研究論文

鳴門教育大学情報教育ジャーナル No.16 pp.21-26 2019

ティンカリングとしてのプログラミング

藤原伸彦[*1]，阪東哲也[*2]，曽根直人[*2]，長野仁志[*3]，山田哲也[*4]，伊藤陽介[*2]

平成 29 年に学習指導要領が告示され，2020 年度以降小・中・高等学校でプログラミング教育が実施されることとなった。プログラミング教育の円滑な実施に向けて，様々な実践事例が提案されている。だがその大半は論理的思考の育成を目的としたものである。プログラミングは創造的思考を育成するのに非常に適した活動であり，かつ，その育成は期待されているものである。本稿では，プログラミングをティンカリングとして捉え，創造的思考力―試行錯誤の方法やその良さへの気づき，物事に根気よく取り組む気持ちなど―の育成の場として考える。プログラミングによって，論理的思考と創造的思考の両面が育成されることが期待される。

[**キーワード**：プログラミング教育，プログラミング的思考，創造的思考，ティンカリング]

1. はじめに

平成29年に学習指導要領が告示され，2020年度以降，小・中・高等学校でプログラミング教育を実施することが示された。着目すべきは，技術や情報といったプログラミングを扱う教科のない小学校でもプログラミング教育が導入されることになった点である(文部科学省，2017a)。新たに始まる小学校におけるプログラミング教育に関して，文部科学省(2018)は「小学校プログラミング教育の手引き(第二版)」や「未来の学びコンソーシアム」Web サイト[1)]を公開するなど，円滑な実施に向けて準備を進めている。とはいえ，プログラミング教育をどのように進めていくべきかは，これから議論を深めるべき課題である。

特に，プログラミング教育を意味のあるものにするためには，そこでどのような力を育てようとするのかについて十分に検討しなければならない。「小学校段階における論理的思考力や創造性，問題解決能力等の育成とプログラミング教育に関する有識者会議」(以下，有識者会議とする)は，その報告「小学校段階におけるプログラミング教育の在り方について(議論の取りまとめ)」において，プログラミング教育の目的は，プログラムを記述するスキル，すなわち「コーディング」を覚えることではなく，「将来どのような職業に就くとしても，時代を超えて普遍的に求められる力としての『プログラミング的思考』などを育成する」ことであると述べている(文部科学省，2016)。そしてその「プログラミング的思考」については，

> 自分が意図する一連の活動を実現するために，どのような動きの組合せが必要であり，一つ一つの動きに対応した記号を，どのように組み合わせたらいいのか，記号の組合せをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力

としている。文部科学省によるこの定義は，いわゆる Computational Thinking(CT)を踏まえつつプログラミングと論理的思考の関係を整理したものとされている。これは，プログラミング教育を実践する際に一定の方向性を示しており，現在「未来の学びコンソーシアム」Web サイトなどに蓄積され始めた実践事例の多くもこの定義にそって行われている。だが，阪東ら(2017)によれば，有識者会議の議論の取りまとめには，CT の考え方がどのように踏襲されたかについて言及されていないこと，また林(2018)によればCT等に関する言説の状況は，Wing(2006)以降日本国内での検討作業が十分ではないことが指摘されている。プログラミング教育が始まったばかりの今，この定義そのものについても検討していくべきであろう。

本稿では，文部科学省が柱の一つとして掲げてい

*1 鳴門教育大学 地域連携センター
*2 鳴門教育大学 情報基盤センター
*3 鳴門教育大学 附属小学校
*4 鳴門教育大学 附属中学校

る「プログラミング的思考」をどのように捉えるべきかについて論じる。特に，創造的思考の方法論である「ティンカリング」という概念に着目し，論理的思考以上のものとしてプログラミング的思考を捉えたい。

2. 論理的思考としてのプログラミング的思考

プログラミング的思考あるいはCTが含んでいる要素については，様々に整理されている(阪東ら，2017；林，2018)。例えば，イギリスのComputer At School(CAS)は，「CTは論理的推論に関わる認知的あるいは思考過程であり，問題を解決したり，人工物や手続き，システムをよりよく理解したりするのに使われる。」とした上で，①アルゴリズム的に考える能力，②分解の観点から考える能力，③一般化して考える能力(パターンを見出す)，④抽象化して考える能力，⑤評価の観点から考える能力，の5つを含んでいると述べている(CAS，2015)。ベネッセ(2018)は，文部科学省のプログラミング的思考の定義をもとに，①分解，②抽象化，③一般化，④組合せの4つの要素を挙げている。

これらを参考に，ここではプログラミングにおける論理的思考の要素として，(1)分解，(2)アルゴリズム的に考える，(3)抽象化，(4)一般化を考える。

2.1 分解

プログラミング中の思考法の一つとして，実現したいことを下位のパーツに分解する，ということが挙げられる。例えばワープロソフトを考えてみてほしい。一言にワープロ，といっても，それは文字の入力，画面への表示，漢字変換，保存，印刷…といった複数の機能が組み合わさっている。また，それぞれのパーツも，さらに下位のパーツが組み合わさって作られている。プログラミングをする際には，どのような機能があるのかに分割して，それぞれを作る。この大きな課題を下位課題に分割して個々に取り組んでゆくという方法は，一般的な問題解決方略の一つである。プログラミングは，そのような思考の方法を学ぶ良い場となる。

また，個々のパーツは，最終的に使用しているプログラミング言語で準備された命令群(コード)で表現できるレベルにまで分解していく必要がある。このレベルにまで分解することは，物事や行為を分析的に記述することに関わっている。物事や行為を分析的に記述するのは，実は難しい。行為が意識しないものであったり，物事が我々にとって日常的であったりすると，なおさらである。そのことを感じることができる良い題材として，NHK Eテレの番組「デザイン　あ」の一コーナーである「考えていない」[2)]がある。このコーナーでは，水を飲んだり上着を着たりといった日常の何気ない行為をスローで再生して，ナレーションでやっていることを説明していく。例えば「本をめくる」では，文庫本のページをめくる映像に次のようなナレーションがついている。

> まず左の親指が少しずつ外側へずれていきます。指の腹の感覚を研ぎ澄ませて，見事に一枚だけを浮かせます。そして両手で本を閉じると見せかけてその反動で浮かせた一枚を右側へ送ります。右手の親指は向かってくる一枚をスッとかわし上から押さえこみます。無事に一枚をパスし終えた左の親指はゆっくりとまた元の位置に戻ります。親指どうし，素晴らしいコンビネーションです。

日常的な行為について同じように説明してみるとわかると思うが，その行為を分析的に見て言葉にしていくことが求められる。プログラミングは，1つの機能を下位の機能に分解するのにとどまらず，さらにパーツに分解して記述していくことが求められるため，分析的に物事を見る力の育成につながると予想される。

2.2 アルゴリズム的に考える

分解されたステップは，想定された機能を果たすために，一定の順序で並べられる必要がある。そのようにして作られたものを，その機能を果たすためのアルゴリズムと呼ぶことができる。ステップの順序が変わると，想定していた機能を実現することができなくなる。Scratchを使ったプログラミングに関するNHK Eテレの番組「Why!?プログラミング」の一コーナー「ジェイソンをプログラミング」[3)]では，スイカ割りの動きを分解し，スイカに近づく途中で手を振り下ろすとスイカは割れないが，スイカに近づいた時点で手を振り下ろすとスイカを割ることができる，という例を使って説明している。この例はいわゆる「順次」というプログラムの処理の構造のうちの1つに関わるものであるが，同じ処理を何度か「反復」(繰り返し)したり，「もし～なら…する」のような条件によって「分岐」したり，といった構造にステップを並べることもある。

人間の問題解決方略の1つに，ヒューリスティクスがある。ヒューティスティクスとは，「必ず正解にたどり着くわけではないにしても大抵は正解を得

られ，簡単に利用可能な，近道として使える手段」のことである(日本教育工学会，2000)。素早く利用できるため日常生活では有効に働くが，バイアスがかかり時として正確な回答が得られない場合がある。アルゴリズムは，人間の問題解決の他の方略である。ヒューリスティクスと補完的に働かせることによって，より良い問題解決を行えるようになる。プログラミングによって，そのようなアルゴリズム的に思考する力の育成が期待できる。また，アルゴリズム的に考えることは，教科の学習の中の，数学の証明問題や，家庭科において効率的に料理の手順の考える，といったことと関連している。

2.3 抽象化

抽象化という頭の使い方も，プログラミングの最中にしばしば行われる。例えば，あるプログラムを作っていて星形を描く必要があったとする。一般的なプログラミング言語には星形を描くというコードは含まれていないが，線を描くというコードが含まれている。ここで，画面上の具体的な 5 つの点をつなぐプログラムを書くこともできるが，「ある点から一定距離を移動して 144 度回転する」を 5 回繰り返すプログラムでも星形を描くことができる。これは，例えば算数で「リンゴが 3 つあって，さらに 2 つ買ってきたら合計で 5 つになりました」という事象を，リンゴという具体的な表象を考えずに「3 つのものに 2 つのものを加えると 5 になる」さらには「3+2=5」を考える，というのに似ている。

抽象化とは，現実の事象から，その問題解決に関係のある情報を取り出し，取り出した情報間の関係性(構造)を捕まえることだといえるだろう。そのような頭の使い方は，問題場面を把握するのに役立つ。CAS(2015)をはじめ，抽象化の例としてしばしば取り上げられるのは，電車の路線図である。路線図は，実際の地理的な情報から，駅間の距離や(正確な)方向といった情報を捨象し，駅と，駅と駅の関係性のみを取り出して描いたものである。切符を購入したり，目的地に行くのにどこで乗り換えれば良いかといった『問題』を解決したりするのに利用できる。

2.4 一般化

一般化は，事象間の類似性に関係している。ある事象の特徴(抽象化されたものである場合が多い)が他にも適用していくような思考の働きである。プログラミングにおいて，一般化は随所に見られる。例えば，「反復」は，一連のプログラムの中に共通性を見出し，同様の手順をまとめることである。また，プログラミングでは，しばしば一連の作業の集合を「関数」という形でまとめることがあるが，これも一般化である。先の星型を描くプログラムで，一辺の長さと最初の点の位置を変数とした関数にまとめ，星を描く際にそれらを具体的に設定するようにすれば，いろんな星を描くことができるようになる。さらにそれを，特定の点から，一定距離進んで，一定角度曲がる，というレベルにまで抽象化すれば，正三角形や正方形をはじめとする正多角形を描くのに一般化して利用できるものとなる。

一般化は，人間の基本的な認知機能の一つである。例えば，イスにはいろんな形や色のものがあるが，我々はどれも「イス」であると認識する。これは，我々がイスを一般化して理解できるおかげである(藤原，1998)。問題解決の場面でも，過去に経験したことの中から似たものを想起して当てはめる，といったことは，我々にとって自然な行動である。これも一般化の例である。先の算数の例を学習した子供は，「子供が 5 人いて，そこに 2 人やってきたら 5 人になりました」という問題に出会ったら，りんごの例から学んだことを思い出しながら解くことができるだろう。これは「最初の数＋増えた数＝合計」という形で一般化された知識を利用できた結果であると言えるだろう。

3. 創造的思考としてのプログラミング的思考

3.1 プログラミングと創造的思考

前節では，論理的思考としてのプログラミング的思考について概観した。プログラミングをする際にどのような思考を働かせているのか，また，日常的な問題解決とどのように関わっているのかも示した。プログラミング的思考は，プログラミングの際だけに使われるのではなく，普遍的に求められる力であることも理解していただけたと思う。

ところで，人はプログラミングをする際に，論理的思考だけを働かせているのだろうか。答えは否である。阿部(2018)は，小学生がビジュアルプログラミング言語 Scratch を使って活動する際に，非常に創造的であった様子を報告している。有識者会議の正式名称の中にも，創造性の育成という言葉が入っていることからも，プログラミング教育において論理的思考だけでなく創造的な思考の育成も期待されているのは間違いない。ところが，福井・黒田・森山(2018)も指摘するように，プログラミング教育に関する研究は論理的思考の育成に偏っている。そこで以下では，プログラミングを通してどのような創造性が学びうるかについて考察したい。

プログラムを作る際には，全てを綿密に設計して

おいてから作ることはない。もちろん設計は詳細に行われるが，プログラムを作り出したら，少し作っては，試してみて上手く動くかどうかを確認し，思ったように動かなければ原因を考えて修正を加えてさらに試してみて，が繰り返して行われる(宮田・大隅・林，1997；文部科学省，2018)。プログラミングでは，それがうまく行くか行かないかはプログラムを走らせてみればよくわかる。うまく行っているか否かについてのフィードバックがすぐに得られることは，それに基づいて即座に修正を加えることを可能にしてくれる。このことは，プログラミングによる学習での非常に重要なポイントである(市川，1994)。

3.2 ティンカリングとしてのプログラミング

小さな試行錯誤を積み重ねてすすめていくということこそ，プログラミング的思考として児童生徒に身につけて欲しいものの核である。Wilkinson & Petrich(2015，p.13)は，「現象，道具，素材をいろいろと直接いじくりまわして遊ぶこと」をティンカリング[4)]と呼び，「何かが動く仕組みを推測し，疑問を抱きながら，自分なりの方法で探っていくものです。自分で自分に，あれこれいじくりまわす許可を与えるのです。そうすることで，自分自身でも思いもよらなかった素晴らしいものが生まれてくるのです。」と述べている。また，ティンカリングは，それを行うことで「人はデザインセンスを磨き，問題解決の力を高めることができる」(Wilkinson & Petrich，2015，p.10)ものであり，「どのような教科を，何歳の生徒たちに教えるかに関わらず，全ての教室で使われるべき『知るための方法』」の一つ(Martinez & Stager，2015，p.35)であるとされる。

ティンカリングに似た概念として，デザイン思考が挙げられる(ケリー，D.・ケリー，T.，2014；藤原ら，2018；ブラウン，2014)。デザイン思考とは，デザイナーが実践している創造的な思考の方法や心構えのことであり，いくつかの特徴がある。特徴のうちの一つ，「ラピッド・プロトタイピング」は，(間違っていてもいいので)できるだけ素早くアイデアを形にし，それを使ってアイデアを検証する，という方法のことであり，そのプロセスを何度も繰り返し，なるべく多くのプロトタイプを作って，それを基に考えることが推奨される。

先に述べたように，プログラミングはティンカリングあるいはデザイン思考を行うのに非常に良い活動である。その活動を通して児童生徒は，試行錯誤することの方法(探索する，試す，失敗から次を考える，など)や試行錯誤の良さを学ぶことができるだろう。また，物事に根気よく取り組む気持ちや，「人間の基本的資質は努力次第で伸ばすことができる」という信念(ドゥエック(2016)のいう「しなやかマインドセット」)を育むこともできるだろう。

このような資質能力は，有識者会議の議論の取りまとめにおいて，「定められた手続を効率的にこなしていくことにとどまらず，自分なりに試行錯誤しながら新たな価値を生み出していくこと」が「現在，社会や産業の構造が変化していく中で，私たち人間に求められる」と述べられているところから，創造性の具体的な姿としてその育成が期待されているといえよう。

3.3 プログラミング体験の質と遊び

ティンカリングを通して創造性を身につけるためには，プログラミングを体験しさえすれば良い，というわけではなく，その体験が質の高いものでなければならない。それは，幼児教育における遊びの質についての議論に似ている。幼稚園教育要領(文部科学省，2017b)，第 1 章「総則」の第１には，「幼児の自発的な活動としての遊びは，心身の調和のとれた発達の基礎を培う重要な学習である」とあり，幼稚園教育は「遊び」が重要な学習であるという認識に基づいて行われている。保育現場では，子どもたちが遊びを通して学ぶことができるよう，子どもたちが活動に没頭する状況，すなわち「遊び込む」状況をいかに作り出すか，ということに注力されている。

藤原(2018)は，エンデ(1973)の「モモ」などを引用しながら学びにつながる遊びについて言及している。

> ごくさいきん始まったばかりのことなのですが，子どもたちがそんなものを使ってもほんとうの遊びはできないような，いろいろなおもちゃを持ってくることが多くなったのです。たとえば，遠隔操作で走らせることのできる戦車―でも，それ以上のことにはまるで役に立ちません。あるいは，細長い棒の先でぐるぐる円をかいて飛ぶ宇宙ロケット―これも，そのほかのことには使えません。あるいは，目から火花をちらして歩いたり顔をまわしたりするロボット―これも，それだけのことです。
> (中略)とりわけこまることは，こういうものはこまかなところまでいたれりつくせりに完成されているため，子どもが自分で空想を働かせる余地がまったくないことです。ですから子どもたちはなん時間もじっとすわったきり，ガタガ

タ，ギーギー，ブンブンとせわしなく働きまわるおもちゃのとりこになって，それでいてほんとうはたいくつしていて，ながめてばかりいます―けれど頭のほうはからっぽで，ちっとも働いていないのです。ですからけっきょく子どもたちは，むかしながらの遊びにまたまいもどることになります。これから，二つか三つの木箱とか，やぶれたテーブルかけとか，モグラが盛り上げた土の山とか，ひとすくいの小石とかがあれば十分で，あとはなんなりと空想の力で補うことができるのです。(エンデ，1973，pp.98-99)

そこで学びが生じるようにするには，いたれりつくせりの環境を揃えたり，それ以上は役に立たない遠隔操作で動かす戦車のようなものを用いたりするのではなく，子どもたちが空想を働かせる余地のあることが必要である。そしてそれはプログラミングにも当てはまる。プログラミングを通して創造性を育成する際の遊びの重要性は，Scratch を開発したレズニックが，その著書「ライフロング・キンダーガーデン」(レズニック，2018)などを通して繰り返し主張していることでもある。プログラミング体験でも，子どもたちが遊び込めるよう，空想を働かせる余地を持たせることが必要である。

3.4 ティンカリングと論理的思考

ここまで，論理的な思考とはまた別のものとして創造的思考を捉えてきたが，Martinez & Stager(2015)は，論理的な思考をすると考えられている科学者や数学者も，ティンカリングを行なっていると述べている。

> ティンカリングは，科学の実践の場で見られる単なる無知，未成熟な方法ではありません。もちろん，科学者たちは計画を立てます。しかし彼らはまた，直感に従い，繰り返し，ミスをし，再考し，やり直し，議論し，じっくり考え，共同作業し，そしてお茶を飲んで一服もしているのです。(Martinez & Stager，2015，p.55)

藤原(2013)は，試行錯誤を通して対象や現象を理解していくプロセスが，「科学する」ことと遊ぶことの間で共通していると述べている。近年，STEM(Science，Technology，Engineering，Mathematics)教育から，Art を含めたSTEAM教育として考えられるようになったことと関連して，創造的思考は論理的思考と別物なのではなく，融合して働き，科学的思考を形成しているといえる。プログラミングにおいてティンカリングを体験することは，科学的思考の育成につながると考えられる。

4. おわりに

本稿では，プログラミング的思考には論理的思考の側面だけでなく創造的思考の側面も合わせ持っていることについて論じてきた。また，プログラミング教育において育成される創造的思考の性質を，ティンカリングという点から述べてきた。プログラミング教育の実践がますます増えることが予測される昨今，両方の側面が育成されるようになることを期待している。

注

1) https://miraino-manabi.jp/
2) http://www.nhk.or.jp/design-ah/kangaeteinai/
3) http://www.nhk.or.jp/sougou/programming/?das_id=D0005180301_00000 (「scene 04 ジェイソンをプログラミング(順次)」を参照)
4) Wilkinson & Petrich(2015)によれば，ティンカリングという言葉は，1300 年代に家財道具を修理してまわった流しの修理屋を示すものとして登場した。

引用文献

阿部和広(2018) 初等教育における構築主義を用いたプログラミング教育，システム／制御／情報，62(7)，pp.254-259.

市川伸一(1994) コンピュータを教育に活かす，勁草書房.

Wilkinson，K. & Petrich，M.(2015) ティンカリングをはじめよう―アート，サイエンス，テクノロジーの交差点で作って遊ぶ (Make:Japan Books)，金井哲夫(訳)，オライリー・ジャパン.

Wing，J.M.(2006) Computational Thinking，Communications of the ACM，49(3)，pp.33-35.

エンデ，M.(1973) モモ―時間どろぼうとぬすまれた時間を人間にかえしてくれた女の子のふしぎな物語，大島かおり(翻訳)，岩波書店.

ケリー，T.・ケリー，D.(2014) クリエイティブ・マインドセット―想像力・好奇心・勇気が目覚める驚異の思考法，千葉敏生(訳)，日経BP社.

Computer At School(2015) Computational thinking: A guide for teachers. https://community.

computingatschool.org.uk/resources/2324/single(最終アクセス日：2019年2月17日).
ドゥエック，S.C.(2016) マインドセット-「やればできる！」の研究，今西康子(訳)，草思社.
日本教育工学会(2000) 教育工学事典，実教出版.
阪東哲也・黒田昌克・福井昌則・森山潤(2017) 我が国の初等中等教育におけるプログラミング教育の制度化に関する批判的検討，兵庫教育大学学校教育学研究，30，pp.173-184.
福井昌則・黒田昌克・森山潤(2018). ゲーム・パズルを題材に高校生の創造的態度の育成を図るプログラミング教育の試み. 日本教育工学会論文誌，42(Suppl.)，pp.21-24.
藤原伸彦(1998) カテゴリー分類に利用される知識，大阪大学人間科学部博士論文.
藤原伸彦(2013) 遊誘財データベース事例集(科学的思考)の構築，鳴門教育大学附属幼稚園研究紀要，第47集，pp.354-360.
藤原伸彦(2018) 子供集団歩き遍路と「遊び」，山﨑洋子(編著)「子ども集団歩き遍路」の可能性―6つの能力に基づくインフォーマル教育―，公益財団法人前川財団 平成29年度家庭・地域社会教育助成事業報告書，pp.141-154.
藤原伸彦・木下光二・森康彦・若井ゆかり・仁木稔明(2018) 教員養成における「アクティブ・ラーニング」を実践する力量形成の試み，鳴門教育大学学校教育研究紀要，32，pp.191-198.
ブラウン，T.(2014) デザイン思考が世界を変える-イノベーションを導く新しい考え方，千葉敏生(訳)，早川書房.
ベネッセ(2018) 図で解説「プログラミング的思考」とは，https://beneprog.com/2018/07/13/computationalthinking/(最終アクセス日：2019年2月17日).
Martinez，S.L. & Stager，G.(2015) 作ることで学ぶ―Makerを育てる新しい教育のメソッド. 阿部和広(監修)，酒匂寛(訳)，オライリー・ジャパン.
宮田仁・大隅紀和・林徳治(1997) プログラミングの教育方法と問題解決能力育成との関連―Process-oriented Approach と Content-oriented Approachとの比較を通して―，教育情報研究，12(4)，pp.3-13.
文部科学省(2016) 小学校段階における論理的思考力や創造性，問題解決能力等の育成とプログラミング教育に関する有識者会議「小学校段階におけるプログラミング教育の在り方について(議論の取りまとめ)」，http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm(最終アクセス日：2019年2月17日).
文部科学省(2017a) 小学校学習指導要領，東洋館出版社，2018.
文部科学省(2017b) 幼稚園教育要領，フレーベル館，2018.
文部科学省(2018) 小学校プログラミング教育の手引き(第二版)，http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_02_1.pdf(最終アクセス日：2019年2月17日).
林向達(2018) Computational Thinkingに関する言説の動向，日本教育工学会研究報告集，18(2)，pp.165-172.
レズニック，M.(2018) ライフロング・キンダーガーデン―創造的思考力を育む4つの原則，酒匂寛(訳)，日経BP社.