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 正典[1,a)]　永井 保夫[1,b)]

概要：アルゴリズムをプログラムするためには，その処理がどのように行われていくのかを理解しておく必要がある．また，配列 (ランダムアクセスコンテナ) における要素交換のように，データ構造ごとに効率的な基本操作があることを知るのも重要である．しかしながら，初心者のうちは反復処理の中に分岐処理が入っているアルゴリズムは理解するのが難しい場合もある．また初心者はアルゴリズムを理解したつもりでも誤解していたり，間違った内容でプログラムしてしまうこともある．本研究では，初等的なアルゴリズムの理解を助けるための視覚化と，その上でプログラミング演習として，アルゴリズムの動作する様子をあらかじめ動的視覚化によって提示し，プログラミングを促す演習システムを提示する方法を提案する．学習者は，自分でプログラムした内容と正解版の両方を動作する様子を動的表示によって確認・比較することもでき，自らの誤りに気づきやすくなる．これによって，自習における学習効率の向上も期待できる．

キーワード：プログラミング教育，視覚化，アルゴリズム，Eclipse，Java

Elementary algorithm visualization system for programming learning for beginners

Abstract: In order to program the algorithm, it is necessary to understand how the process is carried out. It is also important to know that there is an efficient basic operation for each data structure like element exchange in an array (random access container).However, among beginners, it is sometimes difficult to understand algorithms that include branching processing in iterative processing.Also, even if they think they understand the algorithm, they may code with incorrect content. In this research, we propose a method of deepening understanding of algorithms and presenting exercise systems to improve coding by presenting the behavior of algorithms in advance by dynamic visualization as exercises to be coded on them. Learners can also check and compare how they act both on their own coded content and on their correct versions by dynamic display, making themselves more aware of coding mistakes themselves. With this system, improvement of learning efficiency in self-study can be expected.

Keywords: programming education, visualization, algorithm, Eclipse, Java

1. はじめに

アルゴリズムをプログラムするためには，その処理がどのように行われていくのか，そのためにはどのような変数を用意する必要があるのか，理解しておかなければならない．しかし，プログラミング初心者を対象とした場合には，これらの要素をどの様に学習者に示して理解してもらうか．また，その上で学習者がプログラムを書けるように補助するかが困難な点である．例えば，特定のプログラミング言語や入門者用の疑似言語で書かれたソースコードでアルゴリズムの動きや仕組みを示されても，多くのプログラミング初学者は理解できない場合が多いと考えられる．さらにアルゴリズムの動きや仕組みを理解しても，慣れないプログラム言語を使用してアルゴリズムを記述することは難しいと言える．本来なら，アルゴリズムを効率良く理解するには学習者自らプログラムし，実行してアルゴリズムの動

[1] 東京情報大学
Uniersity of Information Sciences, 4–1 Onaridai Wakaba-ku, Chiba, Chiba, Japan
[a)] ohshiro@rsch.tuis.ac.jp
[b)] nagai@rsch.tuis.ac.jp

作を確認するのが良いと言えるが，プログラミング初学者のためにアルゴリズムを自分で記述・作成・実行するという試行を行うことができず，そのためにアルゴリズムの理解が難しくなってしまう．また逆に，アルゴリズムの基本 (変数，逐次処理，選択処理，繰り返し処理) を理解できないために，プログラミング言語の学習も非効率になる，という悪循環がこの種の困難の根底にあると考えられる．

1.1 アルゴリズムの視覚化と関連研究

アルゴリズムの動きや仕組みをプログラミング初心者に理解させる手段として，視覚化があげられる．アルゴリズムの視覚化を行う研究には，以下の様な先行研究がある．

Bell ら [1], [2] による研究では，低年齢層にもアルゴリズムをはじめとするコンピューターサイエンスを理解できる教育法として，コンピュータを用いず，かわりに実物教材を使用した方法論が提案されている．しかし，コンピュータを使用しないことが前提であるので，プログラミング言語を用いてアルゴリズムを記述する，という部分は省かれている．

今泉ら [3] は，プログラム教育において，制御フローに着目し，プログラム構造を可視化し，プログラムの動作過程とプログラム構造を対応させて示すツールを開発している．

Java Visualizer[4] では，Java プログラムの実行状況を可視化表示している．初学者はプログラム実行中の変数とその参照関係やプログラムの構造情報を表示されることでプログラムの理解が容易になる．

武田ら [5] は，プログラムのデバッグや，初心者のプログラム学習を支援することを目的として，汎用的なアルゴリズムの可視化システムを開発している．アルゴリズムを可視化するために，プログラム構造のフローチャートによる自動表示と各変数値の変化を可視化することを基本として，変数の動的な様子の表示，関数の呼び出し関係の表示，フローチャートの簡易表示を提供している．

JIVE[6] は，プログラム開発時に，各オブジェクトの変数の値，参照先，実行時のメソッドなどの情報を図により表示する Eclipse プラグインとして開発された．これにより，プログラムの実行時の状態を可視化することを可能としている．

Jeliot 3[7], [8] は，プログラミングの初学者の教育支援を目的としたプログラムアニメーションシステムである．Java プログラムの実行を可視化することで，様々な概念をアニメーションとして図示することで，初学者がデータ構造やプログラムのふるまいの理解を支援している．

松崎 [9] らは，初学者のアルゴリズム理解支援を目的としたプログラムの動作過程の可視化ツールを提案している．ここでは，任意のプログラムの実行時に生成されるオブジェクトを，学習者が持つデータ構造のイメージに近い形態で可視化するために，オブジェクトの配置をレイアウトできるようにしている．

VisuAlgo[10] では，プログラミング学習に必要となるアルゴリズムとデータ構造を可視化して提示している．プログラミング学習では不可欠となる概念である，アルゴリズムとデータ構造を初学者に可視化することでわかりやすく学習できる機能を提供している．特に，代表的なアルゴリズムである，ソートアルゴリズムからグラフアルゴリズムを取り上げている．

1.2 本研究のねらい

筆者らは，学習用にはどのようなプログラムの視覚化が適しているかを検討し，視覚化機能を持つプログラミング学習支援システムを Eclipse プラグインとして研究・開発してきた [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]．本システムは Java のソースプログラム上の構造を視覚化する静的視覚化と，プログラムが実行される様子を視覚化する動的視覚化の機能を持つ．また，プログラムを視覚化するだけでなく，学習者のソースプログラムを書く能力を向上させることを目的として，静的に視覚化された文法的構造 (クラス，コンストラクタ，メソッド) とインストラクションを表示して，対応するソースコードを構造毎に書かせる演習方法を提案した [20], [21]．

本研究では，この視覚化システムの動的視覚化機能を用い，アルゴリズムの動きと仕組みをシームレス性 [25] をともなったアニメーションによって提示してアルゴリズムを初学者に理解しやすいように配慮すると同時に，理解したアルゴリズムをプログラミング言語によってプログラムさせることによって，学習者自身がアルゴリズムをプログラミング言語で記述・作成・実行するという試行を行える様にするシステムを提案する．また，アルゴリズムの理解を補助するために新たな視覚化機能を追加する．

2. アルゴリズムの視覚化機能

著者らは，本研究の目的のために，これまで開発してきた視覚化システムに新たな視覚化機能を追加した．

- 処理描写に注力しメインクラス・main メソッドを非表示にする
- 条件判定の内容を強調して表示
- 添え字によって指定されている要素の強調
- 変数に値を累積する処理のための表示
- 累積変数の値の履歴
- 変数の履歴表
- 配列要素の交換操作の強調

具体的な例とともにこれらの表示を解説する．なお，以前の論文で述べたとおり，動的視覚化の表示中は，エディタウィンドウで実行の該当箇所がハイライトされる．また，

1 行ずつ実行したり，実行を一時停止させることができる．対象となるアルゴリズムは，プログラミング初心者を対象とした，変数，配列と逐次処理・選択処理・繰り返し処理を組み合わせたもので，表示用以外はサブルーチンを呼ばない初歩的な処理を前提としている．

2.1 配列要素の合計値を求める例

図 1 は，典型的な配列要素の合計値を求めるプログラムである．合計値は式

```
s = s + a[ i ];
```

によって，s に累積する形で求められる．このプログラムを実行すると，JDT を利用した解析により，単一クラスのプログラムで，他のクラスの利用は，static メンバだけに限られる．このことより，このプログラムは手続き的な処理が主な目的で，クラス間関係やオブジェクト間関係はほぼ取り扱われていないと判断し，メインクラスおよび main メソッドの視覚化は行われないこととなる．このプログラムを実行すると，まず初期値が null の配列変数 a が表示され，続いて未初期化の int 型要素 8 個からなる配列本体が表示され，その配列本体への参照値が変数 a に格納される．次いで，int 型変数 s が宣言される．ここまで実行されたときの動的表示は図 2 のようになる．

for 文の実行に入ると，配列の添え字として使用される目的で宣言された for 文専用の int 型変数 i が表示される．そして，繰り返しの条件判定式を実行するが，その際には図 3 のような条件判定の内容を表示する．

for 文の内部の処理に入ると，変数 s に配列 a の i 番目の要素の値が足し込まれる式を表す専用の表示が行われる (図 4)．まず，配列要素 a[i](この場合は a[0]) がハイライトされる．次に=演算子と+演算子が表示され，その上に変数 s と要素 a[0] が分身するように移動する．元の変数 s はこの時点で薄く表示される．移動した変数 s と要素 a[0] の上に，加算と代入を表す矢印のガイドラインが表示される．

次に，移動してきた変数 s と要素 a[0] に格納されている値が，矢印のガイドラインにそってそれぞれ分身するように移動し，接合点で合計される (図 5)．このとき，移動してきた変数 s の表示は薄くなり，かわりに元の変数 s の表示が通常表示に戻る．そして，元の変数 s に向かって引かれた矢印に沿って合計値が移動し，変数 s に入る (図 6)．

変数 i が 2 の時の繰り返しが終わった時点 (i が 3 にインクリメントされる直前) の表示が，図 7 で，値を累積させている変数 s をクリックすると，それまで累積されてきた値の履歴が表示される．また，繰り返し文が実行されている間，動的視覚化ビューの下部より繰り返しの条件判定結果や繰り返し文内で使用されている変数の履歴を引き出すことができる (図 8)．

```
class Sum {
  public static void main( String [ ] args ) {
    int a[ ] = { 21, 7, 11, 3, 1, 19, 8, 4 };
    int s = 0;
    for( int i = 0; i < a.length; i++ ) {
      s = s + a[ i ];
    }
    System.out.println( s );
  }
}
```

図 1 配列要素の合計値を求める例.
Fig. 1 An example of calculating the total value of array elements.

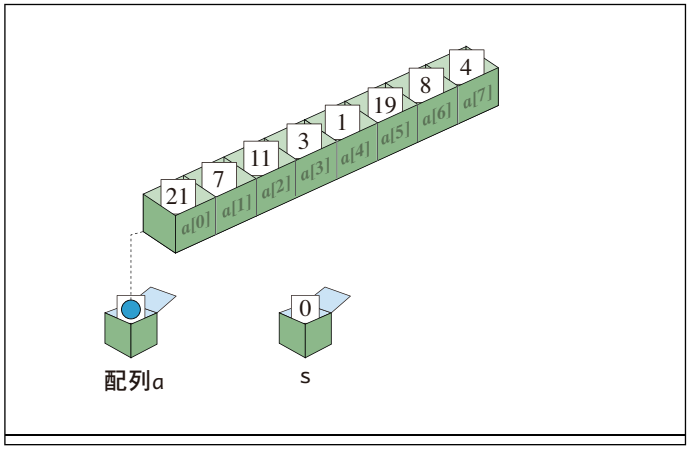


図 2 変数 s の宣言まで実行した状態.
Fig. 2 State executed until declaration of variable s.

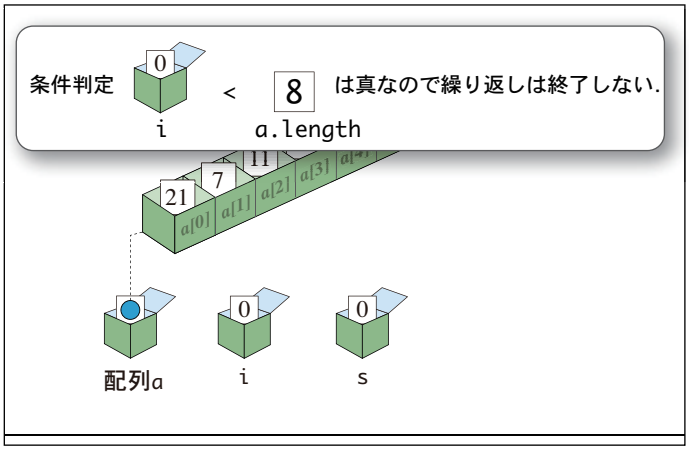


図 3 繰り返しの条件判定式を実行した際の表示.
Fig. 3 Display when execution of condition judgment of iteration statement is executed.

2.2 バブルソートの例

配列のようなランダムアクセスコンテナでは，要素値の移動をともなう処理において，効率が良い要素値の交換処

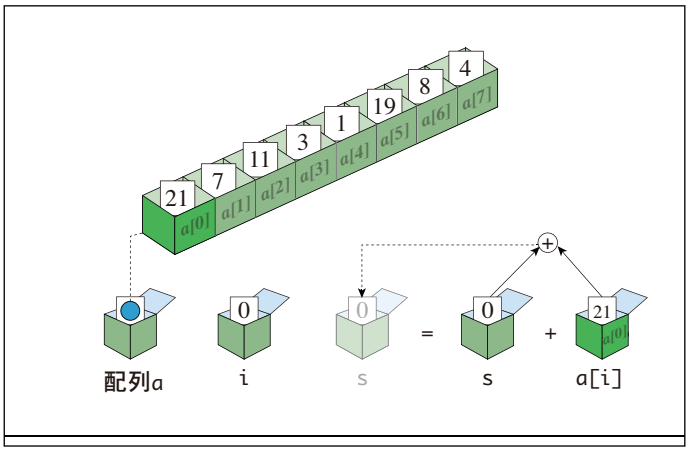


図 4　最初に for 文の中を実行するときの表示.

Fig. 4　Display when executing the content of for statement for the first time.

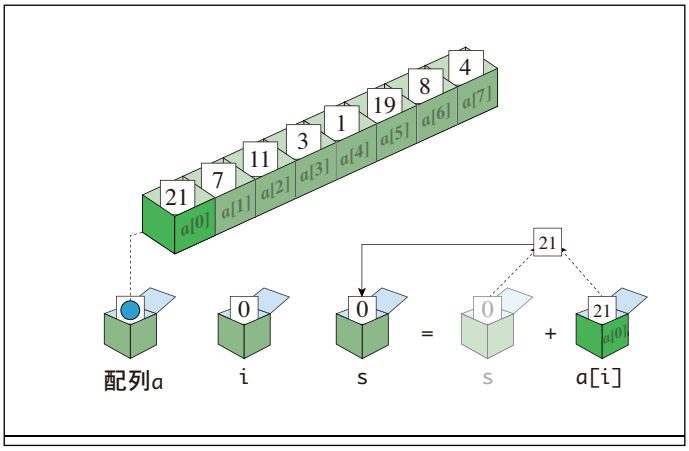


図 5　変数 s と要素 a[i] の値を合計するアニメーション.

Fig. 5　An animation that sums the values of variable s and element a [i].

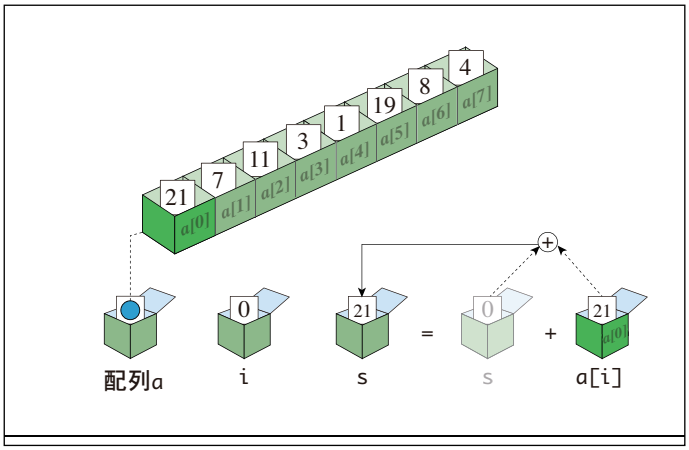


図 6　変数 s に合計値が代入されるアニメーション.

Fig. 6　An animation in which the total value is assigned to the variable s.

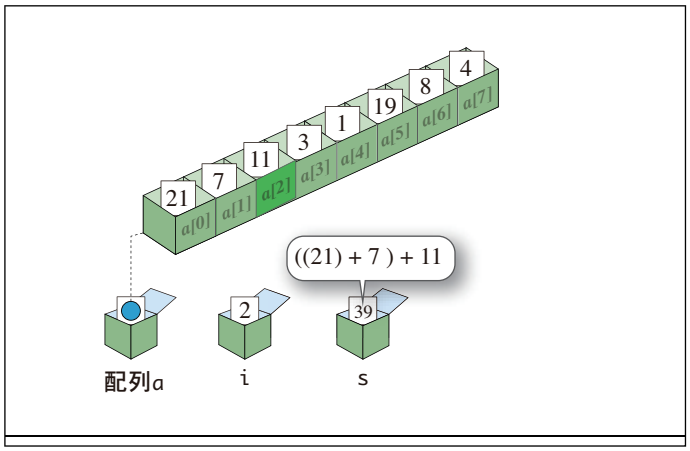
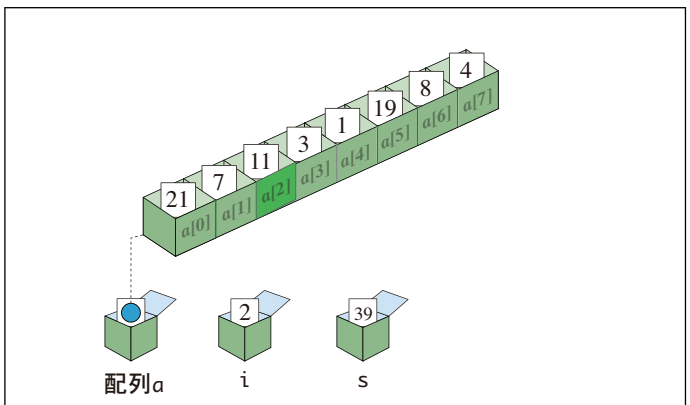


図 7　変数に累積された値の履歴表示.

Fig. 7　History display of values accumulated in variables.



i < a.length	i	s	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
真	0	21	21	7	11	3	1	19	8	4
真	1	28	21	7	11	3	1	19	8	4
真	2	39	21	7	11	3	1	19	8	4

図 8　繰り返し文で使用された全変数の履歴表示.

Fig. 8　Display history of all variables used in iteration statements.

理を基本操作として用いることが重要である．そのため，JDT でコードを解析し，要素値の交換処理が行われるときには，その旨を強調した表示を行う機能を用意した．

図 9 は，典型的なバブルソートを行うプログラムである．バブルソートはソートアルゴリズムとしては遅いが，それでも配列を操作する際の基本にそって，要素の交換を基本操作として用いている．最初に内側の for 文を実行する際，a[0] と a[1] の値が比較された結果，両者の値を交換する処理が実行される．最初に a[0] の値を一時変数 t に保存した時点の表示が図 10 である．続いて，a[0] に a[1] の値を代入した状態が図 11 である．最後に，一時変数 t の値を a[1] に代入し，a[0] と a[1] の値を交換し終えた状態が図 12 である．この処理の間，t,a[0],a[1] は代入操作に沿った矢印で結ばれ，各値はその矢印の上をアニメーションに

```
class BubbleSort {
  public static void main( String [ ] args ) {
    int a[ ] = { 21, 7, 11, 3, 1, 19, 8, 4 };

    for( int j = a.length-1; j > 0; j-- ) {
      for( int i = 0; i < j; i++ ) {
        int t; /* 要素交換作業用変数t */
        if( a[ i ] > a[ i + 1 ] ) {
          t = a[ i ]; a[ i ] = a[ i + 1 ];
          a[ i + 1 ] = t;
        }
      }
    }

    for( int i = 0; i < a.length; i++ ) {
      System.out.println( a[ i ] );
    }
  }
}
```

図 9　バブルソートの例.
Fig. 9　A typical code for bubble sort.

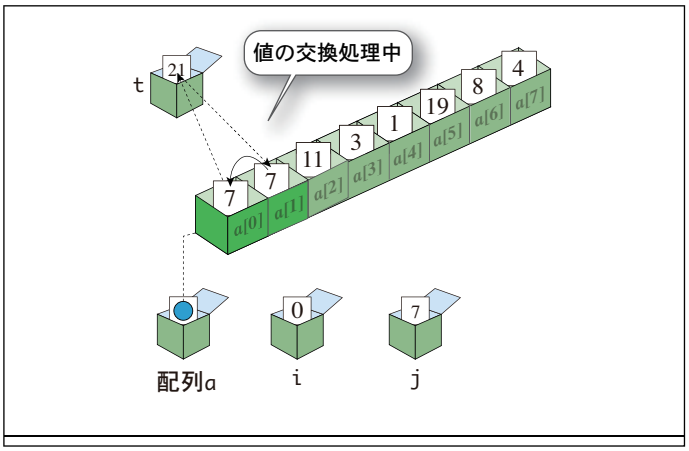


図 11　要素値交換の 2 番目の代入文実行後の状態.
Fig. 11　State after execution of the second assignment statement of element value exchange.

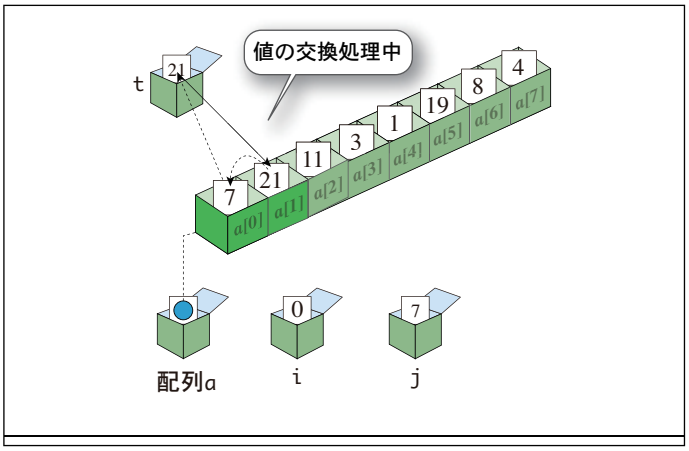


図 12　要素値交換の 3 番目の代入文実行後の状態.
Fig. 12　State after execution of the third assignment statement of element value exchange.

図 10　要素値交換の最初の代入文実行後の状態.
Fig. 10　State after execution of the first assignment statement of element value exchange.

よって移動する．また，要素の交換処理中であることを示す表示が常に表示される．このように，要素値の交換のために専用の表示を用意することで，学習者が配列操作の多くのアルゴリズムが，要素値の交換を基本操作としていることを理解する手助けになると考えられる．

3. 演習システムの実際

「int 型要素を 5 個持つ配列 a に含まれる偶数の数と奇数の数を調べて表示する」アルゴリズムを例として，本システムの動作と演習方法を説明する．システムは，内部にこのアルゴリズムを実装したコードを保持している．図 13 にこの Java コードを示す．コードは基本的に 1 行に 1 つの式・宣言を書くスタイルに統一しておく．学習者には，文章によるインストラクションと動的視覚化のアニメーションを見せた上で，このコードと同等のものを完成させるように段階的に指示する．なお，学習者にクラス定義や main メソッドの枠組みは書かせない．つまり，現状では学習者に書かせるアルゴリズムは main メソッドの内部だけとなる．そのため，再帰処理を含むアルゴリズムは対象外となる．

図 14 に本システムの基本的な画面構成を示す．上部には例題となるアルゴリズムの概要とインストラクションが表示される．左側にプログラムの記入欄が配置される．記入欄は正解コードに基づいて，変数宣言部分，構造文，連続する式文の各エリアに部分分割して表示される．右側に

は正解コードの動的視覚化を表示するエリアがあり，その上には実行ボタンが配置されている．

3.1 正しいアルゴリズムの実行

正解コードを実行すると，動的視覚化エリアで，正しいアルゴリズムの動きが表示される．図 15 は，正しいコードの 1 行目を実行し終わった状態である．正しいコードをすべて実行し終わった状態を図 16 に示す．このように，学習者は正しいコードの実行を動的視覚化によって何度でも確認することが出来る．

3.2 プログラミングと実行比較

つづいて，学習者はプログラミングと実行比較の作業に入る．図 17 は，インストラクションに従い，1 行目の配列生成・要素初期化の処理を記述した後，コードを実行した例である．学習者が記入したコードと正しいコードが同時に動的視覚化によって表示され，その違いを確認することができる．このように，学習者は，アルゴリズムを宣言・式文・構造文単位で記入するようにインストラクションで指示され，記入したコードが実行可能であれば，途中でも実行させて正しいコードと比較することが可能である．

このプログラムにおける間違いの例として，6 行目で偶数・奇数を判別するために剰余演算子%を使うべきところを，間違って除算演算子/を使ってしまった場合の実行例をあげる．当該の学習者は，この部分がこのアルゴリズムの要所であることを理解していない，%演算子と/演算子の違いをよく理解していない，単なる書き間違い，などの理由で間違えたと考えられる．学習者がすべて入力し終わり，自分のプログラムと正しいプログラムを比較実行した場合，最初に違いに気がつく機会は，添字変数 i の値が 1 のとき，6 行目の if 文を実行したときである (図 18)．2 で割った余りを計算したつもりが，結果が本来はあり得ない値である 3 となるため，単なる演算子の書き間違いの場合は，この時点で気づく可能性がある．しかし，そうでない場合はさらに比較実行を続けることになると思われる (なお，比較実行時における正しいプログラムの視覚化欄では，答えを学習者に教えてしまうことを防ぐために，具体的なプログラム内容を含む表示は抑制されている)．

このような学習者が次に間違いに気づく機会は，添字変数 i の値が 3 のとき，10 行目の代入文を実行したときである (図 19)．この段階で初めて変数 even と変数 odd の値が，正しいプログラムと異なってくる．このように変数の値が正しいプログラムと異なった場合は，変数に格納されている値が赤くハイライトされて，正しいプログラムと違いが出ているということを強調して学習者に伝える．学習者は，この段階で自分で入力したプログラムの見直しに入ることができる．その過程で，比較実行を繰り返すことも

```
01  int [ ] a = { 3, 7, 0, 8, 5 };
02  int even = 0;
03  int odd  = 0;
04
05  for( int i = 0; i < a.length; i++ ) {
06    if( a[i] % 2 == 0 ) {
07      even = even + 1;
08    }
09    else {
10      odd = odd + 1;
11    }
12  }
13
14  System.out.println( even );
15  System.out.println( odd );
```

図 13 プログラムの例.
Fig. 13 A sample code.

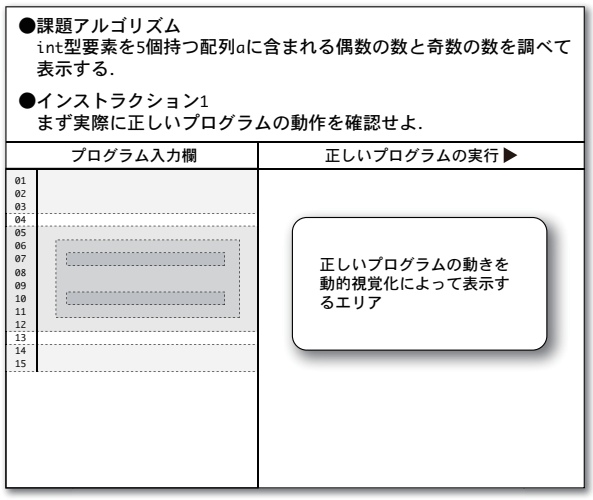


図 14 基本画面構成.
Fig. 14 Basic layout of the system.

でき，図 18 の段階で顕在化している間違いに気づくかもしれない．

4. おわりに

本研究では，初心者がアルゴリズムの動きを直感的に理解しやすいように動的視覚化を行うと同時に，プログラミング言語でアルゴリズムを記述し，正しいコードとの動作比較を行いながらアルゴリズムを理解・記述することができるようにサポートする学習システムの提案を行った．現状では，メソッド定義はサポートしておらず，前述したように再帰処理を含むアルゴリズムはサポートしていない．将来的に，メソッド定義も扱えるように改良したい．同様に，現段階では参照を使ったデータ構造である連結リストや木構造も専用の視覚化を用意していないので，将来的に

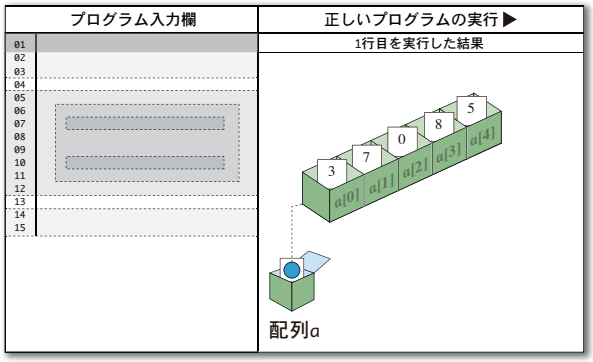


図 15　正しいコードの実行例.
Fig. 15　An example of execution of correct code.

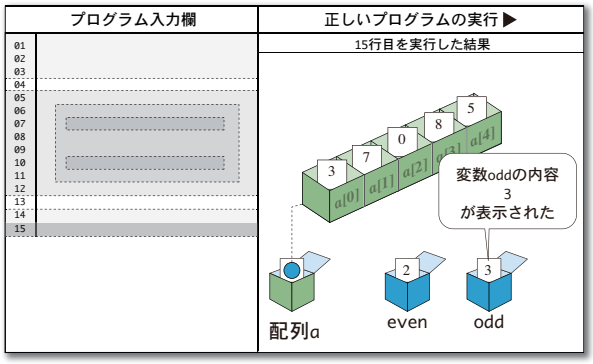


図 16　正しいコードの実行後.
Fig. 16　A last state of execution of correct code.

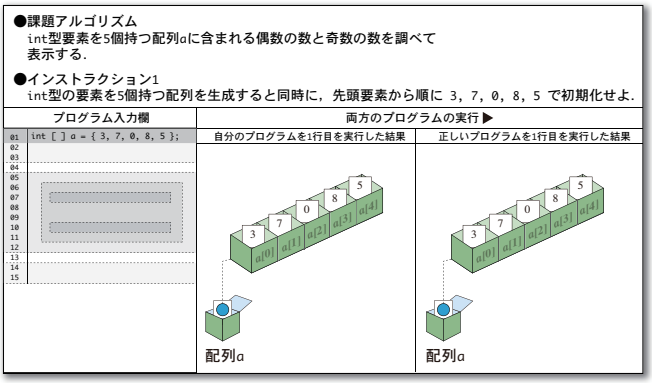


図 17　コードの記入と動作比較.
Fig. 17　Coding and comparison between student's code and the correct code.

それらに適した視覚化を用意したい．また，今回は初心者が，変数 (配列含む)・逐次処理・選択処理・繰り返し処理を組み合わせてアルゴリズムを理解し，自ら使える様になることを目的としたが，同じ目的のためのアルゴリズムの実装の仕方は本来一様ではない．今後は，学習者が習熟するに従って現れるであろう多様な考え方を許容するような工夫も必要になるものと思われる．今後，本システムの実装を行って実際に授業で使用して効果を検証したい．

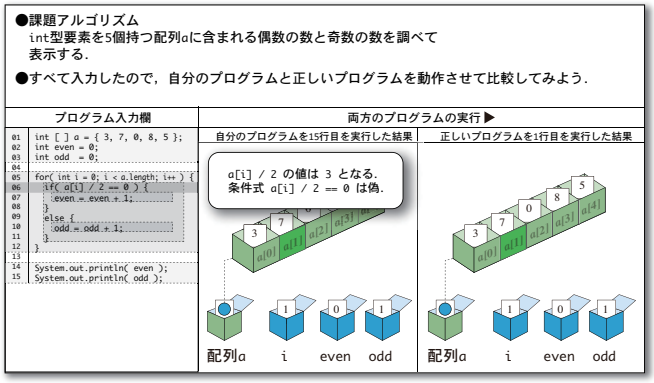


図 18　最初に違いが現れる状況.
Fig. 18　The situation where the difference first appears.

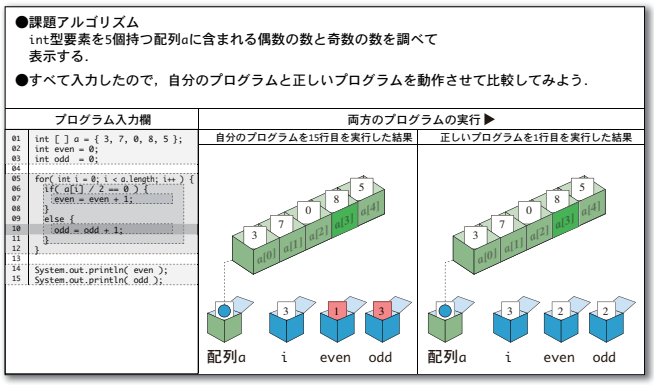


図 19　違いが現れる他の状況の例.
Fig. 19　An examples of other situations where differences appear.

参考文献

[1] CS Unplugged Computer Science without a computer, , available from ⟨http://csunplugged.org⟩

[2] Bell, T., Witten, I. H. and Fellows, M.: コンピュータを使わない情報教育　アンプラグドコンピュータサイエンス，イーテキスト研究所 (2007).

[3] 今泉俊幸，橋浦弘明，松浦佐江子，小宮誠一：ブロック構造の可視化によるプログラミング学習支援環境 azur “関数の動作の可視化”，情報処理学会研究報告ソフトウェア工学，Vol. 167, No. 11, pp. 1–7 (2010).

[4] Java Visualizer, available from ⟨https://cscircles.cemc.uwaterloo.ca/java$_{visualize}$⟩

[5] 武田寛昭，山下英生：C 言語プログラムに対するアルゴリズムの可視化システム，技術報告，広島工業大学 (2008).

[6] Lessa, D. and Jeffrey K. Czyz, a. B. J.: JIVE: A Pedagogic Tool for Visualizing the Execution of Java Programs (2011).

[7] Moreno, A., Myller, N., Sutinen, E. and Ben-Ari, M.: Visualizing Programs with Jeliot 3, *Procs. of AVI'04*, pp. 373–376 (2004).

[8] Moreno, A., Myller, N. and Bednarik, R.: Jeliot 3, an Extensible Tool for Program Visualization, *Procs. of 5th Annual Finnish/Baltic Sea Conference on Computer Science* (2005).

[9] 松崎　駿，酒井三四郎，松澤芳昭：プログラミング教育現場で利用可能な軽量オブジェクトレイアウト言語「DAST」の設計と評価，情報処理学会情報教育シンポジウム 2014

論文集 SSS2014，pp. 233–238 (2014).
[10] Halim, S.: VisuAlgo –Visualizing Data Structures and Algorithms Through Animation, *Olympiads in Informatics*, Vol. 9, pp. 243–245 (2015).
[11] 大城正典，永井保夫：オブジェクト指向プログラムの視覚化によるプログラミング教育システム，電子情報通信学会 2009 総合大会講演論文集 情報システム講演論文集 1，p. 212 (2009).
[12] 大城正典，永井保夫：オブジェクト指向プログラムの静的・動的側面を視覚化するプログラミング教育支援システム，情報処理学会第 71 回全国大会講演論文集 (4)，pp. 4–413, 4–414 (2009).
[13] 大城正典，永井保夫：オブジェクト指向プログラムの視覚化によるプログラミング教育システムの改良，電子情報通信学会 2010 総合大会講演論文集 情報システム講演論文集 1，p. 170 (2010).
[14] 大城正典，山川裕子，J., M. K.，松下孝太郎，布広永示：プログラミング学習支援システム CAPTAIN における学習状況把握機能の開発，日本教育工学会「教育実践を指向した学習支援システム／一般」研究会，pp. 81–84 (2010).
[15] 永井保夫，大城正典：モデリングを考慮したソフトウエア教育のためのオブジェクト指向プログラミング教材の検討，電子情報通信学会 2010 総合大会講演論文集 情報システム講演論文集 1，p. 169 (2010).
[16] 大城正典，永井保夫：初等プログラミングから設計レベルまでを対象としたオブジェクト指向教育のための支援システムの提案，情報処理学会 情報教育シンポジウム 2011 論文集 SSS2011，pp. 59–66 (2011).
[17] 大城正典，永井保夫：オブジェクト指向プログラム視覚化教育システムにおけるデザインパターンの視覚化サポート，電子情報通信学会 2011 総合大会講演論文集 情報システム講演論文集 1，p. 139 (2011).
[18] 大城正典，永井保夫：情報視覚化を活用したオブジェクト指向プログラミング教育支援システムの提案，信学技報教育工学， Vol. 110, No. 453, pp. 131–136 (2011).
[19] 大城正典，永井保夫：Eclipse を用いたオブジェクト指向プログラミング教育支援視覚化システムの設計と実装，信学技報教育工学， Vol. 112, No. 500, pp. 185–188 (2013).
[20] 大城正典，永井保夫：モニタ機能と可視化機能を持った構造指向による漸次的なプログラム作成学習システム，信学技報教育工学， Vol. 113, No. 482, pp. 31–34 (2014).
[21] 大城正典，永井保夫：段階的コーディングガイド機能およびモニタ機能を持つオブジェクト指向プログラミング教育のための視覚化支援システムの提案，信学技報教育工学， Vol. 114, No. 260, pp. 53–58 (2014).
[22] 大城正典，永井保夫：Eclipse 視覚化プラグインによる総合的なプログラミング教育支援システム，情報処理学会情報教育シンポジウム 2015 論文集 SSS2015，pp. 23–30 (2015).
[23] 大城正典，永井保夫：視覚化機能を持つ Eclipse プラグインによる段階的コーディングから動作検証までをサポートするオブジェクト指向プログラミング学習システム，信学技報教育工学， Vol. 115, No. 492, pp. 61–66 (2016).
[24] 大城正典，永井保夫：プログラミング初学者を対象としたオブジェクト指向プログラミング教育システムの提案-オブジェクト指向の基本概念の理解に基づいたプログラムの作成・実行支援機能を中心として-，情報処理学会情報教育シンポジウム 2016 論文集 SSS2016，pp. 114–121 (2016).
[25] 大城正典，永井保夫：シームレス性と文脈依存性を重視した視覚化機能を持つ Eclipse プラグインによるプログラミング学習支援システム，情報処理学会情報教育シンポジウム 2017 論文集 SSS2017，pp. 137–144 (2017).