コンピュータを使わない プログラミング及びプログラム実行環境の提案

六沢 一昭[1,a)]

概要：本稿は, コンピュータを使わない プログラミング及びプログラム実行環境を提案するものである. この環境は, トランプ大の矩形 (エリア) が 8 個程度書かれたシートとカード (例えばトランプ) からなる. エリアには 1 枚以上のカードを重ねて置くことができ,「カードの移動」や「エリアが空であるかの検査」といった基礎的な操作のみを行なうことができる. エリア上のカード群はスタックを形成し,「カードの移動」は, 移動元エリアに対する pop 処理と, 移動先に対する push 処理から構成されると考えることができる. プログラムはこれらの操作を使ってフローチャートにより記述し, 実行は手作業で行なう. 本環境は, 千葉工業大学情報工学科 AO 入学試験において 2008 年秋から 7 年間使用した.

キーワード：CS アンプラグド, プログラミング教育, アルゴリズム, プログラム実行環境

Programming and Program Execution Environment without Using Computers

ROKUSAWA Kazuaki[1,a)]

Abstract: This paper proposes a programming and program execution environment without using computers. The environment exists a sheet with about eight rectangles (areas) and cards. In the environtment only basic operations such as moving a card from one area to another and checking whether an area is empty are defined. The program is described using the operations in the flowchart and executed manually. The environment was used for seven years from 2008 in the AO entrance examination of Chiba Institute of Technology.

Keywords: CS Unplugged, Programming Education, Algorithm, Program Execution Environment

1. はじめに

本稿は, コンピュータで実行することを前提としないプログラムの作成 (プログラミング) と, そのプログラムを実行する環境を提案するものである.

提案する環境は, B4 サイズ程度の大きさの用紙と, トランプのようなカードからなり, 行なうことのできる操作は, カードの移動といった基礎的なものだけである. プログラム (アルゴリズム) はフローチャートにより記述し, 実行は, フローチャートに従って手でカードを操作することにより行なう. このため, プログラムの作成者はプログラムの動きを無理なく理解し, プログラムの間違いに自ら自然に気がつくことが期待できる.

本環境は, 千葉工業大学情報工学科 AO 入学試験においてアルゴリズム作成能力を問うために開発したものである. 2008 年秋から 7 年間の AO 入試における実績がある.

本稿では, 開発した環境を解説し考察する[*1]. 2 節で本環境を開発した背景を述べ, 3 節で本環境を説明する. 4 節では, 本環境を用いてプログラムを作成/実行した課題のいくつかを紹介し, 5 節で本環境の使用実績を示す. 6 節で本環

[1] 千葉工業大学 情報科学部 情報工学科
Department of Computer Science, Chiba Institute of Technology
[a)] rokusawa.kazuaki@it-chiba.ac.jp

[*1] 本稿は [1][2] を発展改訂したものである

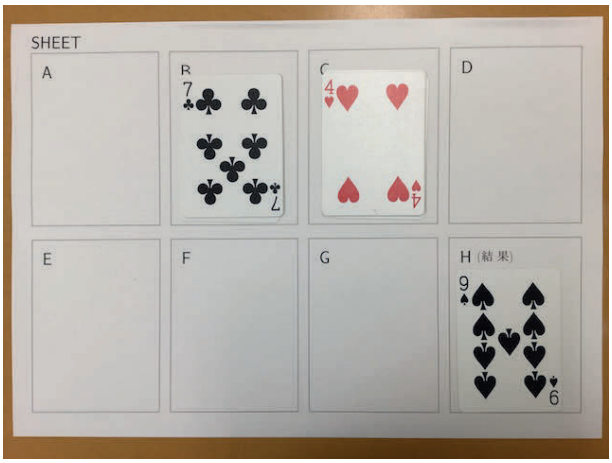


図 1　実行環境の一場面

境を考察し, 関連する研究を 7 節で示す. そして最後に 8 節で今後の課題を述べる.

2. 環境の開発の背景

アルゴリズムの作成能力を問う AO 入試を 2008 年秋に行なうことが決まり, 実施方法を検討することになった.

入試では「限られた時間内に, 受験者がアルゴリズムを作成/実行できる環境」を用意する必要がある. 既存の言語によるプログラム作成/実行環境では以下の問題があると考えた.

> 作成したプログラムの, アルゴリズムとしては本質的ではない部分の誤りのためにプログラムが動かないことがある.

もちろん, これは現在の実際のプログラミング開発においては本質的な課題であるが, 入試の目標である「アルゴリズムを作成する能力を限られた時間内に問う」ためには避けたいことである.

また,

> 既存のプログラミング言語によるプログラミング経験がなくてもそれが直接的な不利にはならないようにしたい.

という方針もあり, これを実現するために

> アルゴリズムを構成する操作を, プログラミング言語とは独立した少数の基本的なものだけにすればよいのではないだろうか.

と考え, 以下を実現する環境を考えた.

> 与えられた課題を実現するアルゴリズムを, 受験者が作成/記述し, 受験者自らが実行してテストすることができる.

3. 提案する プログラミング及びプログラム実行環境

3.1 実行環境の構成

実行環境は, シートと, シートの上で操作するカードからなる. 図 1 に実行環境の一場面を示す.

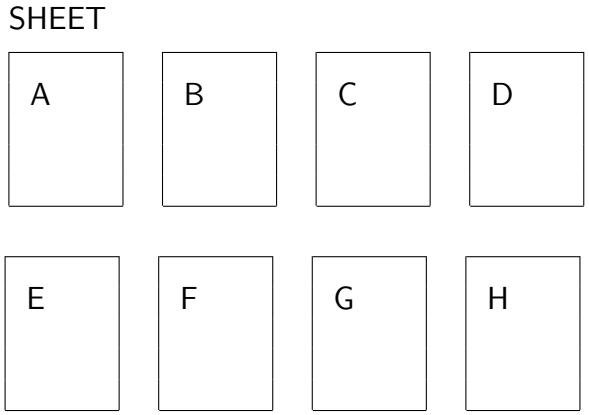


図 2　シートの例

シート　トランプ大の矩形 (エリア) を 8 個程度設けた用紙である. 各エリアは名前を持つ. シートの例を図 2 に示す.

カード　トランプのように数値や記号などの書かれているカードである. エリアに 1 枚以上のカードを重ねて置くことができる.

エリアの大きさや数は本質ではない. 大きさは, 使用するカードが収まればよく, 数は 8 個でなくてもよい.

3.2 実行環境における操作

行なうことのできる操作を以下に示す. いずれも基礎的なものである.

(a) カードの移動 (1)　あるエリアのカードの一番上のカードを, 他のエリアの一番上に移動させる.

(b) カードの移動 (2)　あるエリアのすべてのカードを, 順序はそのままで他のエリアの一番上に移動させる. [*2]

(c) 空であるかの検査　あるエリアが空であるか (カードがあるかどうか) 調べる.

(d) 2 枚のカードの数値や記号の比較　2 つのエリアの一番上のカードの数値や記号を比較する.

(e) カードの数値や記号の比較　あるエリアのカードの数値や記号と, 指定した数値や記号を比較する.

図 3 は, これらの操作をまとめたものである.

3.3 プログラムの記述

プログラムは, 前述した操作を使ってフローチャートにより記述する. フローチャートにおいて使える制御構造は順次と分岐のみとし, 反復は使わないことにした. 以下のように考えたからである.

- 反復は, 順次と分岐で記述することができる.
- 順次と分岐は感覚的に理解できるが, 反復には説明が必要である.

[*2] この操作は, 後述するように, カードの移動 (1) と 空であるかの検査 から構成することができるので本質的に必要な操作ではない.

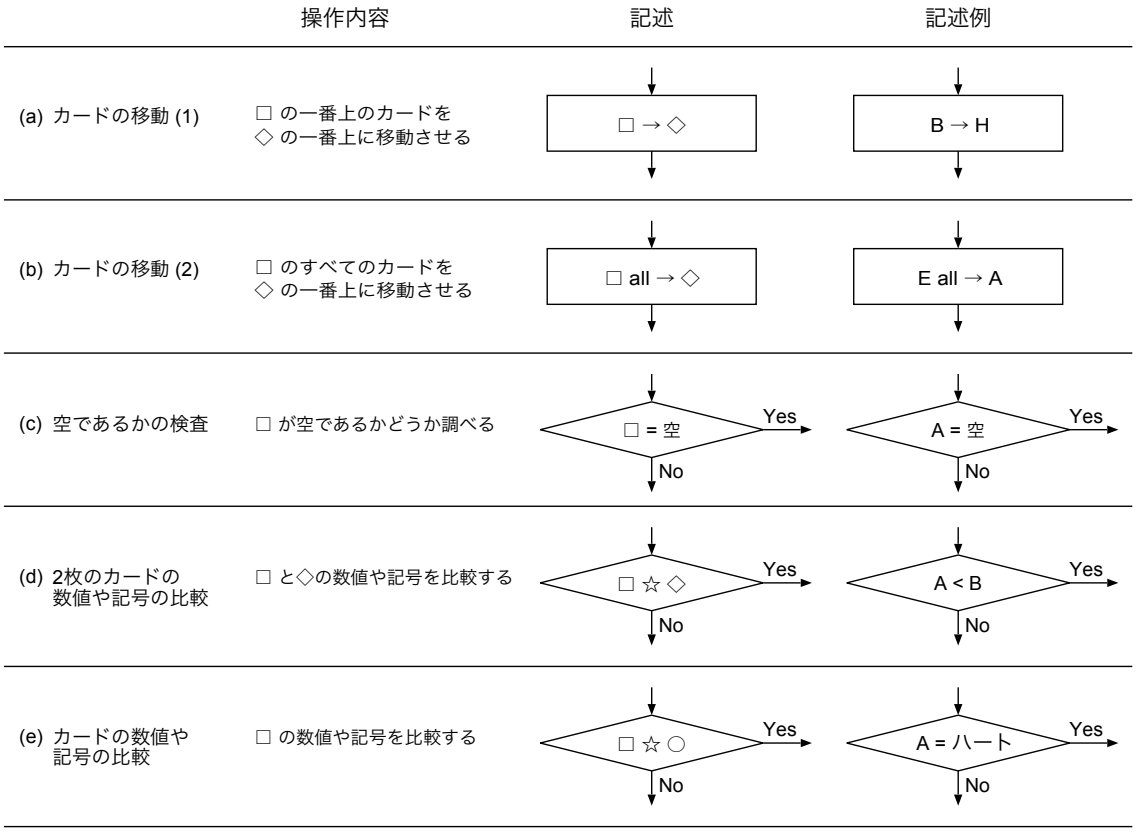


図 **3** 実行環境における操作

以下に簡単なプログラム例を二つ示す.

プログラム例 **1:** あるエリアの全カードの移動

エリア A のすべてのカードをエリア B に移動させるプログラムを図 4 に示す. プログラムの処理内容は以下の通りである.

1. A の一番上のカードを B の一番上に移動させる.
2. A にカードがあるならば (空でないならば) 1. へ戻る.
   カードがないならば (空ならば) 処理を終える.

1. は前述した操作 カードの移動 **(1)** であり, 2. は 空であるかの検査 を使った分岐である.

このプログラムによる移動ではカードの順序が逆になる. 移動を 2 回繰り返すことにより, 前述した操作 カードの移動 **(2)** が実現できる.

プログラム例 **2:** 最大値のカードの検出

エリア A にあるカードから最も大きい数値のカードを見つけて H へ置くプログラムを図 5 に示す. プログラムの処理内容は以下の通りである.

1. まず, A の一番上のカードを B の一番上へ移動させる. (B は, 仮の最大値 (のカード) を保持する.)
2. その後, A にカードがある限り以下を繰り返す.
   A の (一番上の) カードと, B の (一番上の) カードを比較し, A が大きければ A の一番上のカードを B へ移動させ, 大きくなければ A のカードは C へ移動させる (捨てる).
3. B の一番上のカードを H へ移動させる (B の一番上のカードが最大値のカードである).

4. 課題例

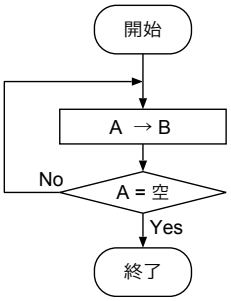
入試などで試みた課題のいくつかを以下に示す.

4.1 カードの数値あるいは記号を考慮するもの

課題 **(1)** A にダイヤとクラブのカードがある. 2 枚ある数値のカードをすべて見つけて H に置く. (A のカードが ♣4, ♢2, ♣8, ♢3, ♣2, ♢6, ♢8 ならば, ♢2, ♣2, ♢8, ♣8 を H に置くことになる.) 解答例を図 6 に示す.

課題 **(2)** A にスペードのカードが, B にハートのカードがある. A, B のいずれか一方にだけある数値のカードをすべて見つけて H に置く. (A のカードが ♠2, ♠4, ♠8, B のカードが ♡3, ♡8, ♡6, ♡2 ならば, ♠4, ♡3, ♡6 を H に置くことになる.)

課題 **(3)** A には, ある数値 x のカードが 1 枚と, x 以外のいくつかの数値のカードがそれぞれ 2 枚あるいは 3



A にあるカードをすべて B へ移動させる.

図 4　プログラム例 1: あるエリアの全カードの移動

枚ずつある. A から数値 x のカードを見つけて H に置く.

(A のカードが ♠5, ◇2, ♣3, ◇5, ♣5, ♡3 ならば, ◇2 を H に置くことになる.)

4.2　エリアのカードの枚数のみを考慮するもの

課題 (4)　A に m 枚の, B に n 枚のカードがある.
m, n とも 1 以上であり, $m > n$ である.
$m - n$ 枚のカードを H へ置く. 解答例を図 7 に示す.

課題 (5)　A に n 枚のカードがある. n は 1 以上の整数であり, 3 の倍数である. カードを操作し, $n/3$ 枚のカードを A から H へ移す.

課題 (6)　A に m 枚のハートの, B に n 枚のスペードのカードがある. m, n とも 1 以上である. $m > n$ ならば $m - n$ 枚のハートのカードを H へ, $m = n$ ならばすべてのカードを H へ, $m < n$ ならば $n - m$ 枚のスペードのカードを H へ置く.

課題 (7)　A に n 枚のカードがある. n は 1 以上の整数である. n を 3 で割った余りを m とする. m 枚のカードを H へ置く. ただし, $m = 0$ の場合は 3 枚のカードを置く.

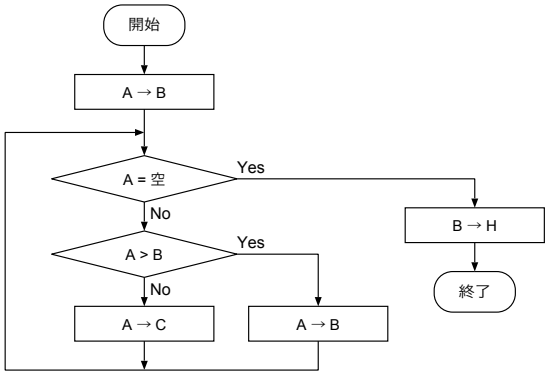
課題 (8)　A に n 枚のカードがある. n は 1 以上の整数である. n が奇数ならば A のすべてのカードを H へ, 偶数ならば D へ移す. 解答例を図 8 に示す.

5.　本環境の使用実績

5.1　入学試験

千葉工業大学 情報工学科 AO 入学試験において, 2008 年秋の試験から 7 年間使用した. 試験における, 本環境を用いた課題実習を以下に示す.

まず, 行なうことのできる操作を説明し, 次に, 最大値のカードの検出といった非常に簡単な課題の解のプログラム (フローチャート) を説明した. 説明には図 3 と図 5 を使い, さらに, 受験生に, フローチャートを見ながらシートの上でカードを操作するという手を使った作業も行なわせた. ここまでの本環境の説明にはおよそ 30 分かけている.



A にあるカードから最も大きい数値の
カードを見つけて H へ置く.

図 5　プログラム例 2: 最大値のカードの検出

次に課題を説明し, プログラム (フローチャート) 作成に取り組んでもらった. プログラムが完成した時には「カードがこのような状態でプログラムを実行して欲しい」というテストデータも何種類か与えた. プログラム作成とテスト実行の時間は 80 ～ 90 分であった.

AO 入試の合否の決定は, アルゴリズムの作成能力だけではなく多くの観点から総合的に行なう. 具体的には, 出願書類に対する審査と, 課題実習の後の面接で決定する. 面接では, 課題の解答についての質疑応答はもちろん行なうが, それだけでなく, 情報工学科の学生としての適性や, 志望の動機, 入学後の抱負なども面接で問う [3].

(課題の解答に対する) 面接の場では,

- 解答の間違いに気がついた.
- 実習時にはわからなかった正しい解を発見した.

といったことが何度もあった. 課題実習を, 入試という審査としてではなく, 受験生のアルゴリズムの学習という観点でとらえると, この面接までが学習過程になると考えられる.

課題の解答がほとんどまったく出来ておらず, 面接でもほとんど何も説明することができないという受験生はいなかった. このことから, 限られた時間 (説明と課題実習でおよそ 2 時間) に受験生が環境の操作などを理解し与えられた課題に取り組むことに大きな困難を感じることはなかったと考えられる.

5.2　大学院の授業

2016 年度と 2017 年度の情報工学専攻 大学院授業「アルゴリズム特論」において, 2 時間 (90 分 × 2 回), 本環境を使ったプログラム (アルゴリズム) の記述と実行を行なった. 環境の説明は入学試験と比べて簡単に行ない, 課題は多く与えた.

- 受講者 15 人ほど (修士 1 年生).
- 学生に感想を求めたところ概ね好評であった.
- カードの枚数をエリア (変数) の値とする課題と, カー

表 1　操作に対する考察

	エリアの値	
	カードの数値 or 記号	カードの枚数
積んだカード	スタックを構成する	枚数のみ意味をもつ
カードを取る	pop 操作	-1
カードを置く	push 操作	$+1$
空であるか	スタックが空であるか	値が 0 であるか

ドの数値あるいは記号を値とする課題の両方を出題した.

- 学生のおよそ半数は, 全問は正答に至らなかった. カードの枚数を値とする課題は比較的容易に解けたようであるが, カードの数値あるいは記号を値とする課題は難しいと感じた学生が多かったようである.
- ほとんどすべての学生が, 自分の手による実行で自分のプログラム (アルゴリズム) の間違いに出会い修正を行なった.

5.3　研究室の学部 4 年生のゼミ

昨年 (2017 年) 研究室の学部 4 年生のゼミにおいて, 大学院の授業とほぼ同じ課題に取り組むことを行なった. ここでも, 環境の説明は簡単に行ない, 多くの課題を提示した.

- 受講者 8 人ほど (学部 4 年生)
- 大学院の授業の時と比べてこちらの方が成績がよく, より短い時間でほとんど全員が全問に対して正答に至った. 研究室にはプログラミングの好きな学部生が集まっており, 大学院生よりもプログラミングの能力が高かったのではないかと思われる.

6.　環境に対する考察

6.1　エリア

エリアは, プログラムにおける変数に, エリアの名前は, 変数の名前に対応すると考えることができる.

エリア (変数) の値として以下の二つが考えられる.

- エリアの一番上のカードの数値あるいは記号を値とする.
- エリアのカードの枚数を値とする.

6.2　操作

エリア (のカード) に対する操作は, 前述したエリアの値によって意味が異なる.

一番上のカードの数値あるいは記号を値とする場合　エリアに積んだカードはスタックを構成すると考えることができる. 従って, エリア (の一番上) のカードを取る操作は, スタックに対する **pop** 操作であり, エリア (の一番上) にカードを置く操作は, スタックに対する **push** 操作である. そして, カードの移動は, 移動元エリアに対する **pop** 操作と, 移動先エリアに対する **push** 操作から構成される.

空であるかの検査は, スタックが空であるかの検査となる.

カードの枚数を値とする場合　エリアに積んだカードは枚数のみ意味を持つ[*3]. カードを取る操作は値をひとつ減らす操作であり, カードを置く操作は値をひとつ増やす操作である.

また, 空であるかの検査は, (変数の) 値が **0** であるかの検査に相当する.

以上の, 操作に対する考察を表 1 に示す.

4 節の課題 **(4)** は, $+1$, -1 及び 0 であるかの検査のみで実現した減算である. 同様に, 課題 **(5),(6),(7),(8)** は, 除算, 大小比較, 剰余の算出, 奇数/偶数の判定である.

6.3　二つのエリアの結合

二つのエリア A と B の, それぞれのデータ列 (カード群) をつなげてひとつのデータ列と考えることができる. A の一番下のカードがデータ列の先頭で, B の一番下はデータ列の最後尾, それぞれの一番上のカードはデータ列の途中の二つとなる. このデータ列は, 一ヶ所であるが途中を参照することができ, 参照位置の移動も, エリア間のカードの移動で実現できる.

これを生かした処理としてバブルソート [4] を考えた. プログラムを図 9 に示す. このプログラムの処理は以下の通りである.

A と B で大小関係が A $<$ B を満たすようにカードを入れ換えながら一枚ずつ A のカードを B へ移動させて行く. A から B への移動が終わると (A が空になると), B の一番上のカードは最小値のカードであり, E へ移動させる. B のカードを, 順序が逆にする形で A へ移動させ, 上記処理を繰り返す. B の一番上のカードを E へ移動した結果 B が空になったら終了である.

6.4　カードの大きさ

カードの数値をエリアの値とした場合, カードの大きさを, そのカードの数値に比例させることが考えられる.

数値の大小関係を扱う課題の場合は, このようなカードの使用はプログラムの実行状況の把握や間違いの発見に役に立つかもしれない. この場合, カード群の置かれたシートの形状はハノイの塔 [5] に似たものとなりそうである.

6.5　フローチャートから処理を導く学習

5 節で述べた本環境の使用例はすべて「まず課題を示し, その課題を実現するプログラムを考える」ものであった.

これとは逆に「まずフローチャートを示し, それが示す処理を考えさせる」といった学習も考えられる. フロー

[*3] ただし, 枚数を数えることはできない. 枚数がゼロであることも, 空であるか調べて初めてわかるのである.

チャート上のひとつひとつの処理は容易に理解できるのであるが, それらが組み合わさって実現している処理を想像するのは簡単ではないことがある. 従って, この「フローチャートから処理を導く」こともひとつの学習形式として成り立つと考えられる.

6.6 CS アンプラグドパターンを用いた検討

本環境は, コンピュータを使わないということにおいてアンプラグドである. CS アンプラグド [6][7] を考えて設計したのではない.

しかし, 5 節で述べた, 本環境を使った実習は, CS アンプラグドの「プログラミング言語」という学習項目におおよそ対応すると考えることができそうである. 「コンピュータはプログラム通りにしか動かない」ということの学習にこの実習はつながるのではないだろうか.

以上のように考え, 本環境を使って CS アンプラグドのアクティビティを設計する上での問題点を探るために, 本環境それ自身の特性を, CS アンプラグドパターン [8] のフォースと解決策の各項目について検討した. 検討状況を以下に示す.

フォース

- コンピュータを使わずに教えられる:
  使用する教具はシートとカードだけであり, コンピュータは使わない.
- 専門家でない教員が教えられる:
  作業の指導だけならば専門知識は不要である. しかし, アルゴリズムや, スタックといったデータ構造を解説するには, 専門知識が必要になる.
- 20 ～ 40 分で学ぶことができる:
  例えば図 4 のプログラムを解説する程度であれば 30 分程度で十分であろう. しかし, それ以上の内容を学習するには 1 時間以上必要であると思われる.

解決策

- 学習にゲームの要素を取り入れる:
  本環境それ自身にはゲームの要素はない. シートやカードを工夫すれば, ゲームの要素を取り入れることができるかもしれない.
- 教える対象ごとに適した教具を使う:
  本環境はアルゴリズム一般を対象としており, 特定のものに特化したものではない.
- 体験による学習を行う 及び 体験の中で思考錯誤しながら考えさせる:
  「作成したフローチャートに従ってシートの上で自らの手を使ってカードを操作し, 正しい結果を得たり, プログラムの間違いに出会う」「プログラムの間違いに気づいてプログラムを修正し, 再びカードを操作する」といった思考錯誤を含む経験をしつつ学習を行なう.
- グループによる共同作業を取り入れる:
  本環境は共同作業を前提としていない. 本環境を使った実習にも共同作業の要素はなかった.
- ワークシートを用意する:
  本環境自身にはワークシートの類は存在しない.
- 教具を手軽で安価に作成できる:
  シートもカードも手軽で安価に用意できる.
- さまざまな場所で行える:
  シートとカードさえあればどこでも実施できる.
- 実際のコンピュータでの利用を解説する:
  本環境は, 実際のコンピュータで使われているデータ (構造) や処理をほぼそのまま使っているため, 特別な解説は不要である.

本環境を使って CS アンプラグドのアクティビティを設計する上での問題点は, 本環境の「ゲームの要素」と「グループによる共同作業」の欠如であると言える.

7. 関連研究

[9] は, 独自に考案したカードゲームを使い, 構造化プログラミングの学習を目指したものである. ゲームは, 順次, 分岐, 反復に対応するフローチャート記号の書かれたカードを使ったものであり, 構造化プログラミングの学習は, 処理を順次, 分岐, 反復により記述することの学習である.

[10] も独自に考案したカードゲームを使い, プログラミング (アルゴリズムの作成) の学習を目指したものである. 色という値を用い, 色を変化させる順次カードと, 色の一致/不一致という分岐条件を持った分岐カード, それに反復を示すカードを並べることによりプログラムを作成する. プログラムは, あみだくじの性質を持ったフローチャートであり, あみだくじと同様に線をトレースすることによって実行する.

[11] は,「まなべー」という小学生向けプログラミング学習教材についてのものである. 画面上のロボットに対する動作指示が定義されており, これを一次元的に並べることにより, ロボットを動かすプログラムを作成する. CS アンプラグドの「プログラミング言語」の学習と, ドリトルによるプログラミングとの間を補間することが「まなべー」の目的である.

以上 3 件ともゲームの形で学習を行なうことができ, 学習者から「楽しかった」という感想が得られている.

[9][10] は, プログラムの記述に本環境と同じくフローチャートを用いている ([11] のプログラムも, 順次だけであるが, フローチャートと考えることはできる). 本環境では手書きでフローチャートを作成するのに対し, カードを並べるという手間のかからない形式をとっている.

8. 今後の課題

エリアと操作

エリアのマトリクス状の配置を生かした処理はまだ全く考えられていない. 配列の添字のようなものを導入することなく, 上下左右といった相対位置のみを用いた記述を考えたい.

前述したマトリクス状の配置にも関係することであるが, 配列あるいは配列に準ずるデータ構造を実現したい. 基本操作が複雑になることなく, 子供からお年寄りまで理解できるような単純で簡単な仕組みのみを用いての実現を目指したい.

関連研究から

ゲームの要素とそれから得られる学習の楽しさは本環境にないものであり大いに学びたいと感じている. また, フローチャート作成についても手書き以外の方法を検討したい.

9. まとめ

コンピュータを使わないプログラミング及びプログラム実行環境を提案した. 本環境の特徴は, 教具が手軽に用意できることと操作が極めて単純であることである.

フローチャートにより記述したプログラムは, 本環境上で手作業により容易に実行することができる. このため, 本環境の使用者は, プログラムの動きの詳細を自然に理解することができ, またプログラムの間違いを自ら自然に発見することが期待できる.

参考文献

[1] 六沢 一昭, “アンプラグド プログラミング及びプログラム実行環境の提案,” 第 10 回全国高等学校情報教育研究会全国大会, ポスター No.20, 2017.

[2] 六沢 一昭, 藤田 茂, “アンプラグド プログラミング及びプログラム実行環境の提案,” 第 59 回プログラミング・シンポジウム, pp.73–77, 2018.

[3] 千葉工業大学 AO 入学試験 学生募集要項, 2008 ～ 2014.

[4] https://ja.wikipedia.org/wiki/バブルソート (2018.6.3 参照).

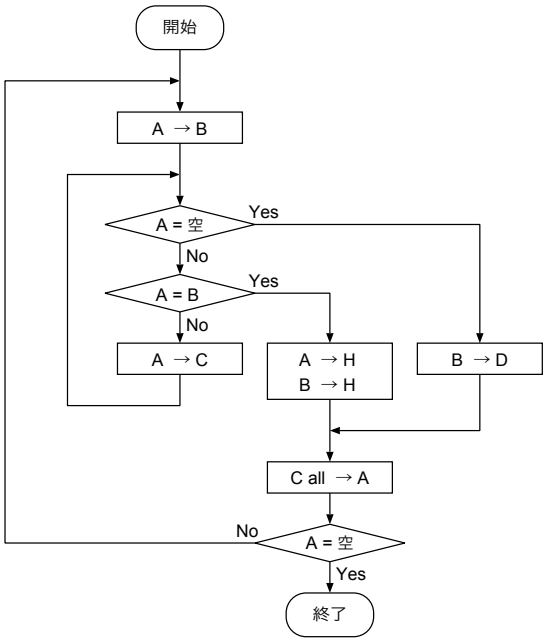
[5] https://ja.wikipedia.org/wiki/ハノイの塔 (2018.6.3 参照).

[6] Computer Science Unplugged Homepage, http://csunplugged.org, http://csunplugged.jp (2018.6.3 参照).

[7] 兼宗 進 監訳, コンピュータを使わない情報教育アンプラグドコンピュータサイエンス, イーテキスト研究所, 2007.

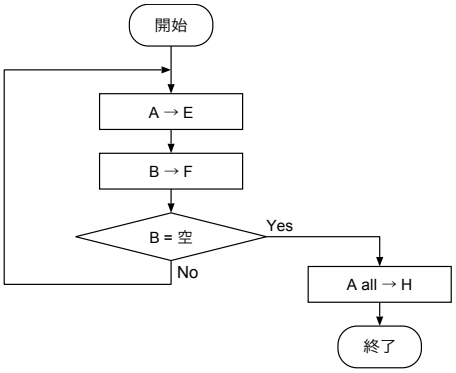
[8] 西田 知博, 井戸坂 幸男, 兼宗 進, 久野 靖, “コンピュータサイエンスアンプラグドの分析と CS アンプラグドデザインパターンの提案,” 情報教育シンポジウム 2008, pp.179–186, 2008.

[9] 望月 博文, 山田 朗, “CS アンプラグドの応用によるカードゲームを用いた構造化プログラミング学習の提案,” 日本教育工学会論文誌 36 (Suppl.), pp.145–148, 2012.



A にダイヤとクラブのカードがある.
2 枚ある数値のカードをすべて見つけて H に置く.

図 **6** 課題 (1) の解答例

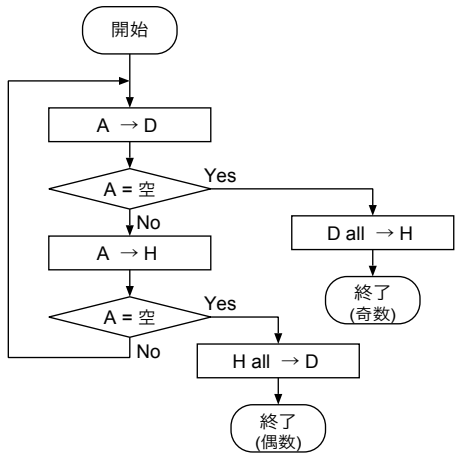


A に m 枚の, B に n 枚のカードかある $(m > n)$.
$m - n$ 枚のカードを H に置く.

図 **7** 課題 (4) の解答例

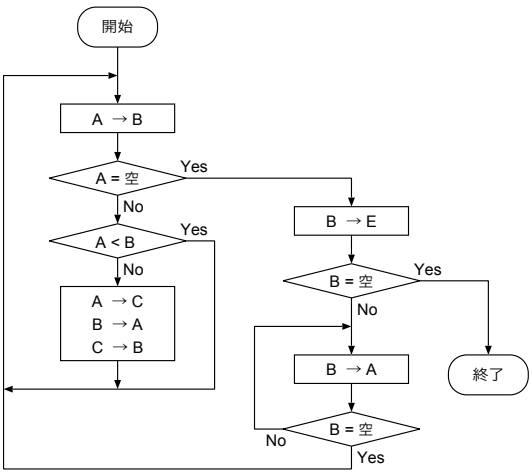
[10] 望月 博文, 山田 朗, “あみだくじを基にしたカードゲームを用いたプログラミング学習の提案 – CS アンプラグドの応用 –,” 日本教育工学会論文誌 37 (Suppl.), pp.73–76, 2013.

[11] 間辺 広樹, 並木 美太郎, 兼宗 進, “小学生向けプログラミング学習教材「まなべー」の開発と教育効果,” 研究報告コンピュータと教育 (CE), 2017-CE-131(6), 2015.



A に n 枚のカードがある. n が奇数ならば
A のすべてのカードを H へ, 偶数ならば D へ移す.

図 **8**　課題 (8) の解答例



バブルソートにより A のカードをソートする.
ソート結果は E である.

図 **9**　バブルソート