プログラムの各実行ステップの可視化による プログラムの動作理解の支援

TRAN THANH TUNG*1・北英彦*1
Email: 420m2b1@m.mie-u.ac.jp

*1：三重大学大学院工学研究科

◎Key Words　　プログラミング演習，可視化，デバッグ

1. はじめに

近年，情報社会の発展とともに，2020 年から小学校においてプログラミング学習が必修化されるなど，プログラミング学習の重要性が高まっている [1]。一般的なプログラミング学習方法として，演習型の授業が行われている。学習者は以下の工程でプログラムの作成を行う。

(1) コーディング
(2) コンパイル
(3) テスト
(4) デバッグ

学習者がテストを行い，プログラムの動作結果が正しければプログラミングの作業は完了する。動作が正しくなければデバッグする必要がある。デバッグは図 1 に示すように 2 つの小工程からなる。

(a) 間違えている位置の特定
(b) 間違えているコードの修正

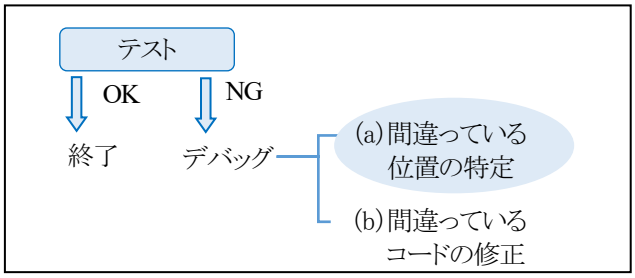

図 1　デバッグの手順

初学者は自分が作成したプログラムの動作が間違っていても，間違っている位置を特定できないことがあるためにデバッグできないことがある。

間違っている位置の特定を支援するために本研究室の院生である伊藤は，Java を学習する初学者を対象としてプログラムの動作中の変数の値や配列の値の移り変わりを可視化するシステムを開発した [2]。提案した機能のうち約半分が未実装として残っている。本研究では，そのうちの変数・配列の値が変化したときにその理由を表示する機能を可視化システムに加えた。また，多人数での運用が可能になるように，サーバの構成を変更した。

2. 先行研究

変数や配列の値を可視化することでプログラムの動作の理解の支援を行うシステムに関する先行研究ついて紹介する。

2.1 Online Python Tutor

Online Python Tutor は，学習者が書いたプログラムに対して変数の値や配列の値を可視化することでプログラムの動作の理解の支援を行うシステムである [3]。Online Python Tutor の使用画面を図 2・図 3 に示す。ソースコードの指定した行の実行後の状態（変数・配列の値）を表示する。1 行ずつ進んだり戻ったりすることができる。しかし，実行後の状態のみを表示するため，プログラムの動作の理解に必要な 1 行の実行の前後の変数・配列の値の変化を把握するのに手間がかかる。

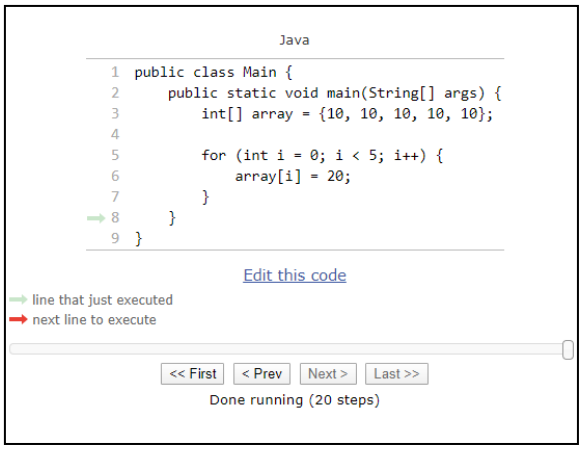

図 2　Online Python Tutor のソースコード部分

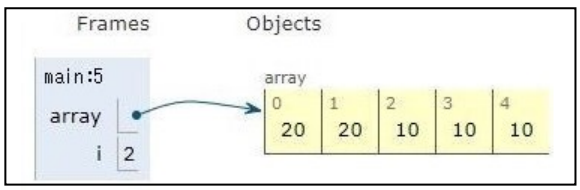

図 3　Online Python Tutor の可視化部分

2.2 伊藤の可視化システム

伊藤は，既存の可視化システムである Online Python Tutor を拡張して，1 行の実行の前後の変数・配列の値の変化を 1 つの画面で閲覧できるようにした[2]。また，変数・配列の値の変化している部分に色づけをして強調表示を行う。可視化部分に実行前後の状態を表示して変化を示すことで，プログラムのソースコード 1 行の実行前後の動作を確認できるようになっている。伊藤の可視化システムの使用画面を図 4・図 5 に示す。

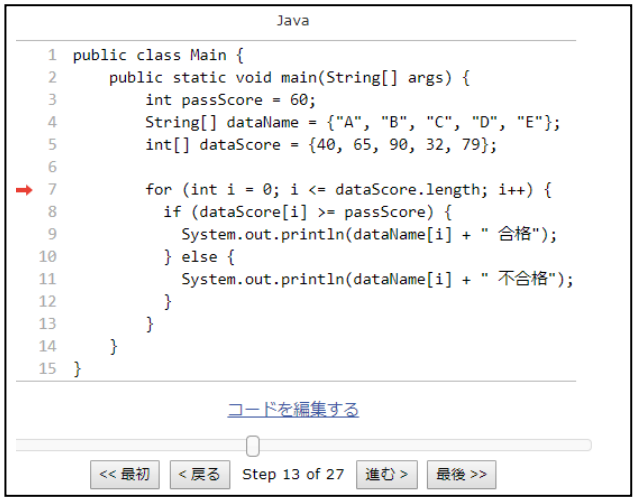

図 4　伊藤のシステムのソースコード部分

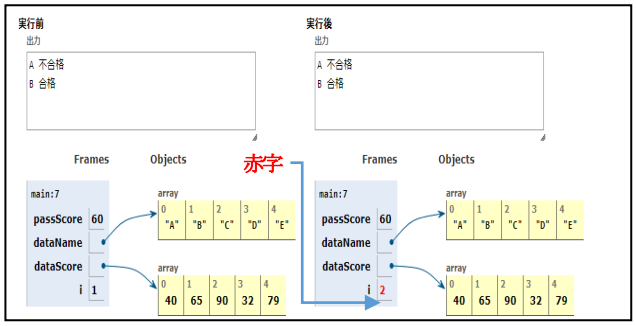

図 5　伊藤のシステムの可視化部分

伊藤が提案した可視化システムの機能を以下に示す。

(1)　学習者のプログラムで使用
(2)　図を用いて表現
(3)　図を自動生成
(4)　前の実行状態を表示
(5)　実行前後の状態を比較
(6)　変化部分を強調表示
(7)　変化した理由を表示
(8)　ブロック毎の変化の可視化
(9)　繰り返し動作の可視化

伊藤は，(1) から (6) までの機能を実装したが，(7)，(8)，(9) の機能については実装していない。

3.　運用上の問題点

伊藤の可視化システムの構成図を図 6 に示す。コーディング画面で学習者が書いたソースコードは実行トレース結果生成サーバに送られる。このサーバは，学習者のソースコードを受け取り，JSON 形式 [4] の実行トレース結果を Web ブラウザに返すことで可視化を行う。

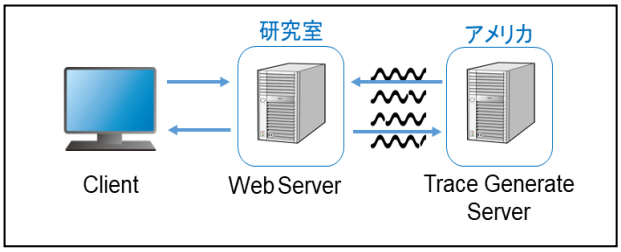

図 6　伊藤の可視化システムの構成図

しかし，伊藤の可視化システムの実行トレース結果生成サーバはアメリカのサーバを借りているため，サーバの応答が遅く，プログラミング演習の講義で多人数では同時に使用できない。

4.　改善内容

本研究では，上記の伊藤が実装しなかった機能を追加する。また，可視化システムのサーバの構築方法を変更する。具体的な内容を以下に示す。

4.1　変化した理由の表示の実装

学習者から受け取ったソースコードより，抽象構文木を生成する。生成した抽象構文木を分析して，受け取ったソースコードの要素とその宣言の間の関係を見つけることで，プログラムの変化の理由を把握し，変化した理由の可視化を実装する。図 7 に「if (a == 5)」の時の抽象構文木を示す。

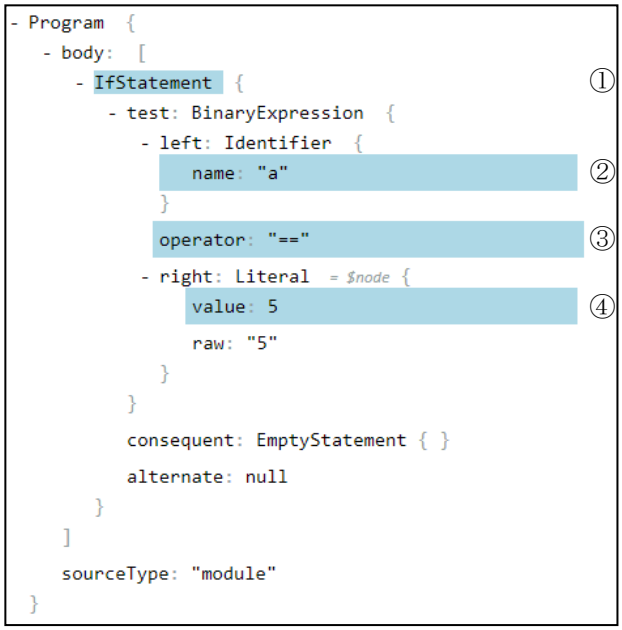

図 7　抽象構文木のイメージ

図 7 で示す例では，①から実行命令は制御文であり，②，③，④から変数名，式，変数の値を取得できることが分かる。取得した情報より，可視化を行っている命令に関係のある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，計算式の内容を可視化部分に表示する。この機能により，命令の実行前後の変化の理由を把握することを支援する。

図 8・図 9 に示す実際の画面を 1 つの例として，式の評価の可視化機能について説明する。ステップ 13 で実行する for 文は最初に i++ の式により，i の値を更新し，次に i<dataScore.length の条件式をチェックする。この実行の可視化を行うと i=1+1 で i の値が 2 に変化し，条件をチェックした後に TRUE の判定結果が表示される。これらの表示により，学習者は変数・配列の値が変化した理由を理解することができる。

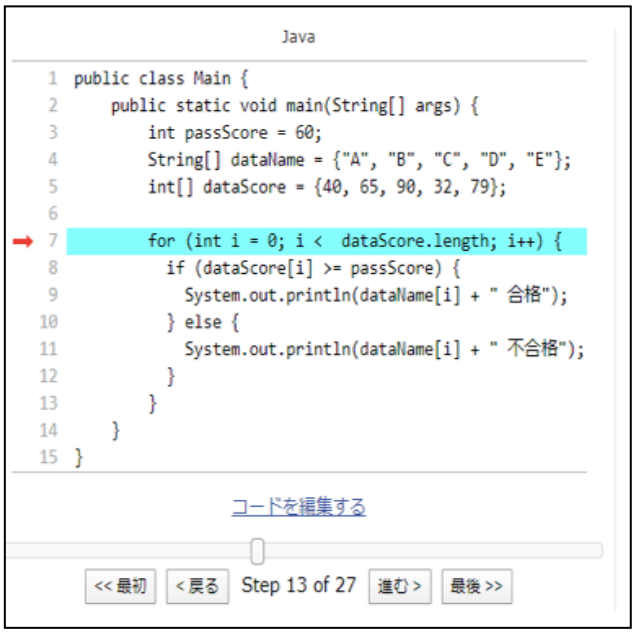

図 8　本システムのソースコード部分

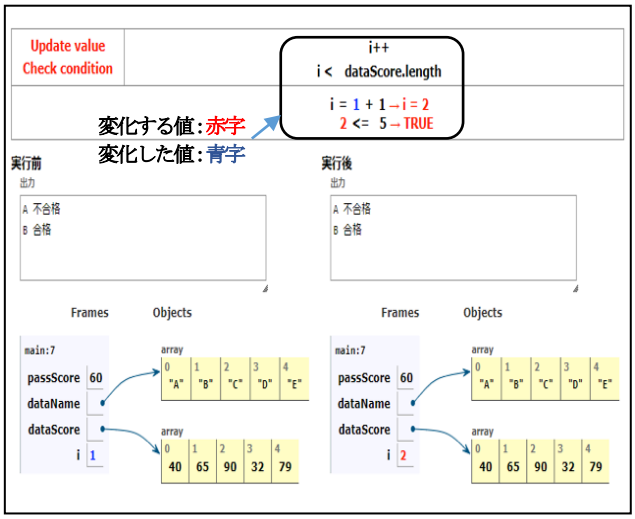

図 9　本システムの可視化部分

4.2　伊藤の可視化システムの改善

3 章で述べたように，伊藤のシステムでは実行トレース結果を生成するサーバはアメリカのサーバを借りているため，サーバの応答が遅く，多人数で同時に利用できない。プログラミング演習授業で多くの学習者が同時に使用できるようにするために，Web サーバと実行トレース結果を生成するサーバを同じ研究室に設置する。

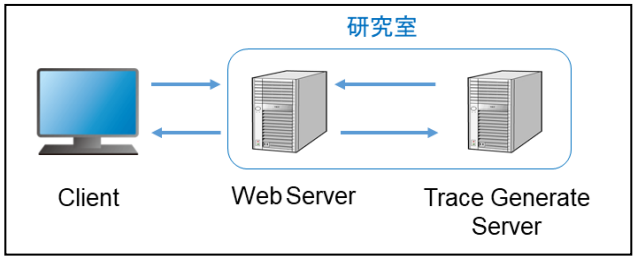

図 10　可視化システムの改良の構成図

図 10 に伊藤の可視化システムの改良の構成図を示す。同じ研究室に設置することで，リクエストしてから応答があるまでの時間が短くなると期待できる。

5.　サーバの構成を変更した結果

4.2 節で述べた改善の効果を確認するため，伊藤システムと改善システムにそれぞれ実行時間が非常に短い奇数・偶数の判定プログラムと実行に少し時間のかかる Bubble Sort プログラムを用いて，以下の実験を行なった。

実験 1 では，実行時間を 15 秒に制限し，1 リクエストに対してサーバからレスポンスが戻って来るまでの時間を計った。実験 1 の実験結果を表 1 に表す。

実験 2 では，実行時間を制限せずに，100 リクエストに対してサーバからレスポンスが戻って来るまでの時間を計った。実験 2 の実験結果を表 2 に表す。

表 1　伊藤システムと改善システムの比較
（1 リクエスト，実行時間制限：15 秒）

実行プログラム	伊藤システム	改善システム
奇数偶数の判定	4.4 秒	1.9 秒
Bubble Sort	×	7.2 秒

表 2　伊藤システムと改善システムの比較
（100 リクエスト，実行時間制限：なし）

実行プログラム	伊藤システム	改善システム
奇数偶数の判定	121.9 秒	23.2 秒
Bubble Sort	△	24.4 秒

実験 1 について，Bubble Sort の場合，伊藤システムは制限時間の 15 秒以内には応答がなかったのに対して，改善システムでは 7.2 秒で応答があった。実験 2 については，Bubble Sort の場合，6 分経っても応答がなかったので実行を強制的に中止した。

実験 2 では，100 リクエストを同時に送った。実際の利用を考えると，100 人規模の授業でも 100 リクエストを一瞬の間に同時に送ることはないため，改善システムは 100 人規模の授業で使用可能である。また，初学者が書くような短いプログラムの場合は 30 秒以内に応答があることが分かった。

6.　今後の実装の予定

6.1　ブロック毎の変化の可視化

繰り返し文や条件分岐等の条件文を 1 つのブロックとしてみなし，1 つのブロックの実行前と実行後の変数の値や配列の可視化を行う。1 つのブロックの実行前後の変化を表示することで，マクロな視点でプログラムの動作を把握することを支援する。イメージ図を図 11 に示す。図 11 (A) に実行前，図 11 (B) に実行後の状態を表示する。

6.2　繰り返しの動作の可視化

繰り返し文は 1 ブロックの実行ごとに応じて変数と配列の状態を表示する。繰り返し文のブロックの状態を可視化することにより，学習者が繰り返し文のブロック実行前後での変数の値や配列の値の変化を把握することを支援する。イメージ図を図 12 に示す。図 12 の (A)，(B)，(C) に 3 回の繰り返しの状態を表示する。

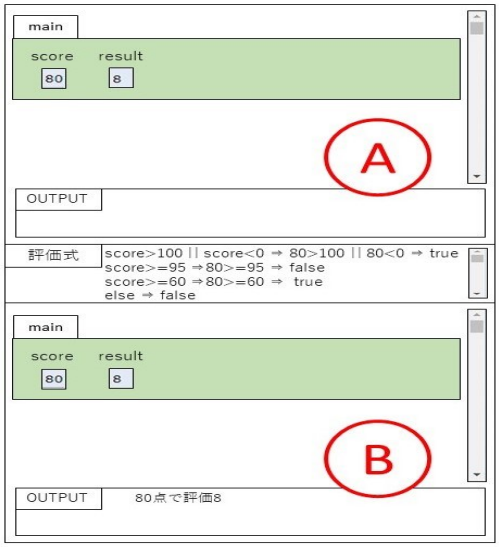

図 11 ブロック実行前後の状態表示

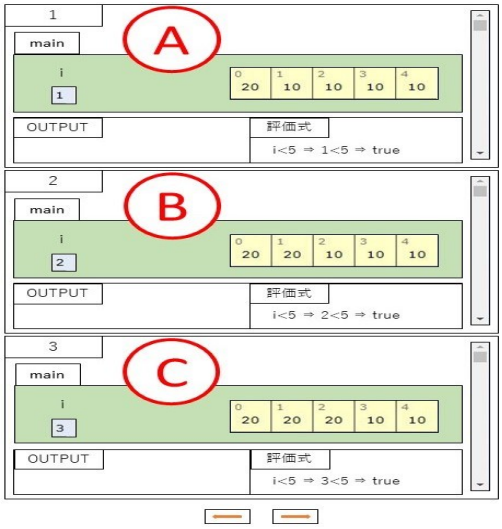

図 12 繰り返し文の可視化表示

7. おわりに

本研究では，変数・配列の値の変化した理由の表示の実装を行った。また，多人数の演習でも利用できるようにサーバの構築方法を変更した。

今後，以下の機能の実装を行う予定である。

- まとまりの実行前後状態の表示
- 繰り返し処理に気づきやすい表示

これらのすべての機能を実装すれば，プログラミングの初学者であってもプログラムの動作の確認が行えると予想される。完成後に実際の授業で利用して，その効果を確認する予定である。

参考文献

(1) 文部科学省：小学校プログラミング教育の手引（第二版）
(2) 伊藤 福晃，北 英彦：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，2018PC カンファレンス論文集，pp.29-32，2018
(3) Philip J. Guo：Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education，Proceeding of the 44th ACM Technical Symposium on Computer Science Education，pp. 579 -584，2013
(4) ECMA-404：The JSON Data Interchange Format (1st ed.). Geneva: ECMA International. October 2013