なぜプログラミングは難しいのか？
繰り返しの理解構造とCの教科書分析からのアプローチ

保福　やよい†, 西田　知博‡, 長　慎也§, 兼宗　進¶

概要

本発表では、プログラミングの学習における理解過程のモデルを提案する。そして、C 言語の学習においてつまづく学生が多いと言われる「繰り返し（ループ）」の理解過程を考察し、市販テキストの説明順序での検証結果を報告する。

Why programming is so difficult? – A proposal of a model for programming learning process and an analysis of C textbooks

Yayoi Hofuku†, Tomohiro Nishida‡, Shinya Cho§, Susumu Kanemune¶

Abstract

In this presentation, we propose a model of an understanding process while learners study programming. Especially, we focus on a process while learners understand repetition (looping) structres in C because many learners have trouble while they understand them. We also analyze the difference of description order between several C programming textbooks based on the model.

1　はじめに

コンピュータの裏にはプログラムがあることを知るのは情報社会を理解する上で有益であり、将来の職業でプログラミングを必要としなくても体験することには意味がある。

そこで、「嫌にならず、短時間で体験を通して本質を理解できる」学習が重要と考えられる。

本稿では大学のプログラミング教育で多く行われている C 言語を取り上げ、さらにその中でつまづく学生が多いと言われている「繰り返し（ループ）」の理解過程について考察する。

2　繰り返しの例

最初に、「繰り返し」について、初心者向けの教育用言語での 2 通りの扱いを見た上で C など汎用言語での扱いを比較する。

2.1　教育用言語での繰り返しの扱い

教育用言語の例として、小学校低学年や就学前児童からの体験が可能な「Viscuit」[1] を紹介する。Viscuit では、「めがね」と呼ばれる 2 つの円の中に操作対象となるオブジェクトのアイコンを置くことで、左側のパターンにマッチしたときの右側の動作を記述する。図 1 では、右側のオブジェクトの位置は左側より右にあるため、オブジェクトは右に移動する。

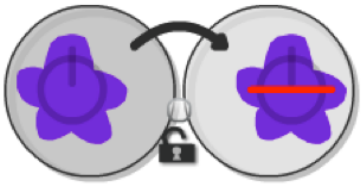

図 1: Viscuit のプログラム記述要素（めがね）

「めがね」の動作は、1 秒間に 4 回の間隔で繰り返し実行される。ここでは、暗黙の繰り返しが使われている。

このように、入門用の言語ではアニメーションを表現するために、間欠的な（繰り返しに wait を入れた）暗黙の繰り返しが使われることが多い。

このような「暗黙の繰り返し」は、繰り返しという制御構造を意識させずに作品のプログラムを作成するためには有効であるが、制御構造が明示されて

† 神奈川県立相模向陽館高等学校
Sagami Koyokan High School
‡ 大阪学院大学 Osaka Gakuin University
§ 明星大学 Meisei University
¶大阪電気通信大学
Osaka Electro-Communication University

いないため、「繰り返し」を教える目的としては適していない。

次に、小学生から使用可能な言語の例として、「ドリトル」[2] と Scratch の繰り返しの例を紹介する。

図 2 はドリトルの繰り返しである。「」で囲まれた範囲を 5 回繰り返すということがわかりやすく書かれている。図 3 は Scratch の繰り返しである。対象のブロックを繰り返しのブロックで挟むことにより、繰り返しをわかりやすく表現している。

```
「かめた！１０　歩く」！５回　繰り返す。
```

図 2: ドリトルの繰り返しの例

図 3: Scratch での繰り返しの例

これらは文字による記述とブロックによる記述のように表現方法は異なるが、どちらも「指定した範囲を指定回数だけ繰り返す」ことをわかりやすく表現することができ、「繰り返し」の学習に適している。

2.2　汎用言語での繰り返しの扱い

C などの汎用言語では、繰り返しの構文として「for」「while」「do...while」など、複数の構文が用意されていることが多い。本来は「while」などの基本的な繰り返しから学習を進めることが望ましいが、おそらく「条件を比較しながらの繰り返し」よりも「指定回数の繰り返し」のほうが例題を作りやすかったり概念的に易しいという理由から、for を中心に繰り返しを学習する教科書や授業が見られるようである。

しかし、C での繰り返しは前述の言語ほど簡単ではない。図 4 に、C の代表的な繰り返し構文である for と while の例を示す。このプログラムでは、画面に「こんにちは」を 10 回出力する。

```
int i;
for(i=0; i<10; i++) {
        printf(”こんにちは”);
}
```

```
int i=0;
while(i<10) {
        printf(”こんにちは”);
        i++;
}
```

図 4: C の繰り返しの例（for, while）

これらのプログラムを理解する前提としては、「変数」「数値のデータ型（int）」「変数の初期化」「値の代入」「比較演算子」「真偽値」「インクリメント」の概念をあらかじめ理解している必要がある。

今回は、このような「数多くの概念がいちどに現れ、それらをすべて理解する必要がある」点に C など汎用言語の繰り返し構文の理解が難しい点があるのではないかと予想した。

なお、このような C 言語の for 文に見られる特徴は、必ずしもすべての言語で共通ではないが、多くの高等教育現場で教えられていることから題材として着目した。

そこで我々は現在、「if」「while」「for」などの構文について、それらを理解するために必要な基礎概念の理解構造の検討を進めている。図 2 に if 文の理解構造の例を示す。

3　C 言語テキストの分析

プログラミング言語を理解していくための概念の理解構造の分析と並行して、市販の C 言語の入門用テキストに出現する概念の順序についても分析を進めている。

表 1 に 3 冊のテキストの分析例を示す [3][4][5]。for を中心とする繰り返しに着目した場合、2 つの点が観察された。

表 1: 市販教科書の分析例

テキスト A	テキスト B	テキスト C
変数、代入	変数、代入	変数、代入
—	配列	—
演算と型	各種演算子	演算子、型変換
if, switch	if, for, while	関係演算子
do, while, for	do, switch	for, while, do

ひとつは、理解構造を積み重ねていく学習から外れた概念を扱っていることである。具体的には、学習の流れから見て必須でない概念を、言語仕様の説明のために扱ってしまっている。たとえば、テキスト A では if の説明だけでなく switch も説明している。テキスト B ではデータ型の説明において配列を扱っており、演算子についても C の演算子をすべて説明している。

もうひとつは、説明の順序が理解構造の積み重ねと合致していないものである。たとえば、テキスト B とテキスト C では for の後で while を扱っており、for を学習するために必要な「条件比較」と「繰り返し」を理解するための if と while が、必ずしも for の前に扱われていない。また、テキスト A では、条件分岐の例題が「x % 5」のような形から説明されており、「真偽値」と「C 言語における 0 の扱い」という 2 つの概念が同時に出現している。表 2 の代替案のように、比較演算子を用いた一般的な条件比較の概念を学んだ後に、C 言語に特有の「真偽値と 0 の関係」を学んでいくほうが理解しやすいと思われる。

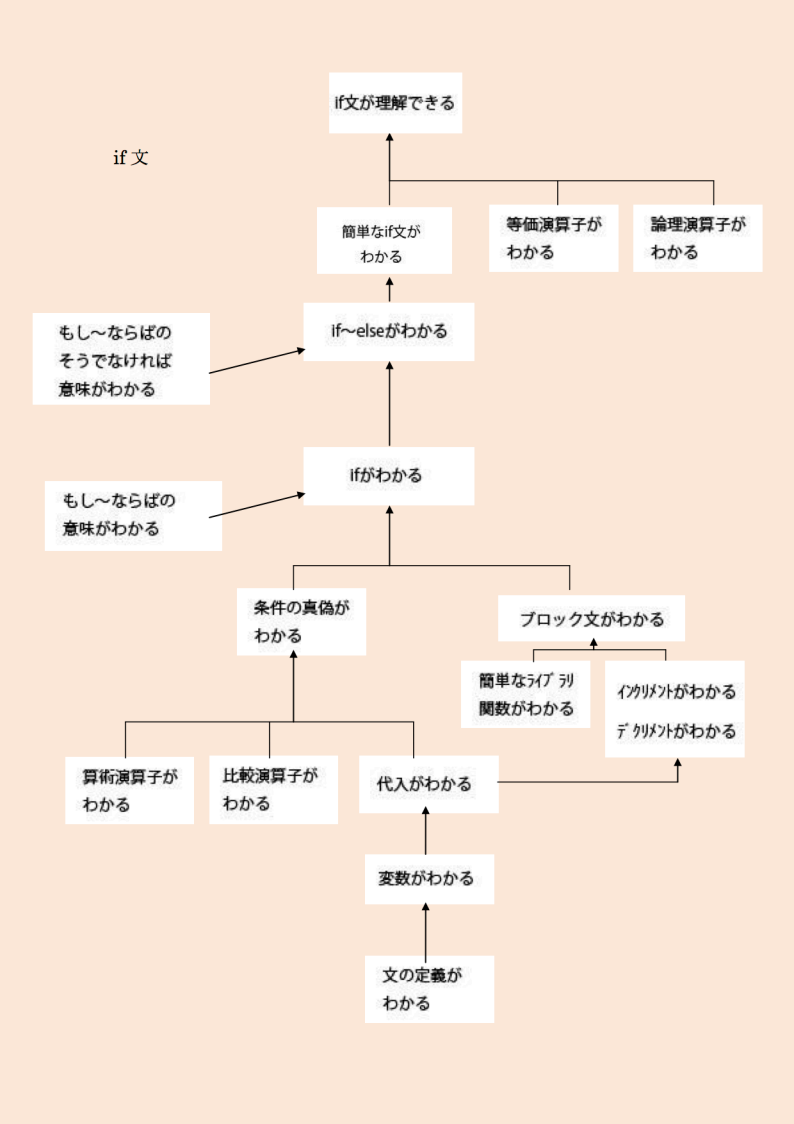

図 5: if 文の理解構造の例

表 2: 比較演算子と条件分岐の説明順の例

例題	テキスト A	代替案
3-1	`if (x % 5)`	`if (x1 == x2)`
3-2	`if (x % 2)`	`if (x1 > x2)`
3-3	`if (x % 5) else`	`if (x1 > x2) else`
3-4	`if (x % 2) else`	`if (x != 0) else`
3-5	`if (x) else`	`if ((x % 2) != 0)`
3-6	`if (x1 == x2) else`	`if (x % 2) else`

このような点を意識することにより、市販のテキストを使う場合にも、言語の理解構造に合わせて説明順序などを入れ替えることで、学習者がスムーズに理解できる授業が可能になる可能性がある。

なお、テキストの説明順については、著者の思想が反映したものであり、そのテキストを用いるための標準的な学習順序と考えられる。一方、実際にテキストで学ぶ学習者や授業で使う教授者にとって、「学びやすい」または「教えやすい」順序がそれと同じかどうかが本研究の興味である。

4 例題の差分分析ツール

我々は現在、テキストのサンプルプログラムにおける初出概念の分析を行うために、例題ごとに初出の概念を検出するツールを開発中である。これは例題ごとに C 言語の構文木を生成し、木のトップ（ルート）からのパスの差分を検出するものである。

たとえば、表 2 のテキスト A の if 文に関する 3 章の例題に適用したところ、次の結果が得られた。

- 例題 3-1 では、1,2 章の例題と比較して「if」と「if の中での剰余演算」が検出された。

- 例題 3-2 では、例題 3-1 までと比較して初出概念はなかった。
- 例題 3-3 では、例題 3-2 までと比較して「else」が出現した。
- 例題 3-4 では、例題 3-3 までと比較して初出概念はなかった。
- 例題 3-5 では、例題 3-4 までと比較して「if 内での変数」が出現した。
- 例題 3-6 では、例題 3-5 までと比較して「if 内での比較演算子」が出現した。

5 おわりに

今回は C などの汎用言語における繰り返しの理解に着目し、プログラミング概念の理解構造の検討と、それを元にした市販テキストの分析を行った。

関連研究としては、プログラミング教育に関する各種の研究に加え、学習者がプログラミングに関して獲得すべきスキルを例題化した研究 [6] などが存在する。

今回試作したツールでは、例題の解析に構文解析の情報だけを使用した。今後も引き続き、繰り返し以外の構造を含め、検討と調査を進めていきたい。

参考文献

[1] 兼宗進, 阿部和広, 原田康徳: プログラミングが好きになる言語環境, 情報処理, Vol.50, No.10, pp.986–995, 2009.

[2] 兼宗進, 久野靖: ドリトルで学ぶプログラミング, 第 2 版, イーテキスト研究所, 2011.

[3] 高橋麻奈: やさしい C 第 3 版, ソフトバンククリエイティブ, 2007.

[4] 柴田望洋: 新板 明快 C 言語 入門編, ソフトバンククリエイティブ, 2004.

[5] 林晴比古: 改訂新 C 言語入門 ビギナー編, ソフトバンククリエイティブ, 1998.

[6] 山本三雄, 関谷貴之, 山口和紀: プログラミングのスキル階層に関する研究, 情報処理学会, コンピュータと教育研究会, CE104, 2010.