段階的な設計能力の向上を目的とした ブロック育成型学習支援システムの提案

Proposal of Scalable Modular Block Style Learning System for Gradual Programming Skill Up.

古池　謙人[1]　東本　崇仁[1]

Kento KOIKE[1], Takahito TOMOTO[1]

[1] 東京工芸大学工学部コンピュータ応用学科

[1] Department of Applied Computer Science,
Faculty of Engineering,
Tokyo Polytechnic University

Abstract: Beginners tried to learn programming, it is rare to learn ideally. In that case, it is difficult for the beginners to skill up. Therefore, in this paper, improvement of programming skill is considered to be a challenge. We thought that is important, it is learning for programming to gradually by beginners. Therefore, we did proposal of scalable modular block process. This process was divided programming, standard-block with the meaning of a single line of code, and advanced-block with the meaning of function. We are using this process, we have proposed of learning support system.

1 はじめに

プログラミング学習において，初学者が自ら学ぶ場合には，書籍やウェブ上に掲載されている解説を読むと共にサンプルコードを実行し，動作を理解し，自由に改変をした結果から経験を得ることで学習を行うことが理想であり，調査（計画），実行，調整によって学習するというプロセスは，プログラミングのみならず多くの学習において非常に有意義である．しかし，現実にはサンプルコードを読む段階で学習が止まってしまう場合や，サンプルコードの切り貼りによってプログラムを作成するだけにとどまることがある．同様に，教師によるプログラミング学習では教師がサンプルコードの提示を行い（調査），生徒が提示されたコードを入力して動かす（実行）というプロセスは踏むものの，自由な調整を支援する機会が与えられない場合が多く存在する．そういった場合には，初学者は提示されたコードの各行に対する具体的な動作や，コードにおける入力と出力の関係がわからないなど十分な理解ができないことで，プログラミングにおける知識構造が不十分となり，初学者が自らプログラムを作成する場合において設計を行うことが難しくなる．特に，近年主流となっている Java などオブジェクト指向プログラミングには設計能力が必要不可欠であるため，初学者が能動的にコードに対してアプローチと失敗を繰り返し，設計能力の向上を図る必要がある．そういった背景から，初学者がプログラミング学習を行うための支援は多く行われており，なかでも初学者がコードの各行について正しく理解することを支援する研究が，渡辺ら[2]によって行われている．この研究では，金森ら[1]が提案するプログラミングプロセス（図 1）において，プログラムの読解と意味理解を行うことが学習において有意義であるとし，その手段として段階的抽象化プロセスを提案している．段階的抽象化プロセス[2]とは，与えられたコードの中で，一連の操作だと考えられる部分をまとめ，その意味を考えることを繰り返すことで，“段階的に読む” 学習を支援するプロセスである．この金森ら[1]のプログラミングプロセスや，渡辺ら[2]の段階的抽象化プロセスのように，プログラミング学習においてはプログラムを段階的に学ぶことが重要であり，これらによって初学者の設計能力が養われると著者は考える．

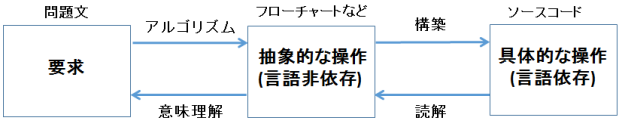

図 1 金森ら[1]によるプログラミングプロセス

また，近年では，初学者がプログラミング学習を

行う際に，ブロックを組み合わせて行うビジュアルプログラミング言語（以下，ブロック型言語）による支援が多く行われている[3]–[5]．ブロック型言語の大半はＧＵＩ上でパズルを組むような感覚でプログラムの作成が可能である[3], [5]．加えて，ブロック型言語には記述ミスがないことから，初学者による調整を容易にしている[3]．こういったブロック型言語による支援は，初学者への敷居を下げプログラミング学習への抵抗を減らすことに有用であると著者は考える．

そこで，本稿では，ブロック型言語の特徴に段階的抽象化プロセスを用いることによる学習手法「ブロック育成」と，それを用いることによる Java を対象とした学習支援システムの提案を行う．

2 ブロック育成

著者が提案するブロック育成は，前述の段階的抽象化プロセスを参考に，初学者がプログラムにおける一行ずつの処理を組み合わせて有意な塊（ブロック）を構築し，追加，及び修正を加えることによって新たに有意な塊（ブロック）の構築を繰り返すことでブロックを育てていく．こうすることで，初学者に一行ずつの処理に対する理解を促し，またその処理を用いて設計を行うことができる．この過程を，“段階的に読み，作る”学習を支援するプロセスと定義した．通常，熟達者はプログラミングの際に他人のプログラムを段階的に読む行為と，自ら段階的にプログラムを作る行為を日常的に行っている．熟達者は他人のプログラムの構造を理解し自らの知識構造に加えることで，加わった知識を用いて自らプログラムを作成することが可能となる．

段階的に読む学習の有用性については金森ら[1]によって主張されているが，加えて，従来の学習行程である段階的に作る学習を組み合わせることができれば，よりプログラミング学習に有用であると著者は考える．例えば，一行ずつの処理を組み合わせて有意な塊を構築し，それを再利用できればオブジェクト指向プログラミングにおける関数やクラスなどの概念の学習に役立てることができる．これらを踏まえて，ブロック育成においては単一的な処理であるスタンダードブロック，複合的な処理であるアドバンスブロックをそれぞれ定義した．

2.1 スタンダードブロック

スタンダードブロック（図 2 左部）は，従来のプログラミングにおける各行に相当するものとし，「int a;」や，「System.out.println(“Hello, World!”);」などの

図 2 スタンダードブロック（左）と
アドバンスブロック（右）の例

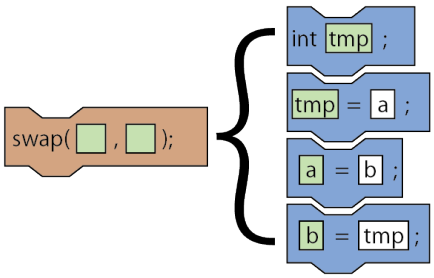

図 3 スタンダードブロック同士を組み合わせた
アドバンスブロックの例

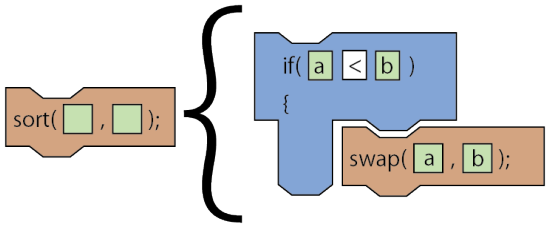

図 4 スタンダードブロックとアドバンスブロックを
組み合わせたアドバンスブロックの例

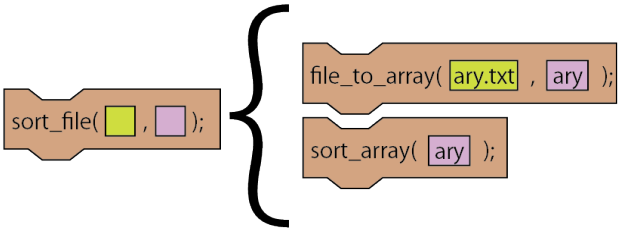

図 5 アドバンスブロック同士を組み合わせた
アドバンスブロックの例

単一な処理を行うものと定義した．

2.2 アドバンスブロック

アドバンスブロック（図 2 右部）は，従来のプログラミングにおける関数やクラスに相当するものとし，スタンダードブロックの組み合わせによる複合的な処理，すなわちアルゴリズム化された一意な塊と定義した．組み合わせの手段は三種類あり，(1) まず，スタンダードブロックのみの組み合わせからなるアルゴリズムの定義（図 3），(2) 次に，スタンダードブロックとアドバンスブロックの組み合わせからなるアルゴリズムの拡張（図 4），(3) 加えて，

アドバンスブロック同士の組み合わせからなるアルゴリズムの統合（図 5）からなる．ブロック育成では，この三種類の組み立て方法を提供し，初学者はその中から選択，構築することで設計能力の向上を目指す．

3 提案システム

本稿では，上述のブロック育成を用いた，Java を対象とした学習支援システム（以下，本システム）の提案を行う．本システムで Java を採用とした理由として，(1) オブジェクト指向プログラミング言語であること，(2) その人気から社会的に有用であることが挙げられるためである．本システムは，ブロック育成の活用として，構築すべきアドバンスブロックを目標としてあらかじめ設定し，ブロックの組み合わせや発想が容易なものから順に習得を目指すことで，具体的なアルゴリズムの理解を促す．そうすることで初学者における設計能力の向上を目的とした，プログラミング学習の支援を図る．本システムでは，大きく分けて学習モードと自作モードの二種類の機能を持ち，学習モードではアドバンスブロックの習得を目的とした学習支援を行い，自作モードでは，それまでに習得したブロックを組み合わせる自由な作成を支援する．

3.1 学習モード

学習モードでは，ブロック育成におけるアドバンスブロックの習得を，習得するアドバンスブロックとその習得順序の提案を行うことで支援する．アドバンスブロックの習得順序を学習ツリー（図 6）で提案し，その中から選択することで習得していく．一度習得したアドバンスブロックは，学習モードと自作モードの両者で利用できるようになる．

3.1.1 学習ツリー

学習ツリーは，アドバンスブロックの習得順序をガイドするために提示する．アドバンスブロックの習得順序はあらかじめ設定されたものを利用する．根元のアドバンスブロックから習得することで，より高い難度のアドバンスブロックを習得できるようになる．根元は一つではなく，ジャンル分けされた多数の根元によって成り立ち，複数のジャンルを跨ぐアドバンスブロックはツリー間に位置される．例えば，「swap」のアドバンスブロックは，データジャンルのスタンダードブロックのみからなり，「sort」のアドバンスブロックはそこに制御ジャンルのスタンダードブロックが加わる．このように，学習ツリーは習得するアドバンスブロックに対して各ジャンルのスタンダードブロックの包含関係から成り立つ．

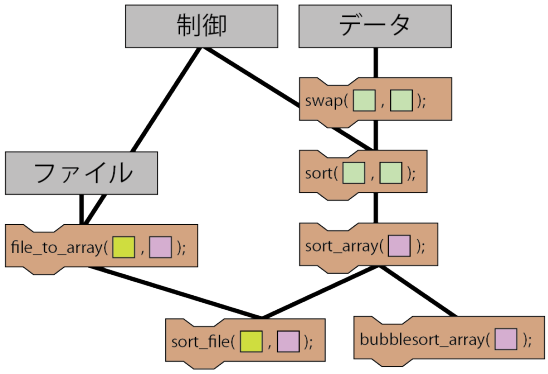

図 6 学習ツリーの一例

3.1.2 アドバンスブロックの習得

アドバンスブロックの習得画面を図 7 に提示する．左上部にはスタンダードブロックを系統的に分別したジャンル分けメニューがあり，そのメニューからジャンルを選ぶことで左下部に選んだジャンルのスタンダードブロックが提示される．提示されるブロックは，習得しようとしているアドバンスブロックの構成に必要なもののみである．右下部ではそれらのスタンダードブロックを用いた構築画面が提示され，カーソルを合わせることでそのブロックの動作が表示される（図 8）．中央上部には習得するアドバンスブロックが表示され，完成した場合には，右上部の完成ボタンを押すことで正答であれば習得し，不正答である場合には不足部分のフィードバックをシステムが提示する（図 9）．フィードバックの提示手法は正答のブロック構成と学習者のブロック構成を比較し，最も上流にあたるエラーの正答を一つ提示するものとした．例えば，図 9 の場合，「int tmp;」のブロックの後に，本来は「tmp = a;」のブロックが必要だが，挿入されていない．この場合には，「tmp = a;」のブロックが足りないというフィードバックを行う．このように，学習者は得られたフィードバックによって改善を行い，再度完成を目指すことでアルゴリズムに対する正しい理解を促す．

3.2 自作モード

自作モードでは，学習モードによって習得したアドバンスブロックや，スタンダードブロックを用いて，自らプログラムの設計を自由に行い，実行，保存できるモードである．このモードで習得したアド

バンスブロックを再利用することで，一度習得した知識構造が自らのプログラム作成に貢献することの理解を促す．

4 まとめ

本稿では，プログラミングの際における設計能力の向上を図るため，ビジュアルプログラミング言語による支援をもとに，読む学習，作る学習に，渡辺ら[2]による段階的抽象化プロセスを用いたブロック育成を定義した．ブロック育成では，従来のプログラミングにおける各行に相当するスタンダードブロックと，スタンダードブロックの組み合わせによって作るアドバンスブロックを提案し，加えて，アドバンスブロックにおける組み合わせの多様性から様々なアプローチでアドバンスブロックの作成が可能であることを示した．それらのブロック育成を用いることで，システム上におけるアドバンスブロックの習得を課題とした学習支援システムの提案を行った．

今後の課題は，ブロック習得における学習ツリーの洗練，本システムの開発である．

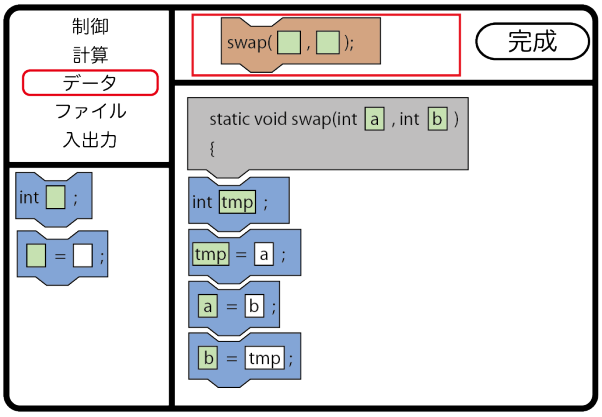

図 7 ブロック習得画面 1

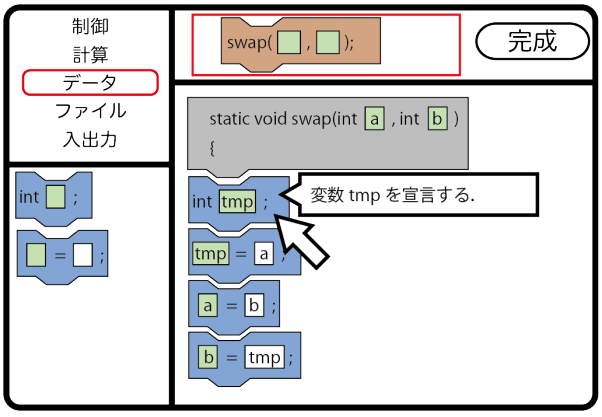

図 8 ブロック習得画面 2（ブロックの詳細表示）

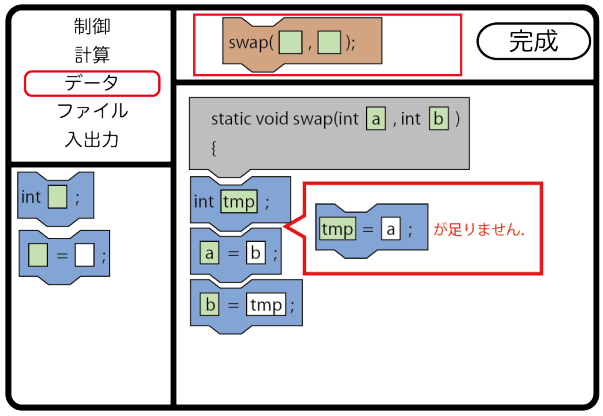

図 9 ブロック習得画面 3（フィードバック例）

謝辞

本研究の一部は科研費・基盤研究（C）（10508435）の助成による．

参考文献：

[1] 金森春樹, 東本崇仁, 米谷雄介, and 赤倉貴子, “プログラミングプロセスにおける「プログラムを読む学習」 の提案及び 「意味理解」 プロセスの学習支援システムの開発,” *電子情報通信学会論文誌 D*, vol. 97, no. 12, pp. 1843–1846, 2014.

[2] 渡辺圭祐, 東本崇仁, and 赤倉貴子, “段階的抽象化を用いたプログラムを読む学習の支援システムの開発とその評価 (教育工学),” *電子情報通信学会技術研究報告= IEICE Tech. Rep. 信学技報*, vol. 115, no. 50, pp. 49–54, 2015.

[3] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. a Y. Silver, B. Silverman, and Y. Kafai, “Scratch: Programming for All.,” *Commun. ACM*, vol. 52, pp. 60–67, 2009.

[4] B. Harvey and J. Mönig, “Bringing ‘No Ceiling’ to Scratch: Can One Language Serve Kids and Computer Scientists?,” *Constructionism*, pp. 1–10, 2010.

[5] 松澤 芳昭, 保井 元, 杉浦 学, and 酒井 三四郎, “ビジュアル-Java相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価,” 情報処理学会論文誌, vol. 55, no. 1, pp. 57–71, 2014.