論理的思考力育成のためのプログラミング学習

八木　　徹*・山口　敏和**

要　約

プログラミングの学習には様々なハードルが存在する。さらに，オリジナルソフトウェアの開発のためにはプログラムの文法的学習だけでなく，ソフトウェアの開発工程全体を知る必要がある。一方，問題解決のために論理的思考力を育成するという立場に立てば，ソフトウェア開発工程は，目標を定め，計画を立ててそれを実行し，成果を振り返るという過程であるため，問題解決力の育成に向けた良い練習素材となり得る。その目的のために，コンピュータ上でのプログラミングから離れ，コンピュータを用いないアンプラグドのソフトウェア開発に取り組んだ。さらに，プログラマ役とコンピュータ役に分かれたロールプレイを行い，プログラムはコンピュータのプログラミング言語ではなく，日常用いる自然言語で記述することとした。これにより，短い期間でソフトウェアの開発工程全体を体験することができた。

キーワード：論理的思考力，プログラミング学習，アンプラグドソフトウェア開発，自然言語プログラム

1. はじめに

プログラミングの学習においてハードルとなるポイントは多様である。著者らがこれまでにプログラミングの指導をしてきた経験では，命令文の書式とその使い方を把握すること以外に，英単語に対する拒否感，覚えられないものは理解できないものだという思い込み，プログラムを読解できず指示内容を把握できないなど，人により難しいと感じる点はさまざまである。

また，プログラミング言語を学習してその文法を修得しても，自分ではプログラムが組めない，すなわち「コンピュータに何をさせたいのかがわからないため，オリジナルのプログラムが組めない」という訴えも多い。これは道具としてコンピュータを活用することを考えた時に，一種の本末転倒が起きているとも言える。

2017 年 11 月 30 日受付
* 江戸川大学　情報文化学科准教授　　情報科学
** 江戸川大学　情報文化学科専任講師　情報教育

コンピュータを使ってやりたいこと，こんなことができたらいいなという希望があり，それを実現するためにコンピュータのプログラミング言語を学ぶとなれば，その目的に向かって試行錯誤を重ねていくことができる。しかし，いくら文法的な要素を学んでも，コンピュータで何をしたいのかという動機が無いと上達は難しい。そういった意味では，使い道のないまま外国語を学習するのと類似している点があるといえる。

このような場合，「プログラミングができるようになりたい」という目標よりも，「○○のゲームを作りたい」という動機の方が，学習の原動力として効果的であると言える。しかし，現在のコンピュータのシステムは GUI で使いやすいよう高度に発展しているため，ゲーム一つを作ろうとした際でも，初学者にとって学ぶべき項目が多く，完成までの道のりは長い。このため，学習している要素を積み重ねても，作りたいゲームにたどり着くというイメージを持ちにくくなってしまう。現在，プログラミング学習の導入として広く用いられている Scratch のようなビジュアルプログ

ラミング言語は，GUI 環境における簡便かつ効果的なプログラミング学習環境と言える。

しかし，ここでコンピュータ上でのプログラミングにこだわらず，問題解決に向けた論理的思考力育成のための手段として，ソフトウェア開発を考える。目標とするソフトウェアの完成に向けた開発工程は，要件定義，設計，実装，テストといった段階から成り立っており，これらは問題解決のフローとしても活用できる。したがってソフトウェアの開発工程を体験することは，問題解決力を養うためのトレーニングになり得る。そこで，本研究では，コンピュータを使わない，いわばアンプラグドのプログラミングに取り組んだ。習得に時間のかかるコンピュータ用のプログラミング言語は用いず，プログラムを自然言語で記述することとし，目標達成に向け，設計からテストまでの開発工程全体を体験する試みを実施した。

2. プログラミング学習時の障壁について

プログラミングの学習をする際，初学者にとって障壁となり得る点を以下に列挙する。

- 命令文の書式とその使い方を把握すること
- 英単語に対する拒否感
- 覚えられないものを理解できないと感じる思い込み
- プログラムを「読む」ことができない
- オリジナルプログラム作成につながらない

ここでは特に，プログラムを「読む」ことと，オリジナルプログラム作成への障壁について詳しく述べる。

2-1. プログラムを「読む」こと

プログラムのソースコードには，コンピュータで実施する処理とその手順が記述されている。したがって，それを読むことで，どのような指示が出されているのかを把握することができる。しかし，初学者にとっては，ソースコードを読み解くには困難が伴う。命令文の書式自体が把握できていることと，そこでどのような処理が行われるのかを読み解くことの間には大きな開きがある。

図 1 に解説のための仮想的なサンプルコードを示す。図 1-(a)は if による条件分岐を用いて処理を選択するものである。初学者がこのようなプログラムを見た時に，if 文の書式に従って命令を見ても，何が行われるのか意味が分からないと感じることがある。

そこで，図 1-(b)のように，命令を日本語で記述する。このようにすることで，処理を「文章」として読むことができるようになる。このため，プログラムで指示された内容を理解しやすくなる。このように，ソースコードを意味のあるものとして「読む」ことができると，どのような作業が指示されているのか把握ができるため，納得しやすい。

もちろん，実際のプログラムは，処理がさらに複雑なものとなるため，このように単純化できないものが多い。しかし，ソースコードを記号の羅列ではなく，意味のある言葉として「読める」ことが重要である。

```
priceOfApple = getPrice( );
if ( priceOfApple ＜ 100 )  {
      buyApple( );
}   else     {
      buyOrange( );
}
```

図 1-(a)　仮想的なソースコード

```
リンゴの値段を設定する;
もし (リンゴの値段が 100 円より安い) ならば {
      リンゴを買う( );
} そうでなければ {
      オレンジを買う( );
}
```

図 1-(b)　処理を日本語に置き換えた仮想コード

プログラムを書くことと，論理的な文章力を持つことの因果関係は不明である。しかしプログラム自体は，作業手順を示した指示書として，矛盾なく論理的に整合性のとれた文章となっている必

要がある。すなわち，プログラムの作成ができるようになるということは，相手に対して的確に指示を伝達する文章が書けるようになるということでもある。

このようにとらえると，図 1-(b)のように，日常用いる自然言語でプログラム作成を練習することで，他者に正しく伝わる指示書を作成する訓練になると考えられる。このため，後述するように，プログラマ役の人間が自然言語でプログラムを記述し，コンピュータ役の人間がその指示に従って処理を行うという，ロールプレイによる作業を実施した。

2-2. オリジナルプログラム作成への障壁

プログラミングの学習を一通り行い，基礎的な文法を理解したとしても，具体的にコンピュータで何かができるようになる気がしないという問題もある。これにはそもそもコンピュータで何かをしたいという「やりたいこと」が希薄である，という点も含まれるが，基礎的なプログラミングと「やりたいこと」がつながらず，オリジナルのソフトを作れるような気持になれないという思いを持つ学習者も多い。

例えば，学習のサンプルとして，掛け算の九九の一覧表を作ったとしても，それが初学者の好奇心を喚起する例題にはなりにくい。そこで，グラフィカルな UI を持つゲームを簡便に作成できれば，それは興味をそそるサンプルになり得る。しかし，Window の処理や画像関連のライブラリ利用など，さらなる学習を要す項目が増えてしまう。

このように，プログラミング基礎の学習と，学習者が興味を持てるオリジナルソフト作成の間には大きな障壁がある。そこで，初学者の学習意欲を喚起しつつ，プログラミングの理解を深めるための工夫が重要となる。

さらに，プログラミング学習の初期では，言語そのものの理解に時間が割かれるため，ソフトウェア開発の全体像が見えにくい。しかしソフトウェア開発の全体においては，「要件定義」や「設計」「実装」「テスト」などの工程が存在する。通常のプログラミング学習で行う「コーディング」は，「実装」の一部分でしかない。

実際，ソフトウェアを開発する際，いきなりコーディングから始まるということはなく，要件定義において作るべきものを定めることから始める。例えば，システムに必要となる機能や，どのような情報をどのように保持するかというデータ構造，そのほか，システムを操作するために必要となるユーザインタフェース（UI）等を定める。次に，要件定義で決定した事柄について，具体的な実現方法を決めるための設計を行う。その上で実装を行い，制作作業に入る。さらに，制作したソフトが，要件で定められた適切な動作をするか検証するためのテストを行う。これらの工程がソフトウェア開発の全体像であり，これらを把握することが，オリジナルソフト作成に必要となる。

ここで，ソフトウェア開発工程を，問題解決のためのステップとして捉える。そのために，開発工程の各段階を，図 2 のように表す。すなわち，要件定義において「何をしたいか」という目標を定め，設計では「どうやって実現するか」という目標実現の手段を考え，実行で「実際の制作（作業）」という具体的な行動を起こし，振り返りにて「思い通りに実現できたか確認」という検証を行う。

このように，ソフトウェアの開発工程のフローは，ソフトウェアの開発にとどまらず，一般的な問題解決に必要な要素に対応している。このため，適切な問題設定をした上で開発工程を体験し，問

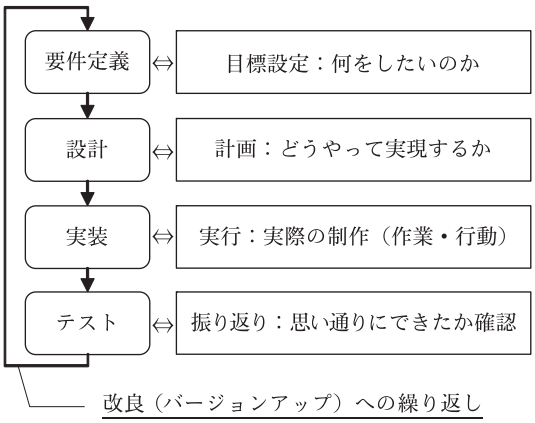

図 2　問題解決に向けた作業工程

題解決力を育成することを考える。

2-3. 問題解決に向けた論理的思考力育成

これまで述べてきたように，今回の取り組みでは，コンピュータ上でのプログラミングやソフトウェア開発から離れ，問題解決に向けた論理的思考力を育成するという視点に立つ。すると，不慣れなコンピュータ用のプログラミング言語を用いる必要は必ずしもない。日本語で書く文章の論理性を高めることが重要であり，相手に正しく処理手順を伝達することができればよい。

そこで，本研究では，学習者をプログラマ役とコンピュータ役に分けたロールプレイを行うこととした。プログラマ役が自然言語でプログラム（指示書）を記述し，それをコンピュータ役が受け取ってその指示通りの行動をする。アイコンタクトや言葉による意思疎通を行わず，プログラムに書かれたことのみで処理を行うことで，プログラムに記載された指示手順が正しく伝達できるかを判定する。

プログラマ役が記述したプログラムに不備があれば正常な作業は実施できないが，正しい指示が出されていたとしても，プログラムを読み取るコンピュータ役が正確に指示に従うことができるか，という問題もある。したがって，一度でうまく指示が伝達できるかと言うよりも，プログラムを実行するたびに振り返り，うまく伝達できた指示とそうでなかったものを検証し，改良していくことが重要である。

このような自然言語で記述したプログラムを用いることで，仮想的なソフトウェア開発を短時間で体験することができる。要件定義からテストまで，一連の開発工程を体験することで，目標に向け筋道立てて一貫した思考をし，必要なことを自分調べ，考え，構築していくという問題解決に向けた力の育成につなげる試みとなる。

3. コンピュータを用いない「アンプラグド開発」

取り扱う題材を，初学者が興味持てるものとするため，いわゆる落ち物系のブロックパズルとした。このブロックパズル開発を目標として，要件定義，設計，実装，テストの各工程を体験した。プログラムは日本語の文章で記述することとし，コンピュータ役の人間がプログラムに従ってゲーム進行の作業を実行した。

各段階の作業ではワークシートを用意し，学習者それぞれが実習を行った。各ワークシートの内容を以下に示す。

○ 「要件定義」のワークシート
- タイトル：何を作るのか決める（仕様）
- 内容：落ち物ブロックパズルの要素を箇条書きで書き出す。

箇条書きにしたブロックパズルの要素を，次の3つの視点で整理する。
- UI（ユーザインタフェース）：どんな操作をするのか，どんな情報を送り込むのか，どんな情報を受け取るのか，という点を箇条書きにする。原則的に名詞で書く。
- 機能：どんな機能を持つか，どんな作業をするのかを書き出す。原則的に動詞で書く。
- データ：どんな情報を記憶する必要があるかを書き出す。

○ 「設計」のワークシート
- タイトル：どうやって作るのかを決める（設計）
- 内容：要件定義で書き出した項目について，実現方法を検討する。

次の3つの分類ごとに具体的な実施の方法を書き出していく。
- UI：要件定義でピックアップしたUIを実現する方法を考える。（例えばブロックの移動回転などの操作に対して，声やゼスチャーでの指示を出すなど）
- 機能：動く仕組み，からくりを考える。（ブロック回転の中心を決めて操作担当の人間が動かすなど）
- データ：必要な情報の記録方法（ゲームのスコアなど）

○ 「実装」のワークシート
・実体としてのゲームの要素作り
　ここでは制作作業が中心となる。ブロックパズルやゲーム盤面を作る。
・指示書（プログラム）の作成
　プログラムは日本語で記述する
　1つの指示が1つの作業に対応するように書く
　分岐や反復の構造を意識する

プログラムについては，いきなり完成形を作るのではなく，次のステップで次第に完成度を高めるようにした。

1） まず，ブロックが出現し，下に落ちて，止まる，このプログラムだけを書き，コンピュータ役の人がその手順を実施できることを確認。
　うまくいかない場合，問題点を特定し，改善のための対策を検討する
2） ブロックの落下中に，左右に移動できるプログラムに改良
3） ブロックの落下中に，回転できるプログラムに改良

○ 「テスト」のワークシート
・タイトル：動作の検証（テスト）
・内容：指示書（プログラム）を実行し，仕様通りに動作できるかを確認する

具体的には，プログラムを実行してその動きを確認する。その際に以下の点に注意する。

✓プログラムを書いた人は実行に口出ししない
✓コンピュータ役（プログラムを読み実行する人）は，書いてあることだけを忠実に実行するよう心がける（プログラムを書いた人の気持ちを忖度しない）

全体の改良に取り組むには，「要件定義」から「テスト」までの工程を行い，テスト結果を踏まえた振り返りを反映させたバージョンアップを行う。バージョンアップのために再度「要件定義」からの工程を繰り返すことになる（図2）。しかし，今回は「要件定義」，「設計」，「実装」，「テスト」の1工程のみを実施した。

4. 実際の作業

図3から図5に，実際に制作したブロックパズルを示す。パズルのフィールドは，ホワイトボード上にテープでマス目をつけたものとした（図3）。ブロックはマグネットを4個組み合わせて作った。ブロックを移動しやすくするために，ブロック形状に合わせた型を作成し，その中にマグネットを並べることとした（図4）。この他，簡易的に折り紙を並べてテストするグループもあった（図5）。

作成したプログラムの一例を図6に示す。【下線部】の文は，当初記述されておらず，テスト実行をしながら追加した。④の処理に引き続き，a, b, cのチェックを行い，これを繰り返すものであるが，これが抜けていたため補ったものである。

ここで注目したいポイントは，④の「メトロノームのリズムに合わせ」という部分である。今回設定した目標が「落ち物パズルゲーム」であったため，タイミングを取る作業が必要であった。そのための工夫として導入されたものである。このような工夫が必要という点では，この素材は難易度が高いものとなってしまっているといえる。より扱いやすい素材とするためには，ターン制のよう

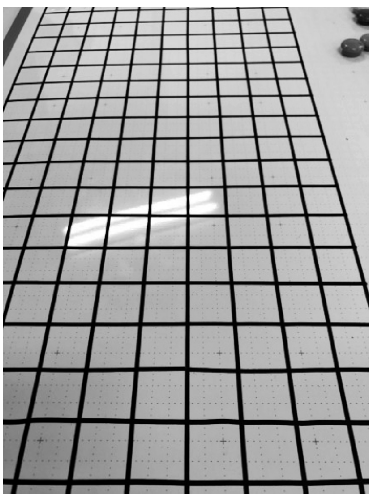

図3　パズルのフィールド

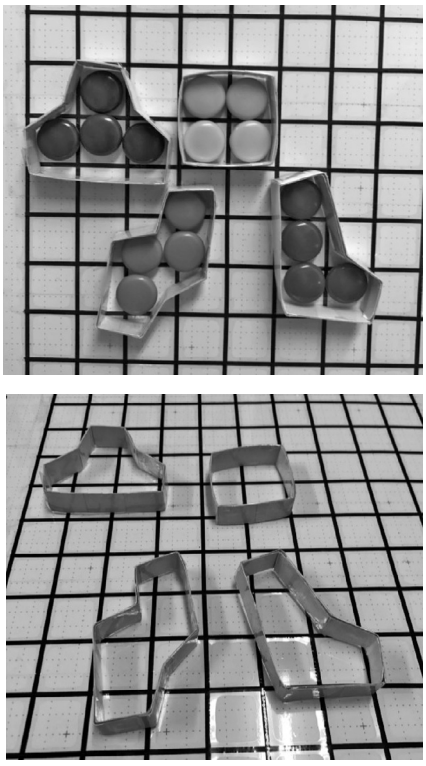

図4　ブロック（マグネット（上）と型（下））

① ブロックの型が入った袋の中から，中を見ずに1つの型を取り出す
② ホワイトボード上の準備場所で，型の中にマグネットを4つ入れる
③ スタートの合図がした，または前のブロックが止まったら，準備場所のブロックをスタート地点にセットする
④ メトロノームのリズムに合わせ，ブロックを1マス分，フィールドの底に向けて移動させる。【その後以下をチェックする。】
　a もしプレイヤーが「←」の札を挙げて振ったならば，振った回数分のマス目だけ左に移動
　b もしプレイヤーが「→」の札を挙げて振ったならば，振った回数分のマス目だけ右に移動
　c もしブロックの底がフィールド底に触れた，または，すでに積まれているブロックに触れたならば，その場で止まる。【そうでなければ④へもどる。】
⑤ もしブロックがフィールドの最上面より上に積まれていたらゲームを終了する。そうでなければ①へ戻る。

図6　ブロックパズル「プログラム」の一例

なリアルタイム処理の不要なものが適している。

実際に，この部分のテストでは，コンピュータ役が現在の処理を見失いやすいという問題が出た。そこで，改善策として，プログラムの進捗を一つずつチェックして進められるようにすると良い（今どの命令を処理しているか明確化する）という対策案が出た。処理の箇所を示すマーカーを，リズムに合わせて移り変えていくようなものである。このほか，プログラムを読む作業をする人物と，処理を実行する人物のように，コンピュータ役の人間を増員する方が良いという意見も出された。

この改善で，ゲームの処理は図7のように行わ

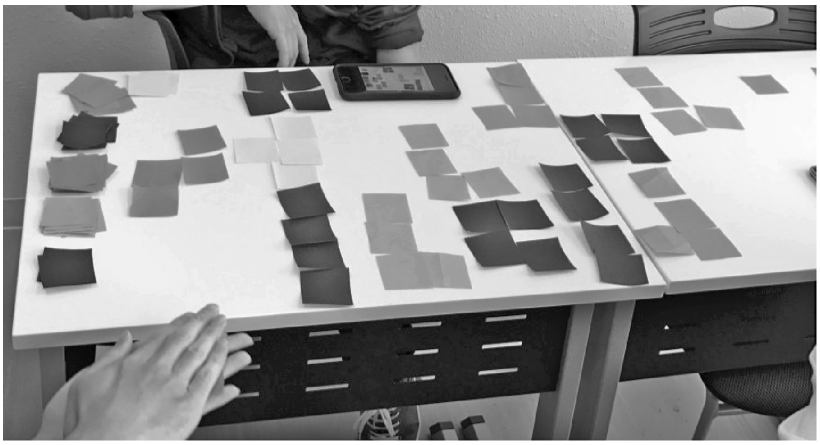

図5　折り紙を用いたブロック

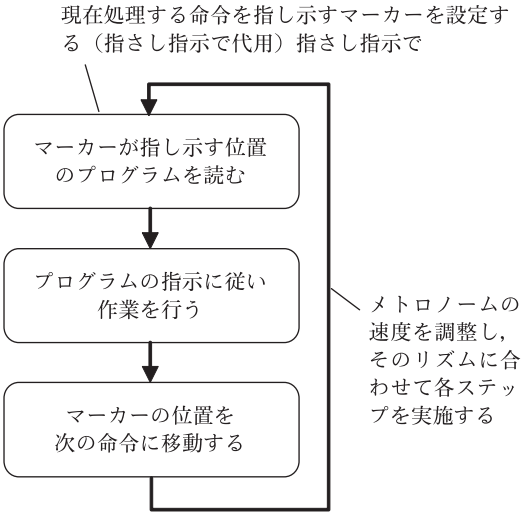

図 7　実行処理の流れ

れることになる。ここで，メトロノームのリズムを CPU の動作クロック，マーカーをプログラムカウンタ（PC）と見立てると，図 7 の動作は，CPU のフェッチ・デコード・実行のサイクルに対応するようで興味深い。

今回の試みに取り組んだ学生からは以下のような意見が出された。

- ただプログラムを組む，という作業ではなく，ゲームを作る過程を簡易的にでも体験できた点が興味深かった
- 相手が理解できるようにプログラムを書くことが難しかった
- 「誰が実行しても同じように動作するプログラム」となるようにする工夫が必要
- 当たり前のように操作できてしまうことを，厳密に記述しようとするのは，非常に難しかった
- 実行すべきルールがわかっていても，いつどこでどの動作をするべきなのかを考えるのが難しかった。
- 目標とするプログラム作成に必要な工程と，プログラムのコーディングそのものの違いが体験できた。このため，プログラムを学んでも，何も作れないなと思っていた理由がわかった
- 今回の制作を経験して，Scratch でなら同じプログラムが作れそうに感じた。

これらの感想から，今回の作業で，必要な指示を適切に相手に伝える工夫の必要性や，ゲーム作成という目標に向けて，段階的に取り組む流れを意識できたことがわかる。

5. ま と め

今回，コンピュータを用いない開発体験の試みを実施した。コンピュータ上でのコーディングを離れることで，短い時間で開発の工程を一通り体験することができた。この体験は，基礎的なプログラミング学習とソフトウェア開発の溝を埋めるものであり，さらに，目標に向けて必要な行動をとるという問題解決に向けた取り組みにもつながるものである。

今回の取り組みでは，プログラマ役とコンピュータ役のロールプレイを行うことで，自然言語でのプログラム記述と，その指示を受けての作業を行うことができた。これは，それぞれの立場で，意図した結果につながる指示の出し方を考えるきっかけとなった。プログラマ役にとっては意味が伝わる指示を工夫すること，コンピュータ役にとってはプログラムを「読む」という訓練になる。いずれにしても，相手に対して的確に指示を伝達する文章を工夫する体験となった。

落ち物ブロックパズルは，リアルタイムでの進行のため実行が難しい。ターン制のゲームの方が現実的である。しかし，副産物として，CPU の動作サイクルを再現するような結果が得られた。

参考文献

［1］ 玉田和恵，神部順子，八木　徹，山口敏和，鈴木哲平，重藤　暁，松村豊子，個に応じたキャリア教育を実現するためのファカルティ・ディベロップメントの取り組みⅨ ── 情報化・グローバル化に対応した人材の育成 ──，江戸川大学紀要，第 27 号，pp. 255–264（2017）