K-033

日本語テンプレートの展開・再構成が可能な初心者向けプログラミング環境
A Programming enviroment for Beginners with expandable and reconstructable template of program structures using Japanese keywords

大川　哲史 *　　永松　礼夫 *
Satoshi Ohkawa　　Leo Nagamatsu

概 要

近年, プログラミング教育においては C 言語や Java などの既存の汎用言語を用いて学習を行う一方で、入門用の言語や環境を使用する手法も提案されている。本学の情報科学科の学生に目を向け、プログラミング初心者が C 言語へと移行しやすいような独自のテキスト記述型言語の処理環境を製作した。C 言語の言語表現を日本語のテンプレートとして表現し、配置を行うことでプログラミングを可能にする。またテンプレートの配置後には、単一のテンプレート、for 文、if 文などの制御構造ごとに入れ替えができる。「変数宣言」「条件分岐」「繰り返し処理」の例題について、他の学習用言語や環境との比較を行った。

キーワード:情報教育、教材開発、教育用および初心者用言語

1. はじめに

現在のプログラミング教育においては、C 言語や Java など既存の汎用言語をそのまま用いて実践するという手法も取られているが、Scratch[1] や Squeak[2] といったビジュアルプログラミング言語やなでしこ [3] や PEN[4] のようなテキスト記述型プログラミング言語を用いた手法も提案されている。これらの言語は実際に森らによる小学生を対象として繰り返しを含むプログラミングを行った報告 [5] や松澤らによる大学生向けの Java 入門教育での実践 [6] など、教育現場において利用されている。Scratch は MIT が Squeak を用いて作成した初心者向けプログラミング言語で、GUI でプログラミングを行うことができるという特徴をもつ。またそれを応用した例として ことだま on Squeak[7] がある。

本論文では、プログラミング初心者が C 言語への学習へとスムーズに移行することをサポートする言語環境を提案し、教育への有用性を検証する。本システムで扱う内容は、C 言語からポインタ・構造体・関数の概念を除き、変数を整数型に限定したサブセットである。プログラミングの最初に学習する「変数宣言」「四則演算」「条件分岐」「繰り返し処理」を想定している。当システムで考える言語の表記方法は初学者にとって分かりやすいように、C 言語上の予約語を日本語のテンプレートとして扱う。テンプレートの配置をした後で、学習者がソースコードの構造・順序を変更できるように、展開・再構成の機能を実装した。また作成したソースコードを C 言語のソースコードへ変換、実行、変数の値の一覧表示ができる。

2. 実装方針

2.1. 機能概要

本論文では日本語テンプレートを用いてプログラミングを行う教材システムを JavaScript を用いて作成する。ブラウザ上で図 1 に示すような「日本語風コード」「C 言語」「変数の値表」の 3 つのペイン上で操作ボタンを用いてソース作成を行う。日本語テンプレートの機能と表現の一覧、テンプレートの構造・順序の変更、日本語風コードから C 言語への変換、実行例について述べる。

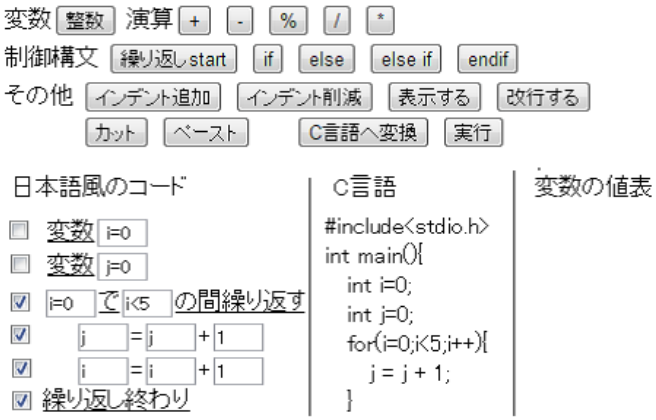

図 1: 本システムの画面例

2.2. 教育のポイント

初心者では二重にネストする for 文や if 文を使用する際に、条件などの部分は正しいが、組み合わせる順番を間違えて、正しく動かないことが見られる。さらに、それを修正する際にエラーの原因がわからず、ソースコードを全部消去してゼロから作り直す傾向がある。そこで、当システムではコードの配置順序の変更機能を用いることで、削除によらないプログラムの修正作業をサポートする仕組みを提案する。

2.3. テンプレートの再構成

本論文の実行環境ではテンプレートを挿入し、変数名や制御構造の条件文を編集することで、プログラムを作成する。また、一度作成した日本語風コードの内容を書き換えること、テンプレートの配置順序を変更することを再構成と呼ぶ。日本語風コードの一部としてテキストボックスが存在し、中身の値を変更することができる。配置順序の入れ替えを行う際には、ボッ

*神奈川大学理学部
Facility of Science,Kanagawa University

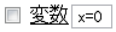

図 2: 変数テンプレート

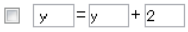

図 3: 演算子に対応するテンプレート

クス前のチェックボックスで選択を行い、選択された複数の行についてカット&ペースト方式で編集をする。

2.4. 各テンプレートの機能と表現

2.4.1. 変数宣言

テンプレートとして図 2 のような変数宣言と初期値設定を一つにしたものを用意した。整数型の変数のみをサポートする。日本語テンプレート側で変数名を自由に変更できる。「変数 1 = 変数 2」のように変数同士の値の代入も可能である。

2.4.2. 四則演算

用意されている各演算子ボタンを押したときに図 3 のようなテンプレートが追加される。演算結果の反映は「値の表示」ボタンで生成される変数の値表により確認できる。

2.4.3. 条件分岐

C 言語の if-else および if-elseif-else の 2 つに対応するテンプレートを用意した。switch 文は扱わない。なお、下線は日本語テンプレートでの予約語を示している。「条件」は「比較対象 1，比較演算子，比較対象 2」である。

- if-else 構造
  もし　条件　ならば　処理$_1$
  そうでなければ　処理$_2$
  条件文終わり
- if-elseif-else 構造
  もし　条件$_1$　ならば　処理$_1$
  また　条件$_2$　ならば　処理$_2$
  そうでなければ　処理$_3$
  条件文終わり

2.4.4. 繰り返し処理

C 言語での for 文に相当するテンプレートで以下のような表記を用いる。

変数と初期値 で 条件 の間繰り返す
　処理
繰り返し終わり

なお「変数と初期値」は「変数＝初期値」の構文で例えば「x=2」などである。「繰り返し終わり」は「繰り返し終了」ボタンを押したときに挿入され、繰り返し処理の終端となる。「変数と初期値で条件の間繰り返す」と「繰り返し終わり」の間に他のテンプレートを挿入する。

2.4.5. テンプレート構造・順序の変更

テンプレートで入力されたコードに対して行単位で順序の変更を行うことができる。各行の前にあるチェックボックスによりその行を選択状態にする。カットボタンを押すことで切り取り、挿入したい位置の行をチェックしてペーストボタンを押すことで、切り取られた内容を直前に挿入することができる。

2.5. 日本語テンプレートから C 言語への変換

「C 言語へ変換」ボタンを押すと、日本語テンプレートにより作成された日本語風のコードを C 言語のコードへと変換する。例題 1 は整数値が頂点の高さになるような二等辺三角形を描くプログラム (二重 for ループ) で、例題 2 は点数による成績わけ (if-elseif-else) である。

2.6. 変換の例

例題 1 のプログラム (図 4) を C 言語に変換した結果を図 5 に、例題 2 のプログラム (図 6) を C 言語に変換した結果を図 7 に示す。

```
変数 i = 0
変数 j = 0
変数 h = 5
i = 0でi < hの間繰り返す
    j = 0でj <= iの間繰り返す
        "*"を表示する
        j = j + 1
    繰り返し終わり
    改行する
    i = i + 1
繰り返し終わり
i = 0でi < h - 1の間繰り返す
    j = iでj < h - 1の間繰り返す
        "*"を表示する
        j = j + 1
    繰り返し終わり
    改行する
    i = i + 1
繰り返し終わり
```

図 4:　例題 1(日本語風コード)

```
int h = 5;
for(i = 0; i < h; i++){
    for(j = 0; j <= i; j++){
        printf("*");
    }
    printf("\n");
}
```

図 5:　例題 1(C 言語に変換後の一部分)

3. 他のシステムとの比較

本システムを他の言語やシステムの機能や特徴と比較した。今回はテキスト型記述プログラミング言語で

```
変数 score = 60
もし score >= 80ならば
     "優"を表示する
また score >= 70ならば
     "良"を表示する
また score >= 60ならば
     "可"を表示する
そうでなければ
     "不可"を表示する
条件文終わり
```

図 6: 例題 2(日本語風コード)

```
int score = 60;
if(score >= 80){
     printf("優\n");
}
else if(score >= 70){
     printf("良\n");
}
```

図 7: 例題 2(C 言語に変換後の一部分)

ある「なでしこ」と、ビジュアル型プログラミング言語の Scratch の 2 つを主な比較対象とする。

3.1. 制御構造の変更方法

初心者にありがちな細部は正しいが全体の組み立てがおかしいプログラムに対して、大きなブロック単位での組み換えが楽にできるかについて検討した。

なでしこ: テキスト記述型の言語のため、修正を行う際に文字単位で文を書き直す必要がある。

Scratch: GUI パーツをドラッグで動かすことで、制御構造の中身の処理を変更することができる。構造単位で動かすことも可能である。

当システム: 単一のソースコードを動かすだけでなく、選択された複数の行についてカット&ペーストすることができる。

3.2. エラー処理

文法的に間違ったプログラム (例えば、閉じていないループ) を作成することがあるか、また入力の文法エラーをいつ検出するかについて述べる。

なでしこ: プログラム入力時にはチェックはなく、コンパイル時に文法エラーを調べる。エラーがあれば別ウィンドウを表示し、エラー内容と修正のためのヒントを表示する。

Scratch: 未定義の変数は操作できないなど、故意にエラーを起こすことはできない。

当システム: 未定義の変数でも使用/参照することはできるが実行時にエラーになる。

3.3. ソースコードの作成方法

各システムにおけるソースコードの作成方法について比較した。

なでしこ: テキストボックス中に一文字ずつ入力する。文章の補正機能などはない。

Scratch: GUI をドラッグ&ドロップすることにより、ブロックを組み立てる。「変数」「値」「式」はテキストボックスの中に入力する。

当システム: 使用したい機能に対応するボタンを押し、テンプレート配置することでソースコードを組み立てる。「変数」「値」「式」はテキストボックスの中に入力する。

4. 考察

4.1. プログラムの再構成の方法

Scratch ではクリック&ドラッグを使用することによりブロックの移動を行うが、for 文,if 文の制御構造について移動する際に、制御構造の中に配置されている上半分のブロック (複数の文の固まり) については、そのまま移動することはできるが、下半分のブロックについては一度制御構造を外さなければ移動できない。

一方当システムでは、移動を行う際にカットボタンとペーストボタンによる操作 (2.4.5 参照) を使用する。エラーチェックは実行時にされるため、ペースト後のコード全体の構成にエラー (閉じていない for 文など) があっても挿入した時点では指摘しない。行単位で選択できるので、制御構造内のつながりによらずに入れ替えることができる。

4.2. エラー検出する時期に関する課題

二重の for 文や if 文に対して、構造の入れ替えを実行する場合を考える。場合によってはエラーが生じる場合も考えられる。この時想定されるのは、制御変数が未定義であったり、使用される変数の値がおかしいなどである。その際の対処方法として、(1) 前回実行した際の変数の値をそのまま利用する (Scratch の場合), (2) 未定義変数の値は undefined として扱う, (3) 構造の入れ替えをする際に構文解析をしてエラーを検知する, の 3 通りが考えられる。

本システムでは、使用している変数の値が間違っているということを明確に利用者に示すため、(2) の方式を採用している。(1) は前回の実行時の値をそのまま利用するので、変数の値が不明確になり、利用者を混乱させるので望ましくない。(3) はコードが変更される毎にチェックを行い、未定義の値を参照しているなどのミスを検出を試みるが、実行時でないと確定しない条件などがあるため完全ではない。

例 1 ありがちな間違いとして、二重ループの内と外で同じ制御変数を使用してしまうケースがある。

```
for(i=0〜....){
  for(i=0;〜....){
   printf("*");
```

```
    }
   printf("\n");
  }
```

この例では、"*"を用いて正方形を描こうとしたが、同じ制御変数 i を使用したため、コンパイルエラーにはならないが、最初の一行分しか"*"を出力されず、予想と違った結果になる。このような場合、本システムでは制御変数部分はテキストボックスになっており、制御構造の内容が編集しやすいため、学習者は容易に誤りを修正することができる。

例 2　内側のループ変数 j の初期値に外側のループの変数 i の値を使っている場合に、内側のループを外側のループの前に移すという制御構造の入れ替えを行う。

入れ替え前

```
for(i=0～1,2...){
    for(j=i ···· ){
    }
}
```

入れ替え後

```
for(j=i ···· ){
}
for(i=0～1,2...){
}
```

ここでは、内側のループ変数 j の初期値が変数 i で与えられており、ループを入れ替えた際に、j の値が予期しない値になって、正しい結果を得られない。ループより前に i が与えられていればコンパイルエラーにはならず、実行されてしまうため、入れ替えを行った際、エラーと判断をするのは難しい。

また、変数の初期値が与えられておらず、そのまま使用されている時の対応も検討するべき点である。C 言語ではそのような場合、コンパイル時、プログラム実行時ともエラーにならず、不定の変数の値で実行される。JavaScript では値を undefined として扱える。このような状況についての警告の方式として、(1) 特に警告はしない,(2) 宣言された時点で自動的に 0 を代入する,(3)undefined 値を入れておく, などが考えられる。当システムでは実装言語を JavaScript にしたことを生かして、初期化されていない変数の値は undefined にし、実行時までは警告はしないということにした。これは初期値を与えないまま実行したときに、学習者に注意を促して誤りを認識させたいからである。

4.3. テンプレートの表記方法

繰り返しの記述方法については、他の学習環境と比べて表記に違いがある。C 言語での for(i=0;i<=5;i++) に対応する表記は、「なでしこ」では

iで1から5まで繰り返す

PEN では

iを1から5まで1ずつ増やしながら

当システムでは

i=0でi<=5の間繰り返す
　処理
　i=i+1
繰り返し終わり

となる。「1 から N まで 1 ずつ」のような典型的な繰り返しを手軽に書けることと、C 言語の for 文がサポートする様々な条件設定の自由度のうちどちらを優先させるかの方針の違いがある。後々C 言語へ移行することを考えると、その文法に近い記述にすべきかもしれない。この点については試用を通じて検討すべき部分である。

5. まとめ

本論文では、初心者向け言語環境の提案を行った。実装中の「変数宣言」「四則演算」「条件分岐」「繰り返し処理」のテンプレートの文法と機能を紹介した。他の言語処理環境と「変更の柔軟さ」「文法エラーの検出」「ソースコードの作成方法」について比較を行って当システムの特徴を示すとともに、提案する再構成機能について述べた。

参考文献

[1] Scratch Team Lifelong Kindergarten Group MIT Media Lab: Scratch -imagine.program.share- http://scratch.mit.edu/ (accessed 2013-4-3).

[2] Dan Ingalls, Ted Kaehler, John Maloney, Scott wallace and Alan Kay ”The Story of Squeak, A Practical Smalltalk Written in Itself”, Proc. of ACM OOPSLA’97 (1997).

[3] クジラ飛行机”日本語プログラム言語「なでしこ」” http://nadesi.com/ (accessed 2013-3-28).

[4] 中村 亮太, 西田 知博, 松浦 敏雄”初学者向けプログラミング学習環境 PEN” http://www.media.osaka-cu.ac.jp/PEN/ (accessed 2013-4-15).

[5] 森 秀樹, 杉澤 学, 張 海, 前迫 孝憲”Scratch を用いた小学校プログラミング授業の実践: 小学生を対象としたプログラミング教育の再考”, 日本教育工学会論文誌, Vol.34,No.4,pp.387-394 (2011).

[6] 松澤 芳昭, 酒井 三四郎”ビジュアル型言語とテキスト記述型言語の併用によるプログラミング入門教育の試みと成果”, 情報処理学会研究報告, コンピュータと教育研究会, CE119,pp1-11 (2013).

[7] 慶応義塾大学 大岩研究室 Squeak 開発チーム ”ことだま on Squeak” http://crew-lab.sfc.keio.ac.jp/squeak/ (accessed 2013-4-9).