修士論文

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

平成 30 年度修了

三重大学大学院工学研究科

博士前期課程　電気電子工学専攻

伊藤　福晃

目次

第 1 章　はじめに 1
第 2 章　初学者のプログラミング学習の現状 3
第 3 章　必要な支援 6
3.1　可視化の自動生成 6
3.2　命令の実行前後の状態を表示 6
3.3　命令の実行前後で変化している部分を強調表示 6
3.4　条件文の評価式とプログラム中の計算式の詳細表示 7
3.5　ブロック表示 7
第 4 章　先行研究 8
4.1　Jeliot 3 8
4.2　Online Pyhton Tutor 9
4.3　静岡大学の考案したシステム 10
4.3.1　プログラムの 1 ステップの実行前後での変化 10
4.3.2　同じプログラムで異なるデータを用いた際の挙動の変化 11
4.3.3　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 11
第 5 章　実装 13
5.1　可視化の自動生成 14
5.2　命令の実行前後の状態を表示 15
5.3　命令の実行前後で変化している部分を強調表示 15
5.4　条件文の評価式とプログラム中の計算式の詳細表示 16
5.5　ブロック表示 17
5.6　使用方法 19
第 6 章　実験 20
6.1　実験期間と対象者 20
6.2　実験方法 20
6.2.1　可視化システムを使わずにプログラムを修正 20
6.2.2　従来の可視化システムを用いてプログラムを修正 21
6.2.3　本システムを用いてプログラムを修正 22
6.3　アンケート調査 25
第 7 章　実験結果と考察 26
7.1　課題の正答率と考察 26
7.2　アンケート内容，アンケート結果および考察（質問Ⅰ～Ⅳ） 26
7.2.1　質問Ⅰの内容と結果 27

7.2.2　質問Ⅱの内容と結果 27
7.2.3　質問Ⅲの内容と結果 28
7.2.4　質問Ⅳの内容と結果 29
7.2.5　質問Ⅰ～Ⅳの結果のまとめと考察 29
7.3　アンケート内容，アンケート結果および考察（質問 1～12） 30
7.3.1　質問 1 と 2 の内容，結果および考察 30
7.3.2　質問 3 と 4 の内容，結果および考察 31
7.3.3　質問 5 と 6 の内容，結果および考察 32
7.3.4　質問 7 と 8 の内容，結果および考察 33
7.3.5　質問 9 と 10 の内容，結果および考察 34
7.3.6　質問 11 と 12 の内容，結果および考察 35
第 8 章　まとめ 37
参考文献 38
実績一覧 41
謝辞 42

第1章 はじめに

プログラミング学習は近年，小学校の授業への組み込みが決定されるなど，情報化社会が発展を遂げる中で更なる注目を集めている．その注目に合わせて今日では，多くの教育機関でプログラミング教育が行われている．一般的なプログラミングの入門教育方法として，演習型の授業方式が取り入れられている．演習型の授業方式とは，講義の初めに基本的なプログラムの文法や処理の流れを講義形式で解説し，その後，指導者があらかじめ用意しておいた演習課題を学習者が各自で解いていくというものである．演習によって自分でプログラムを書くことにより，実践的なプログラミング能力を学習することができる．また，プログラミング教育はマンツーマンによる指導が効果的であるとされている [1]. 演習型の授業方式の問題点として，学習者の視点では，指導者の数が限られているため，対話を行えるのは学習者の一部となり学習者全員が十分な支援を受けることが困難であるという点，指導者の視点では，学習者全員の状況を把握するには時間を要し，指導に大きな負担が発生する点が挙げられる [2]．そのために，演習において初学者はプログラミング学習でつまずくことがしばしばある．初学者がつまずく点の 1 つとして，自身が作成したプログラムの動作が誤っている場合に修正できないという点がある．その原因として，プログラムが複雑になると，自分で作ったプログラムの動作が誤っていても，動作の誤っている原因となる場所を特定できないことが考えられる．

プログラムの動作が間違っている場所を特定する方法として，一般的にデバッガが用いられる．しかし，既存のデバッガは初心者向けに作られていないことが多く，初学者にとって利用法習得の負荷が大きいという問題がある．その一例として，デバッガの機能の 1 つである「ブレークポイント」は，初学者が適切に設定するのが困難であるといわれている [3]．また，プログラムの理解を助けるための手法として，様々なプログラム可視化システムが開発されており，可視化システムはプログラミング学習の初学者の様々な側面を支援するツールとして使用されている [4-6]. また，可視化システムにはプログラミング学習に効果があることが知られている [7, 8].

そこで本研究では，プログラムの動作が誤っている時に，間違えている場所を特定することができない初学者を対象として，変数の値や配列の値の移り変わりを把握する支援をすることを目的とする．そして，可

視化システムを使用することで支援を行う．初学者向けのプログラミング演習担当教員に，変数の値や配列の値の移り変わりを把握するのに必要な支援についての聞き取り調査を行い，必要な支援となる機能を備えた支援システムを構築した．そして，初学者を対象に本システムを使用してもらい，アンケート調査を行うことでその有効性を確かめた．

本論文の構成は以下のとおりである．2 章で初学者のプログラミング学習の現状について述べる．3 章で必要な支援について述べる．4 章では先行研究について述べる．5 章では実装について述べる．6 章では実験について述べる．7 章で実験結果と考察について述べる．8 章で本論文のまとめを述べる．

第2章 初学者のプログラミング学習の現状

プログラミング演習において，学習者はプログラミングをするにあたり，いくつかの工程を経ることでプログラムを作成する．学習者はプログラムを作成する際に以下の工程を行う．

1. コーディング
2. コンパイル
3. テスト
4. デバッグ

学習者は，工程 4 のテストを行い，プログラムの動作が正しければ目的とするプログラムは完成し，正しくなければデバッグを行う（図 1）．

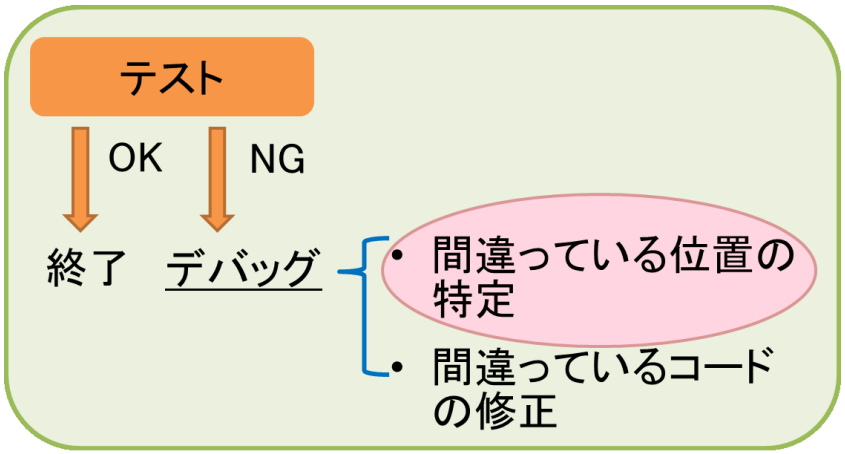

図 1　デバッグの手順

さらに，デバッグには図 1 に示すような 2 つの工程が存在する．

1. 間違えている位置の特定
2. 間違えているコードの修正

学習者はこの 2 つの工程を行うことでプログラムを修正し，再び動作のテストを行って，作成したプログラムが正しいかを判断する．初学者向けプログラミング演習において，デバッグの 1 つ目の工程である間違えている位置の特定ができないような初学者が存在する．その原因とし

て，プログラムが複雑になると変数の値や配列の値の移り変わりを把握することができないことが考えられる．実際に，三重大学工学部電気電子工学科 2 年次が受講対象である「プログラミング演習Ⅰ・Ⅱ」の授業において，図 2 に示すような，単純なプログラムの実行後の変数を答える問題を出題すると，正答率は 94.6%と，ほとんどの初学者が変数の値を最後まで追うことができるということが考えられる．

問 1. 実行後の変数 result の値を書きなさい.

```
int result = 0;
for (int i = 10; i > 0; i --) {
    result = result + i;
}
```

図 2　プログラムの変数を追う問題 1

しかし，図 3 に示すような少し複雑なプログラムにおいて，同じ問題を出題すると正答率は 58.1%となり，変数の値を最後まで追うことができない初学者が存在することが考えられる．

問 2. 実行後の変数 result の値を書きなさい.

```
int result = 0;
for (int i = 0; i < 10; i ++) {
    for (int j = 0; j < i; j ++) {
        result ++;
    }
}
```

図 3　プログラムの変数を追う問題 2

以上より，実際に，動作の複雑なプログラムになると変数の値や配列の値の移り変わりを把握することができなくなる初学者が増加していることがわかる．

本研究では，自分で作成したプログラムの間違えている位置の特定ができないような初学者を対象として，初学者が変数の値や配列の値の移り変わりを把握する支援を行う．3 章では，初学者が変数の値や配列の値の移り変わりを把握するのに必要な支援について述べる．

第3章 必要な支援

本章では，初学者向けのプログラミング演習担当教員に，プログラミング可視化システムを使用して，変数の値や配列の値の移り変わりを把握するために必要な支援についての聞き取り調査を行った．その結果，必要な支援は以下のようになった．

① 可視化の自動生成
② 命令の実行前後の状態を表示
③ 命令の実行前後で変化している部分を強調表示
④ 条件文の評価式，プログラム中の計算式の詳細表示
⑤ ブロック表示

聞き取り調査の結果である上記 5 点について，詳細を以下に述べる．

3.1 可視化の自動生成

1 章で述べたように，プログラミング演習において，講師の人数に対する学習者の人数が多いために，学習者一人一人に対応することが難しい．そのため，学習者自身が書いたプログラムを可視化して支援する必要がある．

3.2 命令の実行前後の状態を表示

学習者は，変数の値や配列の値の移り変わりを把握するために，ソースコード 1 行の実行前後において，変数の値や配列の値がどのように変化しているのかがわかるようにする必要がある．そのために，ソースコード 1 行の実行前後の変数や配列の値を示して支援する必要がある．

3.3 命令の実行前後で変化している部分を強調表示

ソースコード 1 行の実行前後で変数の値や配列の値を可視化して表示するだけでは，学習者は変化している部分に注目することが難しい．そのため，ソースコード 1 行の実行前後で変数の値や配列の値が変化している部分を強調表示して支援する必要がある．

3.4　条件文の評価式とプログラム中の計算式の詳細表示

学習者は条件文の評価式の内容や，プログラム中の計算の内容を読み取り間違えていることがある．そのため，評価式や計算式の詳細を示して支援する必要がある．

3.5　ブロック表示

ソースコード 1 行の実行前後での可視化だけでは，プログラムの大きな視点から変数の値や配列の値の変化を把握することが難しい．そのため，繰り返し文と条件分岐文を 1 つのブロックとしてみなし，ブロックの実行前後の変数の値や配列の値を表示して支援する必要がある．特に，繰り返し文はブロックの前後の表示だけでは繰り返し文のループが進むごとの変数の値や変数の値の変化を把握することが難しい．そのため，1 回分のループを 1 ブロックとしてみなし，ブロック実行後の状態を 3 回分表示して支援する必要がある．

第4章 先行研究

3 章では，変数の値や配列の値の移り変わりを把握するのに必要な支援について述べた．本章では，3 章で述べた必要となる支援に類似した機能をもつ代表的な可視化システムを紹介する．また，以下に示す可視化システム以外にも，様々な可視化システムは存在している [9-12].

4.1 Jeliot 3

Jeliot 3 は，学習者が書いたプログラムを可視化して動画で表示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 [13-16]．Jeliot 3 の画面を図 4 に示し，可視化部分だけを拡大したものを図 5 に示す．Jeliot 3 はプログラミング初学者を対象としている．デバッガのように動作し，実行，ステップ実行を行うことができる．学習者は，Jeliot 3 にプログラムを入力し，「Play」ボタンを押すことで，動画を再生できる．ステップ実行の際には，1 ステップずつ動画が再生されていく．しかし，1 つ前の実行後の状態の再生をすることができず，前の実行状態を見るには最初から再生を行う必要がある．学習者は変数の値や配列の値の移り変わりを把握する際に，再生した命令の実行前の表示を行うことが煩雑で，把握することが難しいと考えられる．

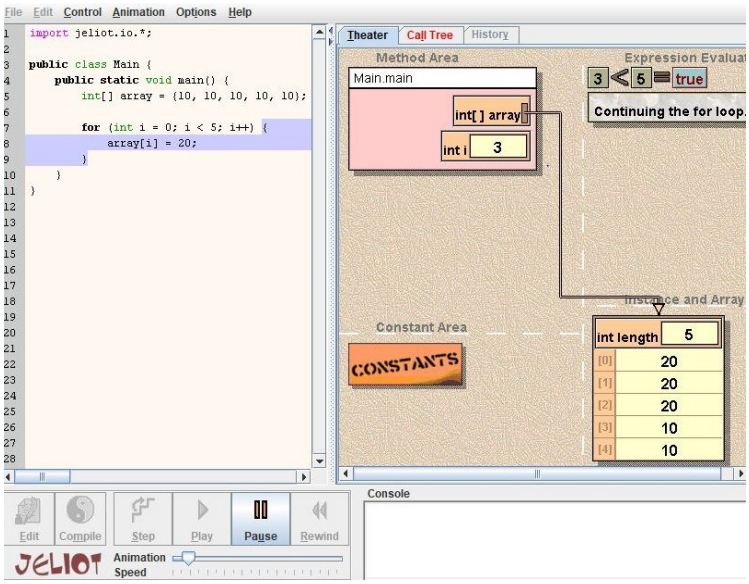

図 4 Jeliot 3

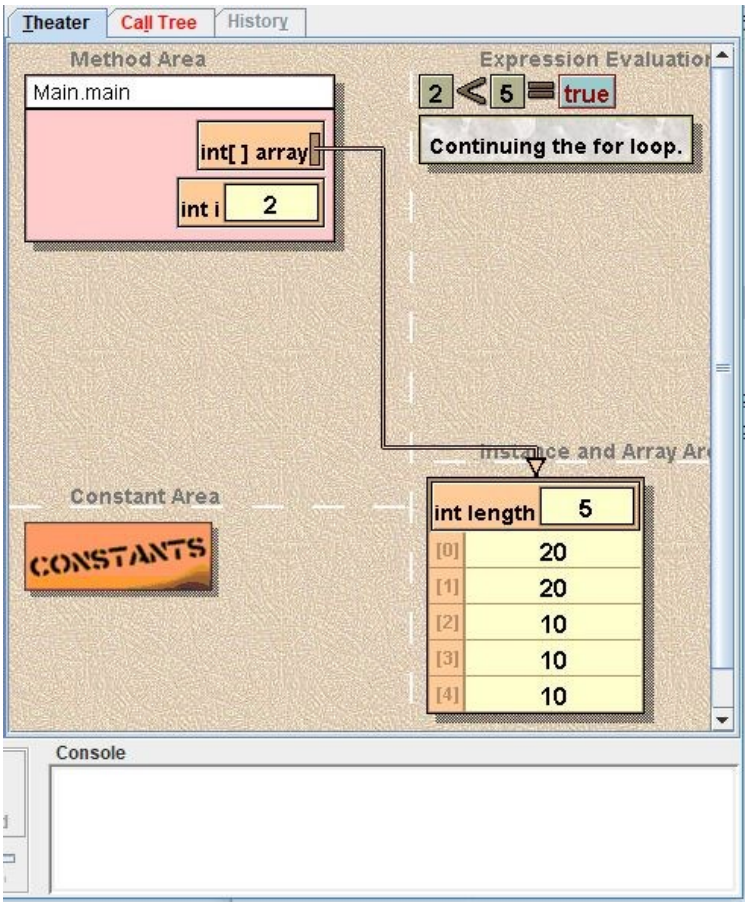

図 5　Jeliot 3 の可視化部分

4.2　Online Pyhton Tutor

Online Python Tutor は，学習者が書いたプログラムを，変数の値や配列の値を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 [4]. Online Python Tutor を図 6 に示し，可視化部分を図 7 に示す．ソースコード 1 行ずつの実行後の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．学習者は，プログラムを入力し，「Visualize Execution」ボタンを押すことで可視化を行い，プログラム中で選択した行の実行後の変数の値や配列の値を見ることができる．しかし，一度に 1 つの実行状態のみを表示するため，プログラム命令の実行前後で変数の値や配列の値の移り変わりを把握することが難しい．

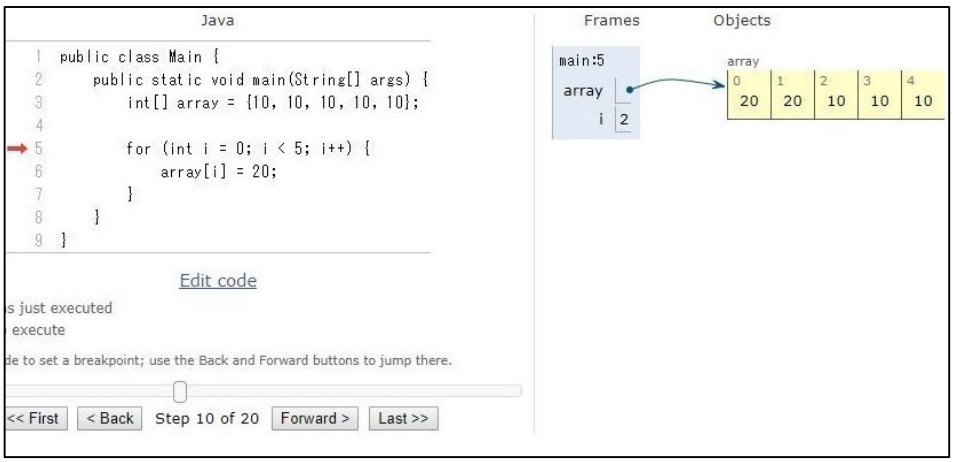

図 6　Online Python Tutor

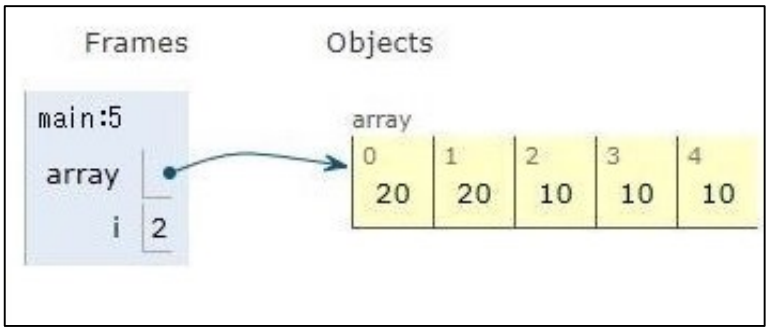

図 7　Online Python Tutor の可視化部分

4.3　　静岡大学の考案したシステム

SCBCR（Scenes Comparison Based Code Reading）と呼ばれる比較表示による学習を用い，TEDViT（Teacher Explaining Design Visualization Tool）を拡張したシステムである [5]．アルゴリズムをプログラムで実装した際，アルゴリズムの特長をつかむために可視化を行うシステムである．アルゴリズムをプログラムで実装したときに，以下の可視化を行う．

4.3.1　プログラムの 1 ステップの実行前後での変化

図 8 に示すように，①にソースコード，②にシステムを操作するボタン，③に選択したソースコードの命令の実行前の変数の値と配列の値，④に選択したソースコードの命令の実行後の変数の値と配列の値，⑤に学習者へのメッセージが表示される．

4.3.2　同じプログラムで異なるデータを用いた際の挙動の変化

図 9 に示すように，①と②にはプログラム動作の説明のタイトル，③と④にはソースコード，⑤と⑥には選択したソースコードの命令の実行後の変数の値や配列の値，⑦にシステムを操作するボタン，⑧には学習者へのメッセージが表示される．

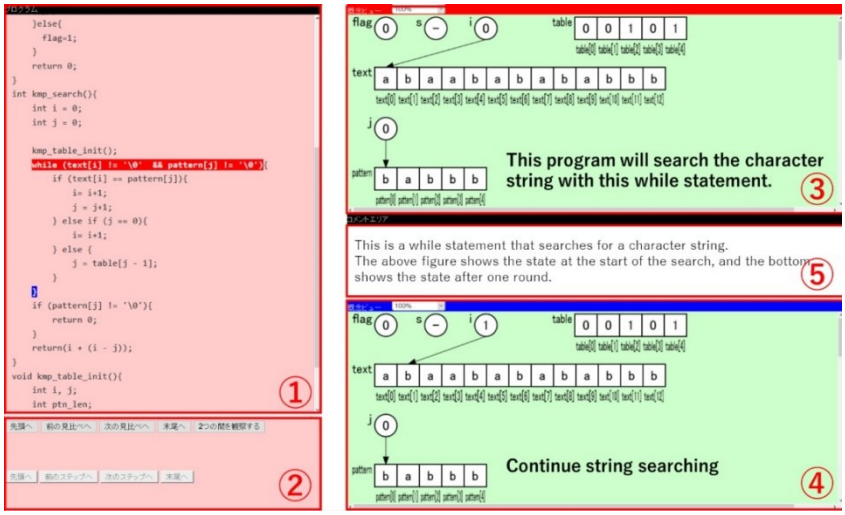

図 8　静岡大学のシステム①

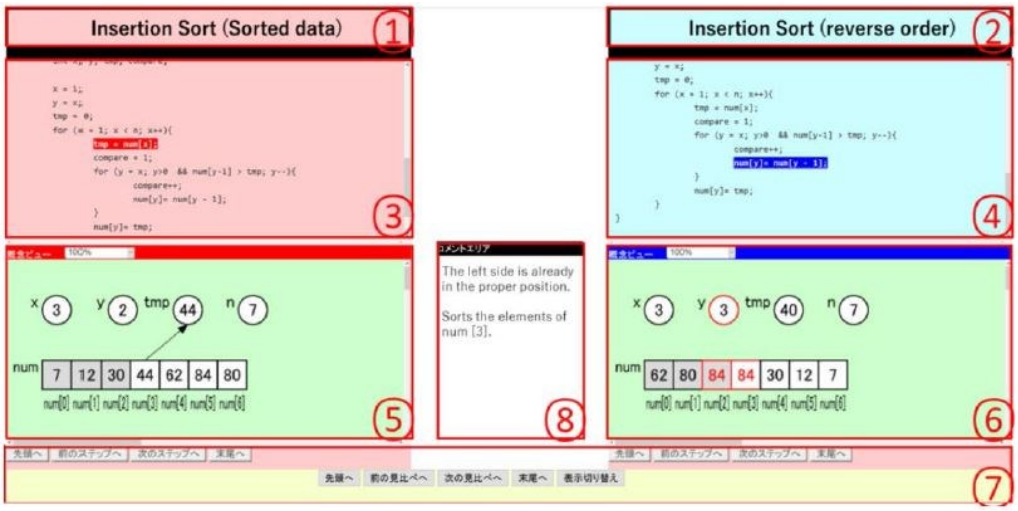

図 9　静岡大学のシステム②

4.3.3　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

4.4.2 と同様の方法で表示される．

これらの表示を行うことでアルゴリズムをプログラムで実装した際の理解を支援するシステムである．プログラムの 1 ステップの実行前後の状態を表示しているため，学習者は変数の値や配列の値の移り変わりを把握しやすいと考えられる．

このシステムは，学習者が書いたプログラムを自動で可視化するのではなく，講師があらかじめ説明したいプログラムの可視化表示を用意し，表示することでアルゴリズム理解を支援するシステムである．従って，学習者は自分で作成したプログラムについての可視化を行うことができない．

第5章 実装

本研究室では，プログラミング演習中に学習者が書いているプログラムの状況をリアルタイムで把握することができるプログラミング演習支援システム PROPEL（PROgramming Practice Easy for Leaners） [19-21]を開発し，実際に三重大学電気電子工学科のプログラミング演習Ⅰ・Ⅱの授業 [22, 23]で運用を行っている．本研究の提案するシステムをサブシステムとして付け加える．なお，本システムは，既存の可視化システムである Online Python Tutor を拡張することで実装を行った．

本システムの構成図を図 10 に示す．コーディング画面でコーディングされたソースコードは Web サーバに送られる．そして，JSON 形式の実行トレース結果となり Web ブラウザに返されることで可視化を行う．

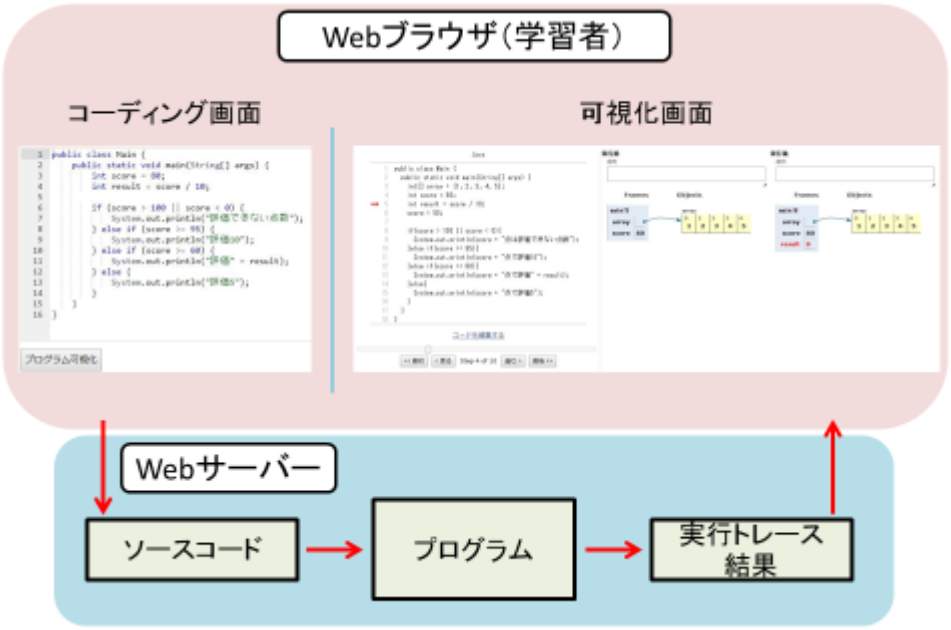

図 10　本システム構成図

3 章で述べた，以下の 5 つの必要な支援についての実装と仕様の決定を行った．

1. 可視化の自動生成
2. 命令の実行前後の状態を表示
3. 命令の実行前後で変化している部分を強調表示
4. 条件文の評価式とプログラム中の計算式の詳細表示
5. ブロック表示

これらの機能について，1-3 については実装を行い，4 と 5 は仕様の決定までを行った．以下にそれぞれについて説明をする．

5.1 可視化の自動生成

本システムは，既存の可視化システムである Online Python Tutor を拡張することで実装を行い，Online Python Tutor と同様に可視化を自動生成する．本システムの可視化画面を図 11 に示す．画面左側には可視化を行うプログラムのソースコードを，右側に変数の値や配列の値を可視化表示する．

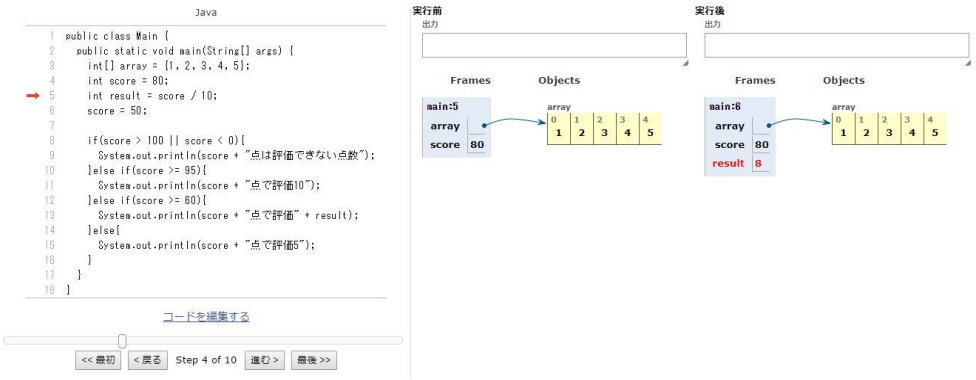

図 11　本システムの可視化画面全体

5.2　　命令の実行前後の状態を表示

図 12 のように，命令の実行前後の変数の値や配列の値の表示する．

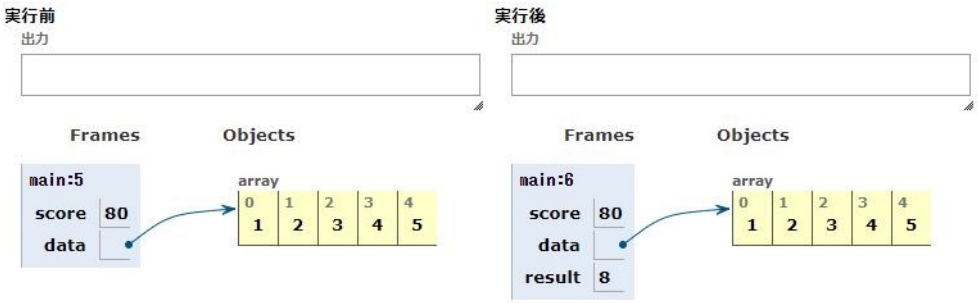

図 12　命令の実行前後の状態表示

5.3　　命令の実行前後で変化している部分を強調表示

命令の実行前後で，変数の値や配列の値が変化している部分に色付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．

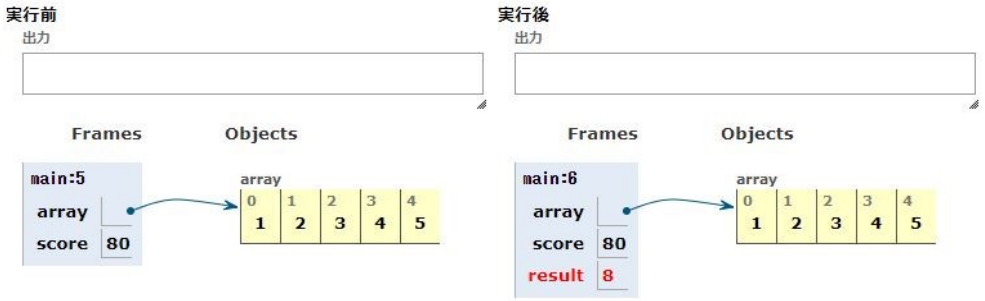

図 13　本システムの可視化部分

5.4　　条件文の評価式とプログラム中の計算式の詳細表示

繰り返し文や条件分岐文の評価式や，プログラム中の計算式を表示する機能は，仕様の決定までを行った．イメージ図を図 14 に示す．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，その true，false を表示する．同様に，計算式の内容を可視化部分に表示する．これらの機能により，命令の実行前後での変化の理由を把握することを支援する．イメージ図を図 15 に示す．

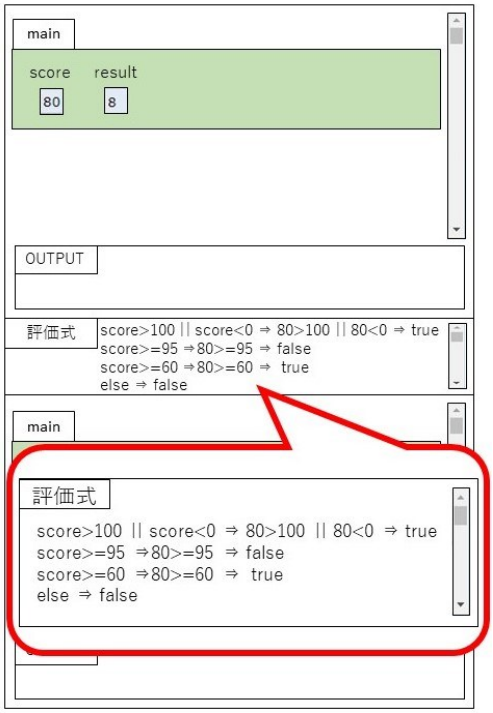

図 14　評価式の詳細表示

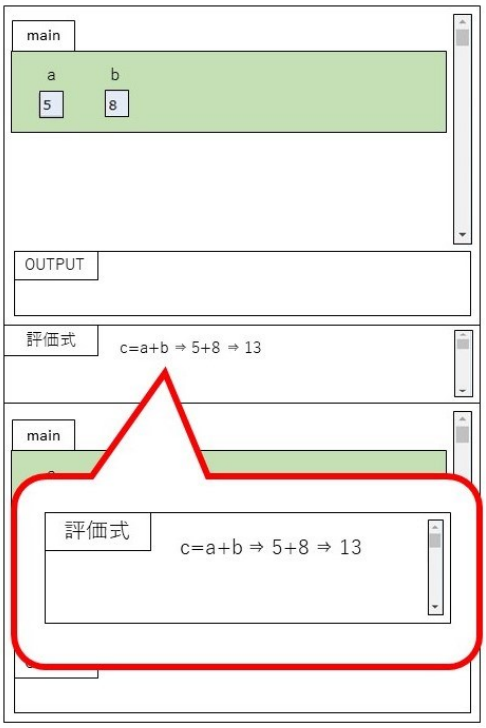

図 15　計算式の詳細表示

5.5　　ブロック表示

ソースコード 1 行の実行前後での可視化だけでなく，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機能は，仕様の決定までを行った．イメージ図を図 16 に示す．条件文を 1 つのブロックとして捉え，そ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．

さらに，繰り返し文は 1 ブロックの実行ごとに 3 回分の変数と配列の状態を表示する．イメー図を図 17 に示す．3 回分のブロックの状態を可視化表示することにより，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．

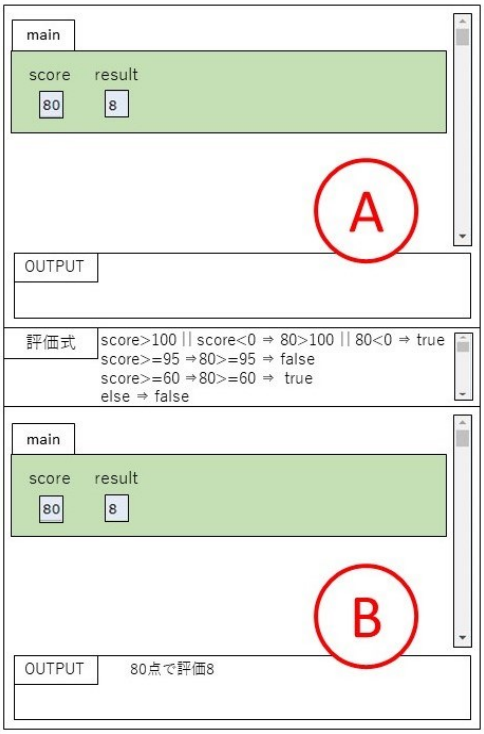

図 16　ブロックの実行前後の状態表示

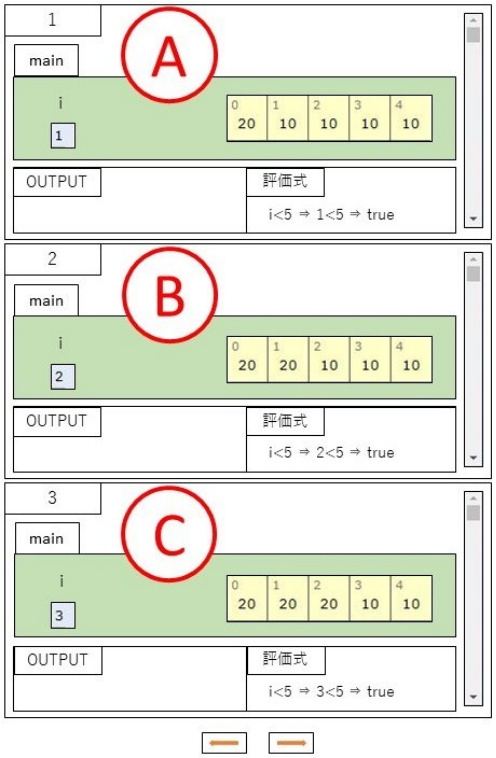

図 17　繰り返し文の可視化表示

5.6　使用方法

図 18 のソースコード入力画面にソースコードを入力し，入力欄下の「可視化する」ボタンを押す．図 11 の可視化画面が現われたら，左側のソースコード表示の下の「戻る」「進む」のボタンを押して，ソースコードを 1 行ずつ進めていくことができる．「最初」「最後」ボタンは，ソースコードの最初や最後に実行状態を移すことができる．上記ボタンの上にはスクロールバーが表示されており，ソースコード中の任意の位置に移動させることで，任意の実行状態を確認することができる．

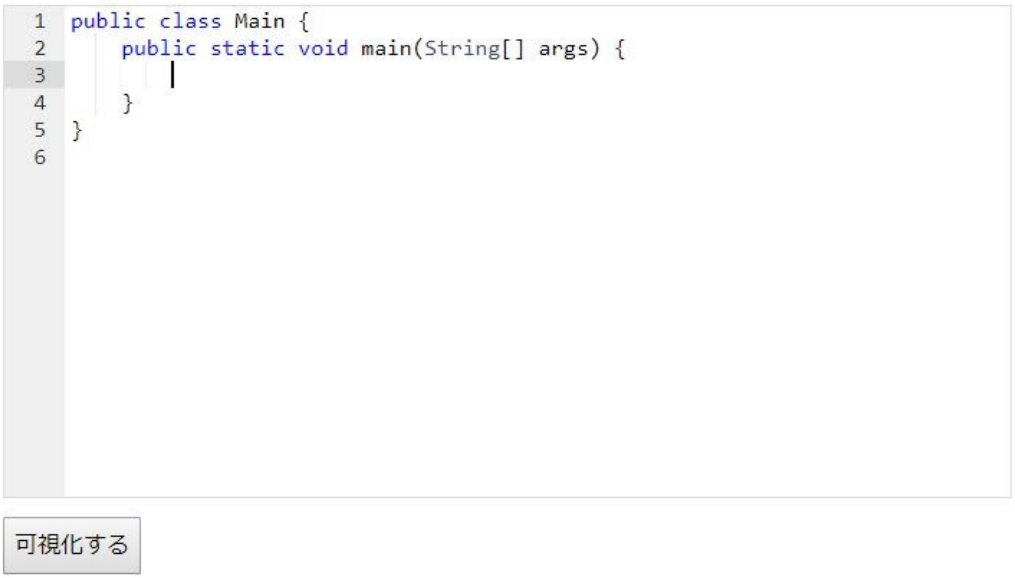

図 18　本システムの入力画面

第6章 実験

本システムにより，学習者が，変数の値や配列の値の移り変わりを把握する支援できているかを確認するため，以下の実験を行った．

6.1 実験期間と対象者

- 実験期間：2019 年 1 月 24～2019 年 1 月 31 日
- 対象者：三重大学工学部電気電子工学科在学中のプログラミング演習Ⅱの受講者
- 対象者数：64 人

6.2 実験方法

動作の誤っているプログラムを初学者に与え，修正を行うという形式で，以下の 3 種類の課題で実験し，正解率を確認した．

6.2.1 可視化システムを使わずにプログラムを修正

可視化システムを使わずにプログラムを修正する問題を出題した．問題文を図 19 に，修正させるプログラムを図 20 に，修正後の正解のプログラムを図 21 に示す．

・表示されるプログラムを修正する。
・支援システムは使用しない。
プログラムの動作は以下の通り：
整数型の配列が渡される。配列の各要素には自然数が格納されている戻り値なし。新しい配列は使わずに渡された配列の中の要素を逆順する。例えば「1, 2, 3」が格納された配列が渡された場合には「3, 2, 1」を格納してメソッドを終了する。

図 19　問題文 1

```java
public class Main {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        for (int i = 0; i < array.length; i++) {
            System.out.printf("%d ", array[i]);
        }
        System.out.println("");
        reverseArray(array);
        for (int i = 0; i < array.length; i++) {
            System.out.printf("%d ", array[i]);
        }
        System.out.println("");
    }
    public static void reverseArray(int[] array) {
        for (int i = 0; i < array.length; i++) {
            array[i] = array[array.length - i - 1];
            array[array.length - i - 1] = array[i];
        }
    }
}
```

図 20　プログラム 1

```java
public class Main {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        for (int i = 0; i < array.length; i++) {
            System.out.printf("%d ", array[i]);
        }
        System.out.println("");
        reverseArray(array);
        for (int i = 0; i < array.length; i++) {
            System.out.printf("%d ", array[i]);
        }
        System.out.println("");
    }
    public static void reverseArray(int[] array) {
        for (int i = 0; i < array.length / 2; i++) {
            int tmp = array[i];
            array[i] = array[array.length - i - 1];
            array[array.length - i - 1] = tmp;
        }
    }
}
```

図 21　プログラム 1 の正解

6.2.2　従来の可視化システムを用いてプログラムを修正

従来システムである Online Python Tutor を用いてプログラムを修正する問題を出題した．使用した問題とプログラムは 6.2.3 でまとめて説明する．

6.2.3　本システムを用いてプログラムを修正

本システムを用いてプログラムを修正する問題を出題した．使用した問題とプログラムは，受講者を半分（A グループ，B グループ）に分け，表 1 のようにして実験した．

表 1　6.2.2 と 6.2.3 の実験方法

	問題	
	図16	図18
従来システム	Aグループ	Bグループ
本システム	Bグループ	Aグループ

問題 2 の問題文 2 を図 22 に，プログラム 2 を図 23 に，プログラム 2 の正解を図 24 に示す．また，問題 3 の問題文 3 を図 25 に，プログラム 3 を図 26 に，プログラム 3 の正解を図 27 に示す．それぞれのプログラムに対して，間違いの 1 つ目を間違い①，2 つ目を間違い②とした．

・メインメソッドは変更せずに修正する。
目的：配列の要素を昇順に並べる。選択ソート。
手順の考え方：
最初の要素の添え字を覚えておく。
最小値を見つけ、最初の要素と値を入れ替える。
2 番目以降の要素に対して再度同じことをする。
要素が一つになるまで繰り返す。

図 22　問題文 2

```java
public class Main {
    public static void main(String[] args) {    //メインメソッド.修正しない
        int[] data = {8, 6, 9, 4};
        sort(data);
        for(int i = 0; i < data.length; i++) {  //ソート後の配列を表示。
            System.out.print(data[i] + " ");
        }
    }

    public static void sort(int[] array) {      //修正するメソッド
        for (int i = 0; i < array.length - 1; i++) {
            int min = i;
            for (int j = 0; j < array.length; j++) {
                if (array[min] > array[j]) {
                    min = j;
                }
            }
            array[i] = array[min];
            array[min] = array[i];
        }
    }
}
```

図 23　プログラム 2

```java
public class Main {
    public static void main(String[] args) {    //メインメソッド.修正しない
        int[] data = {8, 6, 9, 4};
        sort(data);
        for(int i = 0; i < data.length; i++) {  //ソート後の配列を表示。
            System.out.print(data[i] + " ");
        }
    }

    public static void sort(int[] array) {    //選択ソートメソッド
        for (int i = 0; i < array.length - 1; i++) {
            int min = i;
            for (int j = i + 1; j < array.length; j++) {
                if (array[min] > array[j]) {
                    min = j;
                }
            }
            int tmp = array[i];
            array[i] = array[min];
            array[min] = tmp;
        }
    }
}
```

図 24　プログラム 2 の正解

・メインメソッドは変更せずに修正する。
目的：配列の要素を昇順に並べる。バブルソート。
手順の考え方：
最初の要素から隣り合う要素の大小を比較して、
順番が正しくないときにいれかえる。
一通りいれかえると、一番大きな値が配列の最後の要素になる。
最初の要素から同じ処理をしていく。
ただし、2 回目は最後の要素は比較の対象としない。
これを、要素数が 1 つになるまで繰り返す。

図 25　問題文 3

```
13 public class Main {
14     public static void main(String[] args) {    //メインメソッド.変更しない
15         int[] data = {7, 5, 8, 3};
16         sort(data);
17         for(int i = 0; i < data.length; i++) {  //ソート後の配列を表示。
18             System.out.print(data[i] + " ");
19         }
20     }
21
22     public static void sort(int[] array) {      //修正するメソッド
23         for (int i = 0; i < array.length - 1; i++) {
24             for (int j = 0; j < array.length - i; j++){
25                 if (array[j] > array[j + 1]) {
26                     array[j] = array[j + 1];
27                     array[j + 1] = array[j];
28                 }
29             }
30         }
31     }
32 }
33
```

図 26　プログラム 3

```
13 public class Main {
14     public static void main(String[] args) {    //メインメソッド.変更しない
15         int[] data = {7, 5, 8, 3};
16         sort(data);
17         for(int i = 0; i < data.length; i++) {  //ソート後の配列を表示。
18             System.out.print(data[i] + " ");
19         }
20     }
21
22     public static void sort(int[] array) {    //バブルソート
23         for (int i = 0; i < array.length - 1; i++) {
24             for (int j = 0; j < array.length - i - 1; j++){
25                 if (array[j] > array[j + 1]) {
26                     int temp = array[j];
27                     array[j] = array[j + 1];
28                     array[j + 1] = temp;
29                 }
30             }
31         }
32     }
33 }
34
```

図 27　プログラム 3 の正解

6.3　アンケート調査

本システムが変数の値や配列の値の移り変わりを把握する支援をするのに有効であるかを確認するため，6.2.2 と 6.2.3 についてのアンケート調査を行った．アンケートの概要を以下に示す．

- アンケート I～Ⅳ：
  - プログラム中の間違っている位置に気づいたかについての質問
- アンケート 1～12：
  - 本システムの機能が，変数の値や配列の値の移り変わりを把握するのに役立ったかについての質問．
  - 従来システムよりも機能が増えたことによって，見やすさと操作性が低下していないかについての質問．
  - その他意見・感想等

第7章 実験結果と考察

学習者の正解率，アンケート結果を以下に示す.

7.1 課題の正答率と考察

6.2.1～6.2.3 それぞれの方法の正答率を表 2，表 3，表 4 に示す.

表 2 可視化システムを使用していない際の正答率

	正答率（可視化システムなし）			
点数	0	1	2	合計
人数	55	5	12	72
百分率	76%	7%	17%	100%

表 3 従来システムを使用した際の正答率

	正答率（従来システム使用）			
点数	0	1	2	合計
人数	56	9	13	78
百分率	72%	12%	17%	100%

表 4 本システムを使用した際の正答率

	正答率（本システム使用）			
点数	0	1	2	合計
人数	31	27	15	73
百分率	42%	37%	21%	100%

3 つの表から，1 点以上の百分率が，可視化システムを使用しない場合は 24%，従来システムを使用する場合は 28%，本システムを使用する場合は 58%となっており，本システムを使用すると，学習者は変数や配列の値の移り変わりを把握する支援をできていると考えられる.

7.2 アンケート内容，アンケート結果および考察（質問Ⅰ～Ⅳ）

質問Ⅰ～Ⅳの内容，アンケート結果および考察を以下に示す.

7.2.1　質問Ⅰの内容と結果

質問Ⅰの内容を図 28 に，その結果を表 5 に示す.

質問Ⅰ

従来システムを使用した際，課題の間違い①に気づきましたか.

1. 気づけなかった
2. 気づいたが直せなかった
3. 気づいて直せた

図 28　質問Ⅰの内容

表 5　質問Ⅰに対する回答

	従来システムで間違い①に気づいたか			
選択肢	1	2	3	合計
票数	30	16	17	63
百分率	48%	25%	27%	100%

7.2.2　質問Ⅱの内容と結果

質問Ⅱの内容を図 29 に，その結果を表 6 に示す.

質問Ⅱ

従来システムを使用した際，課題の間違い②に気づきましたか.

1. 気づけなかった
2. 気づいたが直せなかった
3. 気づいて直せた

図 29　質問Ⅱの内容

表 6　質問Ⅱに対する回答

	従来システムで間違い②に気づいたか			
選択肢	1	2	3	合計
票数	27	20	16	63
百分率	43%	32%	25%	100%

7.2.3　質問Ⅲの内容と結果

質問Ⅲの内容を図 30 に，その結果を表 7 に示す.

質問Ⅲ

本システムを使用した際，課題の間違い①に気づきましたか.

1. 気づけなかった
2. 気づいたが直せなかった
3. 気づいて直せた

図 30　質問Ⅲの内容

表 7　質問Ⅲに対する回答

	本システムで間違い①に気づいたか			
選択肢	1	2	3	合計
票数	26	14	24	64
割合	41%	22%	38%	100%

7.2.4　質問Ⅳの内容と結果

質問Ⅳの内容を図 31 に，その結果を表 8 に示す.

> **質問Ⅳ**
> 本システムを使用した際，課題の間違い②に気づきましたか.
>
> 1. 気づけなかった
> 2. 気づいたが直せなかった
> 3. 気づいて直せた

図 31　質問Ⅳの内容

表 8　質問Ⅳに対する回答

	本システムで間違い②に気づいたか			
選択肢	1	2	3	合計
票数	15	17	32	64
百分率	23%	27%	50%	100%

7.2.5　質問Ⅰ～Ⅳの結果のまとめと考察

質問Ⅰと Ⅲの結果より，従来システムを使用した時よりも本システムを使用した時の方が選択肢 2 と 3 の合計の百分率が 7 ポイント上がっている．これは，間違い①に気づくことができた回答者が，従来システムよりも本システムを使った人の方が 7 ポイント増えたことを示している.

質問Ⅱ とⅣの結果より，従来システムを使用した時よりも本システムを使用した時の方が選択肢 2 と 3 の合計の百分率が 19 ポイント上がっている．これは，間違い②に気づくことができた回答者が，従来システムよりも本システムを使った人の方が 19 ポイント増えたことを示している.

これらの結果より，従来システムを使用するよりも本システムを使用した方が，間違いに気づくことができる人数が増加したことがわかる．これは，従来システムに比べて本システムを使用した方が，プログラム中の誤っている場所を特定できるようになったためだと考えられる.

7.3　　アンケート内容，アンケート結果および考察（質問 1～12）

質問 1～12 のアンケート内容，アンケート結果および考察について以下に記述する．

7.3.1　質問 1 と 2 の内容，結果および考察

質問 1 と 2 の内容を図 32 に，その結果を表 9 に示す．

質問 1

変数や配列の値の移り変わりを把握する際に，
「本システムの表示：実行前と実行後の 2 状態を表示」は
「従来システムの表示：実行後の状態のみ表示」に比べて役立ちましたか．

1. 役に立たなかった
2. どちらかといえば，役に立たなかった
3. ふつう
4. どちらかといえば，役に立った
5. 役に立った

質問 2

質問 1 で「1（役に立たなかった），2（どちらかといば，役に立たなかった）」を選択した方に質問です．
どのような点で変数や配列の値の移り変わりを把握するのに役立たないと感じましたか．

図 32　質問 1 と 2 の内容

表 9　質問 1 に対する回答

	質問1					
選択肢	1	2	3	4	5	合計
票数	1	1	27	18	17	64
百分率	2%	2%	42%	28%	27%	100%

質問 1 のアンケート結果（表 9）より，「4（どちらかといえば役に立った），5（役に立った）」の回答が 55%と過半数に達した．これより，本

システムの機能である実行前と実行後の 2 つの状態が表示されることは，変数や配列の値の移り変わりを把握することに有効であると考えられる．

質問 2 の回答として「結局，どう直せばいいかわからなかった」があったが，意図した問に対する回答ではないため，考慮する必要はないと考えられる．

7.3.2　質問 3 と 4 の内容，結果および考察

質問 3 と 4 の内容を図 33 に，その結果を表 10 に示す．

質問 3

変数や配列の値の移り変わりを把握する際に，
「本システムの表示：実行前後の変化部分に色がつく表示」は
「従来システムの表示：実行後の状態のみ表示」に比べて役立ちましたか．

1. 役に立たなかった
2. どちらかといえば，役に立たなかった
3. ふつう
4. どちらかといえば，役に立った
5. 役に立った

質問 4

質問 3 で「1（役に立たなかった），2（どちらかといえば，役に立たなかった）」を選択した方に質問です．
どのような点で変数や配列の値の移り変わりを把握するのに役立たないと感じましたか．

図 33　質問 3 と 4 の内容

表 10　質問 3 に対する回答

	質問3					
選択肢	1	2	3	4	5	合計
票数	1	0	29	16	18	64
百分率	2%	0%	45%	25%	28%	100%

質問 3 のアンケート結果（表 10）より，「4（どちらかといえば役に立った），5（役に立った）」の回答が 53%と過半数に達した．これより，本システムの機能である，実行前後の変化部分に色をつけて示す表示は，変数や配列の値の移り変わりを把握することに有効であると考えられる．

質問 4 に対する回答はなかった．

7.3.3　質問 5 と 6 の内容，結果および考察

質問 5 と 6 の内容を図 34 に，その結果を表 11 に示す．

質問 5

今後，自分でプログラムを作成する際，どちらの可視化システムを使いたいと思いますか．

1. どちらも使いたくない
2. 従来システムを使いたい
3. どちらかといえば，従来システムを使いたい
4. どちらかといえば，本システムを使いたい
5. 本システムを使いたい

質問 6

質問 5 で「1（どちらも使いたくない），2（従来システムを使いたい），3（どちらかといえば，従来システムを使いたい）」を選択した方に質問です．

どのような点でそう感じましたか．

図 34　質問 5 と 6 の内容

表 11　質問 5 に対する回答

	質問5					
選択肢	1	2	3	4	5	合計
票数	2	4	6	36	16	64
百分率	3%	6%	9%	56%	25%	100%

質問 5 のアンケート結果（表 11）より，「4（どちらかといえば，本システムを使いたい），5（本システムを使いたい）」が 81%と過半数に達した．これより，学習者が自身でプログラムを作成する際，可視化システムを使用しない方法と従来システムを使用する方法よりも，本システムを用いて取り組む方法を行いたいという学習者が多いことがわかる．

質問 6 に対する回答として，「変化部分を示す赤色が見にくかった」という意見があった．本システムでは配列表示を，赤色の背景の上に黒色の文字で表示しており，文字が読み取りづらいことが考えられる．今後，配列表示の配色を検討する必要がある．また，「自分の中でコードを見て判断できるので，このシステムを使うと手間がかかるから」という回答があった．この回答者はソースコードを見て判断することができているため，本研究で対象とする初学者ではなく，初学者に対して本システムは有効であると考えれる．

7.3.4　質問 7 と 8 の内容，結果および考察

質問 7 と 8 の内容を図 35 に，その結果を表 12 に示す．

質問 7

操作性に関して，本システムは従来システムと比べてどうでしたか．

1. 使いにくかった
2. どちらかといえば，使いにくかった
3. 変わらない
4. どちらかといえば，使いやすかった
5. 使いやすかった

質問 8

質問 7 で「3（変わらない）」以外を選択した方に質問です．
どのような点でそう感じましたか．

図 35　質問 7 と 8 の内容

表 12　質問 7 に対する回答

	質問7					
選択肢	1	2	3	4	5	合計
票数	0	1	40	12	11	64
百分率	0%	2%	63%	19%	17%	100%

質問 7 のアンケート結果（表 12）より，操作性に関して，「3（変わらない），4（どちらかといえば，使いやすかった），5（使いやすかった）」が 98%と過半数に達し，ほとんどの使用者が変わらないまたはそれ以上を選択した．これより，操作性に関して，本システムは従来システムに比べて低下していないことが考えられる．質問 8 の回答には「実行前と実行後の表示があることで，可視化させる位置を変えた際に前の値が残っており急激な変化が抑えられるから」という回答があった．これは，本システムは命令の実行前後の状態を表示しているため，表示している実行状態から 1 つ前の状態や 1 つ後の状態を確認する際に，戻るボタンまたは進むボタンを押す必要がなくなったためだと考えられる．

7.3.5　質問 9 と 10 の内容，結果および考察

質問 9 と 10 の内容を図 36 に，その結果を表 13 に示す．

質問 9

見やすさに関して，本システムは従来システムと比べてどうでしたか．

1. 見にくかった
2. どちらかといえば，見にくかった
3. 変わらない
4. どちらかといえば，見やすかった
5. 見やすかった

質問 10

質問 9 で「3（変わらない）」以外を選択した方に質問です．
どのような点でそう感じましたか．

図 36　質問 9 と 10 の内容

表 13　質問 9 に対する回答

	質問9					
選択肢	1	2	3	4	5	合計
票数	1	4	33	14	12	64
百分率	2%	6%	52%	22%	19%	100%

質問 9 のアンケート結果（表 13）より，「3（変わらない），4（どちらかといえば，見やすかった），5（見やすかった）」が 92%と過半数に達し，ほとんどの使用者が，変わらないまたはそれ以上を選択した．これより，本システムの可視化表示は従来の可視化システムよりも情報量が増えているが，見やすさは低下していないことが考えられる．質問 10 の回答には「実行前と実行後の両方が表示されていること．処理の流れがわかった．変化した部分に色がついたことにより違いが一目でわかった.」という回答があった．これは，命令の実行前後の表示があり，変化部分が強調表示されていることにより，プログラム全体の中で注目すべきところがわかりやすくなったためだと考えられる．

7.3.6　質問 11 と 12 の内容，結果および考察

質問 11 と 12 の内容を図 37 に示す．

質問 11
従来システムに関して，その他意見・感想等ありましたらお聞かせ下さい．

質問 12
本システムに関して，その他意見・感想等ありましたらお聞かせ下さい．

図 37　質問 11 と 12 の内容

質問 11 と 12 のアンケート結果より，「本システムのほうが前回のシステムよりもわかりやすかった．比較するというのが大きなポイントだった.」という回答が得られた．また，その一方で「本システムの方が難しく感じた．どこが間違っているのかは分かったが，どのように修正すべきかが分からなかった」という回答が得られた．後者の意見より，プログラム中の誤っている場所を特定できなかった学習者が，誤っている位置を特定できるようになったと考えられる．また,「どのように修正すべきか」という支援は本研究の範囲外であり，今回対象としている支援には有効であったと考えられる．

第8章 まとめ

プログラミング初学者が，自身で作成したプログラムの変数の値や配列の値の移り変わりを把握する支援をすることを目的として，以下の機能を備えた可視化システムを提案した．

- 可視化の自動生成
- 命令の実行前後の状態を表示
- 命令の実行前後で変化している部分を強調表示

そして実験結果より，本システムが変数の値や配列の値の移り変わりを把握する支援に有効であることを確認した．また，従来システムよりも本システムを使用したいという学習者が 80%に達するという結果になった．

参考文献

[1] 都倉信樹，情報処理専門教育について：情報処理教育における実験・演習，情報処理学会研究報告，vol. 32，no. 10，pp. 11011108，1991

[2] 松井大成，伊藤恵：プログラミング行動中の思考分析に基づいた初学者支援システムの提案，日本ソフトウェア科学会第 35 回大会 (2018 年度) 講演論文集

[3] 吉村巧朗，亀井靖高，上野秀剛，門田暁人，松本健一：ブレークポイント使用履歴に基づくデバッグ行動の分析，電子情報通信学会技術研究報告：信学技報，vol. 109，no. 307，pp. 8590，2009

[4] Sanja Maravić Čisar，Robert Pinter and Petar Čisar：Effectiveness of Program Visualization in Learning Java: a Case Study with Jeliot 3，International Journal of Computers，Communications & Control，vol. 6，no. 4，pp. 668–680，2011

[5] Erkki Kaila，Teemu Rajala，Mikko-Jussi Laakso and Tapio Salakoski：Effects of Course-Long Use of a Program Visualization Tool，Proceedings of the Twelfth Australasian Computing Education Conference，vol. 103，pp. 97106，2010

[6] Oscar Karnalim and Mewati Ayub：The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor, Communication & Information Technology Journal, vol. 11，no. 2，pp. 6776，2017

[7] Sassi Bentrad and Djamel Meslati：Visual Programming and Program Visualization — Toward an Ideal Visual Software Engineering System —，ACEEE International，vol. 1，no. 3，pp. 56‐62，2011

[8] Christine Alvarado，Briana Morrison，Barbara Ericson，Mark Guzdial，Brad Miller and David L. Ranum：Performance and use evaluation of an electronic book for introductory Python programming，Technical Report GT-IC-12-02，Georgia Institute of Technology，2012

[9] 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—，情報処理学会研究報告，ソフトウェア工学研究会報告，vol. 2014，no. 5，pp. 18，2014

[10]中原進，紫合治：オブジェクト指向プログラムの動作の可視化，情報処理学会研究報告，ソフトウェア工学研究会報告，vol. 2010，no. 9，pp. 18，2010
[11]中村亮太，西村知博，松浦敏雄：プログラミング入門教育用学習環境 PEN，情報処理学会研究報告，pp. 6571，2017
[12]長慎也，甲斐宗徳，川合晶，日野孝昭，前島真一，筧捷彦：プログラミング環境 Nigari−初学者が Java を習うまでの案内役，情報処理学会論文誌：プログラミング，vol. 45，pp. 2546，2004
[13]Andrés Moreno，Niko Myller，Erkki Sutinen and Mordechai Ben-Ari：Visualizing Programs with Jeliot 3，Proceedings of the Working Conference on Advanced Visual Interfaces，pp. 373–376，2004
[14]Ronit Ben-Bassat Levya，Mordechai Ben-Ari and Pekka A.Uronen：The Jeliot 2000 program animation system，Computers & Education，vol. 40，pp. 115，2003
[15]Christopher D，Hundhausen，Sarah A. Dougals and John T. Stasko : A Meta-Study of AlgorithmVisualization Effectiveness ， Journal of Visual Languages & Computing ， vol. 13，pp. 259290，2002
[16]Andrés Moreno and Niko Myller：PRODUCING AN EDUCATIONALLY EFFECTIVE AND USABLE TOOL FOR LEARNING, THE CASE OF JELIOT FAMILY，Proceedings of International Conference on Networked e-learning for European Universities，vol. 15，no. 5，pp. 795825，2003
[17]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. 579584，2013
[18]Daiki Ihara，Satoru Kogure，Yasuhiro Noguchi，Koichi Yamashita，Tatsuhiro Konishi and Yukihiro Itoh：Algorithm Learning by Comparing Visualized Behavior of Programs，pp. 385390，2017
[19]伊冨昌幸，北英彦，高瀬治彦，林照峯：コーディング状況に応じたアドバイスを可能にするプログラミング演習システムに関する研究，コンピュータ利用教育協議会，2010PC カンファレンス，2010
[20]小島佑介，高橋功欣，北英彦：プログラミング演習における効率のよい指導のためのエラー早期指摘，コンピュータ利用教育協議会，2011PC カンファレンス，2011

[21]小川正，西口大亮，北英彦：プログラミング演習における iPad などの携帯デバイスの利用による指導の円滑化，コンピュータ利用教育協議会，2011PC カンファレンス，2011
[22]三重大学ウェブシラバス　プログラミング演習Ⅰ：https://syllabus.mie-u.ac.jp/?action=display&id=19567（2019 年 1 月参照）
[23]三重大学ウェブシラバス　プログラミング演習Ⅱ：https://syllabus.mie-u.ac.jp/?action=display&id=19574（2019 年 1 月参照）

実績一覧

国際会議発表

[a] Fukuaki Ito，Hidehiko Kita : Improvement of a Programming Aptitude Test，Proceedings of The 7th International Symposium for Sustainability by Engineering at Mie University (IS^{2}EMU 2017)，pp. 4142，2017

[b] Fukuaki Ito，Haruhiko Takase，Hidehiko Kita，Hiroharu Kawanaka : Program-Flow Visualization for Novice Programmers，Proceedings of The 8th International Symposium for Sustainability by Engineering at Mie University (IS^{2}EMU 2018)，pp. 78，2018

全国大会・支部大会

[c] 伊藤福晃，北英彦：プログラミング素養診断テストの改良，2017PC カンファレンス論文集，pp. 3740，2017

[d] 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. 2932，2018

謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学大学院工学研究科の北英彦准教授に深く感謝いたします．

また，貴重な時間をさいて本論文を査読して頂いた，三重大学大学院工学研究科の高瀬治彦教授，川中普晴准教授に深く感謝致します．最後に，日頃熱心に討論していただいた計算機工学研究室の皆様方にお礼申し上げます．