Vol.2022-CE-164 No.8
2022/3/12

プログラミング初学者のための読み下し文提示システムの開発と評価

立花 昂己[1]　松澤 芳昭[1,a)]

概要：本研究では, 初学者のソースコードの読解を補助することを目的とする，ソースコード読み下し文提示システムの開発を行った．本システムは JavaScript のソースコードを入力すると，入力に対応する日本語の読み下し文を出力するものである．本システムの利用シナリオは，a) 読み下し文を読むことによるプログラムのデバッグでシステムを使用する，b) 読み下しを用いたサンプルコードの読解によりプログラム実行の流れを理解する，である．プログラミング初学者を対象にした授業において，本システムを導入し，使用者へのインタビュー，受講者へのアンケートの実施とシステムの操作ログの解析によるシステムの有効性評価を行った．その結果，1) システムの使用は任意であったにも関わらず，使用率は約 50%と一定の支持を得たこと，2) 利用者へのインタビュー及びアンケートにおいて，想定した 2 つの利用シナリオで使用している学生を確認できたこと，3) 変数，条件分岐，繰り返し，関数，引数付き関数，どの単元においてもシステムが用いられており，特に効果的であったのは，for 文の入れ子，関数，引数付き関数を学ぶ単元であったこと，が分かった．

キーワード：初学者，プログラミング教育，ソースコード，日本語，読み下し文

1. はじめに

プログラミングの学習を進めていく中で，プログラムを制作する際にソースコードが読めず，備えている機能が分からずに足踏みしてしまう学生は少なくない．我々はプログラミング学習はコードを覚えるためではなく，論理的思考力の育成をすることが重要であると考える．コードの文法の読解に時間をかけてしまうと，論理的思考力の育成という目的にまで届くこと無くプログラミング学習を終えてしまうことになる．

この問題の解決を試みるいくつかの提案がある．一つはふりがなプログラミングである [1]．ふりがなプログラミングはプログラミング言語で書かれたコードに対して日本語のふりがなをふることによって，コードを日本語で読めるようにする試みである．これにより，「各コードが何を意味していて，どう動くのかがわかる」ことが意図されている．二つ目は札場らの研究 [2] である．この研究では，人手による日本語の疑似コードをソースコードと一緒に提示してプログラムを学ぶ手法を提案している．特にプログラミング言語の経験が浅い場合，プログラムの読解の補助となることが示唆されている．

そこで本研究ではソースコードの読み下し文に着目した．ソースコード読み下し文とは，ソースコードを，読み下せる日本語の文章にしたものである．JavaScript のコードで例を挙げると，

var x = 1;

であれば

変数 x を宣言し，1 に書き換える

というような日本語になる．

読み下し文を導入することのメリットは，そのプログラムがどんな流れで動いているのかが分かりやすくなる点，プログラムが備えている機能がすぐ分かるようになる点が考えられる．これによりプログラミング初学者が陥りやすい，手段もしくはプログラムの動きの流れが分からず，プログラムが作れないという問題を無くすことが出来る可能性がある．

本研究では，プログラミング初学者に向けたソースコード読み下し文提示システム「Leia」を提案する．本研究の目的は，(1) 日本語の読み下し文は初学者のソースコードの読解を補助出来るのか，(2) ソースコードの読解を補助しやすい読み下し文はどのような形式なのか，という二つのリサーチクエスチョンに対して，提案システムを利用して回答することである．

[1] 青山学院大学 社会情報学部
School of Social Informatics, Aoyama Gakuin University
[a)] matsuzawa@si.aoyama.ac.jp

リスト 1 JavaScript ソース

```
1  a();
2  var x = 0;
3  var y = a();
4  var z = a() + random(10) + 10;

5  println(x);
6  function a() {
7    x = 10 + 10 − 10 ∗ 10 / 10;
8    return x;
9  }
```

リスト 2 読み下し文

```
1  関数aを呼び出す。
2  変数xを宣言し、0に書き換える。
3  変数yを宣言し、関数aを呼び出した結果に書き換える。
4  変数zを宣言し、関数aを呼び出した結果+0以上10未満の乱数
       +10に書き換える。
5  変数「x」の中の文字列を表示する。
6  以下を実行する関数aを定義する。{
7    変数xを10+10−10×10÷10に書き換える。
8    戻り値としてxを返す。
9  }
```

図 1 代入と式の読み下し例

2. 先行研究

日本語プログラミングに関連する研究やシステムの提案はこれまでにも多く進められてきた．

その例として，日本語プログラミング言語がある．「Mind」[3] は片桐により開発された日本語プログラミング言語である．「Mind」において，ソースコードは逆ポーランド記法で記述されており，この表記法はソースコードを自然言語として読む場合の日本語の「目的語+動詞」の語順に適合しているが全部がひらがなの単語を使用すると単語認識で誤作動を起こすというコードの自由度の面で問題があった．中川らの研究 [4] ではプログラミング言語の文法の範囲内で母語を自由に使用するという形の日本語プログラミング環境を構築している．その環境でソフトウェアの開発を進めて行った結果，約 5 年間に合計 20 万行以上のプログラムを作成し，それを管理者が毎年入れ替わる環境でも保守することに成功している．「なでしこ」[5] は一般的な事務作業を主な仕事とする人が扱いやすいように設計された日本語プログラミング言語である．利用者には学生や事務，コンピュータ系の職業に就いている人の割合が多く，プログラミングの学習に活用されていること，事務の自動化や効率化に貢献していることが確認されている．犬童 [6] は Prolog を用いて日本語の論理プログラミング化の実行方法を提案している．

プログラミング教育への活用を目的とする日本語プログラミング研究も提案されている．西田ら [7] は，日本語でプログラミングをする初学者用プログラミング学習環境「PEN」を開発し，実際に大学の授業で導入してシステムの評価を行っている．兼宗ら [8] は，学校教育用オブジェクト指向言語「ドリトル」を開発している．「ドリトル」は初中等教育向けの日本語プログラミング言語であり，句点や日本語の括弧も用いることが可能で，階層的な構文を避けることで初中等教育においても生徒が理解しやすいような設計になっている．高橋らの研究では [9] 日本語のプログラミング言語である「プロデル」を用いて授業教材を開発，授業での実践を行っている．岡田ら [10] は日本語プログラミング言語「言霊」を開発している．これは，日本語からプログラミング言語への翻訳という無駄な作業を取り除くことで文法教育の必要性を無くし、プログラミング教育の効果を上げることを目標に設計されている．

ソースコードやその動きの内容を日本語にするといった形の研究も進められている．小田らの研究 [11] では Python のプログラムをソースコード構文木から統計的機械翻訳を用いて最適な翻訳結果を自動で提示するシステムを開発している．小山らの研究 [12] では変数の値の動きを可視化することでプログラムの理解支援システムを開発している．

このように日本語でプログラミングをする環境を構築し，教育や実際の開発の場で役に立てる研究や，ソースコードを日本語にするシステムの開発は今までに例がある．本研究は既存のプログラミング言語を読み下し文にするシステムの開発に加え，実際に授業に導入し，プログラミング初学者のソースコードの読解の補助が出来るのかを実践するところに新規性がある．

3. 読み下し文の仕様

3.1 デザイン原則

読み下し文は以下の原則に則ってデザインされている．

- 違和感の無い日本語にする．
- プログラムの動きの流れに沿っている文章にする．
- 変数や関数といった専門用語はそのまま扱う．

3.2 代入と式

代入と式の読み下し例を図 1 に示す．

式の読み下しは，複雑な式でもどのような計算をしているか分かるような設計にしてある．式の読み下し文においては，関数を実行する読み下し文を処理に合わせて「呼び出した結果」にしている．四則演算については，乗法と除法の記号はプログラムで用いるものではなく，数学で用いられる記号を使うことで何の計算をしているのか分かりやすいようにしている．

代入については全てコードの左から右へ読み下す形式に

リスト 3 JavaScript ソース

```
1  var x = random(2);
2
3  if (x == 0) {
4    println('0');
5  }
6  else if (x == 1) {
7    print('1');
8  }
9  else {
10   print('error');
11 }
```

リスト 4 読み下し文

```
1  変数xを宣言し、0以上2未満の乱数に書き換える。
2
3  もしxが0と等しい場合、以下を実行する。{
4    「0」を表示する。
5   }
6  そうでなければ、もしxが1と等しい場合、以下を実行する{
7    「1」を表示する。
8  }
9  そうでなければ、以下を実行する。{
10   「error」を表示する。
11 }
```

図 2 if 文の読み下し例

リスト 5 JavaScript ソース

```
1  var i = 0;
2
3  while (i <= 3) {
4    println(i);
5    i = i + 1;
6  }
7
8  for (var x = 0; x <= 3; x++) {

9    println(x);
10 }
11
12 for (var y = 0; y <= 2; y++) {

13   for (var z = 0; z <= 5; z++) {

14   println(z);
15 }
16   println(y);
17 }
```

リスト 6 読み下し文

```
1  変数iを宣言し、0に書き換える。
2
3  iが3以下である限り、以下を実行する{
4    変数「i」の中の文字列を表示する。
5    変数iをi+1に書き換える。
6  }
7
8  宣言した変数xを0に書き換え、xが3以下である間、以下を実行
       する。{
9    変数「x」の中の文字列を表示する。
10 }(8: 変数xに1を足す。)
11
12 宣言した変数yを0に書き換え、yが2以下である間、以下を実行
       する。{
13   宣言した変数zを0に書き換え、zが5以下である間、以下を実行
         する。{
14   変数「z」の中の文字列を表示する。
15 }(13: 変数zに1を足す。)
16   変数「y」の中の文字列を表示する。
17 }(12: 変数yに1を足す。)
```

図 3 繰り返しの読み下し例

設計している．「書き換える」という表現にすることで，扱うデータに関わらず違和感の発生しないような日本語にしている．例えば，サンプルコード 2 行目の「var x = 0」の 0 を「こんにちは」という文字列にした場合，「代入する」という表現を用いると「〜を宣言し、こんにちはを代入する」といった形になる．数値を用いないのに，「代入する」という読み下し文は違和感があると考え，このような設計にしている．

3.3 if 文

if 文の読み下し文はソースコードのミスを見つけやすくする設計にしてある．サンプルコードと読み下し文の例を図 1 に示す．

if 文の読み下し文においては else を使用すると「そうでなければ」と付く．else をつけ忘れてしまったせいで条件を全て満たす結果が出た際に if 文が全て実行されてしまう，初学者の陥りがちなミスを見つけやすくする狙いがある．

3.4 繰り返し

for 文と while 文の読み下し文はプログラムの実行の流れが分かりやすくなるように設計してある．

サンプルコードと読み下し文の例を図 3 に示す．

while 文の読み下し文は条件文のみなので内部の処理でループを終わらせることが殆どである．「〜の限り」という読み下し文にすることによって単体では上限の設定が出来ないイメージを持たせ，for 文との差別化を図る狙いがある．

for 文の読み下し文を見ると，最終式の処理が for 文で実行されるブロックの外に「(12: 変数 i に 1 を足す)」という形になっている．システムの提示する読み下し文はプログラムの実行の流れに沿ったものになっており，最終式はブロックの中を実行した後に実行されるため，3 行目の読み下しでは表示されず，ブロックの後ろに表示される．これにより，for で実行される範囲が分かりやすくなり，for 文の入れ子のコードにおいても，どの繰り返しでどこまでが

リスト 7 JavaScript ソース

```
1  main();
2
3  function main() {
4    var aishow = judge('木村拓哉', '工藤静香');
5    println('相性は' + aishow + 'です');
6  }
7
8  function judge(name1, name2) {
9    var x = name1.hashCode() + name2.hashCode();
10   x = x % 100;
11   return x;
12 }
```

リスト 8 読み下し文

```
1  関数mainを呼び出す。
2
3  以下を実行する関数mainを定義する。{
4    変数aishowを宣言し、実引数木村拓哉,工藤静香を渡す関数
       judgeを呼び出した結果に書き換える。
5    「相性は+aishow+です」を表示する。
6  }
7
8  仮引数name1,name2を渡し、以下を実行する関数judgeを定義
       する。{
9    変数xを宣言し、変数name1の文字列をハッシュコードに変換
       した結果+変数name2の文字列をハッシュコードに変換し
       た結果に書き換える。
10   変数xをxに100を割った余りに書き換える。
11   戻り値としてxを返す。
12 }
```

図 4 関数の読み下し文の例

実行されているかを読みやすく出来るという狙いがある．

for 文の読み下し文は，for 文の最終式の読み下し文を他の for 文の処理と同じ位置に出る仕様を考えた．しかしこの仕様では快適さに欠けるため現在の仕様になっている．例えばソースコードの 8 行目なら「宣言した変数 x を 0 に書き換え、x が 3 以下である間、以下を実行した後、変数 x に 1 を足す」といった文になっている．処理の順番を考えるとこの形式では一度読んだ文章からまた戻って読み直さなくてはならない．

3.5 関数

関数の読み下し文については，プログラムの実行の流れが分かりやすくなるような設計にしてある．サンプルコードと読み下し文を図 4 に示す．

関数を実行する際の読み下し文は，「実行する」ではなく「呼び出す」という表現を用いている．これは，コード上では離れている関数でもその関数がある場所まで飛ぶというイメージを持たせる表現を選んだためである．関数を定義する際は「定義する」という読み下し文にすることで関数の定義と実行の区別しやすくしている．

引数付き関数については仮引数と実引数で読み下し文の表現を変えている．表現を分けておかないと初学者にとっては仮引数と実引数の違いが伝わらない．そのため，引数の仕組みを理解させることを目的に，このような読み下し文にしている．

4. 提案システム

4.1 システム概要

本研究では JavaScript のコードを読み下し文にして提示するシステム「Leia」を提案する．「Leia」はポルトガル語で読むという意味の単語を英語読みして「レイア」と呼ぶ．本システムは，「Sumatra」に組み込まれている．「Sumatra」は，松澤研究室が独自に開発して，授業で導入している JavaScript の学習用開発環境である．

システムは node.js のライブラリである esprima を用いて JavaScript のソースコードをパースすることで AST(Abstract Syntax Tree) を生成し, AST をツリーウォークすることで日本語読み下し文を生成，表示する．

Sumatra 上で Leia を使用した際の外観を図 5 に示す．ブラウザ右上に Leia と書かれたボタンがあり，クリックするとウインドウが表示される．プログラムを実行した後，ウインドウ左上の生成ボタンを押すことで読み下し文が表示される．読み下し文にはソースコードに対応する行番号が付き，ソースコードに合わせてインデントが自動で付くようになっている．

Leia のインタフェース設計には，ソースコードと読み下し文の比較をしやすくする狙いがある．このようなインタフェースで読み下し文を提示するのは，ふりがなプログラミングの長所である，各コードが何を意味していて，どう動くのかがわかるだけではなく，ソースコードが読み下し文になることによりプログラムの実行の流れの把握にも役に立つという長所も付加する，という意図がある．

4.2 利用シナリオ

本節では，実際にシステムを使用する際の 2 つにシナリオを説明する．

4.2.1 シナリオ 1: 読み下し文を読むことによるプログラムのデバッグ

一つ目の利用シナリオは if 文の条件式のミスを見つけることに役立ったというシナリオである．

学生 A さんは if 文を用いて入力された数値によってコンソールに表示される文字を変えるプログラムを書く課題に取り組んでいる．A さんの書いたコードを図 5 の左側に示す．完成したプログラムを動かしたところ，上手く動か

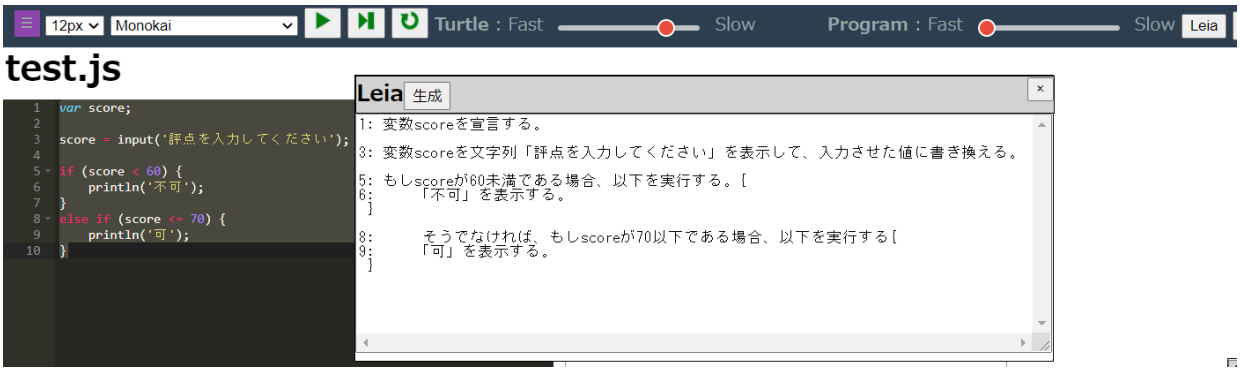

図 5 Leia の外観とシナリオ 1 の例

ない．A さんは比較演算子の意味がまだ分かり切っておらず，条件文が成り立っていないプログラムを書いてしまっている．

そのような時に読み下し文提示システムを使用する．書いたソースコードを読み下し文にした結果をリスト図 5 の右側に示す．読み下し文を見ると条件文の箇所が，「もし score が 60 未満である場合」，「もし score が 70 以下である場合」，のような形の読み下し文になっており，比較演算子がどのような意味を持っているかがすぐに分かる．こうして，A さんのソースコードの読解を補助し，修正が必要な箇所の発見に繋がった．

4.2.2 シナリオ 2: サンプルコードの読解によるプログラム実行の流れの理解

二つ目の利用シナリオは引数付き関数のプログラムの実行の流れを理解することに役立てるというシナリオである．

このシナリオは 3.5 節で使用した図 4 のサンプルコードと読み下し文を使用して説明する．

学生 B さんは，引数付き関数を用いたプログラムのサンプルコードを読んでいる．このような引数付き関数のサンプルコードを読む際に，引数がどのように動いているのか，実引数と仮引数が何なのかという引数の仕組みがコードを読んだだけでは理解出来ていない．

そのような時に読み下し文提示システムを使用する．読み下し文によって，関数の呼び出しや定義などコードを見ただけでは分かりにくい各コードの意味の理解，引数が渡る関数が何なのかの把握ができる．こうして，B さんが関数の仕組みを理解するのに役立った．

5. 実験方法

5.1 実験目的

実験の目的は，Leia を使うことによって初学者のソースコードの読解を補助することが可能なのか，そしてソースコードを読解させやすい読み下し文の形式は何なのかを評価することである．

5.2 対象

本実験の対象は青山学院大学社会情報学部社会情報学科 1 年生の必修授業である，コンピューティング実習の受講者の中の 1 年生，211 人分のデータである．受講者はプログラミングの初学者である．初回授業で筆者が授業で Leia についての 2 分程度の説明を行い，使用するかしないかは，学生の任意とした．

5.3 データの収集方法

本研究で分析するデータは，操作ログと，アンケート，使用者へのインタビューの 3 つである．操作ログは，システムの生成ボタンを押した際に記録されるものであり，使用したファイル名と受講者，読み下し文の結果を記録している．アンケートは，第 13 回の授業中及び授業後にとったものであり，回答者数は 141 人である．アンケートの全設問が複数回答可能になっている．使用者へのインタビューは操作ログにおいて Leia を多く使用している学生に対して行った．

6. 結果

6.1 Leia の使用率

Leia を使用した受講者の数を調査するため，操作ログを解析した結果を図 6 に示す．操作ログにおいて，「二つ以上のファイルで Leia を利用した学生」をシステムの利用者とした．操作ログに記録された Leia の使用者は 107 人で，受講者の 50,7%であった．

次に，アンケートにおける，「Leia が役に立ったという経験がありますか」という設問の回答結果を図 7 に示す．設問に対して役に立ったと回答した人は 74 人で，受講生の 35%であった．

受講者のシステムの使用は任意であり，役に立つ確証も無いという状況でシステムを使用し，高く評価している．これらのことから使用率が 50.7%，役に立ったと回答した人が 35%という結果に対し，システムが一定の支持を得ていると考えることができる．

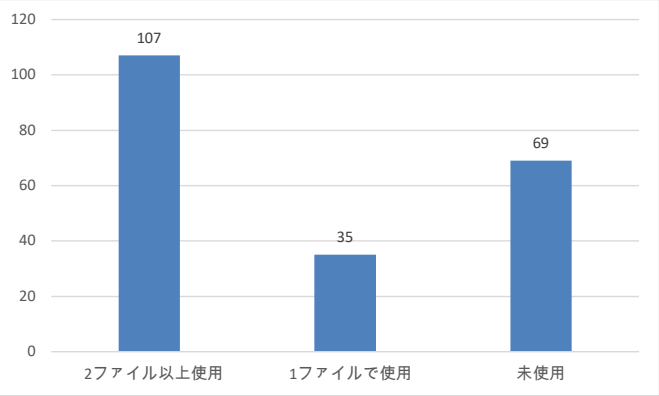

図 6 Leia の使用率

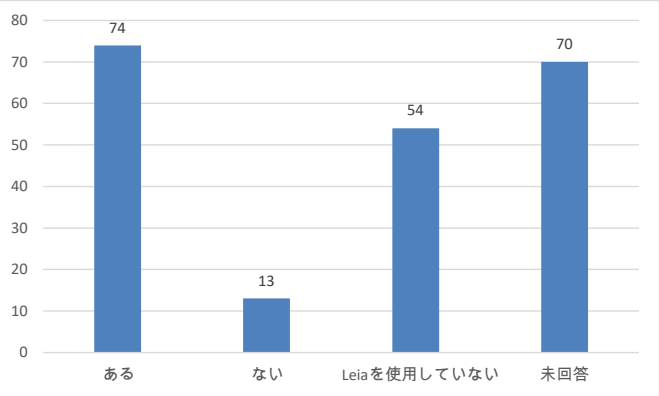

図 7 Leia を使用し，役に立った経験があるか

6.2 システムがどの単元で使用されていたか

操作ログから，Leia が使用されたファイルの個数を単元別に参照し，どの単元において使用率が高いのかを算出した結果を表 1 に示す．括弧内は何週目の授業かを記載している．変数，For 文の入れ子，関数，引数付き関数の単元が比較的使用率が高い．

次に，アンケートにおいて“Leia はどの章で役に立ちましたか”という設問に対しての回答率を表 2 に示す．For 文の入れ子，関数，引数付き関数の単元で役に立ったと回答した割合が高く，変数の単元では比較的回答率が低い．

変数の単元で使用率が高かった．これは導入直後の授業で，システムがどのようなものかを確かめるために使用したのが原因ではないかと考えられる．それに対し変数の単元で役に立ったという回答が少なかった．これは読み下し文が無くてもソースコードの読解に問題はなかったのではないかと考えられる．

後半の単元では使用率が高く，役に立ったという声も多かった．操作ログに引数付き関数の回のサンプルコードでシステムを使用したと思われるログが特に多くあり，システムを使用した学生へのインタビューにおいて，コードが長く複雑になり，日本語で整理することでわかったという声があった．これらのことからコードが長く複雑になるとアルゴリズムも複雑になってくるため，使われたのではないかと考えられる．

表 1 単元別の Leia の使用率

単元とその週	使用率
変数 (第 2 週)	19.5%
If 文 (第 3 週)	12.0%
For 文 (第 4 週)	8.9%
For 文の入れ子 (第 5 週)	18.2%
関数 (第 6 週)	17.4%
引数付き関数 (第 7 週)	23.7%

表 2 Leia がどの単元で役に立ったか

単元とその週	回答率
変数 (第 2 週)	19.1%
If 文 (第 3 週)	30.1%
For 文 (第 4 週)	26.0%
For 文の入れ子 (第 5 週)	39.7%
関数 (第 6 週)	36.9%
引数付き関数 (第 7 週)	43.8%

表 3 Leia を使用した時の状況

選択肢	回答率
課題を進めていたらプログラムが動かない所があったため原因を探るために使った	71.2%
テキストのサンプルコードの意味が分からなかったためコードの機能を理解するために使った	36.9%
プログラムを段階的に解くために、機能を搭載する度に確認のため使った	12.3%
プログラムは分かっているが念のため自分が何をしているか確認するために使った	10.0%

6.3 Leia を使用した時の状況

アンケートにおいて，「Leia をどのような状況で使用しましたか」という設問の結果を表 3 に示す．

利用シナリオ 1 に即した回答である「課題を進めていたらプログラムが動かない所があったため，原因を探るために使った」と回答した受講者が 71.2%，利用シナリオ 2 に即した内容である「テキストのサンプルコードの意味が分からなかったため，コードの機能を理解するために使った」と回答した受講者が 36.9%という結果が得られた．このことから，4.2 節で説明した二つの利用シナリオと同じ状況で使われていることが分かった．

次に，「プログラムを段階的に解くために、機能を搭載する度に確認のために使った」と回答した学生が 12.3%，「プログラムは分かっているが念のため自分が何をしているか，確認するために使った」と回答したが 10%という結果が得られた．このことから，4.2 節で説明した利用シナリオ以外の状況でもシステムが初学者のソースコードの読解に寄与する可能性があると考えられる．

表 4 アンケートにおけるコメントの一部

学生	コメント
B さん	エラーと表示される時は英語が表示されるため，その時点でどこが間違っているのかがわからない状態でしたが， Leia を使用することにより，分からない点が言語化されるので，より理解がしやすくなり，エラー解決にかける時間も短縮することができました．特に for 文から苦手が増えたため，Leia を多く活用させて頂きました．
C さん	Leia を利用することでプログラムの問題点を発見することができました．また，自分の頭の中でプログラムを言語化し理解をする際にも Leia の読み下し文が参考になりました．
D さん	コードだけだと一見わかりにくく感じてしまうものも， Leia の機能を使って日本語での表示を見ると理解できるようになり，授業課題に取り組む際にとても役に立ちました．
E さん	分からなくなったらとりあえず Leia！と友人が言うくらい信頼していたのでとても素晴らしいシステムだったと感じました

6.4 使用者の声

6.4.1 インタビューにおける使用者の声

操作ログにおいて Leia を多く使用していた人にインタビューを行った．その結果の一部を抜粋してシステムが実際にどのように利用されていたのかを説明する．

学生 K さんは授業の課題を進めていく中で，利用シナリオ 1 と同じような形でシステムを使用しており，

- 自分は数学が苦手で，イコール付くかつかないかがどっちだか分からなくなったのですが，そういった問題も解決しつつシステムを使用しました．

とコメントしていた．イコールが付くか分からないというのは比較演算子の以下と未満の符号を覚えていなかったということであった．

学生 A さんは授業のテキストのサンプルコードの多くで使用していた．

- テキストとか動画を見たけど理解しきれなかった時に，訳として出てくることでコードの意味を確認することが出来た
- コードが複雑な時こそシステムが使えた

とコメントしていた．A さんは主に for 文の入れ子以降の後半の単元で使用していたため，このようなコメントが得られたと考えられる．

6.4.2 アンケートにおける使用者の声

アンケートで自由解答欄の記述を抜粋し，表 4 に示す．抜粋したコメント以外にも，システムに対して好意的なコメントが 70 件ほど確認出来た．

6.5 その他アンケート結果

その他，Leia に関するアンケートの回答結果を抜粋したものを，以下に示す．

表 5 読み下し文をどのように見ることで自分の問題を解決したか

選択肢	回答率
コードの読み下し文全体を見てプログラムの動きの流れを理解した	45.0%
コードの分からない部分の読み下しだけを見てコードの備えている機能を理解した	38.8%
undefined や error!が表記されて、自分が書いてるコードが正しくないことに気づいた	30.0%
複数の読み下しを生成して見比べることで自分の書いているコードの問題点を見つけた	15.0%
その他	2.5%

6.5.1 読み下し文をどのように見たか

アンケートにおける，「読み下し文をどのように見ることで自分の問題を解決しましたか」という設問の結果を表 5 に示す．

「コードの読み下し文全体を見てプログラムの動きの流れを理解した」と回答した受講生が 45%，「コードの分からない部分の読み下しだけを見てコードの備えている機能を理解した」と回答した受講生が 38.8%と高い回答率を得た．このことから，ソースコードが文章になることによりプログラムの実行の流れの把握にも役に立つ，という長所と同時に，ふりがなプログラミングの各コードが何を意味していて，どう動くのかがわかる，という長所が読み下し文には備わっていると考えられる．

6.5.2 Leia を利用して役に立たなかった単元はどれか

アンケートにおいて，Leia を使用して役に立った経験がなかったという受講生に使用した単元を回答してもらった．結果を表 6 に示す．最も役に立たなかったと回答されたのは for 文である．for 文からプログラムが単純に上から下の順に実行される訳ではなくなり，アルゴリズムが複雑になったこと，読み下し文に理解を補助する工夫が不足していた可能性があることが考えられる．

アンケートで役に立ったという回答が多かった関数や引数付き関数が高い回答率を得ている．ソースコードが長くなり，読み下し文が複雑になることで読みづらさを感じてしまう人や読み下し文を見てもコードの意味が分からなかった人もいたと考えられる．

役に立ったという回答が少なかった変数や if 文は役に立たなかったという回答も少ない．6.2 節で説明したように，導入直後の授業でシステムがどのようなものか確かめるために使っただけではないかと考えられる．

6.5.3 Leia を利用しなかったのは何故か

アンケートにおいて，Leia が役に立たなかった，Leia を使用していないという受講生を対象に理由を聞いた．回答率を表 7 に示す．「読み下し文が無くてもプログラムの理解ができていたから」という回答が 52.4%と最も高く，既にソースコードを理解している受講生はシステムを使わずとも学習を進められるため，効果が期待できないことが分

表 6 Leia を利用して役に立たなかったという人の使用した単元

単元とその週	回答率
変数 (第 2 週)	11.1%
If 文 (第 3 週)	11.1%
For 文 (第 4 週)	33.3%
For 文の入れ子 (第 5 週)	16.7%
関数 (第 6 週)	27.8%
引数付き関数	22.2%

表 7 Leia を利用しなかった理由

選択肢	回答率
読み下し文が無くてもプログラムの理解ができていたから	52.4%
システムの使い方が分からなかった	27.0%
提示された読み下し文がわかりにくかった	20.6%
システムの UI が分かりにくかった	9.5%
その他	11.1%

かった.

7. 考察

7.1 読み下し文提示システムはソースコードの読解を補助出来たのか

操作ログの解析結果とアンケートの結果から，システムの使用は任意であったにも関わらず使用者からは一定の支持を得た．このことから，読み下し文提示システムによるソースコードの読解を補助することが出来たと評価する．どの単元でもシステムは使われたが，特に for 文の入れ子や関数，引数付き関数の単元で使用されていること，使用者へのインタビューの結果から，プログラムのアルゴリズムが複雑になった際に使われるのではないかと考えることができる．

7.2 ソースコードの読解をさせやすい読み下し文はどのような形式だったのか

操作ログの解析結果とアンケートの結果から，for 文の入れ子や関数，引数付き関数で役に立ったという声が多かった．このことから，ソースコードが長く複雑になることで読み下し文も長くなったとしても，それらの単元では現在のシステムでも意味の伝わる形の読み下し文になっているのではないかと考えられる．

for 文で役に立ったという声が少なく，for の入れ子になった時に役に立ったという声が増えた要因としてはプログラムの流れに沿った読み下し文になっていることから，入れ子の区切りが見やすくなったという可能性がある．

変数や if 文，for 文といった前半の単元においては役に立ったという声が少なかったことから，それらの読み下し文は改善の余地があるか，読み下し文を表示するまでもなく理解が進んだ学生が多かったのではないかと考えられる．

8. まとめ

本研究では読み下し文提示システムを開発し，大学生のプログラミング初学者を対象にした授業で導入実験を行い，システムの評価を試みた．

操作ログやアンケート，使用者へのインタビューの結果，システムは一定の支持を得ており，こちらの想定した利用シナリオの通りに使用してソースコードの読解に役立てている被験者を確認できたことから，プログラミング初学者のソースコードの読解の補助に利用できる可能性が示唆された．

ソースコードの読解をさせやすい読み下し文が何なのかを明らかにするという研究目的については，ソースコードの長くなる for 文の入れ子や関数などの授業後半の単元で使用者から支持を得ていることから，読み下し文の長さは大きな問題にならないということが考えられるのみである．

今後は授業後半の単元で支持を得た理由を明確にする点，現在の仕様では支持を多く得られなかった，前半の単元における読み下し文の更なる改善を進めていく点が課題となる．

参考文献

[1] リブロワークス・著 及川卓也・監修：スラスラ読める JavaScript ふりがなプログラミング (2018).
[2] 札場寛之，小田悠介，Neubig, G.，畑　秀明，Sakriani, S.，戸田智基，中村　哲：機械翻訳を用いた疑似コード生成による学習者支援，教育システム情報学会第 40 回全国大会，pp. 259–260 (2015).
[3] 平賀正樹：日本語プログラミング言語 Mind，コンピュータソフトウェア， Vol. 6, No. 2, pp. 170–178 (1989).
[4] 中川正樹，早川栄一，玉木裕二，曽谷俊男：日本語プログラミングの実践とその効果，情報処理学会論文誌，Vol. 35, No. 10, pp. 2170–2179 (1994).
[5] 酒徳峰章：日本語プログラミング言語「なでしこ」，コンピュータソフトウェア，Vol. 28, No. 4, pp. 23–28 (2011).
[6] 犬童健良：日本語による論理プログラミングの試み，関東学園大学経済学紀要　第 46 集 (2020).
[7] 西田知博，原田　章，中村亮太，宮本友介，松浦敏雄：初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol. 48, No. 8, pp. 2736–2747 (2007).
[8] 兼宗　進，御手洗理英，中谷多哉子，福井眞吾，久野　靖：学校教育用オブジェクト指向言語「ドリトル」の設計と実装，情報処理学会論文誌プログラミング，Vol. 42, No. SIG11(PRO12), pp. 78–90 (2001).
[9] 高橋岳之，近藤　泉，山田果林：教科情報における日本語プログラミング言語を用いた授業実践，愛知教育大学研究報告　自然科学編，pp. 11–14 (2018).
[10] 岡田　健，中鉢欣秀，鈴木　弘，大岩　元：プログラミング言語としての日本語，情報処理学会第 44 回プログラミング・シンポジウム (2003).
[11] 小田悠介，札場寛之，Neubig, G.，Sakriani, S.，戸田智基，中村　哲：ソースコード構文木からの統計的自動コメント生成，情報処理学会研究報告，No. 12, pp. 1–9 (2014).
[12] 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，Vol. 10, No. 4, pp. 1–11 (2017).