C 言語の概念と実行過程を可視化する プログラミング学習用アプリケーションの開発

Development of a Web application for learning programming that visualizes concepts of C language and execution process of C programs

ネットワーク情報学部　松本絵里子，佐藤良樹，佐藤瑞帆，鈴木博登，中谷香凜，
藤木諒斗，馬淵仁輝，森美由紀，松永賢次，望月俊男

School of Network and Information　Eriko MATSUMOTO, Yoshiki SATO, Mizuho SATO,
Hiroto SUZUKI, Karin NAKAYA, Ryoto FUJIKI, Hitoki MABUCHI,
Miyuki MORI, Kenji MATSUNAGA, and Toshio MOCHIZUKI

Keywords: programming, visualization, Web application, C language

Abstract

We developed a Web-based application for learning basic concepts of programming that learners can study at their own pace. The application provides the function to visualize the concept and execution process of the C language, which allows beginners to learn programming with pleasure, when learners face examples and tasks. We conducted two classroom experiments in a high school and a university, revealing that the application is effective for learning programming in some extent, especially for beginner students who felt pleasure to learn with this application.

1. はじめに

1.1. 背景

近年，プログラミング学習は国内外で重要視されるようになってきている．わが国でも，義務教育段階でプログラミング教育を一層推進することが検討されている [1].

しかし，情報学を専攻する大学生であってもプログラミングの基礎を理解することは難しい．我々が在籍するネットワーク情報学部（以下「本学部」）の必修科目である C 言語プログラミングの授業では，基礎的な内容を扱っているにも関わらず，全体の約 3 人に 1 人の学生が単位を修得できていない.

難しいという印象がもたれるプログラミングに対して，楽しく学べるようにしようとする取り組みがいくつかみられる．例えば Tech Kids CAMP[2]という団体による，自らのアイデアを実現する力を身につける小学生のためのプログラミングスクール「Tech Kids School」がある．これは，楽しさを重視したプログラミング教育の取り組みであり，参加した子どもたちの中には，アプリケーションやゲームを製作できるまでになった例もある．また，(株)SCSK は社会貢献事業として，ワークショップを通して楽しくプログラミングを学ぶ「CAMP」という取り組みを行っている[3]．これらを踏まえれば，プログラミング学習に「楽しさ」を取り入れることは良い影響があるのではないかと考えられる．

一方，プログラミングの基礎を理解する「難しさ」についてもアプローチする必要がある．一般的な方法としては，「プログラミング言語の概念の可視化」がある．書籍や授業では，変数を箱に例えるような説明がみられる．またデバッガなどのツールには，実行環境において期待した値が変数に格納されているかを確認するために，格納された値を表示する機能を持つものがある．例えば，Mozilla Firefox の拡張機能である firebug がある．Visucuit[4]や Scratch[5]，LEGO® Mindstorms[6]などは実行過程をアニメーションやロボットの動作で確認できるようになっている．

しかし，プログラミング未経験者向けに，実行過程のみならず，「プログラミング言語の概念」をゆっくりと丁寧に可視化して学習できるものは少ない．「変数への値の代入」などの，プログラミング言語上の概念をゆっくりと丁寧に，その挙動を含めて可視化することは，プログラミング未経験者の学習にとって有用と考えられる．

そこで我々は，プログラミング未経験者でも理解できるように，プログラミング言語上の概念を可視化し，ゆっくりと丁寧に楽しくプログラミングの基礎を学習できる自学自習支援の仕組みを提案することとした．

自学自習に焦点を当てるのは，さまざまなレベルの学習者が自分のペースで学習できることが，よりよいプログラミングの理解につながると考えるからである．

1.2. 目的

本研究では，「C 言語の概念とプログラムの実行過程をゆっくりと丁寧に可視化し，楽しく学習できる Web アプリケーション」を開発し，その学習効果を検証することを目的とする．これをさらに進めて，C 言語プログラミングの自学自習を支援することが最終目標である．

ユーザが学習するプログラミング言語に C 言語を選定した理由は，C 言語は初めて学習するプログラミング言語に選択されることが多く，また本学部の必修科目でも採用しており，我々が十分に理解しているからである．

1.3. 論文の構成

本稿の構成は以下のとおりである．第 2 章ではプログラミング学習に関する調査をもとに，我々の開発する Web アプリケーションのコンセプトについて論ずる．第 3 章では開発した Web アプリケーションの仕様と実装方法，第 4 章ではアプリケーションによる効果の検証，第 5 章ではその考察を論じ，最後に結論をまとめる．

2. コンセプトの設定

2.1.プログラミングの苦手意識に関するアンケート

プロラミングの学習上の困難を明らかにし，開発する Web アプリケーション内でどのような工夫をしたらよいか考察するため，プログラミングの苦手意識に関するアンケートを 2014 年 5 月～6 月に実施した．対象はネットワーク情報学部 1 年次の必修授業である「プログラミング入門」を受講したことのある，本学部の 2 年生と 3 年生であり，計 390 人が回答した．

アンケートは，「プログラミングすることについてどのように考えているか」について 5 段階（とても好き～とても嫌い），「プログラミング入門」で扱った分野（変数，代入，配列等）について 4 段階（とても難しい～とても簡単）のリッカートスケールで質問した．

アンケート結果を集計・整理し，「プログラミング嫌いは，苦手な分野や学習活動が影響している」という仮説を立て，この仮説を重回帰分析で検証した．

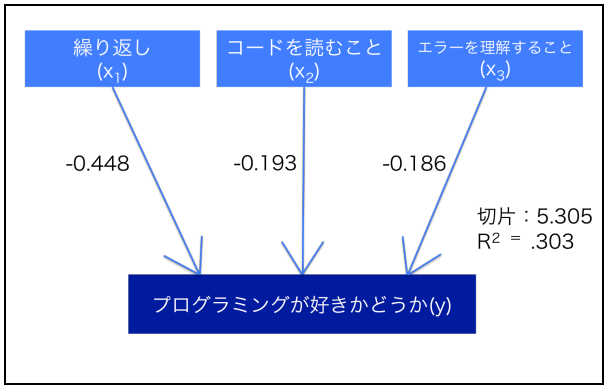

図 1　重回帰分析の結果

重回帰分析の結果を図 1 に示す．得られた結果から，プログラミングが嫌いな人は「繰り返し（for,while 等）」「コードを読むこと」「エラーを理解すること」が難しいと思っていることがわかった．これらの「プログラミングが嫌いであることと関係のある要因」をできる限り取り除くことでプログラミングを好きになってもらおうと考えた．最も係数が高い「繰り返し（for,while 等）」は，プログラミングの学習事項であるため，これは難易度をできる限り下げることとし，「コードを読むこと」「エラーを理解すること」の二つに着目してさらに検討した．

その結果，コードを読めない原因として「変数の値の変化などの，C 言語上の概念を想像できていない」，エラーを理解できない原因として「どこで，なぜエラーが起こっているのかわからない」といったことが挙げられた．

以上のことから，C 言語上の概念とプログラム実行過程をわかりやすく可視化することがプログラミング未経験者にとって有効と考えられる．

2.2. 先行事例の調査から得られたコンセプト

先行事例を調査し，参考にして取り入れたいと考えた部分や，反対に避けようと考えた要素をまとめ，以下に挙げる 7 つのコンセプトを決定した．

（Ⅰ）無料　（Ⅱ）日本語

有料や英語の教材は，自学自習する上でハードルになると考え，無料であること，日本語表記であることを最低限の条件とした．

(Ⅲ)失敗から学べる

アルゴロジック[7]はロボットを移動させて画面内の全ての旗を回収するゲームである．失敗してもキャラクターが移動した軌跡が残るため，どこで間違ったかがすぐにわかり，その部分を修正して，またすぐに実行できる．このように失敗から学べることは重要であると考えた．

(Ⅳ)目標が明確

Scratch[5]や LEGO ® Mindstorms [6]は，ユーザの自主性や創造性を重んじており，具体的な目標は提示されない．そのため，実現したいことがないと何をすればよいかわからず学習に取り組みにくい．

一方，アルゴブロック[8]やアルゴロジック，CODEPREP[9]，Codecademy[10]では，問題が与えられ，目標が明確である．「目標を達成するために学習する」というのは学習者にとって良い動機づけになると考えた．

(Ⅴ)親しみやすいキャラクター

Scratch の猫のキャラクターのように，親しみやすいキャラクターがいるだけで堅苦しさが緩和され，親しみを感じてもらいやすくなる．実際，CodeCombat[11]を体験した学生から，「キャラクターがユーザに語りかけてくることによって世界観に入り込みやすく感じた」という意見がみられた．

(Ⅵ)ゲーム性

CodeCombat では，RPG のように敵を倒して物語を進めていく形式で学習が進む．このような仕組みにより，前項で述べたキャラクターとあわせて世界観に入り込みやすくし，楽しく感じられるようになっていると考えた．

(Ⅶ)段階的

CODEPREP や Codecademy は，ユーザに基礎から順番に問題を解かせ，正解しないと次に進めない仕組みになっている．このような段階的に学習させる仕組みは，ユーザが学習内容を理解しないまま進めることを防ぎ，学習に効果的だと考えた．

3. アプリケーションの開発

3.1. アプリケーションの構成

本アプリケーション「C 言語を学ぼう！～怪盗 C からの挑戦状～」は，ユーザが「ストーリー」「お手本」「問題」を繰り返すことで学習できる(図 2)．

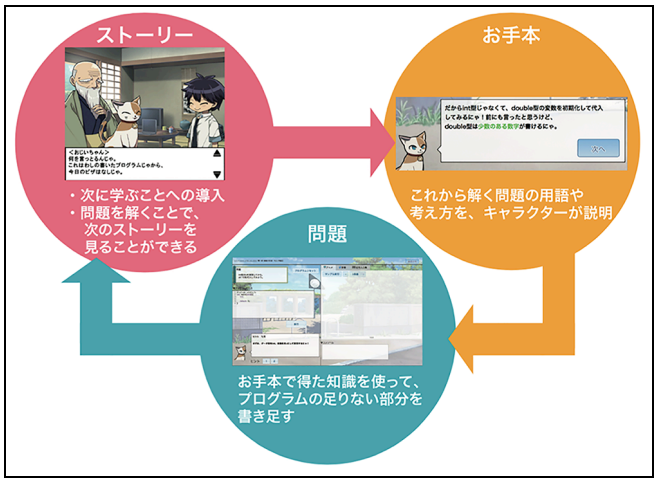

図 2　ユーザの学習の流れ

(Ⅰ)ストーリー

物語を見ながらお手本や問題の学習内容を知ることができる．これはモチベーションを高める狙いがある．

(Ⅱ)お手本

サンプルコードが画面に表示され，C 言語上の概念とサンプルコードの実行過程を可視化したアニメーションとコードの説明を同時に見て学習する．アニメーションによる感覚的な理解と，言葉の説明による知識的な理解の両方ができるようになっている．

(Ⅲ)問題

実際にユーザがプログラムを入力して動作させる．問題は穴埋め形式になっており，問題文に沿ったコードを完成する．問題をクリアしないと次に進めない仕組みになっており，ユーザは順を追って学習することになる．

本アプリケーションには 5 つの章があり，単元ごとに分かれている．第 1 章「導入」から始まり，「変数」「条件分岐」「繰り返し」「配列」を各章で扱っている．

3.2. 各画面の機能役割

メニュー画面(図 3)は，コンセプトの「段階的」の箇所でも述べたように，ユーザが内容を理解しないまま次の問題へ進んでしまうことを防ぐ仕組みになっている．各ユーザの学習の進行度を記録しており，各章の問題をクリアすると目印として猫の足跡がボタンの横に表示され，ユーザは次のお手本や問題に進むことができる．

ストーリー画面(図 4)は，お手本画面に遷移する前に必ず表示される画面である．ここでは物語を見ながら，次のお手本や問題で学ぶ内容を知る．

画面右下部にある三角形のボタンでセリフの戻る/進むの操作をし，物語が終わると次のお手本画面に進める

図 3　メニュー画面

図 4　ストーリー画面

図 5　お手本画面

ようになる．また，ユーザが以前にクリアしたお手本を復習しようとした際は，一度見たことがあるストーリー画面を再度見ることになるので，その対策として，画面右上部にストーリーをスキップできるボタンを設けた．

ストーリー画面で学習内容の導入を済ませた後，お手本画面(図 5)で本格的に学習する．図中⑤には予めコードが用意されており，ユーザは画面左下の説明(③)と，コード，C 言語上の概念のアニメーション(②)を見ながら学習する．アニメーションによる感覚的な理解と，言葉による知識的な理解の両方ができるようにしている．その他を含め画面上の構成要素を以下に述べる．

①例題文

この画面で扱う例題が表示される．

②アニメーション

予め用意されたコードに含まれる C 言語上の概念や実行過程が可視化され，アニメーションとして表現される．

③説明

実行中のコードとそのアニメーションの解説を見られる．「次へ」ボタンを押すことでコードに関するアニメーション又は次の説明が表示される．

④ヘルプ(画面説明)

お手本画面の使い方を見ることができる．

⑤エディタ

お手本画面では予めコードが埋め込まれており，ユーザが書き込むことはできない．アニメーション実行中には，該当する行にハイライトが当てられる．

⑥速度調整/一時停止・再開

アニメーションの速度を変更することができる．初期状態では 1 倍速であり，0.5 倍，2 倍，4 倍，10 倍速に変更できる．アプリケーション後半の繰り返し(for 文，while 文)などでアニメーションが長くなってしまう場合に早回しで学習したり，逆にじっくり確認するために遅回しや一時停止をして学習したりすることができる．

⑦コンソール

一般的な実行環境でプログラムを動かすときのように，入出力を行う場所である．お手本画面ではユーザが入力を行うことはなく，出力の確認用として使用する．

図 6　問題画面

問題画面(図 6)では実際にユーザがコードを書く．この画面では，ユーザがつまずかないように，以下に挙げる 11 の構成要素を用意した．

①問題文

この画面で扱う問題が表示される．

②サンプル実行

問題に正解した場合のアニメーションを見ることができる．問題文からだけではなく，アニメーションからも正解のコードを考えられることを狙いとしている．

③アニメーション

ユーザが書いたコードに含まれるC言語上の概念や実行過程をアニメーションとして可視化表現される．

④ポイント

クリックすると問題のヒントとなる文章を参照できる．

⑤辞書

クリックすると問題を解くときに参考となるプログラミングに関する辞書を見ることができる．問題をクリアしていくと辞書に書いてある内容も増えていく．

⑥記号入力集

記号の入力の仕方を一覧で見ることができる表が表示される．

⑦ヘルプ(画面説明)

問題画面の使い方を見ることができる．

⑧エディタ

ユーザがコードを入力する場所である．

⑨プログラムリセット

コードを初期状態に戻すことができる．

⑩速度調整/一時停止・再開

お手本画面と同様にアニメーションの速度を変更することができる．

⑪コンソール

実際のコンピュータ上でプログラムを動かすときのように，ここでscanf文やprintf文などに応じた実行時の入出力を行う．

3.3. C言語の概念とプログラム実行過程の可視化

本アプリケーションの最大の特徴は「C言語の概念とプログラム実行過程の可視化」である．一般的な実行環境とは違い，コンソールに文字列だけを出力するのではなく，アニメーションを使用し，コードのどの行が何の処理を行っているのかを表現する．それにより，ユーザは書いたコードが意図しない動きをした場合，どこで間違えたのか，どこを修正すればよいかを確認できる．

アニメーションの表示の具体例を，代入式の場合をもとに示す．図7のコードの4行目「x=x*3」は，数学における等式と似た書き方ではあるが，プログラミングでは代入式を表している．xの値が5の場合，この代入式は，まず右辺が評価されることで「x*3」，すなわち「5*3」が計算され，その結果である15が左辺のxに代入される．

この概念を数学における等式と混同せずにユーザに理解してもらうため，本アプリケーションでは変数を表すキャラクター(図8)を用意した．キャラクターの頭部に書かれた「x」は変数名，ホワイトボードに書かれた「5」は変数の値を意味する．図9～図10のようなキャラクターのアニメーションによって代入を表現する．また同時に，図7のように，現在アニメーションで表現されている部分を表すために，コードの実行箇所をハイライトして強調している．

このようにして，変数の格納する値が変化していく様子などをゆっくりと丁寧に確認できるようにしていることが，本アプリケーションの最大の特徴である．

```
1 #include <stdio.h>
2 int main(void){
3     int x = 5;
4     x = x * 3;
5     return 0;
6 }
7
```

図7　コード例および実行部分を強調する様子

図8　変数xに5が格納されている様子

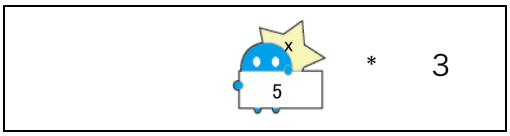

図9　変数xを利用してx*3の計算をしようとする様子

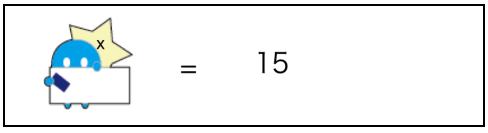

図10　15を変数xに代入しようとする様子

3.4. 実装方法

本アプリケーションの主な開発言語はHTML，CSS，PHP，JavaScriptである．これらを使用した理由は，本アプリケーションをなるべく多くの人に使ってもらえるようWebブラウザ上で動作するものにするためである．ユーザ管理・学習進行度の記録にはMySQLを用いた．

コードの実行過程をアニメーションによって表現する機能は，ユーザが入力したコードを Javascript で開発した構文解析機にかけ，コードに含まれる変数宣言や演算処理を判別し，意味解析を行い，その後に実行すべきアニメーションを Javascript で生成し，そのコードを実行することで実現している．

4. 実証実験

本アプリケーションがプログラミング学習に役立つかを検討するため，二つの実証実験を行った．それらの目的は，1 つは楽しく学習できるかの効果検証，もう 1 つは自学自習だけで進められるかどうかの検証である．

4.1. 学習効果の検証

2015 年 9 月～10 月に C 県立 S 高等学校 1 年生の普通科 6 学級 240 名，情報コミュニケーション科 1 学級 40 名，計 280 名を対象に，情報の授業の一環として実証実験を行った．

4.1.1. 手続き

4.1.1.1. 事前調査

第 1 回の授業時にプログラミングに対する意識調査(以下「事前アンケート」とする)を行った．内容は「プログラミングをやったことがあるか」「プログラミングに対して興味があるかどうか」の 2 項目であった．

4.1.1.2. 授業の進め方

授業は計 13 回行われた．初回授業には我々が高校へ赴き，本アプリケーションの登録の仕方や進め方を生徒に説明した．その後の授業は本アプリケーションの進め方や問題の解き方について教員からの指導を受けることなく，生徒が自分の力で進めていく形式で行った．授業時間以外でも問題を解くことは可能であり，自宅での自習は許可されていた．また，生徒がいつでも書き込める掲示板を設け，質問を受け付けた．

4.1.1.3. 事後調査

授業の最終日では，学習到達度を調査するためのテストと，テスト後に「本アプリケーションが楽しかったかどうか」「本アプリケーションが難しかったかどうか」の 2 項目について，5 段階リッカートスケールで問うと同時に，理由を自由記述で質問するアンケート(以下「事後アンケート」とする)を実施した．テストは本アプリケーションの内容に対応した 34 問で構成され，1 問あたり配点 1 点，合計 34 点満点のテストであった．このテストは学習効果を検証するためのものであり，テストの出題内容に対応する本アプリケーションの箇所(以下「アプリケーションの内容との対応」とする)まで進んだ人がその問題を正解したかどうかを，本アプリケーションの学習効果の指標とした．テストの出題内容と出題数，アプリケーションの内容との対応を表 1 に示す．

表 1　テストの出題内容とアプリケーションとの対応

学習内容とテストの出題内容	出題数	アプリケーションの内容との対応
問 1《変数の宣言と出力》変数と printf を用いて出力を行う問題	6	第 2 章 2 節 問題 2
問 2《変数の入出力》 scanf で数値を入力し，その数値を使って計算を行い，結果を printf で出力する問題	4	第 2 章 4 節 問題 2
問 3 《条件分岐(if)》 if 文を使って，条件に合った入力の場合のみ出力を行う問題	2	第 3 章 1 節 問題 4
問 4《条件分岐(else, elseif)》 条件分岐を用いて閏年を判定する問題	5	第 3 章 3 節 問題 1
問 5《繰り返し(for)》for 文を使って 1 から入力された数字までの和を出力する問題	3	第 4 章 1 節 問題 2
問 6《繰り返し(while)》while 文を用いて「12345」と出力する問題	4	第 4 章 2 節 問題 2
問 7《二重ループ》出力結果に応じた二重ループを扱う問題	3	第 4 章 3 節 問題 1
問 8《一次元配列》配列の初期化とアクセスを行う問題	2	第 5 章 1 節 問題 2
問 9《二次元配列》出力結果に応じた二次元配列を扱う問題	5	第 5 章 2 節 問題 2
合計	34	

4.1.2. 結果

4.1.2.1. 事前アンケート

事前アンケートの結果を表 2 に示す．プログラミング経験者(ある・少しあると回答した生徒)は全体の 26.3%であった．また，全体的にプログラミングに興味がある生徒(ある・少しあると回答した生徒)が全体の 69.5%と多く，少し興味があるという回答が最も多かった．

事前のプログラミング経験とプログラミングへの興味が，普通科と情報コミュニケーション科で大きな違いがないかどうかを検討するため，プログラミング経験の有無，プログラミングの興味の有無について肯定的回答と否定的回答をまとめ，それぞれフィッシャーの直接確率法を行ったところ，有意な差がみられた(プログラミング経験：p <.05，プログラミングの興味：p <.01)．このこ

とから普通科と情報コミュニケーション科の生徒の事前経験や事前の興味は異なっており，情報コミュニケーション科の方がプログラミングの経験者が相対的に多く，興味があると答えた生徒が相対的に多かった．

表 2　事前アンケートの結果

		ある	少しある	ない
プログラミング経験	普通科	8	49	182
	情コミ科	0	16	23
	合計	8	65	205
プログラミングへの興味	普通科	29	129	81
	情コミ科	10	25	4
	合計	39	154	85

4.1.2.2.事後調査

(1) アンケート結果の概要

事後アンケートの結果を表 3，表 4 に示す．表 3 を見ると半分以上が楽しかったと答えているものの，表 4 を見ると半分以上が難しいとも答えている．そこで，楽しさと難しさの相関を検討するため，この 2 項目についてスピアマンの相関係数を求めた．結果は r_s = -.346 であった．散布図(図 11)を見ると，つまらないと答えている人で簡単だったと答えた人はいなかったが，楽しいと答えている人には難しかったと答えた人もみられた．

なお，プログラミングの経験の有無で楽しさや難しさについて差があるかどうかを検討するため，t 検定を行ったところ，どちらも有意差はみられなかった(楽しさ(t (94) = 1.19, *n.s.*)，難しさ(t (90) = -0.89, *n.s.*))．

表 3　アプリケーションが楽しかったかどうか

	人数
とても楽しかった	22
楽しかった	120
どちらともいえない	74
つまらなかった	17
とてもつまらなかった	6

表 4　アプリケーションが難しかったかどうか

	人数
とても簡単だった	1
簡単だった	5
どちらともいえない	32
難しかった	125
とても難しかった	76

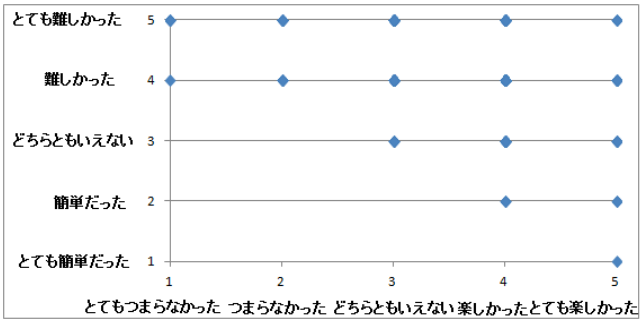

図 11　楽しさと難しさの散布図

(2) 学習進行度の結果の概要

本アプリケーションの学習進行度(図 12)を見ると，普通科と情報科について進行度には大きな差はみられなかった．条件分岐(else, elseif)までは全体のほとんどの生徒が進められているが，繰り返し(for)以降減少していき，ほとんどの人が最後までたどり着くことができていない．とくに条件分岐(else, elseif)と繰り返し(for)の間，二重ループと一次元配列の間の差が大きくみられる．

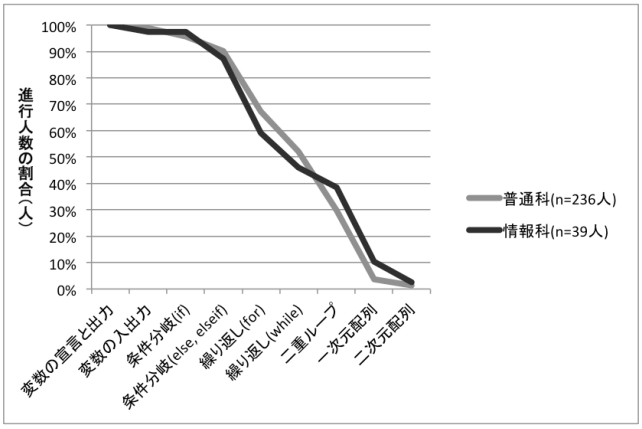

図 12　アプリケーションの進行度

(3) テスト結果の概要

全体の平均点は 16.7 点(SD = 4.47)であり，普通科は 16.8 点(SD = 4.44)，情報コミュニケーション科は 16.0 点(SD = 4.66)で普通科の方がやや高い傾向がみられたが，その差は有意ではなかった(t (276) = 1.02, *n.s.*)．

また，テストの平均点はプログラミング経験者が 17.02 点(SD = 4.37)，未経験者が 16.59 点(SD = 4.51)であったが，その差は有意ではなかった(t (130) = 0.73, *n.s.*)．したがってプログラミングの事前経験の有無が以後テストの成績に与える影響は大きくなかったといえる．

表 3 にそれぞれの問題の全体の平均正答率を示す．この表をみると，問 3 までは正答率が 70%以上と高いが，

問 4 からは徐々に正答率が下がっており，問 7 からさらに大幅に正答率が下がっている．

(4) 学習進行度と学習期間の関連

(2)の結果から，進行度に差ができてしまったのは学習を進めるための時間が不足したことによる可能性を考え，週ごとに本アプリケーション内で出題した各問題を解答した人数を調べることで，学習の進行度を確認した．

図 13 を見ると時間が進むごとに山が等間隔で移動しており，十分な時間が取れていればこの山が最後まで移動することが予想される．このことから，もう 1 週間あるとより学習が進行したのではないかと考えられる．

しかし，5 週目の解答人数を見てみると問 37 を境に人数が大きく減っている．問 37 は二重ループを扱うまとめ問題であり，この問題の難易度が他の問題よりも高いため，解くことができなかった人が多かったおそれがある．

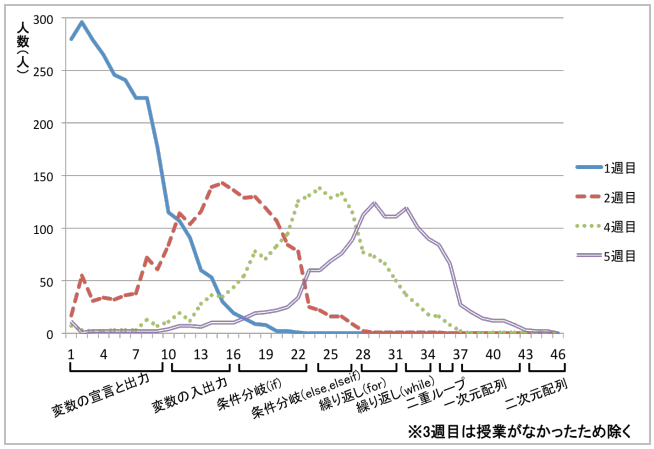

図 13　週ごとのアプリケーションの問題の解答人数

(5) 事前の興味，経験と学習の進行との関連

事前の学習経験が学習の進行にどの程度関連しているかを調べるため，プログラミング経験と学習の進行度を集計した．図 14，図 15 がその結果である．

進行度別の事前アンケートの結果を見ると，プログラミング経験が「ある」と答えた人は，「ない」と答えた人と比べて学習を進められた人の割合が多いことがわかった．とくに経験があることで優位に進行できたのは「繰り返し(for)」の分野までであり，繰り返し(while)以降は進行人数の割合が低減している．プログラミングに興味が「ある」と答えた人も，「ない」と答えた人と比べて本アプリケーションによる学習を進められた人の割合が多いことがわかる．また経験と興味どちらとも「ない」と「少しある」では結果に差がないことがわかる．

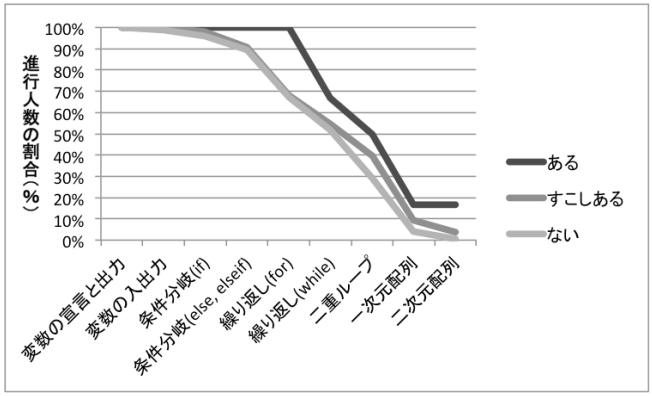

図 14　プログラミング経験の有無と学習進行度

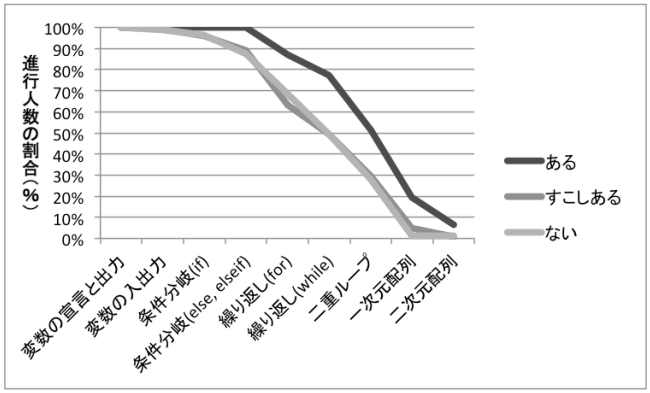

図 15　プログラミングへの興味と学習進行度

(6) 学習進行度とテスト結果の関連

本アプリケーションの進行具合には個々人に差があり，表 5 の平均正答率には，テストの問題に対応する箇所まで学習を進められなかった生徒の数値も含まれている．そこで，テストの対応箇所まで学習を進められた生徒とそうでない生徒別にテストの対応箇所別の平均正答率を出した(図 16)．

図 16 によると，アプリケーションとの対応箇所まで学習を進められた人の方が正答率は高く，学習を進めることでプログラミングの理解を深められたのではないかと考えられる．対応箇所まで進められた人の平均正答率は，「二重ループ」「一次元配列」以外の問題は 50%を超えている．したがってそれぞれの単元の学習内容を学んだ生徒にとっては学習効果が相応にあったといえる．しかし「二重ループ」「一次元配列」の問題は 50%を下回っており，十分な学習効果を得られたとはいえないだろう．

表 5　平均正答率

問 1	問 2	問 3	問 4	問 5	問 6	問 7	問 8	問 9
75.9%	71.0%	86.7%	57.3%	49.9%	45.0%	21.0%	17.5%	8.8%

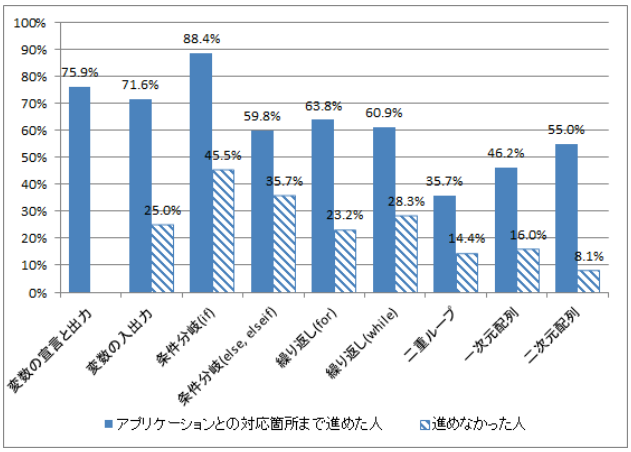

図 16　進行度別テスト平均正答率

(7) 楽しさ・難しさと学習進行度の関連

次に，アプリケーションを進行しているときに感じた楽しさや難しさが，学習の進行に影響を与えているのではないかと考え，(5)と同様に楽しさや難しさ別に学習進行度をグラフにまとめた(図 17，図 18)．

これらを見ると「楽しかった」と答えた人は，「つまらなかった」「どちらともいえない」と答えた人と比べて学習を進めている割合が多かった．「つまらなかった」と答えた人は繰り返しでつまずいている人が多かった．また，「簡単だった」と答えた人は，「難しかった」と答えた人と比べて学習を先に進めている人の割合が多い．「簡単だった」と答えた 5 人は，感想を求めた自由記述で全員が「ヒントやお手本があったから簡単だった」と答えていた．一方で「難しい」「とても難しい」と感じていた人は「ヒントやお手本が足りなかった」と答えている人がみられた(201 人中 23 人)．ヒントやお手本の活用は，学習を進める上で有効な方略と考えられる．

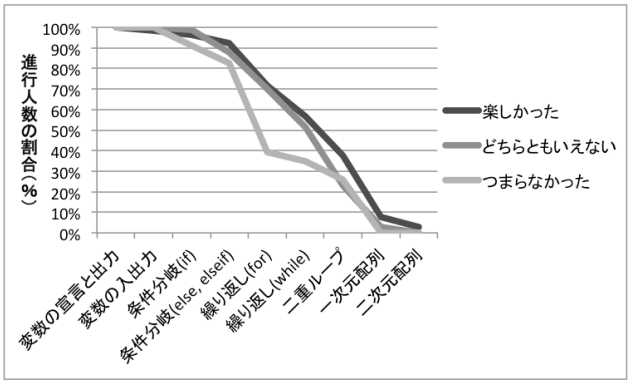

図 17　アプリケーションの楽しさと学習進行度

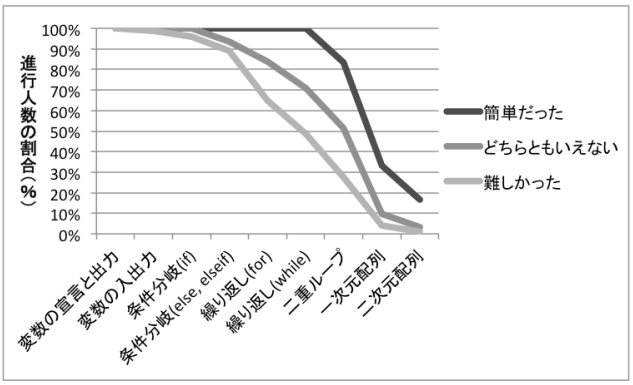

図 18　アプリケーションの難しさと学習進行度

(8) 楽しさ・難しさとテスト結果の関連

次に，テストの結果と事後アンケートの結果に関係があるのではないかと考え，学習中に感じた楽しさや難しさとテストの結果と関連を調べた(表 6)．進行度と同様に「楽しかった」「簡単だった」と答えた人ほどテストの点数が高いことを示している．

表 6　楽しさと難しさによるテスト得点の違い

楽しさ	平均点(SD)	難しさ	平均点(SD)
とても楽しかった	18.23 (4.90)	とても簡単だった	21.00 (0.00)
楽しかった	17.41 (4.46)	簡単だった	22.60 (2.88)
どちらともいえない	16.04 (4.28)	どちらともいえない	19.06 (4.33)
つまらなかった	15.76 (3.85)	難しかった	17.41 (3.97)
とてもつまらなかった	12.83 (5.49)	とても難しかった	14.50 (4.44)
無回答	15.95 (4.22)	無回答	15.95 (4.22)

楽しさや難しさ，事前のプログラミング経験がテストの成績に及ぼす影響について検討するため三要因の分散分析を行うこととした．楽しさの「とてもつまらなかった」，難しさの「とても簡単だった」の回答者が少なかったため，「どちらともいえない」の中立的回答と，肯定的回答，否定的回答に分けて検討した(表 7)．その結果，難しさの主効果が有意であった($F(2, 238) = 5.04$, $p < .01$, $\eta_p^2 = .042$)(図 19)．Bonferroni の方法で多重比較したところ，「簡単だった」生徒の点数が「難しかった」生徒の点数に比して有意に高くみられた．

表 7　分散分析表（* p<.05, ** p<.01）

	自由度	平方和	平均平方	F
プログラミング経験	1	22.38	22.38	1.25
楽しさ	2	56.98	28.49	1.59
難易度	2	180.49	90.25	5.04**
プログラミング経験×楽しさ	2	135.20	67.61	3.78*
プログラミング経験×難易度	2	53.61	26.80	1.50
楽しさ×難易度	1	60.67	60.67	3.39
誤差	228	4083.10	17.91	
全体	238			

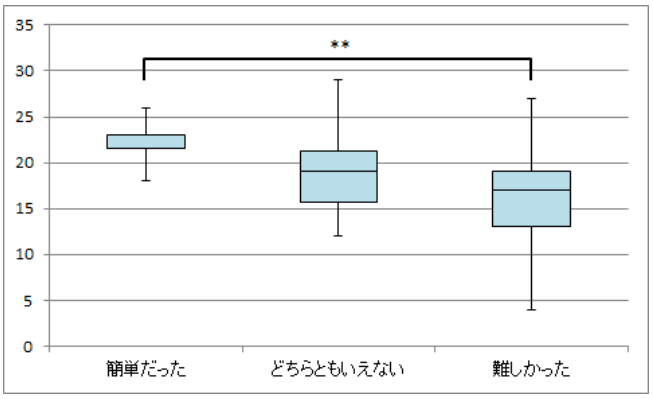

図 19　難しさによるテスト得点の違い(** p<.01)

また，プログラミング経験と楽しさとの 1 次交互作用が有意であった($F(2, 238) = 3.78$, $p < .05$)(表 8)．そこで単純主効果の検定を行ったところ，プログラミング経験のない場合に有意な主効果がみられた($F(2, 228) = 10.69$, $p < .01$, η_p^2=.086)．Bonferroni の方法で多重比較したところ，「楽しかった」と感じた方が「どちらともいえない」「つまらなかった」と感じるよりも得点が有意に高いことがわかった(表 8)．したがってプログラミング経験のない場合に，アプリケーションを楽しく学習できたかどうかが有意にテストの成績に影響することがわかった．

したがって，総じて簡単と感じられた生徒ほど点数が高く，またプログラミング経験のない場合には楽しく学習できた生徒ほど得点が高くなることがわかった．

表 8　プログラミング経験と楽しさの交互作用

プログラミング経験	楽しさ	平均点(SD)
あり	楽しかった	19.51(1.01)
	どちらともいえない	17.75(1.06)
	つまらなかった	16.83(1.73)
なし	楽しかった	19.86(0.89)
	どちらともいえない	14.86(0.99)
	つまらなかった	14.35(1.03)

4.1.3. 考察

まず，総じて本アプリケーションで学習することに一定の効果があるが，「二重ループ」と「一次元配列」の分野では十分な学習ができなかったといえよう．したがって，それらの分野を学びやすくする可視化方法やヒントの与え方を検討する等の改善が必要である．本アプリケーションを簡単に感じた生徒ほどテストの点数が高く，そのように感じるような教材の充実が必要である．

一方，本アプリケーションを開発するにあたって重視した「楽しさ」に関してみると，とくにプログラミング経験がない人にとっては「楽しい」ということが学習に有効であったことがわかった．本アプリケーションはプログラミング初心者を対象として開発しており，この点に関しては有効だったといえよう．

学習の進行に関してみると，プログラミングに興味がある生徒ほど学習を先に進められており，プログラミングへの興味は学習を進める上で大きな要因であると考えられる．一方，プログラミング経験があればよいかというとそうではなく，学習の前半は経験者の方が先に進むが，後半になるにしたがってその他の学習者との差がなくなってくる．

したがって，プログラミングへの興味の有無が本アプリケーションで学習を進める上で重要だと考えられる．

4.2. 自学自習効果の検証

4.2.1. 手続き

自学自習の効果を検証するために，2015 年度にネットワーク情報学部の「プログラミング演習 2」の履修者に対して調査を実施した．まず，我々が授業に訪問して学生 270 人に対して本アプリケーションに関する説明を行った．本アプリケーションは授業時間外の自学自習用としてプログラミング学習に役立つものとして説明した．また，ユーザに本アプリケーションに自分でユーザ登録する方法を教示した．その後，1 ヶ月後に授業支援システムを用いて本アプリケーションによる学習を促す説明

を行った．ただし，学生に対して強制はせず，自分の意志でユーザ登録し取り組んでもらうことを前提とした．また，S高校での検証時と同様に，9月18日から12月14日まで質問用の掲示板を設けた．また，我々のメールアドレスが記載された紙を配布し，メールでの質問も受け付けた．

4.2.2. 結果

図20が登録したユーザの進行度別の人数を示した図である．12月15日の段階で26人が登録していた．

まず，履修者数270人に対して登録人数が26人であり，約10%と少ない人数となった．さらに，26人のうち半数の13人がユーザ登録後の初めの章で進行が止まった．第1章ではプログラミングがどういうものかを説明するのみの内容となっており，半数は本アプリケーションの内容に踏み込まずに終了してしまった．

また，本アプリケーションの本編に入り，「変数の宣言と出力」まで学習した学生は26人中5人にとどまっており，「二次元配列」まで到達しアプリケーションの内容を完了できた学生は1人だけであった．

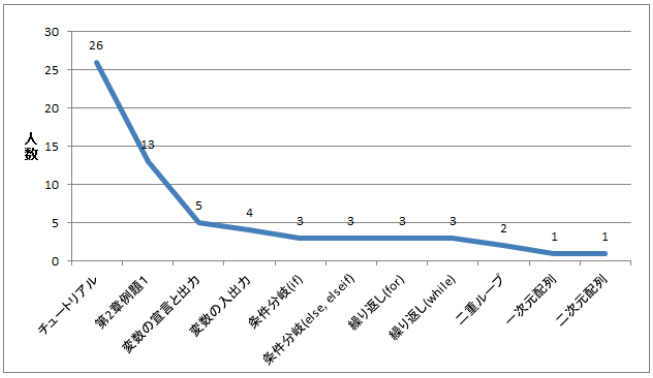

図20　登録したユーザの進行度別の人数

5. 総合考察

学習効果の検証については，アプリケーションの内容との対応箇所まで学習を進めることができた人の方が，テストの正答率が高く，本アプリケーションで学習をすることでプログラミングの理解を深めることができるとわかった．以上から，本アプリケーションには一定の学習効果があると考えられる．また，プログラミング未経験者には，とくに本アプリケーションで楽しく学習できることで，よりよい理解が得られることも示唆された．

一方，学習を進める上では，プログラミングの経験よりもプログラミングへの興味が重要であることが示唆された．授業内で学習を進めることができたS高校では，学習がある程度進んだが，完全に自学自習をすることを促した大学生の場合には，多くの人が進められなかった．これには，いくつかの要因が考えられる．

第一に，自学自習そのものが学習者にとって非常に難しいということである．eラーニングに関する諸研究が示すとおり[12]，学習者が単独で学習を継続するのには自己調整学習と呼ばれる認知機能を働かせ，強い目的意識と動機づけを持つだけでなく，自己の学習に責任を持ち主体的に時間を配分し，自分に効果的な学習方略を働かせることが必要である．S高校では，学習時間を割り当てるのに対面の授業が機能していたと考えられるが，大学生対象の実証実験でみられたようにWebアプリケーションだけで自学自習を促すことは相応のハードルがある．

第二に，学習者のプログラミングに対する興味である．今回大学生には事前調査を行わなかったが，S高校の生徒への調査では，事前にプログラミングに対して興味を持っていることが，本アプリケーションを使った学習の進行に効果的であることが示唆された．Webアプリケーションの改善の方向性としては，学習者の興味をより駆り立てるようなチュートリアルの実装などが少なくとも必要であろう．一方，学習者のプログラミングへの興味関心を学習の事前にいかに高めるかということは，本研究の範疇を超えるが，プログラミング学習環境をデザインする上で，今後の大きな課題である．

6. まとめ

本研究では，C言語上の概念をゆっくりと丁寧に可視化し，楽しく学習できるWebアプリケーションを開発し，学習効果と自学自習に適しているかを検証した．

高等学校での実証実験の結果からは，一定の学習効果はあることが確認できた．また，プログラミング経験がない人にとっては「楽しい」ということが学習に良い影響を与えるという結果が得られた．

大学生に対して行った「自学自習に適しているか」の検証では，本アプリケーションだけでは目指していた効果は得られなかった．これらの原因を考察し，今後の課題として，以下の2点を挙げた．

① 正答率の低くなる分野の内容を改善し，学習をよりスムーズに行えるようにすること．

② ストーリーおよびチュートリアルを改善し，学習を継続できるように興味関心を駆り立てる工夫を行うこと．

謝辞

本研究に際して，学習効果の検証に協力してくださった，千葉県立袖ヶ浦高等学校の眞山和姫先生，永野直先生ほか諸先生方，並びに生徒のみなさま，千葉県立八千代東高等学校の谷川佳隆先生と生徒のみなさま，専修大学附属高等学校の服部竜也先生と生徒のみなさまには，大変お世話になりました．深く御礼を申し上げます．

また，本研究を始める際の先行事例の調査にてご協力いただいた，茨城大学の鈴木栄幸先生，放送大学の加藤浩先生，(株)SCSK CAMP のみなさま，本学部 OB の菊池裕史さんからは，貴重なお話を聞かせて頂くことができました．心より感謝致します．

付記

このアプリケーションは2016年1月現在,以下のURLで公開している．

http://www.ne.senshu-u.ac.jp/~proj26-19/kaitoC/

参考文献

[1] 内閣府（2013)「日本再興戦略- JAPAN is BACK-」，〈http://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/saikou_jpn.pdf〉，（参照 2016-1-11）

[2] Tech Kids CAMP, 〈http://techkidscamp.jp/〉, (参照 2015-12-21)

[3] 森秀樹，北川美宏，向田順子，村田香子，市橋由紀，田村拓（2004)「CAMP ワークショップの開発と実践」日本教育工学会第 20 回大会講演論文集，505-506

[4] 原田康徳（2015)「ビジュアルプログラミング言語 ビスケット(Viscuit)の紹介」，コンピュータソフトウェア， 32(1)，18-26

[5] マサチューセッツ工科大学　メディアラボ「Scratch」，〈https://scratch.mit.edu〉, (参照 2015-12-14)

[6] LEGO (n.d.)「レゴ®マインドストーム」，〈http://www.lego.com/ja-jp/mindstorms〉, (参照 2015-12-14)

[7] 電子情報技術産業協会　情報・産業社会システム部会(2013)「アルゴロジック」, 〈http://home.jeita.or.jp/is/highschool/algo/prm〉, (参照 2015-12-14)

[8] 鈴木栄幸，加藤浩（1995)「共同学習のための教育ツール『アルゴブロック』」，認知科学，2(1)，36-47

[9] 株式会社ギブリー (n.d.)「CODEPREP」，〈https://codeprep.jp〉, (参照 2015-12-14)

[10] Codecademy(2015) 「Codecademy」，〈https://www.codecademy.com>, (参照 2015-12-14)

[11] CodeCombat(2015)「CodeCombat」，〈https://codecombat.com〉, (参照 2015-12-14)