**２０２１　年度　　卒業論文**

卒業論文題目

|  |
| --- |
| ブロックプログラミングを用いた論理的思考と コーディングを身に着けるための学習環境 |

　指導教員　　　　　　鷹野　孝典　教授

神奈川工科大学　情報工学科

　　　　　学籍番号　　　　1821121

　　　　　学生氏名　　　　島岡慎也

　提出日　２０２１年　１２月２３日　指導教員　　印

　受理日　２０２１年　１２月２３日　情報工学科長　　印

# **論文要旨**

# 目次

[**論文要旨** 2](#_Toc89172924)

[目次 1](#_Toc89172925)

[**1.** **まえがき** 1](#_Toc89172926)

[1.1 はじめに 1](#_Toc89172927)

[1.2 本論文の構成 1](#_Toc89172928)

[2. 関連研究 2](#_Toc89172929)

[2.1 2](#_Toc89172931)

[**3.** **提案システム** 3](#_Toc89172932)

[3.1 提案方式の概要 3](#_Toc89172934)

[3.2 提案方式の構成 3](#_Toc89172935)

[3.2.1. 問題を自動生成する 3](#_Toc89172945)

[3.2.2. 問題の正誤を評価 3](#_Toc89172946)

[3.2.3. 整合性の評価 3](#_Toc89172947)

[3.3 提案方式を構成する機能 3](#_Toc89172948)

[**4.** **実装システム** 4](#_Toc89172949)

[4.1 実装システムの概要 4](#_Toc89172951)

[4.2 実装システムの実行環境 4](#_Toc89172952)

[4.3 実装システムの機能説明 4](#_Toc89172953)

[4.3.1. Visual Program Language 4](#_Toc89172958)

[4.3.2. Node.jsについて 5](#_Toc89172959)

[4.3.3. コードジェネレート機能 5](#_Toc89172960)

[4.3.4. コードの実行 5](#_Toc89172961)

[4.4 実装システムの穴あき問題の実装 5](#_Toc89172962)

[4.4.1. 穴埋め問題の生成 5](#_Toc89172964)

[4.5 実装システムの穴あき問題の評価の実装 5](#_Toc89172965)

[4.5.1. 穴埋め問題の解答の評価 5](#_Toc89172967)

[4.6 Blocklyの変更点 7](#_Toc89172968)

[**5.** **実験** 8](#_Toc89172969)

[5.1 実験目的 8](#_Toc89172971)

[5.2 実験環境 8](#_Toc89172972)

[5.3 実験 8](#_Toc89172973)

[5.4 実験の考察 8](#_Toc89172974)

[4.5.2. 実験の感想一覧 8](#_Toc89172975)

[**6.** **むすび** 9](#_Toc89172976)

[**謝辞** 10](#_Toc89172977)

[**参考文献** 11](#_Toc89172978)

[その他 11](#_Toc89172979)

# **まえがき**

## はじめに

2020年以降小学校，中学校，高等学校にてプログラミング教育の必修が全面実施される．

文部科学省の発表によると，プログラミング教育を必修とする目的は，現代社会で普遍的に求められる力としての論理的思考などを育むことである．論理的思考とは，「思考や論証の組み立て，思考の妥当性が保証される法則や形式」に則って思考を組み立てることとする．学習者が教科書を読み進めるだけではなく，プログラミングに実際に触れることで，論理的思考を身に着けられる環境を作成したいと考えた．ブロックプログラミングを用いることで，学習の初期段階でプログラム言語の構文への理解が浅い状態でもコードの実行が可能である．Blocklyのコードジェネレート機能を用いることで，プログラムの構文への理解につながる．論理的思考がすでに培われていることを前提として問題が提示される教育環境となっている．学習者が論理的な思考をもって問題文を読み取ることができる必要がある．プログラミング言語における構文を理解するには，コーディング経験も重要となる．Blocklyによって生成されたコードを見るだけではなく，実際に学習者がコーディングする環境でないと，プログラミング言語の構文への理解の定着にはつながらない．

課題

穴あき問題を自動で生成する際に，正答だと想定した選択肢以外にも実行できる選択肢が存在する場合などに，それを正解だと判断せず，既定の選択肢以外が不正解だと認識してしまう。

提案

　問題としての正解以外でも，既定の選択肢ではないが，実行結果から問題の解答として正しい，問題の解答としては正しくないがコードとしては実行できるなど，問題の正誤だけでなくコードとしての整合性も考慮した採点をする．

## 本論文の構成

　本論文は全６章で構成されている。本章では研究の背景や動機，目的について述べる．第２章では関連研究について述べる．第３章では本研究の提案方式とその特徴述べる。第４章では提案方式を構築した本システムの設計及び実験システムの実装について述べる．第５章では実験を行う。実験の概要と，比較実験後の結果を述べる．第６章では結論と今後の展望について述べる．

# 関連研究



## 

# **提案システム**

　本研究では，穴あき問題の解答の規定の正答だけでなく、正答以外の選択の際にも「構文上の正しさ」と「問題文との整合性」の２点の評価をすることにより，不正解となった場合の学習効率の向上に焦点をあてた学習支援を提案する．なお，提案方式の概要を3.1節，提案方式の構成を3.2節，提案方式を構成する機能を3.3節に示す．



## 提案方式の概要

提案方式は，ブロックプログラミングとコードの比較をする部分，穴埋め問題を利用し複数のコードを見比べる部分，問題と解答から，構文上正しいかどうか，問題文と比べ正しいかどうかを評価する部分に分かれている．

## 提案方式の構成



### 問題を自動生成する

事前に準備されたコードを全て単語毎に区切り，別に用意した規定のワードと比較し，マッチングした箇所を穴あき問題へと変換する．この時，穴あき問題の問題数を変更することで難易度の調整をする．

### 問題の正誤を評価

　3.2.1章にて作成した穴あき問題にて，選択肢中に正答かどうかを判別するオプションを設定しておき，それにより問題の正誤を判定する．

### 整合性の評価

問題の規定の正解以外でも、実行結果から問題の解答として正しい選択肢、あるいは問題の解答としては正しくないが，コードとしては実行できる状態となる選択肢など、問題の正誤だけでなくコードとしての整合性も考慮した評価をする。前者の評価は、事前に用意した結果と、解答から作成される実行結果との比較によって評価をすることが可能である。後者の評価は、実際にコードを実行した結果、事前に用意した正答の結果とは異なるが、エラーが発生しないまま終了した場合などにどのような間違いだったのかを評価する。

## 提案方式を構成する機能

・問題自動生成

・正誤評価

・整合性評価

# **実装システム**

　本研究では，穴あき問題の解答の規定の正誤だけでなく、規定の正答以外の選択の際にも「構文上の正しさを評価する機能」と「問題文との整合性を評価する機能」の２点により，不正解となった場合の学習効率の向上に焦点をあてた学習支援のための機能を実装した．なお，実装システムの概要を4.1節，実装システムの実行環境を4.2節，に示す．



## 実装システムの概要

　実装システムの構築環境を表4.1に示す．

表 4.1 実装システムの構築環境

|  |  |  |
| --- | --- | --- |
| 利用用途 | 名称 | バージョン・型番 |
| OS | Windows 10 |  |
| 開発言語 | Javascript |  |
| パッケージ管理 | npm |  |
| VPL | Blockly |  |

## 実装システムの実行環境構築

### ・本研究のプログラムのダウンロード

以下のgithubのリンクよりprojectの内容をすべてダウンロード．

|  |
| --- |
| https://github.com/kait-takanolab/1821121-shimaoka-thesis/tree/main/project |

### ・node.js

　Node.jsとはサーバ上で動作するJavascript環境．ほとんどのJavascriptと異なり，ウェブブラウザ上でなく，サーバサイドでの動作となる．本研究ではサーバ上での実行と，npmの利用のためにNode.jsを利用する．

npmとはパッケージ管理システムの一種で，Node package managerの意である．Node.jsを用いたシステムの管理のために作成され，Node.jsのツールやパッケージのインストールや管理はもちろん，それらパッケージを利用することも可能．

　また，Node.jsのWebアプリケーションフレームワークであるexpress.jsを用いてjson環境を構築した．

以下のリンクよりNode.jsをインストール．

　Node.jsをインストールすることで、Node.jsというサーバ上でJavascriptを実行ができる．また，npmも同時に利用が可能になる．

|  |
| --- |
| https://nodejs.org/ja/ |

　npmを用いたパッケージのインストールにはnpm install “パッケージ名称”というコマンドでインストールが可能である。本研究のシステムの利用にはいくつかのパッケージのインストールが必要である。下記にインストールのコマンドを示す。（途中）nodemon,morgan,mongoose,parser

|  |
| --- |
| npm install nodemon |

### ・サーバの実行コマンド

　4.2.1項，4.2.2項にて本研究のシステム、Npde.jsのインストールが完了した後にprojectの位置で下記のコマンドを実行することでサーバが起動される．その後loaclhost:3000にアクセスすることで本システムのindex.htmlにアクセスすることができる．

|  |
| --- |
| npm start |

## 実装システムの機能説明



### Visual Program Language

ビジュアルプログラミング言語（VPL）とは，プログラムを視覚的なオブジェクトの組み合わせによって作成するプログラミング言語である．ブロックプログラミング，グラフィカルプログラミング言語とも言う．VPLとは総称であり，どのように視覚的に示すかにはいろんな種類があるが，本研究ではBlocklyを利用する．本研究では一部改変したデータを用いるため、改変後のデータを本研究のproject内にBlocklyのファイルも同封されているが、Blocklyの公式の状態のデータを扱うためには、google/Blocklyのgithubにてファイルをインストールした後、npmによるインストールによって可能となる。

google/blockly github

|  |
| --- |
| https://github.com/google/blockly |

blockly install

|  |
| --- |
| npm install blockly |

### コードジェネレート機能

generateJS(),indent(),

コードジェネレート機能とは，Blocklyによって提供されるモジュールの一つで，組み合わせたブロックからコードを生成する機能を提供している．Blockly公式より提供されるコードジェネレート機能はJavaScript，Python，PHP，Lua，Dartの５つのプログラミング言語に対応したものが存在し，関連研究にはC言語など他の言語に対応させたものも存在する．本研究ではJavascriptに関連した穴あき問題の生成・評価をしている．

コードジェネレート機能を利用するためには、Blocklyを用いるためのスクリプトをコード内に含める必要がある、下記に示す。

|  |
| --- |
| <script src="blockly\_compressed.js"></script>  <script src="javascript\_compressed.js"></script> |

コードを生成する際には下記のコードを呼び出すことで可能となる。

|  |
| --- |
| var code = Blockly.JavaScript.workspaceToCode(workspace); |

リアルタイムでコードを表示する際には下記の様に組み合わせることで可能である。

|  |
| --- |
| function myUpdateFunction(event) {  var code = Blockly.JavaScript.workspaceToCode(workspace);  document.getElementById('textarea').value = code;  }  workspace.addChangeListener(myUpdateFunction); |

### Javascriptコードの実行

jsrunque(),

jsrunout(),

jsruntst()

　4.3.2項にて作成されたコードは無限ループが発生していないかを判断した後にコードを実行する関数を呼び出す、また実行した際にエラーが発生する場合にはそのエラーを表示する。下記のコードによって実装されている。jsrunque(),jsrunout()はどちらもコードを実行する関数であるが、実行するコードを読み取るエリアが異なる以外はほぼ同様のため片方を示す。

jsrunout(),

|  |
| --- |
| function jsrunout() {  // Generate JavaScript code and run it.  window.LoopTrap = 10000;  Blockly.JavaScript.INFINITE\_LOOP\_TRAP =  'if (--window.LoopTrap == 0) throw "Infinite loop.";\n';  var code = document.getElementById("output").innerText;  Blockly.JavaScript.INFINITE\_LOOP\_TRAP = null;  try {  execution(code);  } catch (e) {  document.getElementById('kaitou').textContent = e;  }  } |

　また、index.htmlではBlocklyを用いらず、自身で書いたJavascriptコードの実行も実装されているが、そちらも同様な書き方によって実装されている。jsruncode()という関数。

test.htmlでは穴あき問題からユーザが選んだ選択肢によって完成したコードを実行することが可能である。これはセレクトボックスによって作成された穴埋め問題の選ばれた解答を置換し、実際のコードとなる文字列を作成した後、コードを実行している。

jsruntst()

|  |
| --- |
| function jsruntst() {  var count = 0;  window.LoopTrap = 10000;  Blockly.JavaScript.INFINITE\_LOOP\_TRAP =  'if (--window.LoopTrap == 0) throw "Infinite loop.";\n';  var code = document.getElementById("test").innerHTML;  code2 = code.split(/<select|<\/select>/);    for (let i = 0; i < code2.length; i++) {  if(code2[i].indexOf("sele") == -1){  }else{  while(1){  if(code2[i].indexOf(`sele${count}`) != -1){  break;  }else{  count++;  }  }  var obj = document.querySelector(`select.sele${count}`);  var idx = obj.selectedIndex;  var txt = obj.options[idx].text;  code2[i] = txt;  count = 0;  }  }  code2 = code2.join('');  //code2 = (String(code2)).replace(/,/g, '');  code2 = (String(code2)).replace(/\<br\>/g, '');  code2 = (String(code2)).replace(/&nbsp;/g, '');  code2 = (String(code2)).replace(/&gt;/g, '>');  code2 = (String(code2)).replace(/&lt;/g, '<');  code2 = (String(code2)).replace(/;/g, ';\n');  code2 = (String(code2)).replace(/{/g, '{\n');  code2 = (String(code2)).replace(/}/g, '}\n');  code2 = sengen(code2);  //console.log(code2);  Blockly.JavaScript.INFINITE\_LOOP\_TRAP = null;  try {  execution(code2);  }catch (e) {  document.getElementById('kaitou').textContent = e;  }  } |

上記までで利用している、作成されたJavascriptコードを実行し結果を確認するために、本システムではFunction関数を用いる。実際には下記の様に実装している。codeとは実際にコードがはいっている文字列を引数とする。

execution(),

|  |
| --- |
| function execution(code){  kaitouran();  Function(code)();  } |

実行結果を表示する欄を一度初期化するためにkaitouran()という関数を呼び出している。

kaitouran(),

|  |
| --- |
| function kaitouran(){  document.getElementById("kaitou").textContent = "";  } |

## 実装システムの穴あき問題の実装



### 穴埋め問題の生成

めも：コード全体ではなく、ステップごとにコード片をつけるようにする

本システムでは、問題の例となる完成されたコードから、穴埋め問題を自動で生成している。手順として以下のステップによって生成される。

1. コードを単語で区切り、既定のワードとマッチするかを判断し、ヒットした場所とワードを保存するリストを作成する。
2. 問題の生成数の数とワードがヒットした数を比べ、問題数を決定する。
3. 既定の数よりも多くワードがヒットしていた場合、規定数分だけランダムで数字を作成し、以下のステップに進む。
4. リストより、ヒットした場所をセレクトボックスに置換する
5. セレクトボックスの内容に穴あき問題の解答を挿入し、オプションとして、正答の場合にはans,それ以外には数字を設定する。

問題の生成数は簡単、普通、難しいの３種類用意し、それぞれ3,6,9個の問題を生成する。初期では３問生成される。

実際のコードは以下の通りである。

match()

|  |
| --- |
| function match(code,mode) {  //全選択肢となる部分　もしかしたらDBなどにいれこむかも  var list1 = [  ['for', 'while', 'do'],  ['if', 'else', 'which'],  ['print', 'alert'],  ['break','continue'],  ['<','>','<=','>=','==']  ];  //ワードがマッチした場所を保存する  var lista = [];  //マッチしたワードを含む選択肢を保存する  var sbox = [];  //マッチしたワードを保存する  var mbox = [];  //何回ワードがマッチしたか保存する  var count = 0;  //選択肢になるリスト  var qbox = [];  //?  var btext = code;    //文字列を配列に変更  ctext = btext.split(/(\s|\(|\)|\;|\.|\n)/);  // console.log(ctext);    //問題とのマッチングループ    for(let i = 0; i < list1.length; i++){  for(let j = 0; j < list1[i].length; j++){  for(let k = 0; k < ctext.length; k++){  if(ctext[k] == list1[i][j]){  sbox[count] = list1[i];  mbox[count] = ctext[k];  lista[count] = k;  count += 1;  }  }  }  }  if(mode > count) {  mode = count;  }  //選択肢に変更する個数と変更する箇所を設定する mode,lista  for(var q = 0; q < mode; q++){  while(true){  var tmp = getRandomInt(count);  if(!qbox.includes(tmp)){  qbox.push(tmp);  break;  }  }  }  //問題数の順番を上から順番になるようにソートする  for(var i=0;i<qbox.length;i++){  for(var j=0;j < qbox.length;j++){  if(lista[qbox[i]] < lista[qbox[j]]){  var tmp = qbox[i]  qbox[i] = qbox[j]  qbox[j] = tmp  }  }  }  //マッチしたワード部分を選択肢に変更  for(let l = 0; l < qbox.length; l++){  ctext[lista[qbox[l]]] = `<select class="sele${l}"></select>`;  }    // 配列から文字列に変更して表示  ctext = (String(ctext)).replace(/\n/g, '<br>');  ctext = (String(ctext)).replace(/,/g, '');  //文字列を表示する  document.getElementById("test").innerHTML = indent(ctext);      // ワードがマッチした場所に選択肢を生成する  for(let m = 0; m < qbox.length; m++){  var select = document.querySelector(`select.sele${m}`);  for(let n = 0; n < sbox[qbox[m]].length; n++){  var option = document.createElement('option');  option.innerText = sbox[qbox[m]][n];  if(sbox[qbox[m]][n] != mbox[qbox[m]])  option.value = n;  else  option.value = "ans";    select.append(option);  }  }  } |

## 実装システムの穴あき問題の評価の実装



### 穴埋め問題の解答の評価

　穴埋め問題の解答から、解答者が何問正解したのかを判断する。正解数と、正解した問題番号を表示する

answer2(),

|  |
| --- |
| function answer2(){  //正解数をカウントする  var count = 0;  //ループ回数をカウント  var loop = 0;  //正解した問題番号を保存する  var kotae =[];  //セレクトボックスがいくつあるのかをカウント  //セレクトボックスがなかったらBreak  while(1){  var select = document.querySelector(`select.sele${loop}`);  if(select != null){  //セレクトボックスのValueを入手（正解ならans, それ以外だと数字が返る）  if(select.value == "ans"){  //何問正解しているかを合計  kotae[count] = loop+1;  count++;  }  }else{  break;  }  loop++;  }  //判定結果を表示  document.getElementById("answer").textContent = count + "問正解\n正解した問題は:" + kotae;  console.log(kotae)  } |

　実行結果から、問題の解答例と比較し、正誤を評価する。複数文字列を表示する問題の場合には、何個目の表示が間違っているのかも表示する。下記のコードによって実装。

check2(),

|  |
| --- |
| function check2(){  var code1 = document.getElementById('kaitou').textContent;  var code2 = strcode[2];  var hyouka = document.getElementById("answer").textContent;  if(code1 == code2){  hyouka = "実行結果は正しい";  }else{  if(strcode[3]=="num"){  hyouka = numq(code1,code2);  }else{  var hyouka = "実行結果は誤り\n";  }  }  document.getElementById('answer').textContent = hyouka;  } |

numq()

|  |
| --- |
| function numq(code1,code2){  hyouka = "";  c1 = code1.split(' ');  c2 = code2.split(' ');  console.log(code1,code2,c1,c2);  if(c1.length != c2.length){  hyouka += "ループに間違いがあります、"  if(c1.length > c2.length){  hyouka += "回答が多いです\n"  }else{  hyouka += "回答が少ないです\n"  }  }  for(var i=0; i < c2.length; i++){  if(c1[i] != c2[i]){  hyouka += i+1 + "個目は誤り\n"  }  }  if(c2.length < c1.length){  hyouka += c2.length + "以降は誤り\n"  }  if(hyouka != ""){  return hyouka;  }else{  return "実行結果は正しい";  }  } |

## Blocklyの変更点

Blocklyを公式の状態のまま利用すると本システムでは扱いづらい部分が存在したため、一部を改変し、利用している。変更箇所はblocks\_compressed.js , javascript\_compressed.js, en,jsの３ファイルで、printとfor文の利用のために行った。

### text\_printを変更

　text\_printとはBlocklyによって文字列のアウトプットを扱うために利用されるもので、初期設定では、文字列をalertにて表示するものであった。textareaに実行結果を表示するように変更することで、実行結果を学習者がみやすいと考え下記のように変更した。

|  |
| --- |
| Blockly.JavaScript['text\_print'] = function(block) {  // Print statement.  var msg = Blockly.JavaScript.valueToCode(block, 'TEXT',  Blockly.JavaScript.ORDER\_NONE) || '\'\'';  return "document.getElementById('kaitou').textContent += " + msg + " + \" \";\n";  }; |

### for文を変更

　Blocklyの実装では最初は”<=”の形で以上または以下という書き方でのfor文のみを考慮されていた。この書き方だけでもプログラムは作成できるが、本研究では学習支援を目的としているため、”<”というより多い、あるいは未満という形で扱えるように変更した。下記に変更を示す。

blocks\_compressed.js

|  |
| --- |
| {type:"controls\_for1",message0:"%{BKY\_CONTROLS\_FOR\_TITLE1}",args0:[{type:"field\_variable",name:"VAR",variable:null},{type:"input\_value",name:"FROM",check:"Number",align:"RIGHT"},{type:"input\_value",name:"TO",check:"Number",align:"RIGHT"},{type:"input\_value",name:"BY",  check:"Number",align:"RIGHT"}],message1:"%{BKY\_CONTROLS\_REPEAT\_INPUT\_DO} %1",args1:[{type:"input\_statement",name:"DO"}],inputsInline:!0,previousStatement:null,nextStatement:null,style:"loop\_blocks",helpUrl:"%{BKY\_CONTROLS\_FOR\_HELPURL}",extensions:["contextMenu\_newGetVariableBlock","controls\_for\_tooltip"]},  {type:"controls\_for2",message0:"%{BKY\_CONTROLS\_FOR\_TITLE2}",args0:[{type:"field\_variable",name:"VAR",variable:null},{type:"input\_value",name:"FROM",check:"Number",align:"RIGHT"},{type:"input\_value",name:"TO",check:"Number",align:"RIGHT"},{type:"input\_value",name:"BY",  check:"Number",align:"RIGHT"}],message1:"%{BKY\_CONTROLS\_REPEAT\_INPUT\_DO} %1",args1:[{type:"input\_statement",name:"DO"}],inputsInline:!0,previousStatement:null,nextStatement:null,style:"loop\_blocks",helpUrl:"%{BKY\_CONTROLS\_FOR\_HELPURL}",extensions:["contextMenu\_newGetVariableBlock","controls\_for\_tooltip"]}, |

javascript\_compressed.js

|  |
| --- |
| Blockly.JavaScript.controls\_for1=function(a){var b=Blockly.JavaScript.variableDB\_.getName(a.getFieldValue("VAR"),Blockly.VARIABLE\_CATEGORY\_NAME),c=Blockly.JavaScript.valueToCode(a,"FROM",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"0",d=Blockly.JavaScript.valueToCode(a,"TO",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"0",e=Blockly.JavaScript.valueToCode(a,"BY",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"1",f=Blockly.JavaScript.statementToCode(a,"DO");f=Blockly.JavaScript.addLoopTrap(f,a);if(Blockly.isNumber(c)&&Blockly.isNumber(d)&&  Blockly.isNumber(e)){var g=Number(c)<=Number(d);a="for ("+b+" = "+c+"; "+b+(g?" <= ":" >= ")+d+"; "+b;b=Math.abs(Number(e));a=(1==b?a+(g?"++":"--"):a+((g?" += ":" -= ")+b))+(") {\n"+f+"}\n")}else a="",g=c,c.match(/^\w+$/)||Blockly.isNumber(c)||(g=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_start",Blockly.VARIABLE\_CATEGORY\_NAME),a+="var "+g+" = "+c+";\n"),c=d,d.match(/^\w+$/)||Blockly.isNumber(d)||(c=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_end",Blockly.VARIABLE\_CATEGORY\_NAME),a+=  "var "+c+" = "+d+";\n"),d=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_inc",Blockly.VARIABLE\_CATEGORY\_NAME),a+="var "+d+" = ",a=Blockly.isNumber(e)?a+(Math.abs(e)+";\n"):a+("Math.abs("+e+");\n"),a=a+("if ("+g+" > "+c+") {\n")+(Blockly.JavaScript.INDENT+d+" = -"+d+";\n"),a+="}\n",a+="for ("+b+" = "+g+"; "+d+" >= 0 ? "+b+" <= "+c+" : "+b+" >= "+c+"; "+b+" += "+d+") {\n"+f+"}\n";return a};  Blockly.JavaScript.controls\_for2=function(a){var b=Blockly.JavaScript.variableDB\_.getName(a.getFieldValue("VAR"),Blockly.VARIABLE\_CATEGORY\_NAME),c=Blockly.JavaScript.valueToCode(a,"FROM",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"0",d=Blockly.JavaScript.valueToCode(a,"TO",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"0",e=Blockly.JavaScript.valueToCode(a,"BY",Blockly.JavaScript.ORDER\_ASSIGNMENT)||"1",f=Blockly.JavaScript.statementToCode(a,"DO");f=Blockly.JavaScript.addLoopTrap(f,a);if(Blockly.isNumber(c)&&Blockly.isNumber(d)&&  Blockly.isNumber(e)){var g=Number(c)<=Number(d);a="for ("+b+" = "+c+"; "+b+(g?" < ":" > ")+d+"; "+b;b=Math.abs(Number(e));a=(1==b?a+(g?"++":"--"):a+((g?" += ":" -= ")+b))+(") {\n"+f+"}\n")}else a="",g=c,c.match(/^\w+$/)||Blockly.isNumber(c)||(g=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_start",Blockly.VARIABLE\_CATEGORY\_NAME),a+="var "+g+" = "+c+";\n"),c=d,d.match(/^\w+$/)||Blockly.isNumber(d)||(c=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_end",Blockly.VARIABLE\_CATEGORY\_NAME),a+=  "var "+c+" = "+d+";\n"),d=Blockly.JavaScript.variableDB\_.getDistinctName(b+"\_inc",Blockly.VARIABLE\_CATEGORY\_NAME),a+="var "+d+" = ",a=Blockly.isNumber(e)?a+(Math.abs(e)+";\n"):a+("Math.abs("+e+");\n"),a=a+("if ("+g+" > "+c+") {\n")+(Blockly.JavaScript.INDENT+d+" = -"+d+";\n"),a+="}\n",a+="for ("+b+" = "+g+"; "+d+" >= 0 ? "+b+" < "+c+" : "+b+" > "+c+"; "+b+" += "+d+") {\n"+f+"}\n";return a}; |

　また、Blocklyの表示のためにen.jsというメッセージを管理するファイルを修正した。

|  |
| --- |
| Blockly.Msg["CONTROLS\_FOR\_TITLE1"] = "for(<=) with %1 from %2 to %3 by %4";  Blockly.Msg["CONTROLS\_FOR\_TITLE2"] = "for(<) with %1 from %2 to %3 by %4"; |

# **実験**



## 実験目的

## 実験環境

## 実験

## 実験の考察

### 実験の感想一覧

# **むすび**

# **謝辞**

　本研究を行うにあたり，多くの助言をくださった鷹野孝典教授に心から感謝いたします．

研究を一緒に進めながら様々な意見を出し合い，本研究の方向性の確立にご協力くださった鷹野研究室の４年生のメンバーに深く感謝いたします．

# **参考文献**

# その他

* ソースコード等を記述するときは，表内に挿入し，行間を「固定値：12pt」に設定する．行間を制する者はwordを制する．
* 研究でやったことはすべて，基本的に「本研究では」と称する
* ファイル名は，「1821~-name-thasis.docx」で統一する．
* 目次では、謝辞や参考文献のページ番号は載せない
* 要旨には，背景，課題，提案，実験で確認したことを書く
* 図表番号とラベルの間は半角スペースを入れる
* 節と節の間は改行する．章と章は改ページする
* 本文は日本語：MS明朝　　英語：Cenutty
* ただし数式は「挿入→数式」から入力し，本文中で数式を用いる場合はフォントの形が似ているTimes New Romanを使う．
* 付録にあるプログラムの説明がしたいとき本文の方に，例えば，“（付録３参照）”，“……を付録３に示す．”等と参 照先を記しておく．
* 章の終わりに改ページを行う．
* 改ページはCtrl+Enter
* 今後の課題ではなく展望．修正点とか書かない