Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

# Haskell によるパーサーを用いた C 言語用 Blockly プログラム生成の自動化

山形悠人[†1]　香川考司[†2]

**概要**：プログラミング学習の特に初期段階では，プログラミング言語の文法に慣れていないためのミスが発生する．これを防ぐために，Blockly を用いた C 言語学習支援環境を実装した．しかし，この学習支援環境は Blockly のブロック定義の作成，Blockly のプログラムの GUI での作成に時間がかかるという問題点がある．本研究ではこの問題点を解決するため，Haskell によるパーサーを用いて C 言語のソースコードから Blockly のブロック定義と Blockly のプログラムに対応する XML 作成の自動化を行う．指導者が初学者に提示する練習問題の作成に使用するだけでなく，初学者がブロックベースからテキストベースの学習に移行する際に理解度の確認を行うためにも使用することができる．

**キーワード**：プログラミング学習，C 言語，Blockly，学習支援，Web ベース

# Automatic Generation of Blockly Programs for the C Language using a Haskell Parser

YUUTO YAMAGATA[†1]　KOJI KAGAWA[†2]

## 1. はじめに

プログラミング学習の特に初期段階では，プログラミング言語の文法に慣れていないためにセミコロンのつけ忘れなどが起こる．また，制御構造の条件式の間違いや，冗長なコードを作成してしまっても，エラーが発生しないために指摘されるまでその冗長な処理に気づかないという場合も多い．最初の学習であまり理解をせずにプログラミングに取り組むと，同じ間違いが何度も起きてしまう場合もある．こういったことを積み重ねていくと，指導者だけでなくプログラミング初学者にとっても非常に負担が大きいものとなる．これを解決するために，あまり時間を書けず簡単に解くことのできる練習問題を提示するシステムが有用であると考えられた．それに取り組んだ研究が Blockly を用いた C 言語学習支援環境[1]である．しかし，この学習支援環境は Blockly のブロック定義の作成，Blockly のプログラムの GUI での作成に時間がかかるという問題点がある．また，近年 Blockly を学習支援に使う研究がいくつか知られている．例として，ぶろっくらみんぐを基盤とした学習教材[2]や，Block Code[3]などが挙げられる．これらも，Blockly を学習支援に使用しているが，指導者が練習問題を作成する際に起こるブロックと XML 作成の負担解消には触れられていない．これを解決するためには，ブロックと XML 作成の作成の簡略化を行う必要がある．

本研究ではこれを Haskell によるパーサーを用いて C 言語のソースコードから Blockly のブロック定義と Blockly のプログラムに対応する XML 作成の自動化により行う．また，パーサーを利用して C 言語のソースコードから XML と汎用的なブロックを作成し，作成者の負担の軽減を目指す．

本論文の構成は以下の通りである．2 章では Blockly を用いた C 言語学習支援環境についてとそれらの問題から必要とされるシステムの設計方針，3 章では Haskell のパーサーを使用した XML とブロックの自動生成の実装とその説明を述べる．4 章ではまとめを，5 章では今後の課題について述べる．

## 2. Blockly を用いた C 言語学習支援環境について

本章では，Blockly および Blockly を用いた C 言語学習支援環境について問題点の説明を行う．また，それらから導き出される問題点の解決方法と，本研究の目標をあげる．

### 2.1 Blockly について

Blockly [4]とは Google が提供するビジュアルプログラミング言語である．JavaScript を用い，ブラウザー上で使用できるよう作成されている．そのため，環境に左右されずに

†1 香川大学大学院工学研究科
Graduate School of Engineering, Kagawa University
†2 香川大学工学部
Faculty of Engineering, Kagawa University

Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

使用することができる．キーボードを使用してコードを記述するのではなく，図 1 に示す通り，ブラウザー上に表示されるブロックを組み合わせてコーディングを行う．

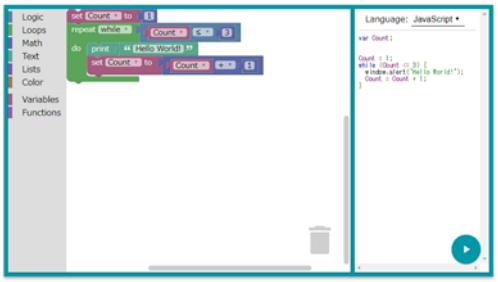


図 1　Blockly のイメージ

Blockly は，プログラミング初学者でも直感的な操作が可能で，プログラミングの基礎を簡単に学ぶことができる．ブロックのデザインを自然言語に近いものにすることで，それらをスムーズに行える．また，ブロックのデザインは変更することができるため，よりプログラミング言語に近い形にすることで各プログラミング言語の学習にも使用できる．Blockly は，組み合わせたブロックで表されたブロックベースの言語から，JavaScript，Python，PHP，Lua，Dart の 5 つのテキストベースの言語に変換することができる．しかし，本研究では C 言語への学習支援を対象とするため，Blockly を C 言語に対応させる必要がある．

### 2.2　Blockly を用いた C 言語学習支援環境について

プログラミング初学者がすぐにコーディングを行うと，プログラミング言語の文法に慣れていないためにセミコロンのつけ忘れなどが起こる．また，制御構造の条件式の間違いや，冗長なコードの問題などが発生してしまう場合がある．そのため，資料を読んですぐにコーディングに入るのではなく，それを行う間に簡単な練習問題を解くことで，より理解を深めコーディングの際にミスを減らすようにする．この簡単な練習問題を，Blockly を用いた C 言語学習支援環境で行う．

先行研究に，尾崎の行った研究[5]がある．図 2 のように，ビジュアルプログラミング言語である Blockly を C 言語に対応させたものである．しかし，完全に C 言語に対応させたものではなく，自然言語に近い表現となっている．そのため，プログラミング初学者が使用するとブロックベースの言語である Blockly とテキストベースの言語である C 言語の違いに混乱してしまう場合がある．

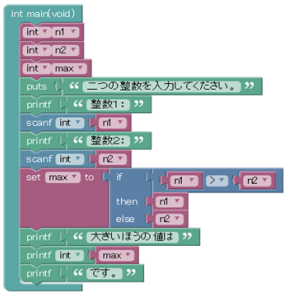


図 2　C 言語に対応させた Blockly

先行研究の問題点を踏まえ，Blockly を用いた C 言語学習支援環境（図 3）は作成された．C 言語の予習と本格的な実習の間に使用し，提示された練習問題を解くことでスムーズに実習に入ることができる．この学習支援環境では，C 言語に一対一に対応したブロックを用意することにより，初学者がブロックベースの言語である Blockly からテキストベースの言語である C 言語に完全に移行する際の差異を減らすことができる．C 言語に一対一に対応したブロックは，練習問題ごとに特化しており，練習問題ごとに合うように使用する変数や値を変えて作成されたブロックを使用する．これにより，1 つの練習問題に使用するブロックの数が減り，操作可能な部分を減らすことに繋がる．初学者にとってどの部分を操作すればよいのか，という疑問があまり出ないようになっている．この学習支援環境は主に 3 つのエリアで構成されている．

(a)　練習問題選択エリア

このエリアでは，練習問題の問題文と，練習問題の選択を行うことができる．

(b)　ブロック選択エリア

このエリアでは，練習問題の解答に使用するブロックを選択することができる．

(c)　ワークスペースエリア

このエリアでは，練習問題の選択された時点で表示されているブロックと，ブロック選択エリアから選択されたブロックを使用し，組み合わせることができる．

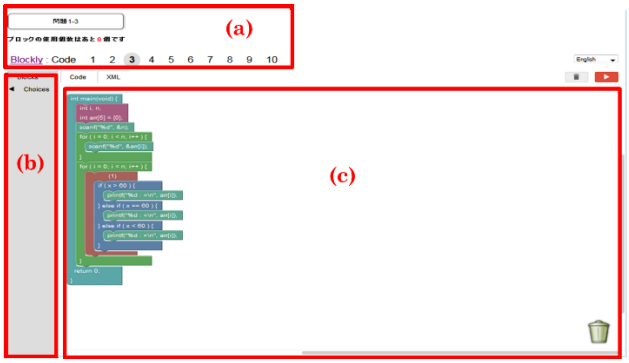



図 3　Blockly を用いた C 言語学習支援環境

Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

### 2.3 問題点

しかし, このBlocklyを用いたC言語学習支援環境には, いくつかの問題点がある．それを以下に挙げる前に，ブロックの作成方法について説明する．ブロックのデザインの定義とコード生成ルールを記述して Blockly のプログラムへ追加することで，ブロックを作成することができる．あまり複雑でないブロックは, Blockly内のアプリケーションで作成することも可能である．

**(1) ブロック作成についての問題点**

練習問題ごとに特化したブロックを使用するということは，毎回ブロックを定義する手間が発生するということである．いくつも練習問題を作成すると，指導者はブロックの定義に大きく時間を取られてしまうという問題点がある．

**(2) 練習問題ごとに作成する XML についての問題**

Blocklyでは, ブロックの組み合わせはXMLで表される．練習問題ごとに対応した XML を設定しておくことで，ブラウザー上のページを開いた時から組み合わせたブロックを最初から表示することができる．例えば“HelloWorld!”を出力するブロックの XML を加えておくことで，“HelloWorld!” を出力するブロックの組み合わせを表示することができる．しかし，この XML を取得するには，自分の手でブロックを組み合わせて Blockly に XML を表示させるか，一から全て記述するしかないという問題点がある．

2 つの問題点についてまとめると，練習問題の作成にはブロックと XML の作成に時間がかかってしまうという問題点がある．

### 2.4 解決方法

この2つの問題点を解決するためには，練習問題作成に必要となっているブロックと XML の作成時間を短縮する必要がある．これを行うために，パーサーを使用しC言語のソースコードから Blockly のブロックと XML の自動生成を行うプログラムを作成する．これまででは，練習問題の問題文の作成，ブロックの作成，ブロックの組み合わせを表示するための XML の作成が必要であったが，これにより練習問題の問題文の作成とC言語のソースコードを作成し，入力するだけとなる．

### 2.5 システムの設計方針

以上から，システムの設計方針は以下の3つにまとめられる．

**(1) 教師の負担軽減**

Haskellのパーサーを使用することで, プログラムをできるだけ短いものにする．それにより，追加で記述を行う際の負担を軽減する．

**(2) 練習問題の作成のしやすさ**

汎用的なブロックを使用することで XML 生成部が短くなる．XML 作成の手間を省くことで, 練習問題の作成時間の短縮につながる．

**(3) ブロックベースとテキストベースの言語の差異をなるべく減らす**

C 言語の構文にあったブロックの作成をする．それにより，Blockly のブロックベースと C 言語のテキストベースの言語の差異が軽減され，ブロックベースからテキストベースの言語への移行がスムーズになる．

## 3. 実装

本章ではHaskellのパーサーを使用したXMLとブロックの自動生成について詳しく説明する．

### 3.1 システム概要

本研究では，Blockly の XML とブロック作成の自動化を行うために以下のような実装を行った．まず，本システムの概要を図 4 に示す．

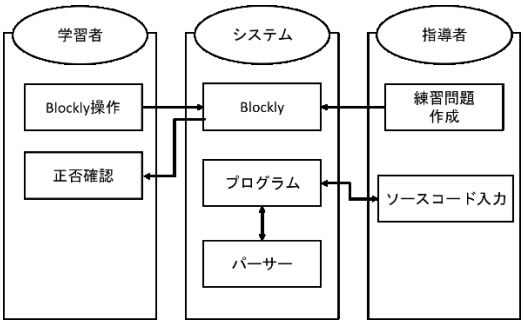



図 4　システムの概要

最初に，問題の作成者である指導者がパーサーを利用するプログラムにソースコードを入力すると汎用的なブロックの組み合わせを表示するために必要な XML が出力される．指導者はそれを使用して練習問題を作成する．次に，指導者は作成した練習問題を, Blocklyを使用した学習支援環境に組み込み，学習者がその学習支援環境を使用し，正否の確認を行う．

### 3.2 汎用的なブロックの作成

使用するブロックは，Blockly を用いた C 言語学習支援環境で作成された練習問題ごとに特化したブロックと異なり，汎用的なブロックを作成する．特化したブロックは練習問題ごとに作成するたびプログラムに記述を加える必要があり非常に手間がかかるためでもある．例えば，図 6 の下側の for のブロックであれば条件式は別のブロックで補うためにこのブロック 1 つあればそれ以降プログラムに記述を加える必要はないが，図 6 の上側のブロックではこのパターンにしか対応できないため，練習問題の追加などで対応できないパターンが発生するたびにブロックを作成しプログラムに加える必要がある．

図 5 の右上の if 文のブロックと，左下の printf のブロックには，どちらも歯車のアイコンがついている．これは

Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

mutator と呼ばれる機能で，この機能を追加されたブロックは，動的に形が変化する．この 2 つのブロックの場合，このアイコンをクリックすることで表示されるウインドウ内を操作することで else-if, else の追加や出力に使用する引数の増減などを行うことが出来る．また，テキストに特定の文字列が入力された場合にブロックの形を動的に変化させる，ということも可能である．

主に以下のブロックを作成した

- 汎用的な for, while 文のブロック
- 汎用的な printf, scanf 関数のブロック
- 汎用的な if, switch 文のブロック
- 宣言時に型の定義を行うブロック

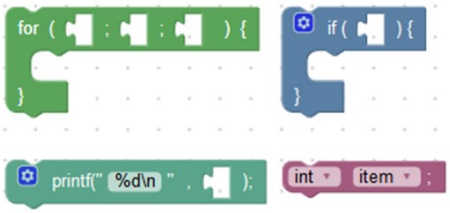
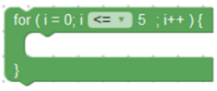



図 5　C 言語の構文にあった汎用的なブロック



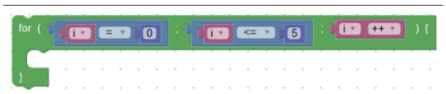




図 6　問題に特化したブロック

### 3.3 ソースコードから XML への変換

Haskell とそのライブラリである language-c-quote[6]のパーサーを利用して C 言語のソースコードを Blockly 用の XML へ変換する機能を実装した．Haskell はパターンマッチングが可能という点と，リストに対して柔軟に処理が行えるという点から選択した．図 7 から，パターンマッチングで for 文の条件式内の変数や定数などに合わせて出力する内容を細かく変更することができるようになっていることが分かる．

```
distributionBlockStm (For (Right (Just (Assign (Var (Id var1 a1) a2) JustAssign expr1 a3)))
                          (Just (BinOp op (Var (Id var2 a4) a5) expr2 a6))
                          (Just (PostInc (Var (Id var3 a7) a8) a9)) stm a)
  | var1 == var2 && var2 == var3 = …
distributionBlockStm (For (Right (Just (Assign (Var (Id var1 a1) a2) JustAssign expr1 a3)))
                          (Just (BinOp op (Var (Id var2 a4) a5) expr2 a6))
                          (Just (PostDec (Var (Id var3 a7) a8) a9)) stm a)
  | var1 == var2 && var2 == var3 = …
distributionBlockStm (For (Right (Just (Assign (Var (Id var1 a1) a2) JustAssign expr1 a3)))
                          (Just (BinOp op (Var (Id var2 a4) a5) expr2 a6))
                          (Just (Assign (Var (Id var3 a7) a8) AddAssign expr3 a9))  stm a)
  | var1 == var2 && var2 == var3 = …
distributionBlockStm (For (Right (Just expr1)) (Just expr2) (Just expr3) stm a) = …
```

図 7　for 文のパターンマッチング

次に，この機能の手順の例を図 8 に示す．自動生成の手順は 4 つに分かれている．

(1)　C 言語のソースコードの入力

最初に，C 言語のソースコードを入力する．図 8 の例では，「HelloWorld!」を出力する C 言語のソースコードを入力している．

(2)　ソースコードからリストへの変換

入力された C 言語のソースコードがプログラム内部で language-c-quote のパーサーを通してリストに変換される．

(3)　リストから XML とブロックへの変換

プログラム内部で生成されたリストを利用して，パターンマッチングを行い，XML とブロックに変換する．

(4)　Blockly を用いた学習支援環境に XML を追加

プログラムにより変換された XML を Blockly に入力することでブロックが表示される．図 8 の例では main 関数ブロックとそのブロック内に「HelloWorld!」を出力するブロックが表示されている．

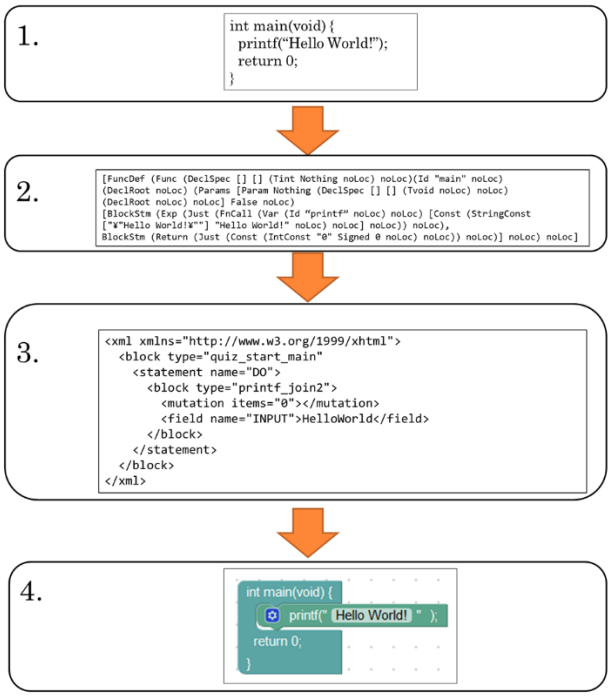



図 8　XML 変換の流れ

これを実装したことにより，例えば for 文を使用し 1 から 10 の値を出力する C 言語のソースコードを XML へ変換することで図 9 のようなブロックの組み合わせを出力することができる．通常の場合，この 12 個のブロックを組み合わせて XML を取得しなければならないが，パーサーに C 言語のソースコードを入力すればブロックを組み合わせる手間が省ける．どれだけ長いソースコードでもブロックの組み合わせに変えることができるが，一定以上の長さのソースコードを変換すると，ブラウザーにブロックの組

Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

み合わせが長くなりすぎて入らなくなるために注意が必要である．

また，テキストベースの言語に移行したプログラミング初学者が理解度の確認に使用することができる．プログラミング初学者がブロックベースの言語である Blockly からテキストベースの言語である C 言語に移行した後，初学者が記述したソースコードをこのプログラムに入力することで，慣れているブロックベースの言語ではどういったものになっているかを確認することができるということである．ブロックベースの言語で確認することで，ソースコードはブロックの組み合わせとしてそれぞれの処理が区切られて表示されるため読みやすく，処理が長ければ汎用的なブロックは非常に大きくなってしまうため冗長なコードがどこか明確に分かりやすいという利点もある．

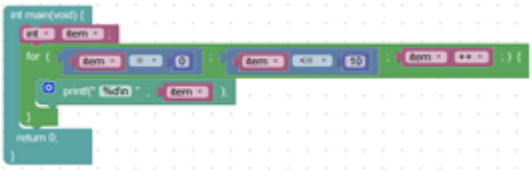


図 9　ブロックの組み合わせ

### 3.4 ブロックの自動生成

ここでブロックの自動生成とは，JavaScript で表される，ブロックの定義に必要なデザインの定義と，コード生成ルールを記述する処理の定義の 2 つを自動生成することを意味する．XML の自動生成と同じく，C 言語のソースコードを使用してブロックの定義に必要なコードを生成する．3.3 節の手順(3)より，XML と同じ段階で XML と組になって，ブロックの定義が出力される．

しかし，3.2 節「汎用的なブロックの作成」で挙げたように，本研究で使用するブロックは汎用的なブロックになっており，汎用的なブロックは全体的に使用されるブロックの数が多くなってしまう．それにより操作可能な部分が多くなってしまい，プログラミング初学者が混乱してしまう，または，操作可能な部分の多さに学習に飽きてしまう可能性もある．そこで，ブロックの自動生成を行うことで特化したブロックを作成し，使用されるブロックの数を減らす．操作可能な部分の減少と指導者がブロックを作成する時間の短縮につながる．

次に，変数のスワップの処理を例に挙げる．

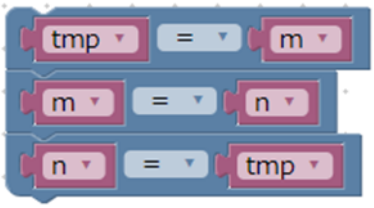



図 10　汎用的なブロックでの変数のスワップ

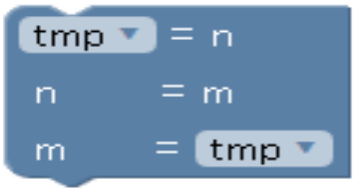



図 11　特化したブロックでの変数のスワップ

図 10 が汎用的なブロックを使用したブロックの組み合わせで，図 11 が特化したブロックである．汎用的なブロックは使用しているブロックの数が 9 個で，操作可能な部分も 9 個あるが，指導者が注目してもらいたい部分に初学者は気づきにくい．一方，ブロックの自動生成により作成された特化したブロックでは，使用しているブロックは 1 個，操作可能な部分も 2 個と，どこが注目してもらいたい部分なのかが一目で分かる．操作可能な部分が少ないために，初学者も誤操作をしにくいという利点もある．こういった処理も C 言語のソースコードから特定できるように，Haskell を使用している．

### 3.5 ブロックの自動生成のルール

ブロックの自動生成は，基本的に C 言語のソースコードにコメントをつけた直後の処理にだけ行われるようにしてある．language-c-quote ではコメントとその直後の statement が一つの組になっているためである．コメントに自動生成用のルールに従った文字列を記述することで，ルールに合ったブロックを生成する．自動生成を行うブロックの種類は，基本的な処理と複数行に渡る式の 2 種類になる．基本的な処理では，for, if～else, switch などが自動生成の対象となる．変数のスワップなど複数行に渡る式はコメントに行数と表 1 のようにどの部分を操作可能にするかをルールに従って記述しそれを取得することでブロックの自動生成を可能とする．

自動生成用のルールは，おおまかには次の通りとなる．

表 1　自動生成用のルール

| 値 | 処理 |
|---|---|
| 0 | 文字のみ |
| 1 | プルダウンメニューに変更 |
| 2 | 変数ブロック挿入可 |
| 3 | テキストボックスに |

基本的にはこのルールを式ごとで区切り，振り分けることになる．

## 4. まとめ

本論文では，Haskell を用いた Blockly のブロック定義の

Vol.2017-CE-142 No.21
Vol.2017-CLE-23 No.21
2017/12/10

作成と Blockly のプログラムに対応する XML 作成の自動化を行った．Blockly では C 言語に対応したブロックがないことと，プログラミング初学者がブロックベースからテキストベースの言語に移行する負担を考え，C 言語の文法に一対一に対応するブロックを作成した．また，練習問題を作成するたびに指導者が XML の記述を行う負担と，ブロックの操作可能な部分が多い場合プログラミング初学者が学習に取り組む意欲が下がるために，パーサーを利用した C 言語から Blockly 用のブロックの定義と XML に変換するプログラムを作成した．これにより，指導者の負担を軽減できるようになっただけではなく，プログラミング初学者がブロックベースの言語である Blockly からテキストベースの言語である C 言語に移行後，C 言語の理解度の確認のためにこのプログラムを使用することも可能である．

## 5. 今後の課題

### 5.1 関数定義，配列の対応

現在，ブロック，パーサー含め関数定義と配列には対応していない．関数定義と配列に対応することで，提示する練習問題の種類を増やすことができるようになる．

### 5.2 Arduino への対応

C 言語のソースコードを，Blockly を用いた C 言語学習支援環境で使用するブロックの定義と XML に変換するだけでなく，Arduino[7]にも対応できるようにする．使用する Block は BlocklyDuino[8]をベースにする．Arduino とはマイコンボードと開発環境がセットになったもので，BlocklyDuino とは Blockly で Arduino をプログラミングすることができるツールである．

**謝辞** 本研究は JSPS 科研費 15K01075 の助成を受けたものである．

## 参考文献

[1] 山形悠人，香川考司. Web ベースグラフィカルプログラミングエディタを用いた C 言語学習支援環境への問題提示機能の実装．教育システム情報学会第 41 回全国大会, I2-6, 2016.
[2] 渡邉 誠，吉村 光令，菱田 隆彰. C 言語初級者にやる気と達成感を与える学習教材の構築．情報処理学会第 77 回全国大会, 3ZF-03, 2015.
[3] 末吉 春一，佐藤 喬．ビジュアルプログラミングを用いたテキストベースプログラミング学習支援システム．情報処理学会第 78 回全国大会, 5ZC-07, 2016.
[4] Fraser, N. “Google Blockly” – a visual programming editor. https://developers.google.com/blockly/, (参照 2017-11-08).
[5] 尾崎陽一. Web ベースグラフィカルプログラミングエディタを用いた円滑な移行が可能な C 言語学習支援環境の開発．香川大学大学院工学研究科信頼性情報システム工学専攻. 2014 年度修論, 2015.
[6] “language-c-quote”. https://hackage.haskell.org/package/language-c-quote, (参照 2017-11-08).
[7] “Arduino”. https://www.arduino.cc/, (参照 2017-11-08).
[8] “BlocklyDuino”. https://code.makewitharduino.com/, (参照 2017-11-08).