側の可視化部分を図 9 に示す．

3.1　比較表示

学習者が書いたプログラムを，ソースコード 1 行単位またはブロック単位で実行前後の変数や配列，評価式の可視化を行う．ソースコード部分のチェックボックスで選択された行またはブロックの実行前後を，右側の可視化部分上部（A）に実行前，可視化部分下部（B）に実行後のプログラムの状態を可視化表示する．学習者が選択したソースコードの実行前後のプログラムの状態を視覚的に比較することでプログラムの挙動の理解が簡単にできることを期待する．

3.2　ブロック表示

繰り返しの for 文と while 文，条件分岐の if 文をブロックとしてみなし，ブロックの実行前後での配列の変化の比較表示を行う．ブロックの部分はソースコードの背景色を変えて強調表示され，ブロックの始まりの 1 行には通常のブロックに加えてもう 1 つのチェックボックスが用意される．追加されたチェックボックスを選択するとブロックの実行前後のプログラムの実行状態を表示する．また，通常のチェックボックスを選択すると，ソースコード 1 行の実行前後のプログラムの実行状態を可視化表示する．以下に可視化画面の例を挙げる．

- 図 9 は，if 文の部分をブロックとして扱い，ブロック前後での変化の可視化を表示している画面である．if 文の部分の背景色を変え強調表示され，チェックボックスが 2 つになる．ブロック表示のチェックボックスを選択することで，ブロックでの実行前後でのプログラムの状態を可視化し，比較表示している．

- 図 10 は for 文の部分をブロックとして扱い，ブロック内の実行回数ごとに連続して 3 回分表示している画面である．if 文同様，ブロック部分には強調表示が行われ，ブロックとしての実行前後での状態を表示するためのチェックボックスが追加される．ブロック表示をしているため，繰り返し部分の実行回数ごとに 3 回分比較表示される．連続表示を行うことで，繰り返し文の挙動の変化を理解し易くなっている．可視化部分下の矢印ボタンでブロックとしての繰り返しの実行回数を変更することが可能である．while 文も同様に表示する．

3.3 式と評価式の表示

条件分岐や繰り返し文の評価式，プログラム内の計算図 11 はプログラム中の計算式を表示している画面である．チェックボックスで選択をした行の実行前後に関係のある計算式を可視化部分に表示することで学習者がプログラムの流れを追うのを支援する．

図 12 は if 文，図 13 は for 文の条件判定の評価式を表示している画面である．チェックボックスで選択されたブロック内で，条件判定や繰り返しに必要となる評価式の内容を表示し，その true，false を表示する．while 文は if 文と同様に表示する．

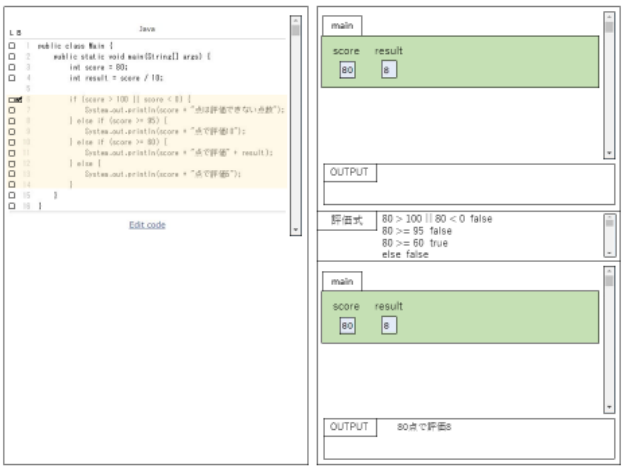

図 7　可視化画面全体

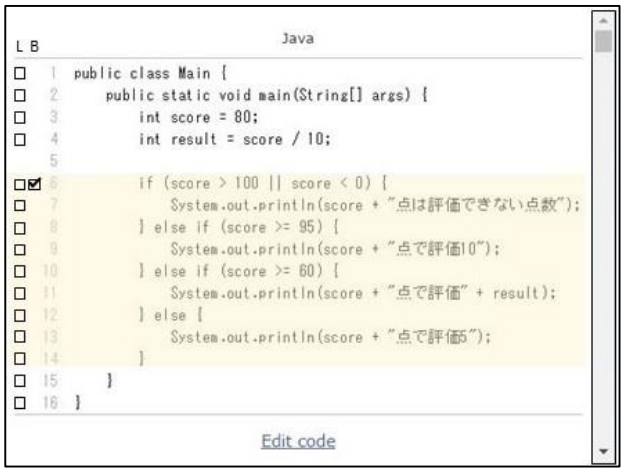

図 8　ソースコード部分

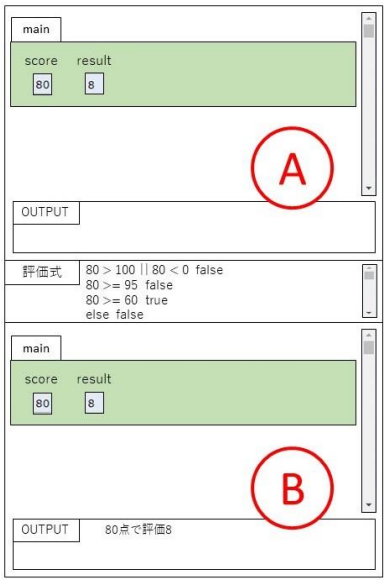

図 9　可視化部分（if 文）

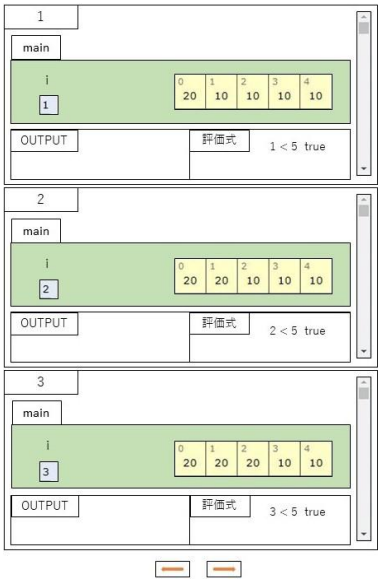

図 10　可視化部分（for 文）

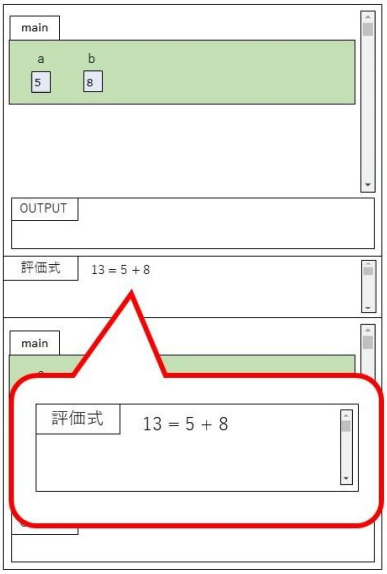

図 11　プログラム中の計算式

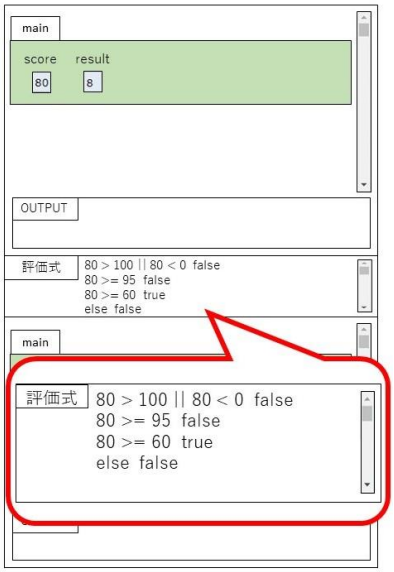

図 12　if 文の条件判定の評価式

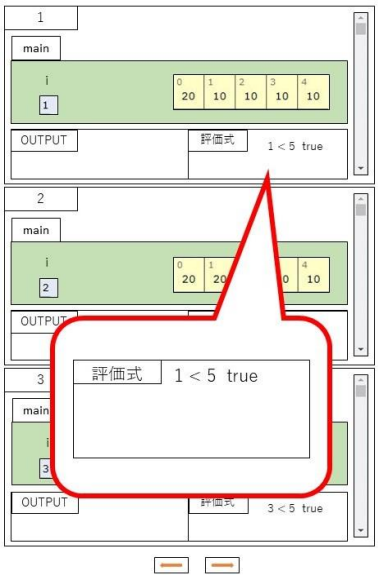

図 13　for 文の条件判定の評価式

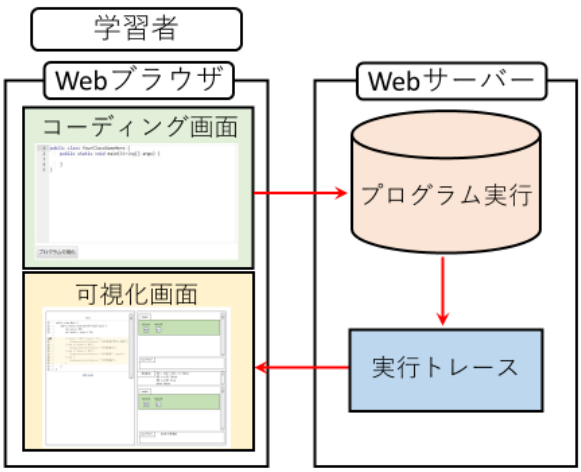

図 14　可視化システム構成図

3.4　実装

著者らの所属する研究室では，学習者のプログラミング状況の把握および学習者への迅速な対処を目的として，プログラミング演習システム PROPEL の開発および運用を行っている．本システムを PROPEL の機能の 1 部として追加し，プログラムの動作を可視化することにより，プログラムの動作が作成者の意図通りであるかを確認できるようにすることで，学習者の更なる支援を行うことを期待する．可視化システムの構成図を図 14 に示す．

4.　評価方法

以下 2 つのテスト問題を用い，本システムの使用する前後での正解数に対して $\chi 2$ 乗検定を行うことにより，本システムが，学習者に対してプログラムの動作が作成者の意図通りであるかを確認するための動作確認システムとして機能しているかを評価する．

- 三角形の種類を，条件分岐の if 文を用いて判別するプログラムを作成する課題.
- 配列を作成し，繰り返し文の for 文を用いて配列の中身を変化させ，その途中の配列の中身を問う課題.

2018 年度 3 重大学工学部電気電子工学科 2 年生を対象として，「プログラミング演習Ⅰ・Ⅱ」授業の受講者 85 人に本システムを使用する．

5.　まとめ

プログラミング学習において，ソースコードとプログラム動作の関連付けが出来ていないことが原因で，簡単なプログラムを書くことはできるが，動作の複雑なプログラムになると書けなくなってしまう初学者が多い．そこで，本システムではプログラムの動作を可視化することにより，プログラムの動作が作成者の意図通りであるかを確認するための動作確認システムを提案した．プログラムの動作が作成者の意図通りであるかを確認するための動作確認システムにより，プログラムの理解を深めることを期待している．

参考文献

(1) Andrés Moreno, Niko Myller, Erkki Sutinen: “Visualizing Programs with Jeliot3”, ACM, 2004.
(2) Philip J. Guo: “Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education”, ACM, 978-1-45-3-1775 - 7, 2013.
(3) Daiki Ihara, Sstoru Kogure, Yasuhiro Noguchi, Koichi Yamashita, Tatsuhiro Konishi, Yukihiro Itoh: “Algorithm Learning by Comparing Visualized Behavior of Programs”, Proceedings of the 25th International Conference on Computers in Education, 2017.
(4) Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, Tapio Salakoski: “VILLE-A Language-Independent Program Visualization Tool”, the Seventh Baltic Sea Conference on Computing Education Research, Vol.88, 2007
(5) Oscar Karnalim, Mewati Ayub: “The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor”, CommIT Journal 11 (2) , 67-76, 2017.
(6) Osku Kannusmäki, Andrés Moreno, Niko Myller, Erkki Sutinen: “What a Novice Wants: Students Using Program Visualization in Distance Programming Course”, Third Program Visualization workshop, 2004.