Vol.2015-IS-132 No.5
2015/6/13

機械式計算機のメカニズムを用いたプログラミング教育の提案

伊藤永悟† 藤本貴之†

近年、コンピュータは高機能かつ多機能なものとなった。開発環境も洗練されたため、多様なプログラムを低労力で実現することが可能である。しかし、プログラミング初心者にとっては、開発環境を整えること自体が大きな障害となる。また、短い記述で多くの処理を実現可能であるため学習のために求められるプログラムが複雑化しやすい。初心者向けに広まっているビジュアルプログラミング言語は、あくまで記述を視覚化して直感的にしたに過ぎない。この問題を解決するため、本研究では単純な計算を機械式計算機のメカニズムを用いて多くの段階を経て計算させることで、単純なプログラムを通じたプログラミング教育を可能とするシステムを提案する。

1. はじめに

近年、コンピュータは高機能化、多機能化している。利用者は多くのメディアやファイル形式を利用することができる。それらのデータを利用するプログラミングは、プログラミング初心者にとって難しい。この問題を解決するためにプログラマはライブラリを利用する。プログラミング初心者は、マルチメディアを利用する前にこれらのライブラリの使用法を学ぶ必要がある。しかし、それは複雑である。

視覚的・聴覚的な情報は自然なものである。ビジュアルプログラミング言語はこの特徴に着目している。例えばビジュアルプログラミング言語であるスクラッチは、はマウスで利用可能な GUI のエディタを有する。 スクラッチの利用者はスプライトも簡単に編集することが可能である。スクラッチはコンソールを標準状態では表示していない。よって､スクラッチの出力はスプライトが中心となっている。これらはマウスイベントが生じたときに処理される。スクラッチは自然な表示を持つ反面、アルゴリズムが複雑になりがちである。

フローチャートはしばしばアルゴリズム学習に用いられる。フローチャートを作るには、フローチャーの書式についての知識が必要となる。これはプログラミングの書式よりは簡単である。そのためプログラミング初心者は柔軟に考えることができる。プログラミング初心者は、例えば計算方法のように、現在学習している内容のフローチャートをしばしば作ろうとする。しかし、学習内容は、それにフローチャートよりも見合った言語がある。よって、プログラミング初心者にとって、フローチャートはその柔軟性のために複雑化しやすい。

プログラミング初心者は、混乱しないように学習していくべきである。しかし、既存の言語や手法は、とても複雑である。これこそがプログラミング学習の最も大きな問題である。本論文では、この問題を解決するための新たなシステムを提案するものである。

2. 計算道具

電子式ディジタル計算機が普及する前、人々は機械式計算機のような計算道具を利用していた。これらの道具は多くの種類メカニズムを有する。本章ではその種類や特徴について述べる。

(1) そろばん（アバカス）

最古の計算機のひとつがそろばん（アバカス）である。世界中で開発された様々な種類のそろばんが存在する。そろばんは基本的に串と珠を有する。それぞれの珠には穴がある。串は珠を貫通している。人々は珠を串に沿って移動させる。珠の位置が数値を表す。それぞれの串はひとつの桁を意味する。人々は串ごとに計算し、珠へ反映させる。そろばんは学習しやすい計算機である。珠の位置や串から数値へと変換しやすい。串はひとつだけの桁を示す。このことから操作手順は容易になる。しかし、そろばんは利用者の計算能力を必要とする。そろばんは操作手順以外の能力を必要とするものである。

(2) ネピアの骨

ネピアの骨はそろばんに似ている。この計算機はいくつかの棒を利用する。その棒には数が書かれている。人々はそれぞれの桁ごとに棒の数を利用して計算をしていく。ネピアの骨は乗算や除算にも適している。 しかし、そろばんのように利用者の計算能力が必要となる。ネピアの骨は機械的な計算に向いていない。

(3) 計算尺

計算尺はアナログ計算機である。多くの計算尺は、長さが等しい 3 つの直線的な細切れで出来ている。人々は中央の細切れを動かし、細切れに書かれた目盛りから値を読み取る。値を読み取るとき、たいていはカーソルを利用する。計算尺は多くの種類の計算を実現できる。しかし、多くの目盛りを利用しているため使い方が複雑である。アナログな仕組みであるので、人々は注意深く操作をしないといけない。これは欠点となる。

† 東洋大学大学院工学研究科情報システム専攻
Dept. of Information System, Toyo University

(4) オドナー型機械式計算機

機械式計算機はライプニッツの計算機がベースとなっている。特にオドナー型機械式計算機は、その小ささから普及した。これには Stepped Drum 機構の代わりに Pinwheel 機構を用いている。通常、これらはハンドルに接続されている。オドナー型機械式計算機の利用手順は、主にこのハンドルの操作となる。回転のさせかたを変えることで、オドナー型機械式計算機は自動的な四則演算を可能とする。

(5) 電動機械式計算機

オドナー型機械式計算機は完全に自動的な計算を実現していない。電動機械式計算機はハンドルの操作を改善したものである。人々は数字や演算キーを使うだけである。モーターが自動的に Pinwheel 機構を回転させる。電動機械式計算機は電子式計算機に似ている。

3. 学習に適した計算アルゴリズム

(1) 学習向けアルゴリズム

プログラミング初心者は、学習のために単純なアルゴリズムを利用するべきである。単純なアルゴリズムとは、次の点を満たすものと考える。

- 演算の目的が明確
- 演算結果の推測が容易
- 視覚的情報量が多い

これらを満たすアルゴリズムとして、機械式計算機を利用した演算が適する。機械式計算機は、四則演算を中心とした演算機能しか持たない。四則演算は、学校教育で日常的に扱うため、結果の推測に問題はない。演算機能が制約されると演算の目的も限定される。また、機械式計算機では、入出力を機械を通して行っているため、そのサポートのための視覚情報が多い。

プログラミングのためのアルゴリズムとしては、人間の演算能力は利用しないことが望ましい。そのため、そろばんやネピアの骨の演算手法は、学習のためのアルゴリズムとして適さない。

(2) 操作手順の視覚化

演算を行うとき、その経過を表示することはデバッグのために重要である。これはデバッガにブレークポイントが実装されていることからも明らかである。計算機を模したアルゴリズムを利用する場合、その操作手順の視覚化は計算機の動きとして行うべきである。

計算尺の場合は、細切れとカーソルの動きである。計算尺はアナログ計算機のため、目盛りの読み取りと操作を交互に繰り返す。演算の基準となる状態がたびたび変わる。よって、操作の内容が分かりにくくなる可能性が高い。

機械式計算機は、数値の入出力と演算の実行が明確に分かれている。ダイヤルは数値の入出力に用いる。ハンドルは演算に用いる。また、オドナー型機械式計算機では、入力値、計算結果、乗数・除数が同時に表示されているため、視覚的情報量が多い。

電動機械式計算機は、数値の入力や演算の実行を全てキーによって行うものである。インタフェースが統一され説明が容易となった。しかし、物理的な特徴が排除されたため、視覚的な意味の分かりやすさが低減している。

(3) 計算手順の増幅

計算機で計算結果を得るには様々な操作の方法がある。プログラミング教育には、計算手順はできるだけ多い方が良い。ただし、その計算手順は、全て意味を持つ必要がある。オドナー型機械式計算機の計算手順は、多くの手順を要する。まず、入力や演算の手順が分離している。ハンドルを回転させる操作もたったひとつのダイヤルをリセットするか、一度の加算・減算をするに過ぎない。もし、利用者が乗算・除算をしようとするなら、加算や減算を繰り返し行う必要がある。

(4) オドナー型機械式計算機の計算アルゴリズム

以上の理由により，オドナー型機械式計算機の計算アルゴリズムは、プログラミング初心者のプログラミング学習に適している。このアルゴリズムは、次の 7 種類の命令により構成されている。

- [Input] レバーを通じた数値の入力
- [Read] ダイヤルの値の読み取り
- [Reset] ハンドル・レバーを通じたダイヤルの帰零
- [Shift] レバーを通じた演算桁の移動
- [Mode] レバーを通じた乗算・除算状態の変更
- [Rotate] ハンドルを通じた加算・減算
- [Bell] ベルの音を聞く

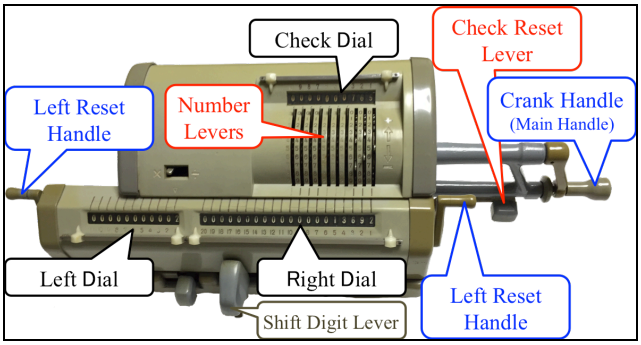

図 1　タイガー計算機（オドナー型計算機の一種）

4. 提案システム

(1) 機械式計算機のアルゴリズムに基づくプログラミング学習

前章までで述べたように、オドナー型機械式計算機の計算

アルゴリズムは、プログラミング教育に適している。プログラミング学習のためのシステムを構築するには、オドナー型機械式計算機で利用されていた 7 種類の命令を実行可能としなくてはならない。また、その命令はオドナー型計算機のハンドルやレバーの動きとして、視覚化する必要がある。
しかしながら、その要件のみ満たしてもシミュレータを作成するに過ぎない。提案システムでは、プログラムとしての可変性を実現する。そのために、提案システムでは下記の命令の実行を可能とする。

- [Scan] キーボードを通じて値を入力する
- [Store] 変数に値を格納する
- [Read] 変数の値を読み取る
- [Print] ディスプレイに文字列や数値を表示する
- [While] while ループ
- [If] if 分岐
- [Exit] 終了

すなわち、本システムを利用する場合、プログラミング初心者はたった 14 種類の命令を覚えるのみでプログラミングを行うことができる。

(2) インタフェース

提案システムのインタフェースは、4 つの領域により構成されている。これらの領域は全てひとつのウインドウ内に収まっている。

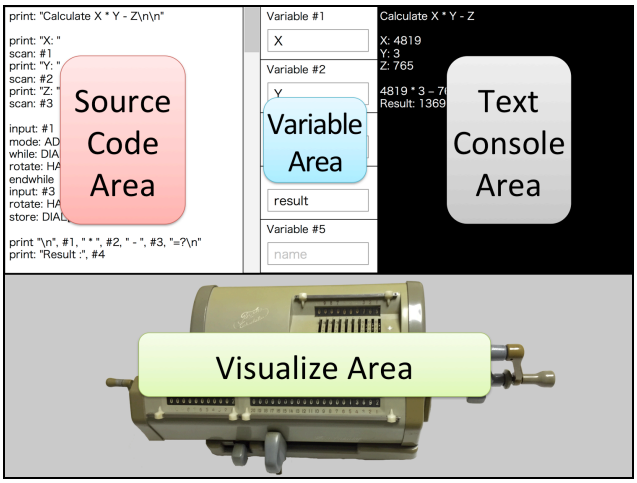

図 2　提案システムのレイアウト

第一に、ソースコード領域が存在する。利用者は、この領域にソースコードを記述する。記述には通常のテキストエリア同様にキーボードやマウスを利用することができる。

```
print: "Calculate X * Y - Z\n\n"

print: "X: "
scan: #1
print: "Y: "
scan: #2
print: "Z: "
scan: #3

input: #1
mode: ADDITION
while: DIAL[left], #3
rotate: HANDLE[main], POSITIVE
endwhile
input: #3
rotate: HANDLE[main], NEGATIVE
store: DIAL[right], #4

print "\n", #1, " * ", #2, " - ", #3, "=?\n"
print: "Result :", #4
```

図 3　ソースコード領域

第二に、コンソール領域が存在する。コンソール領域は、通常の CUI と同様に実行時の文字表示に利用される。ただし、ソースコード領域との競合のため、キーボードによる文字入力は、ダイアログを利用して行う。

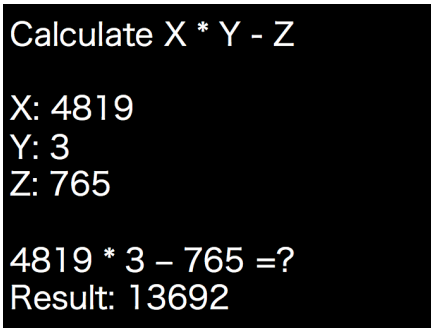

図 4　コンソール領域

第三に、変数領域が存在する。この領域はソースコード編集時および実行時の双方に関わるため、ソースコード領域とコンソール領域の間に位置する。命名方法という風習の学習を不要とするため、決まった数の変数を「#1」などと番号で指定する方法を取る。ただし、宣言文に付随するコメントの代替として、各変数の意味（名前）を付ける欄は存在する。この欄は、既存のプログラミング言語に存在する変数名の命名規則と異なり、自由な記述が可能である。

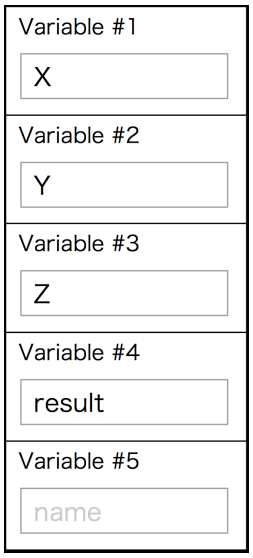

図 5　変数領域

最後に、視覚化領域が存在する。この領域は、全ての領域の下部に位置する。この領域では、オドナー型機械式計算機を表示し、そのハンドルやレバー、ダイヤルの動きを表示するものである。本提案システムでは、オドナー型機械式計算機の一種であるタイガー計算機の写真を加工したものを利用した。

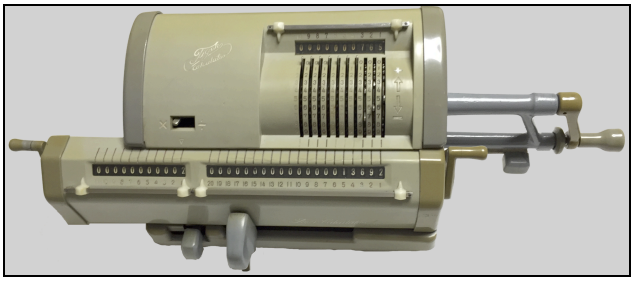

図 6　視覚化領域

(3) プログラミング言語

提案システムでは、ソースコード領域にソースコードを記述する必要がある。このソースコードは、提案に基づき 14 種類の命令を実現するものに過ぎない。そのため、提案システムに合わせた学習用プログラミング言語を作成する必要がある。今回は、簡易なものとして 1 行に 1 命令を記述する言語を作成した。これらの命令は必ず行頭に記述し、コロンの後に引数を記述する方法をとる。
提案システムに実装した命令は下記の通りである。

- input
- reset
- shift
- mode
- rotate
- scan
- store
- print
- while/endwhile
- if/endif/elseif/else
- exit

while ループや if 分岐については、BASIC に倣い括弧を利用せずに endwhile 等で範囲を規定した。
引数として利用される変数は、「#1」「#2」…と番号で表わす。scan 命令では、その変数へ値を格納する。store 命令では、第一引数なら変数の値を読み取り、第二引数では変数に値を格納する。その他の命令では、その変数の値を読み取る。また、ハンドルやダイヤルについては HANDLE[main]、DIAL[check]、DIAL[left]、DIAL[right]という名前で状態を取得可能である。また、ハンドルは POSITIVE や NEGATIVE などを与えることで rotate 命令により回転させることが可能である。ベルの音についても BELL と記述することでその状態（音が鳴ったか否か）を取得できる。

5. おわりに

本提案システムはかつて普及した計算機のメカニズムに基づくものである。これは今日の計算には不向きであるが、その単純さにより学習には向いていると考える。本論文では、よりプログラミング初心者に適したシステムを構築した結果、わずか 14 命令を理解するのみで実行可能なものとした。

参考文献

1) 山本樹, 國宗永佳, 須藤智: ビジュアルプログラミング環境を用いたプログラミング初学者対象の授業実践と評価, 教育システム情報学会研究報告, Vol.29, No.6, pp.23-30 (2015).
2) 西内康裕, 中川洋, 中西通雄: プラレールを用いた学習教材の改良, コンピュータと教育研究会報告, Vol.128, No.16, pp.1-7 (2015)
3) 大駒誠一: コンピュータ開発史, 共立出版 (2005).
4) 幸山直人: タイガー手廻し計算器 (タイガー計算器) シミュレータ
http://kouyama.math.u-toyama.ac.jp/main/computer/personal/tiger/tiger2.htm
5) the Lifelong Kindergarten group at the MIT Media Lab: Scratch, https://scratch.mit.edu/