F１００２
報告 599

プログラミング的思考を育む，授業デザインの在り方

－思考を可視化・意識化することを通して－

木村　祐太（京都市総合教育センター研究課　研究員）

Key Words：プログラミング的思考，メタ認知的活動，アンプラグド，あきらめずに取り組む姿勢

2020年度より全面実施される学習指導要領における改訂の目玉の一つがプログラミング教育である。多くの教員にとってなじみのない分野であるが，そこで育まれる力はこれからの未来社会を生きる児童にとっては必要不可欠なものである。どのように実践すればいいのか，単なるプログラミング体験にせずプログラミング的思考を育むためにはどのようなことに気を付ければいいのか，明らかにすべく研究，実践を行った。

そこで本研究では，メタ認知の理論に着目した。自分がどう考えているのかを明確に認識し，その方法の有効性を実感することで，ある方法は応用の利くものになるという。そのためには，教員がプログラミング的思考を意識し，児童に明示的な発問や有効性を実感させる声掛けをすること，どのような思考かがわかりやすくなるように，可視化することが重要であると考える。

実践を通して，児童がプログラミング的思考を意識できるようになったことや，実践しようという意欲が高まったことが明らかになった。また，課題に対して粘り強く取り組む姿勢を養うことにつながるという，プログラミング体験の効果も明らかになった。

目　　次

＜研 究 担 当＞　　木村　祐太　　（京都市総合教育センター研究課研究員）

＜研究協力校＞　　京都市立　砂川小学校
京都市立　柊野小学校

＜研究協力員＞　　森元　美帆　　（京都市立砂川小学校教諭）
小川　美咲　　（京都市立砂川小学校教諭）
有田　圭佑　　（京都市立柊野小学校教諭）

はじめに

人工知能が〇〇できるようになった，△△の名人に勝った，そのようなニュースを毎年のように聞く。そして，そういった技術は当たり前のように，私たちの生活に入り込んでいる。

技術の進歩は社会の在り方そのものを変革させる。日本がこれまで歩んできた社会は狩猟社会（Society1.0），農耕社会（Society2.0），工業社会（Society3.0），情報化社会（Society4.0）だが，今日本が進もうとしているのはSociety5.0(1)と呼ばれる社会だ。IoT（Internet of Things）により多くのモノと人とがつながる社会である。サイバー空間に集められた情報（ビッグデータ）を人工知能が解析し，例えば渋滞を避けながら最も早く着く道順で車を自動運転したり，あるいは工場で人による指示の必要なくロボットを動かし客の注文通りに製品を作り続けたりする。IoT，人工知能，ロボット，ビッグデータの処理という先端技術があらゆる産業や社会生活に取り入れられ，経済発展を促し社会的課題を解決する，そのような社会を目指すことが内閣府によって提唱されたのである。そしてこの社会への変化は，技術の変化の影響を受けたものであり，技術進歩に合わせてもう始まっている。

このような社会において子どもたちに必要な資質・能力を育むために，プログラミング教育が始まることとなる。これまでも中学校の技術・家庭科や高等学校の情報科で扱われてきたが，小学校において必修化したということは，プログラミング教育で身に付けるべき資質・能力の重要性がこれまで以上に高まったということであろう。

本研究ではこのような現状認識に立脚し，小学校のプログラミング教育において，子どもたちにどのような力を育めばいいかをより具体的にしながら，学校現場においてより充実したプログラミング教育が行われるような手立てを提案したい。

(1) 内閣府『Society5.0』https://www8.cao.go.jp/cstp/society5_0/index.html 2019.4.26

第１章　プログラミング教育の必要性

小学校でのプログラミング教育必修化と聞いた時に多くの人が思い浮かべるのは，小学生がパソコンの前に座って，英語や数字の文字列（java やC 言語などのいわゆるプログラミング言語）を打ち込んでいる姿であろう。プログラミングのプログラムとは，コンピュータを動かす命令のことであり，その命令を作る作業をプログラミングというからだ。その姿を前提として，小学校でプログラミングの授業についていけるようにプログラミング教室に通わせる家庭が増えたというニュースや，小学校ではプログラミング言語を覚える必要はないといった批判を目にする。果たして，小学校で行われようとしているプログラミング教育が目指すのはそういった姿なのだろうか。違うとすれば，一体どのような姿なのか。1 章では，プログラミング教育が小学校で必修化された背景とそこから読み取れる将来の子どもたちに必要な力を整理し，プログラミング教育がなぜ必要なのかを明らかにしていく。

第１節　求められた背景

小学校でのプログラミング教育必修化が議論されるきっかけとなったのは，平成 25 年の政府発表「日本再興戦略」において世界最高水準の IT 社会の実現を目指すと宣言されたことであろう(2)。そこでは IT（情報技術）を活用してよりよい生活が可能となる社会の実現を目指すことと，その実現のためにハイレベルなIT人材の育成と確保を推進することが示されている。また経済産業省の「IT人材の最新動向と将来推計に関する調査結果」(3)により，2020 年に 36.9 万人，2030 年に 78.9 万人のIT人材が不足すると予測されたことも影響を与えていると考えられる。なお，「日本再興戦略」における世界最高水準の IT 社会とは，先に述べたSociety5.0というビジョンとして内閣府により鮮明に打ち出されているものである。進化した人工知能が様々な判断を行ったり，身近なものの働きがインターネット経由で最適化されたりする社会である。その到来は，社会の在り方を大きく変えていくと予想されており，様々な課題に新たな解決策を見いだし，新たな価値を創造し，私たちの生活に便利さや豊かさをもたらすと期待されている。その主役となる人材として，IT 人材の需要が高まったと言える。

ここでいう IT 人材は 2 通りに分けられる。そのことを示す資料として，経済産業大臣が行う IT 分野の国家試験である情報処理技術者試験の種類を図 1-1 に示す。

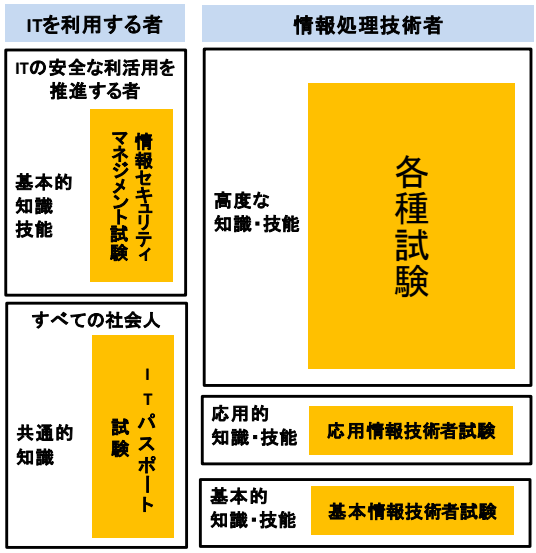


図 1-1　情報処理技術試験種類（情報処理技術推進機構）
図は一部抜粋

右側の情報処理技術者というのは，アプリやソフトといったシステムを作る側である。いわゆる IT 企業で働いている人たちだが，そういった人たちにどのようなスキルが必要なのかを示している。また，図の左側に示されているのは，IT を利活用する者，つまり使う側である。その試験内容を見ると使う側に必要とされるスキルがわかるのだが，IT のセキュリティに関するスキルや，AI などの新技術に関するスキルを問う出題がされている。これらのスキルがあれば，例えば情報漏洩事故や情報消去事故が防げるので確かに必要なスキルと言える。

では，このシステムを作る側のスキルがプログラミングなのかというとそうではない。岡島が述べているプログラミングに必要な工程(4)を以下に示し，それをもとに説明する。

①お客の悩みを聞く
②悩みの本質を洞察，分析し，解決策を立案する
③解決策のうち，情報システムでやるべき部分を抽出する
④やるべきことのざっくりした設計図を作る
⑤ざっくりした設計図を，もうちょっと詳細な設計図や日本語による指示に細分化する
⑥⑤をさらに，コンピュータにとってちょうど良いサイズに細分化する
⑦適切なサイズになった指示は，日本語で書かれているだろうから，コンピュータが理解できるプログラミング言語に翻訳する

一般にプログラミングといった時に思い浮かべるパソコンの前で英語や数字の文字列を打ち込む作業は（以下，コーディングと言う）部分的なものでしかない。

プログラマが担当するコーディングは⑦に該当し，その前にもいくつかの工程が必要なことがわかる。例えば①②にあるように，顧客の悩みの本質を抽出するためのコミュニケーション能力や情報分析能力も必要になる。

以上のことから，様々な力を駆使して問題解決のための IT を作る側の人材と，人工知能などの IT をブラックボックス化せずに理解しよりよく使う人材とが求められていると言える。経済産業省も，決してコーディングをするプログラマのみを重要視しているわけではないことがわかる。

第２節　日本が目指すプログラミング教育

（１）プログラミング教育で育む資質・能力

第１節で述べた流れの中で，2016 年 4 月に文部科学省が小学校でのプログラミング教育必修化を検討すると発表した。そして「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おける有識者会議」(以下，有識者会議）の中でプログラミング教育の概要が整理された。

議論の中で，人工知能と比べ人間の強みとは何かということが述べられている。これは人工知能が発達し様々な仕事が自動化していく中で，人間が担う仕事は何かを念頭に置いているのだろう。人間の強みは，以下のように示されている(5)。

人間は，感性を豊かに働かせながら，どのような未来を創っていくのか，どのように社会や人生をよりよいものにしていくのかという目的を自ら考えることができる。

人工知能は，与えられた目的の中で，命令に従って処理をすることは得意である。何万時間であろうと，どんなに面倒であろうと文句も言わずに処理し続ける。しかし，目的やその解決のための命令を考えることができるのは人間だけである。解決策を考えるにも現状を捉えたり，関係者の意向を聞いたりと情報を分析する必要がある。宗教も文化も言語も違う人たちが共存する社会において，より多様なニーズや条件が入り混じる中で，目的に応じて必要な情報を抽出し，分析することは人工知能にはできない(6)。分析結果を基に自分のアイデアを創造し，多様な他者と協働しながらよりよい解決策を見いだすことができるのも人間の強みである。

これに加え，IT を活用する力の重要性も指摘されている。多様かつ大量の情報を収集するにも分析するにもコンピュータを使う方が便利である。人間が考えた解決策もコンピュータや人工知能，ロボットなどの技術により実現するものがほとんどであろう。コンピュータの働きについて詳しい方がより実現可能なアイデアを出すことができる。つまり上記の問題解決力，情報活用能力，コミュニケーション能力，創造性といった人間の強みを生かすためにも，コンピュータを動かす命令であるプログラミングについて理解する必要があるということである。そこでプログラミング教育で育む資質・能力は「プログラミング教育の手引」において以下のように示されている(7)。

【知識・技能】
身近な生活でコンピュータが活用されていることや，問題の解決には必要な手順があることに気付くこと。

【思考力・判断力・表現力等】
プログラミング的思考

【学びに向かう力・人間性等】
発達の段階に即して，コンピュータの働きを，よりよい人生や社会づくりに生かそうとする態度。

上記【知識・技能】の身近な生活でコンピュータが活用されていることに気付くというのは，コンピュータ（およびコンピュータを動かしているプログラム）の働きが自分たちの生活をよりよくしていると知ることである。コンピュータの仕組みや便利さを知ることや，実際にコンピュータを活用して問題解決をした経験が，【学びに向かう力・人間性等】にあるような，将来もよりよく活用していこうという姿勢につながると考えられる。

問題の解決に必要な手順というのは，先述した①～⑦のような手順だが，これはコンピュータでプログラムを作って問題解決する時の手順であるので，コンピュータを使う使わないに限らずチームで問題解決する場面を想定して，より一般化してみると以下のようになる。

①問題を発見する
②問題の本質を分析し，解決策を考える
③解決策のうち，チームでやるべき仕事を取り出す
④やるべき仕事の大まかな設計図を作る
⑤大まかな設計図をより具体的な作業に細分化し，適性に応じて協力者に分担，指示する
⑥⑤をさらに協力者が，自分自身が実行できる手順に組みなおす
（⑦はコーディングをする場合に限定される）

この手順はプログラミングをするとき以外にも有効な手順だと言えそうである。そしてこの手順の中で，分析したり，細分化したり，細分化した作業の手順を考えたりするときに必要な思考が，【思考力・判断力・表現力等】にあるプログラミング的思考である。

（２）プログラミング的思考とは

既に述べたようにプログラミング的思考はプログラミングをする場合に限らず有効な思考である。現に議論の取りまとめの「プログラミング教育とは」の中においても，「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」(8)と，その普遍性が示されている。

また学習指導要領総則でも以下の二つの文において，プログラミング的思考が様々な学習の基盤となる資質･能力であることが示されている(9)(10)。

総則　第１章　第２の２の(1)
各学校においては，児童の発達の段階を考慮し，言語能力，情報活用能力（情報モラルを含む。），問題発見・解決能力等の学習の基盤となる資質・能力を育成することができるよう，各教科等の特性を生かし，教科横断的な視点から教育課程の編成を図るものとする。
（下線は筆者による）

総則　第１章　第３の１の（3）のイ
情報活用能力の育成を図るため，（中略）あわせて各教科等の特質に応じて，次の学習活動を計画的に実施すること。（中略）
イ　児童がプログラミングを体験しながら，コンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動
（下線は筆者による）

まず，第１章第２の２の(1)では，情報活用能力が学習の基盤となることが示されている。そして，その育成のためには，コンピュータに意図した処理を行わせるために必要な論理的思考力，すなわちプログラミング的思考の育成が必要であることが第１章第３の１の(3)に示されている。この学習指導要領には，コンピュータに意図した処理を行わせるためと記述してあるが，そのためだけのものではなく，他の様々な問題解決場面にも有

効であろうことは第1章2節(1)に先述した通りである(p.3)。

プログラミング教育の核となるプログラミング的思考だが，学習指導要領解説総則編に示されている定義を以下に示す(11)。

> ①自分が意図する一連の活動を実現するために，
> ②どのような組み合わせが必要であり，
> ③一つ一つの動きに対応した記号を，
> ④どのように組み合わせたらいいのか，
> ⑤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。
>
> **（改行と番号は筆者による）**

①②では問題を発見し，どのように解決したいのか考える。大きな問題であれば，小さな問題に分けてみることで，解決策が見つかるかもしれない。また，分けたことで必要のない部分が見つかり，問題をシンプルにとらえることができる。解決策が浮かんだら，実行するためにどんな作業が必要か，全体を分けて考える必要がある。③の記号というのはコンピュータへの命令であり，人間が問題を解決するのであれば分けられた作業や指示である。④それをどんな順番で行えばいいのか工夫し，⑤実際にやってみて評価し，修正する。場合によってはより汎用性をもたせられるように考える。これがプログラミング的思考である。

本市では上記の定義をもとに，プログラミング的思考の重要ポイントが「問題解決のために，ものごとを分解・構築・試行錯誤して考えること」と示された(12)。文科省の定義とは以下のように対応すると考えられる。

> 問題解決のために：①
> ものごとを分解し：②③
> 構築：④
> 試行錯誤して考えること：⑤

本市の定義においても，プログラミング的思考が問題解決のための思考として示されていることを確認しておきたい。

（3）プログラミング的思考の要素

プログラミング的思考の定義は，「コンピューテーショナル・シンキング」（以下 CT）の考え方を踏まえつつ，プログラミングと論理的思考との関係を整理しながら提唱された定義である(13)。CTとは Wing(2006)によって広まった概念で，Wingが現時点で最適と述べている定義を以下に示す。

> Computational thinking is the thought processes involved in formulating a problem and expressing its solution in such a way that a computer -human or machine- can effectively carry out.
> CTは，問題を明確にし，その解決策をコンピュータ，人，機械が効率的に実行できるよう，表現することに関わる思考である。
>
> **（訳は筆者による）**

ここからもプログラミング的思考が問題解決のための思考であることが読み取れる。また，磯部らはCTを「計算機科学（コンピュータ・サイエンス）の流儀で考えて問題解決をすること」とし，以下のように解説している(14)。

> ・問題を抽象化によって分析すること。
> ・解決するためのプロセスを定型化・自動化できるように構築すること。
> ・コンピュータにプロセスを任せられるようプログラムすることはあくまでオプションであって，CTには含まれない。

計算機科学の流儀にはどのような要素があるのか。イギリスはCTを育むことを目的とし，コンピューティングという独立した教科としてナショナルカリキュラムが作られているので参考にする。その内容については太田らの研究が詳しいが，コンピューティングにおけるCTの要素は表1-1のように示されている(15)。

表1-1　イギリスの教科コンピューティングでのコンピュテーショナルシンキングの要素

分解 (Decomposition)	問題や事象をいくつかの部分に，理解や解決できるように分解する。
抽象化 (Abstraction)	問題を単純化するため，重要な部分は残し，不要な詳細は削除する。
アルゴリズム的思考 (Algorithmic)	問題を解決するための明瞭な手順で，同様の問題に共通して利用できるものである
一般化 (Generalization)	類似性からパターンを見付けて，それぞれを予測，規則の作成，問題解決に使用する
評価 (Evaluation)	アルゴリズム，システムや手順などの解決方法が正しいか，確認する過程である。

この5項目による分類は，日本ではベネッセ(16)や日経BP社(17)，他の複数の教育委員会が使用しており，本稿においても参考にする。

分解は，理解や解決のために問題や事象をより小さな部分に分けることである。例えば，カレーライスをその材料に注目して，玉ねぎ，人参，米，

カレー粉などに分けることである。作業手順に注目して，玉ねぎを切る，人参を切る，いためる，お米を炊くなどに分けることもできる。

アルゴリズム的思考は，分解した手順を目的に合わせてよりよく並べることである。その手順は順次処理，分岐，反復の三つで構成される。カレーライスを作るときに，じゃがいもを切る→玉ねぎを切る→人参を切る→人参を炒める→じゃがいもをそこに入れる，というのは順次処理である。「ご飯がなければ→ご飯を炊く」「ご飯があれば→じゃがいもを切る」というように，条件によって処理が変わることが分岐である。具材に串をさし，通ることを確認するまで「熱を通す→串を刺す」というように，条件を満たすまで同じ作業を繰り返すというのが反復である。

一般化は，ある問題解決のためのパターンを他の類似した問題でも使えるようにしたり，使ったりすることである。つまり過去の問題解決のための手順を目の前の問題解決に応用したり，今作った手順を将来の問題解決に応用したりすることである。先述した具材に熱が通っているかを確認する手順は，他の様々な料理に使うことができる。

評価は，考えた手順が目的の達成に十分かを確認することである。あるいは手順が意図した活動になっているかを確認することである。問題があれば，原因を分析し改善方法を考えることでもある。

抽象化については，定義が分かれている部分である。整理したものを表 1-2 に示し，それらをもとに抽象化の機能を二つ示す。

表 1-2　コンピューティング，ベネッセ，日経 Kids+における抽象化の定義の比較

コンピューティング	問題を単純化するため，重要な部分は残し，不要な詳細は削除する。
ベネッセ	分割した動き（事象）の中から適切な側面・性質だけを選び出し，他の部分を除くこと。いわゆる抽象化。
日経Kids+	物事の似ているものから,共通するものを見つけ出す能力

一つ目は，目的に合わせて必要な部分を抽出することである。抽象化の抽象とは，一部分を抽出するという意味である。同時に，抽出された部分以外は捨てられるので，それは捨象という。コンピューティングやベネッセの定義はまさに抽象であり,「正多角形の作図においては,正多角形の意味や性質の中から,『辺の長さがすべて等しい』『角の大きさがすべて等しい』という意味や性質を選ぶなど」という例がベネッセによって示されている(18)。

二つ目は，詳細な情報を省き具体的ではない表現にするということである。日経 Kids+における「物事の似ているものから，共通するものを見つけ出す能力」というのは，例をみると,「天ぷらうどん」「月見うどん」「きつねうどん」「山菜うどん」など 6 種類の異なるうどんがあった場合に，つまり「うどんである」ということ，と示されている。これは具体的ではない表現にするということでもある。抽象の対義語は具体であるので，具体的ではないようにすることは抽象化と言える。例えば算数科において「底辺４ｃｍ」や「底辺５ｃｍ」を「底辺△ｃｍ」と変数にすることも抽象化にあたる。これによって「底辺△cm×高さ〇cm÷2＝三角形の面積」のような一般化が行われる。

共通点を見つけ出すことは重要な思考である。A という問題の解決のために作った手順の一部を他の B という問題に適応させる場合（一般化）には，A の問題と B の問題の共通点を考えることで類似しているかどうかを判断し，類似していればA の問題の解決策が部分的に適応されることになる。または解決策のヒントを探す場合，類似する別の案件を参考にすることがあるが，類似するかどうかは共通点を見つけ出すことができるかどうかによる。一般化するにも，参考事例を探すにも共通点を見つけ出す思考は重要である。共通点を見つけだすことの問題解決におけるメリットは大きい。

以上のように，抽象化の機能は多様であり，その分，重要であるともいえる。本研究では抽象化について再定義し，5 要素の定義について表 1-3 のように整理する。

表 1-3　本研究におけるプログラミング的思考の要素

分解 (Decomposition)	問題や事象をいくつかの部分に，理解や解決できるように分解すること。
抽象化 (Abstraction)	分割した複数の部分の中から，目的に照らしたり共通するものを見つけたりしながら必要な部分を選んだり，具体性を省いた表現・概念にしたりすること。
アルゴリズム的思考 (Algorithmic)	問題を解決するために，明瞭な手順を組むこと。
一般化 (Generalization)	類似性からパターンを見付けて，別の場合にも利用できるようにすること。
評価 (Evaluation)	目的や意図に対して手順が最適かを確認し，修正・改善していくこと。

これらの５要素は，京都市の「分解・構築・試行錯誤」という重要ポイントにも対応していると考えられる。

つまり，問題解決のために，【分解】ものごとを分解したり抽象化したりしながら捉え，【構築】解決のために必要な手順を，抽象化やアルゴリズム的思考や一般化の思考を活用しながら構築し，【試行錯誤】構築したものを自分の意図に照らして評価し，より意図に近づくように修正していく，というように対応していると考えられる。筆者の解釈による対応関係を表 1-4 に示す。

表 1-4　京都市におけるプログラミング的思考の重点と，筆者の考えるプログラミング的思考の要素の対応

<table>
<tr><th colspan="2">京都市</th><th colspan="2">筆者による定義</th></tr>
<tr><td rowspan="4">試行錯誤</td><td rowspan="2">分解</td><td>分解</td><td rowspan="4">評価</td></tr>
<tr><td rowspan="3">抽象化</td></tr>
<tr><td rowspan="2">構築</td></tr>
<tr><td>アルゴリズム的思考</td></tr>
<tr><td></td><td></td><td>一般化</td></tr>
</table>

以上を踏まえ，プログラミング教育とは，

①コーディングを行うことではない。コンピュータの働きや良さを知ることでよりよく活用できるようになったり，よりよく解決策を考えることができるようになったりすることを目指す。

②問題解決のための思考をプログラミング的思考と言うが，コンピュータでプログラムする場合に限らず様々な問題解決に有効である。

③プログラミング的思考は，分解・抽象化・アルゴリズム的思考・一般化・評価で構成されている。

と言える。

(2)内閣府『日本再興戦略-JAPAN is BACK-』2013.6.14

(3)経済産業省『IT 人材の最新動向と将来推計に関する調査結果を取りまとめ』https://www.meti.go.jp/press/2016/06/20160610002/20160610002.html　2019.5.31

(4)岡島裕史『プログラミング教育はいらない―GAFA で求められる力とは―』光文社新書，2019.2.p.26

(5)文部科学省『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（議論のとりまとめ）』2016.6.p3

(6)文部科学省『小学校段階における論理的思考や創造性，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（第２回）議事録』2016.5.pp.2-6
この会議の中で，人工知能が苦手なことや人間の強みが紹介されているので，それらを参考に記述した。

(7)文部科学省『小学校プログラミング教育の手引（第二版）』2018.11.p.6

(8)前掲(6).p.1

(9)文部科学省『小学校学習指導要領』2017.p.19

(10)同.p.22

(11)文部科学省『小学校学習指導要領解説総則編』2017.3.p.85

(12)京都市教育委員会『京都市小学校プログラミング教育スタートハンドブック』2019.9.p.3

(13)文部科学省『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（議論のとりまとめ）』2016.6.p.8

(14)磯部秀司，小泉英介，静谷啓樹，早川美徳『コンピュテーショナル・シンキング』共立出版株式会社　2016.3.25

(15)太田剛，森本容介，加藤浩『諸外国のプログラミング教育を含む情報教育カリキュラムに関する調査―英国，オーストラリア，米国を中心として―』日本教育工学会論文誌，40(3)2016.p.198

(16)ベネッセ，プログラミング教育情報『図で解説「プログラミング的思考とは」』http://beneprog.com/2018/07/13/computationalthinking/ 2019.5.29

(17)日経 Kids+『子どもと一緒に楽しむプログラミング』日経 BP 社　2017.4.17　pp.22-23

(18)前掲(16)

第２章　プログラミング的思考を育むため

第１節　教員が意識するために

（１）教員への意識調査から

プログラミング教育実施にあたっての最大の課題は，小学校の教員とプログラミングとの距離が遠いことであろう。多くの教員が実際にプログラミングをしたこともなければ，CTなどの考えに触れたこともないと思われる。そのことを示すデータとして，黒田らによる教員のプログラミング教育に対する認知についての全国調査の結果(19)を図2-1に示す。

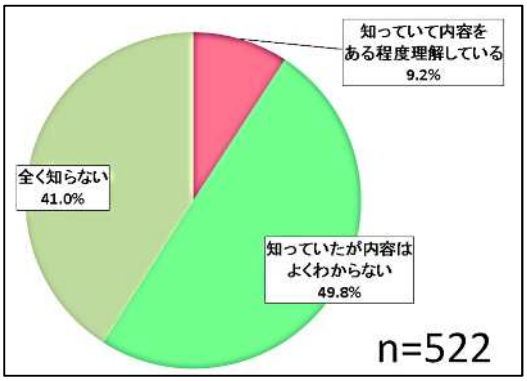


図 2-1　プログラミング教育に対する認識

この調査は有識者会議が行われてすぐの2016年8月～9月に行われたものである。「全く知らない」「知っていたが内容はよくわからない」を合わせると全体の9割であり，そもそも十分認知されていなかったことがわかる。有識者会議の議論が始まったころからメディアで報じられるようになったが，遠巻きに見ている教員が多いというのが実態ではないだろうか。また本市においては令和元年度，プログラミング教育に関する研修が行われたが，全教員に周知，理解されるにはまだまだ時間がかかるであろう。

しかしながら，小学校においては先述した学習指導要領総則にある通り(p. 3)，各教科等の特性を生かし，教科横断的な視点から教育課程の編成を図るものとされている。プログラミング的思考のような〇〇思考は，一回の授業で身に付くものではない。6年間を通して，様々な教科の中で継続的に育んでいく必要がある。教員がプログラミング的思考の要素を意識して教材研究と授業を行えるようになるためには，プログラミング的思考を意識することができる授業デザインシートが必要ではないだろうか。

（２）授業デザインシート

プログラミング的思考を教員が意識できるよう，その要素をわかりやすく端的に示した授業デザインシートを作成した。それを図2-2に示す。

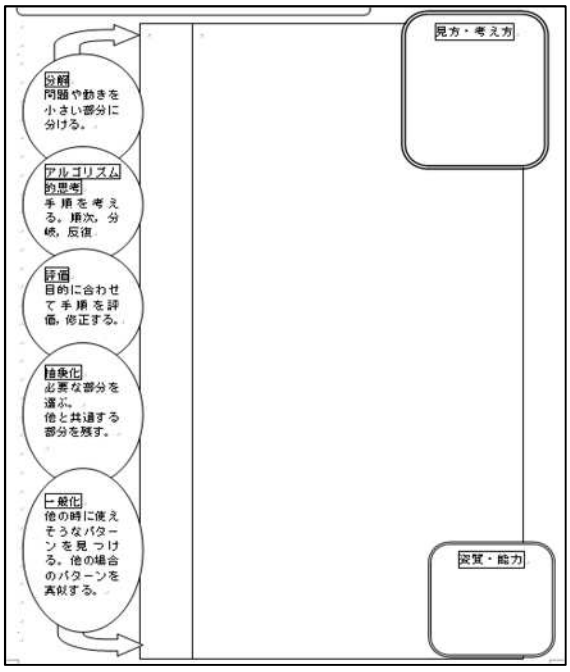


図 2-2　授業デザインシート

授業デザインシートの左側には，プログラミング的思考の5要素が示してある。例えば分解については，「問題や事象を小さい部分に分ける。」と示した。このように示してあることで，授業案を考える時にプログラミング的思考を意識し続けることができる。

シートの右側には教科の「見方・考え方」，単元や本時において育みたい「資質・能力」を書く欄を設けた。これはプログラミング教育のめあてを達成しようとしたあまり，教科としてのめあてが達成されないということに陥らないためである。

教科の見方・考え方のどの部分がプログラミング的思考と一致するのか，あるいは両方1時間の中に盛り込まれているのか，そしてプログラミング的思考を働かせて問題解決を行った結果，教科の資質・能力は育まれたのか，それらを意識しながら教材研究を行う支援になると考える。

（３）算数科における例

本研究においては，２年生と５年生の算数科を中心にプログラミング教育の実践を行う。５年生2月に行われる「円と正多角形」の単元がプログラミング教育に適した単元として学習指導要領に示されており，その単元までにどのような力を培っていけばよいのか系統的に考えるためである。

では，算数科においてプログラミング的思考はどのように発揮されるのであろうか。算数科で育む数学的に考える資質・能力は学習指導要領に以下のように示されている(20)。

> 日常の事象を数学的に捉え見通しをもち筋道を立てて考察する力，基礎的・基本的な数量や図形の性質などを見いだし統合的・発展的に考察する力，数学的な表現を用いて事象を簡潔・明瞭・的確に表したり目的に応じて柔軟に表したりする力
>
> （下線は筆者による）

見通しをもつことは，例えば問題文を読んでいくつかの部分に分け（分解），注目すべき点を抽出し（抽象化），類似性のある既習事項を探し（抽象化），そこで学習した問題解決の手順を目の前の問題解決に利用できないかを考える（一般化）ことである。

基礎的・基本的な数量や図形の性質を見いだし統合的・発展的に考察するというのは，例えば正五角形と正六角形の辺や頂点の数に注目し（抽象化），共通する部分を見いだし（抽象化），正多角形の性質を考える（一般化）ことである。

数学的な表現を用いて事象を簡潔・明瞭・的確に表すというのは，例えば筆算や合同な図形の描き方の手順を考え（アルゴリズム的思考），ステッ

プチャートなどにして明瞭な形で表現する（アルゴリズム的思考）ことである。

このように，数学的に考える資質・能力とプログラミング的思考は共通する部分がある。プログラミング的思考を算数科の授業の中で意図的に浮き上がらせることで，より数学的に考える資質・能力が意識され，子どもたちにどのように思考させたいのかを明確にした教材研究や授業実践が為されるようになるのではないだろうか。その様にして作成した授業デザインシートの例を以下の図2-3に示す。

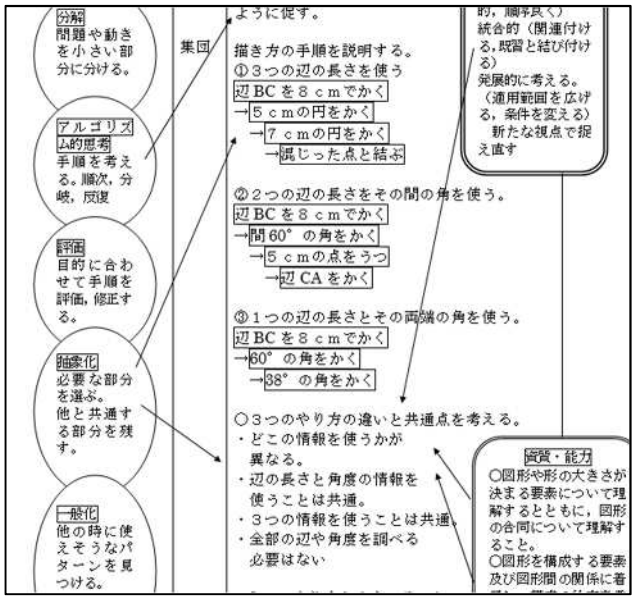


図 2-3　算数科での授業デザインシート活用例

この授業デザインシートは，５年生算数科「合同な図形」のものである。ここでは合同な図形を描く三つの手順を関連付けて統合的に考えることを抽象化することと捉えて，共通点を考える活動としている。

授業の詳細は第3章3節1項で行うが，このようにプログラミング的思考と教科の見方･考え方や育みたい資質・能力の共通点を考えた教材研究の結果,子どもたちへの指示や承認も「考えてみよう。」や「よく考えたね。」といった曖昧なものではなくなり，より具体的になる。「手順を考えて，順番がわかるように表してみよう。」や「共通点を見つけて，きまりを考えよう。」になり，「手順をより簡単に示せたね。」や「面白い共通点を見つけたね。」のようになる。このことは，考えることが苦手な児童にとっては，どのように考えればよいのかのヒントになる。さらに重要なことは，プログラミング的思考を児童に意識させることにつながるということである。そしてそれは，学んだことの活用を促すことにつながる。

第２節　児童への可視化，意識化

（１）転移の困難さ

プログラミング的思考は，プログラミング体験の中で無意識に発揮されればよいというものではない。問題解決能力として，様々な問題を解決する場面に応用されてこそ，その意味がある。では，単にプログラミング体験をすれば応用されようになるのであろうか。

プログラミング的思考のように問題を解決するためのある方略が他の場面に応用されることを転移というが，その困難さが指摘されている。例えば奈須は，平成19年度全国学力・学習状況調査の算数の問題の正答率を例に，転移の困難さを指摘している。以下，図2-4と図2-5にそれらの問題を示す。

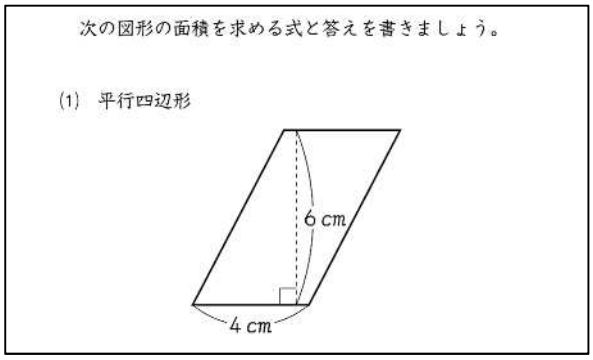


図 2-4　A 問題

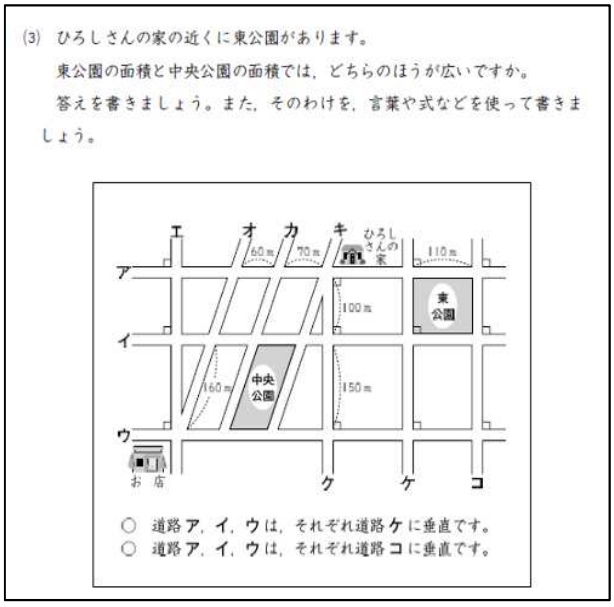


図 2-5　B 問題

A問題の正答率は96%，B問題の正答率は18%であった。B問題の誤答内容との割合を見てみると，中央公園の面積を求める際に公園の隣の道路の長さ150mを高さと認識することができず「底辺×斜辺」として計算している児童が34.4%いる。つまりおよそ3分の1の児童に転移が生じなかったことになる。

その上で奈須は，以下のように述べている(21)。

> 子どもは単に方略を教わっただけでは，①それが本当に有効であるとの実感を持てず，②また，なぜ有効なのかを明晰には理解せず，③したがって，どのような場面で有効なのかを判断することができず，④さらに，教わったのと異なる対象や場面に適合するよう自分でアレンジして実行できるほどにはその使用に習熟してはいないがゆえに，教わった方略を自発的にほとんど用いません。これら 4 つの関門全てをクリアする明示的な教え方をしてはじめて，子どもは学んだ内容をさまざまな問題解決に自発的かつ創造的に活用するのです。

この方略とは，問題解決のための方法である。「アルゴリズム的思考」「抽象化」といったプログラミング的思考が様々な実生活の問題解決に応用されるようになるには,それらの有効性を実感し，なぜそれが有効なのかを理解するように明示的に指導する必要がある。

（２）転移させるためには

プログラミング的思考が応用されるとは，厳密にはメタ認知的活動のプロセスにプログラミング的思考が組み込まれることを指す。

メタ認知的活動とは，対象に対する自分の思考や行動を俯瞰し，自分の思考や行動が目的に合致するか評価し（モニタリング），モニタリングの結果を受け思考や行動を調整する（コントロール）働きである(22)。図に示すと図 2-6 のようになる。

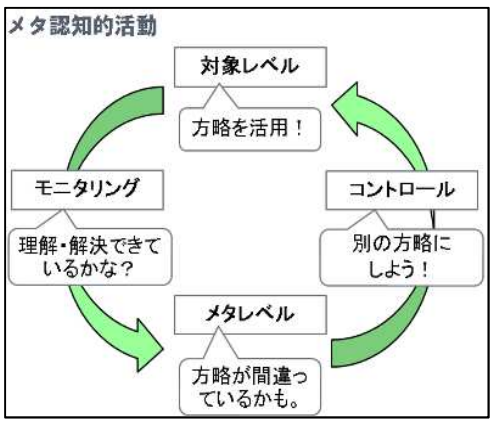


図 2-6　メタ認知的活動のプロセス

このメタ認知的活動でモニタリングされたりコントロールされたりする思考や行動，あるいはモニタリングしたりコントロールしたりする思考そのものの方略に関する知識をメタ認知的知識と呼ぶ。言い換えれば，どう考えればよいかという引出しがメタ認知的知識である。このメタ認知的知識にプログラミング的思考が組み込まれ，かつメタ認知力が育成されれば，プログラミング的思考が状況に応じて適切に選択され働くようになると言える。

では，どのように組み込み，どのように育成すればよいのであろうか。

深谷によると，まず，どのように考えるかという方略の知識をもつことが不可欠である(23)。そもそも方略を知らなければ使うことができない。そこで，メタ認知的活動のプロセスにプログラミング的思考が組み込まれるようになるためのポイントの一つ目として明示的に指導する必要がある(24)。どのように考えているのか，どのように考えることでよりよく理解したのかなど,「やることの順番がわかるように図をかこう。」「比べてみたんだね。」などとはっきりと示すのである。思考に名前をつけ一つの方法として自覚させることにより児童は，今まで無意識的にしていた自分の思考を意識し，方略として獲得することができる。これについては奈須も，以下のように述べている。

> そこで子供たちに手渡したつもりの読解の着眼点なり方略，いわば読解の「お道具」に明確な名前がついていないこと，さらに子どもたちの「お道具箱」が一度も整理されてこなかった点にあります。

つまり方略＝お道具には「分解」「アルゴリズム的思考」「抽象化」などとしっかりと名前を付け，「いくつかの部分に分けることを分解という」のようにどんな道具なのかを整理しておく必要があるというのである。

ポイントの二つ目は，有効性を実感させることとされている(25)。ある方略がどんな時に，どう有効なのかがわからなければ，使おうとはしないであろう。また，方略を変更しようにも，どの方略を用いればよいのか判断ができない。

方略の有効性を実感するためには，まずはその方略を獲得した時の課題と類似の課題を自身で解決することである。その上で，学んだことを土台としてさらに発展的な課題やつまずきやすい課題に対して問題解決を行う授業が有効である。習った方略をそのまま適応しても解決できない事態に追いこまれることで，どう工夫すればよいのか，

自身の方略を評価したり，別の方略を考えたりするようにもなるだろう。そうして自分が選んだ方略で解決できたとき，その方略の有効性が実感される。

また，振り返りをする活動も有効である。単に楽しかったかどうかを振り返るのではなく，自分がどういう方略を活用し，どういう点で有効なものだったのか，あるいは有効でなかったのかをメタ認知させることになる。

ただし，新しい課題に取り組んだりじっくり振り返りをしたりする授業時間が確保できない場合もある。また小学生の特に低学年の児童の場合，ある方略がよかったと実感していてもそれを言語化できない場合がある。それらのような場合は，教員が「～したことで，より～できたね。」のように価値づけることも有効であると考える。

日本の小学校段階におけるプログラミング教育は，各教科の中で行われるものであり，まずその教科の内容に関する振り返りを行う必要がある。さらには，不慣れな教材を使ったことで授業時間に余裕がないことも考えられる。まずは，児童がプログラミング的思考を発揮した瞬間を教員が見逃さず，確実に価値づけることを目指したい。

（３）思考ツールの活用

プログラミング的思考をメタ認知させるには，どのような手立てが有効であろうか。明示的な指導を行うためには，思考を可視化することが有効である。そのために思考ツールを活用する。

思考ツールとは,「アイデアを可視化して考えを生み出したり，共有して協働的に考えたりすることを助けるツール」である(26)。関西大学初等部では，思考スキルの習得・活用を目指す学習が開発・実践されており，児童は，対象とした六つの思考スキル（比較する，分類する，関連付ける，多面的に見る，構造化する，評価する）をそれに適した思考ツールと対応付けて習得し，様々な授業場面で活用している。この学習のプロセスは，どう考えるかを明示的にしながら方略として習得，他の場面で活用するという，まさにメタ認知を育成・促進し思考スキルを転移させるものである。そこでは「思考スキルを課題に合わせて活用する姿が確認された(27)」,「多くの児童が思考スキルの理解とそれに対するメタ認知的知識を習得している(28)」, といった一定の成果を得ていることから，プログラミング的思考の一部を明示的にするために思考ツールを用いて可視化することは，その転移に有効であると考える。

また，すでに述べたように，教員からの指示が明示的であることも重要である。思考ツールを使えば明示的になるのではあるが，思考ツールを使うことが不適切な場合もあり，そのような場合には，教員が，どう考えるのかを具体的に指示する必要がある。このことは第2章1節3項で述べたように，考えることが苦手な児童への支援にもなる。さらに，視覚優位の児童にとっては図にすることでイメージしやすくなる，説明する時の足掛かりになるなど，思考を可視化することは複数のメリットを生むであろう。

なお，このように思考ツールを用いることなどを通して行うコンピュータを使わないプログラミング教育は，アンプラグドと呼ばれている。

（４）本研究の構想

ここまで述べたことを整理し表したものとして，本研究の構想を図2-7に示す。

プログラミング的思考の一つ一つの要素は，なにも真新しいものではない。学習指導要領に示されている通り，情報活用能力であり論理的思考であるから（しかしあくまで計算機科学の流儀で考えることなので,論理的思考のすべてではない。),部分的にはこれまでの学習でも育まれていたものであるし，日常的に散見するものでもある。しかしながら，無意識的に使っていたり部分的に使ったりしているので，必ずしも問題解決場面で有効に働いているとは言えない状況なのかもしれない。

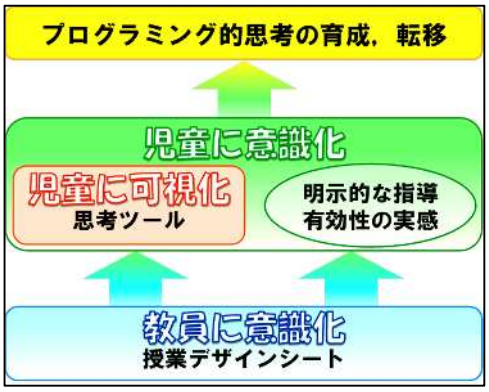


図 2-7　本研究の構想

授業デザインシートを活用することで，教員はプログラミング的思考を意識し,授業構想を練り，児童にどう考えさせるかを具体的に考えやすくなる。そのことで，児童がどう考えればよいか，ど

う考えたのか，発問や評価によって意識させることができる。

思考ツールを活用することで，児童の思考が可視化される。「順序立てる」や「比べる」ということがどういうことか，今自分がどんな風に考えているのか，目に見えてイメージしやすくなる。

また，振り返りや発展的な課題を解決する学習において，あるいは教員の価値付けによって，児童はどう考えたことでどんな良さがあったのかプログラミング的思考の有効性を実感していくことができ，使ってみようと思えるものになる。

プログラミング体験などを通して単にプログラミング的思考を発揮させるだけでなく，教員がプログラミング的思考を意識し，児童に可視化しつつ意識化していくことが，プログラミング的思考を活用できる人間を育てることになるのではないだろうかと仮定し，実践と検証を行う。

(19)黒田昌克，森山潤『小学校段階におけるプログラミング教育に対する教員の意識と意義形成要因の検討』教育メディア研究　Vol.24，No2，43-54　2018.p.48

(20)文部科学省『小学校学習指導要領』2017.3.p.63

(21)奈須正浩『「資質・能力」と学びのメカニズム』東洋館出版社 2017.pp62-66

(22)深谷達史『メタ認知の促進と育成』北大路書房 2016.3.20 p.8

(23)前掲(22)p.168

(24)前掲(22)

(25)前掲(22)

(26)黒上晴夫『初等中等教育におけるシンキングツールの活用』情報の科学と技術 67巻10号，521-526　2017　p.521

(27)小島亜華里，三宅貴久子，泰山裕，黒上晴夫『思考スキル習得のためのシンキングツール活用』日本教育メディア学会第20回年次大会発表論文集 2012.pp.43-44

(28)三宅貴久子，泰山裕，久保田賢一『初等教育における思考力を育成するための授業の実施と評価』日本教育工学会第 28 回全国大会発表論文集 2012.pp.489-490

第３章　実践の実際

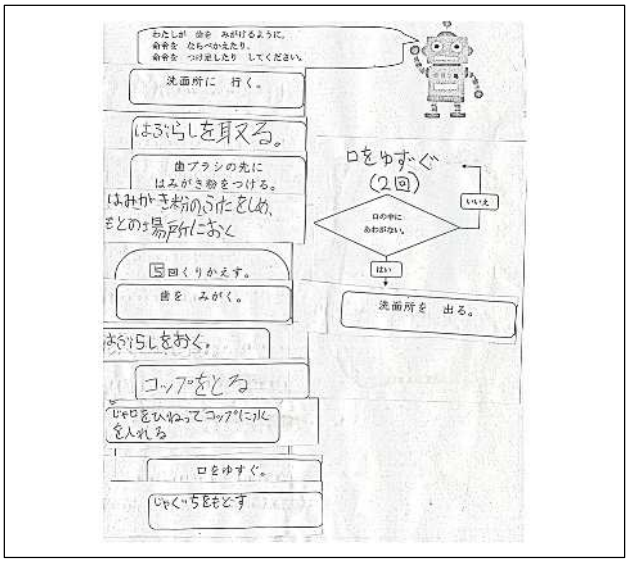
第１節　段階的，系統的に育む

実践は，２年生と５年生において行った。実践計画を立てるにあたっては，プログラミング的思考やそれを促す思考ツールが教科のねらいの達成に寄与するかどうかを第1条件としている。ただし，できるだけプログラミング教育としての系統性が保たれるようにも配慮した。表3-1にその計画を示し，どのような意図に基づいた系統性があるのかを説明する。

表 3-1　実践計画

	2年生	5年生
6月	ロボットにはみがきをさせよう（アンプラグド）	
	たし算とひき算のひっ算①（アンプラグド）	
7月		合同な図形（アンプラグド）
8月		
9月	たし算とひき算のひっ算②（アンプラグド）	整数（Scratch）
10月		
11月		
12月	三角形と四角形（Scratch）	プログラミング体験（Scratch）
1月		
2月		予定：円と正多角形（Scratch）

両学年ともにまずは，プログラミング教育の導入となる活動を行った。それは，ロボットに歯磨きをさせる命令を考え，その命令でロボット役の教員が意図した通りに動くか確かめるものである。この活動を行った理由は，分解やアルゴリズム的思考のイメージをもちやすくするためである。また，これから学習することがコンピュータを動かすこととつながっていると理解してもらうためでもある。先生がロボット役をやり，児童の考えたプログラム通りに動くので，高価なロボット教材が無くてもできる。以下に，５年生が考えたプログラム例を示す。



初めてこのような活動を行ったのだが，予想以上に細かい手順に分解していたので驚いた。分岐や反復のイメージをもてていることも確認できる。なお２年生は5～8個の手順に分解できていたが，学年ごとに分解できる細かさや，扱える手順の量に差があることがわかった。

（１）２年生の実践計画について

２年生は,「三角形と四角形」の単元においてビジュアルプログラミングソフト（本研究においてはScratchを使用しているため，以下すべてScratchと記述する）を使用して正方形と長方形を描くことをゴールとしている。

この学習では，キャラクターに四角形を描かせるための「右に2㎝すすむ→上に２㎝すすむ→左に2㎝すすむ→下に2㎝すすむ」というような順次処理のプログラムを組む活動を行う。つまり，四角形を描くために必要な動きを分解し，アルゴリズム的思考を発揮して正しい順序を構築する必要がある。

そのためにまずは，ある大きな動きを分解し順次処理のアルゴリズムを考えることに慣れる必要があった。そこで,「たし算とひき算のひっ算」の単元において，筆算の手順をステップチャートに表す活動を行った。

「たし算とひき算のひっ算①」では，筆算以外の加減の方法と筆算を結び付けてとらえたり，繰上がりや繰り下がりの仕組みについて，数え棒などを使って理解を深めたりすることが重要である。そこでステップチャートの活用は，指導者が黒板に児童の発言を整理する場面にとどめた。

「たし算とひき算のひっ算②」では，すでに児童が筆算の利便性に気付いていることを前提としたうえで，既知のやり方をどう改善すればよいのかを考えるという課題意識を児童にもたせた。そして課題に合った筆算の手順を構築する自力解決の場面で，児童が考えを表現する手段の一つとして活用した。

なお,「三角形と四角形」の単元でのScratchを用いた活動のための事前の指導は行わないことにした。しかし,ICTスキルに差がある実態を踏まえ,マウス操作と半角入力ができれば使えるような教材を準備し，それら技能を習得する時間15分を含めた60分を授業時間とした。

（２）５年生の実践計画について

５年生は,「円と正多角形」の単元において,Scratchを用いて正多角形を描くプログラムを作成することがゴールになる。まず正方形を描くプログラムを作成し，それを利用して正三角形や正六角形やその他の正多角形を描くプログラムにしていくという活動の流れを想定している。

この活動を算数科の限られた時間数の中で成立させるためには，２年生のプログラミング体験同様一定レベルのICTスキルが前提となる。Scratchを使うためには，半角入力，マウス操作，ファイル管理の技能が必要になる上，扱う内容もブロック（命令）の種類も2年生のそれより高度になるからだ。これら技能の習熟が足りない児童が多いクラスでScratchの活動を行うと,コンピュータの操作に手間取り，その授業のめあてにせまる思考をする時間は減ってしまう。５年生の実態と内容の難しさをふまえて,段階的にScratchに慣れるよう実践を計画した。

そのために行ったのが,「整数」の単元において「ネコが数を1から数えて，ある倍数や公倍数の時に特別なリアクションをする」プログラムを考える活動である。この時間ではまず各ブロックのイメージを児童がつかむ必要があったため，指導者の指示のもと各ブロックの意味を確認しながらプログラムを組むようにした。そして児童の活動の中心は，プログラムの一部分を修正することに限定するようにした。

「合同な図形」の単元は,「円と正多角形」でのプログラミング体験をより効果的なものにするため行った。両単元には共通するプログラミング的思考が働く。それは類似性からパターンを見つけて別の場合にも利用できるようにする「一般化」の思考，そして共通点を見つけ出して必要な部分を選ぶ「抽象化」の思考である。ステップチャートでもプログラミング体験でもそれらの思考を発揮し，可視化して捉えることが，より確実なプログラミング的思考の育成につながると考える。

「円と正多角形」は2月に行う単元であるため,本論文作成時期の都合から本年度の研究に反映することができない。そこで代わりに，長方形や平行四辺形などの四角形を描くプログラムを作る活動を行い,「整数」の単元の学習によって,Scratchを用いた活動がスムーズに行えるようになったかどうかの検証を行った。長方形や平行四辺形の描き方については４年生で学習しており，算数科としては新たに知識や技能を習得することなく取り組める内容であるため，プログラミング的思考や

ICTスキルの面に注目して観察できると考えたからである。

第２節　授業デザインシートを生かして

授業デザインシートは，教材研究の際に，授業中のどの活動においてプログラミング的思考が働いているのか，その活動が各教科での学びをより確実なものにすることにつながっているのか，確認するために使うものである。

本研究に協力してくれた研究協力員は，これまでプログラミング教育もScratchなどのプログラミング体験も行ったことがない教員たちである。研究の初期段階からプログラミング教育としての授業がうまくいったわけではなく，思考は働いたが教科の学びを確実にすることには至らなかった例もある。しかしながら授業デザインシートを活用しながら教材研究を一緒に行った結果として，プログラミング的思考を伴う活動が，教科の学びをより確実なものにした実践となった。

また，授業デザインシートを使ったことでプログラミング的思考の5要素を意識しながら授業をすることができたとの感想を，実践後のインタビューで得ることができた。以下に，その例を示す。

（１）思考を促す発問をした例

２年生「足し算と引き算のひっ算②」において，「65＋78」という繰り上がりが2回ある筆算の方法を考える場面である。前時に「54+72」で百の位に繰り上がる筆算を学んでおり，その手順をもとに考えた。以下，左に前時の手順と右に本時の手順を示す(29)。

図 3-1　筆算の手順を表すステップチャート

二つの手順の違いは，たった一つのブロックである。しかし，筆算はできており繰り上がりが増えたことを理解してはいたが，それを順序だてて説明しようとするとできない，あるいはステップチャートに表すことができない児童が少なからずいた。以下，指導者の発問と，自力解決後の集団解決の場面でのやりとりを示す。

T「これは昨日の図。①今日は昨日よりレベルアップしてるんやな。ということは。」
C「つけたす。」
T「①どこになにを。」
（自力解決）
T「今日は②これが増えたよって気づいた人いる。」
C「どこに増えたかはわからん。」
T「どこに増えたかはわからんけど，どっかに増えたことは気づいた。」
C「あってるかわからんけど，『１繰り上げる』が増えていると思います。」
T「どこに繰り上げるんやろう。」
C「十の位に１くりあげる。」
T「どこに増えたん。」
C「1 の位の計算をするの後だと思います。」
C「なぜなら，十の位の計算のあとに『十の位に繰り上げる』をすると，もう十の位は計算しているから，おかしいし。」
C「③一の位を 5＋8 の計算してから，十の位に１繰り上げなあかんから。」
T「筆算と（図を）あわせながら説明できる。」
C「（前略）④一の位は 5＋8 で計算をして 13 なので，十の位に１繰り上げます。」

下線①②が思考を促す明示的な発問である。既習の手順をもとにして（一般化），本時の問題解決に至るよう必要な動きを考えて（分解），正しい手順を構築する（アルゴリズム的思考）ことを促している。そして新しく「1繰り上げる」が増え，どこに付け足せばいいかを考えることを通して，下線③④のように繰り上がりについて繰り返し説明が行われていた。

結果，本時の理解を見取るための適応題は25名中24名できていた。またこのような実践を繰り返したことで，宿題の練習問題でも繰り上がりや繰り下がりのできない児童はいなかったということが，研究協力員への聞き取りからわかった。

（２）有効性を実感させた例

本節1項における実践では，類似性からパターンを見つけて他の場合にも使う一般化の有効性を実感させた場面があった。

先に示した図3-1を見るとわかる通り，二つの手順には同じパターンが使われており，1手順付け足すだけで問題解決に至ることができる。多くの児童がはじめから本時の筆算の手順を描いた中，1名が前時の手順をまず写し，そこに「1繰り上げる」を挿入した図を描いていた。そこで研究協力員が，「ここに付け足すって思って矢印で入れたんやって。これもわかりやすいな。」と価値付けていた。もちろん児童は一般化したとは思っておらず，もしかしたらただ最初はわからなかったから前時のものを写し，集団解決の中で付け足しただけなのかもしれない。しかし，「使えるパターンは使ってよい」という姿勢を示すことにはなったであろう。

有効性を実感させることは，コンピュータを使ったプログラミング体験でも重要である。５年生で四角形を描くプログラミング体験を行った際の例を示す。

児童はまず正方形のプログラムを作成した。以下の図3-2はその例である。

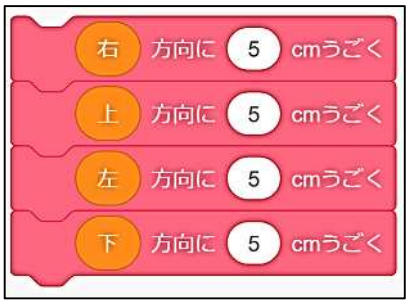


図 3-2　正方形を描くプログラム

作成のための時間をとった後，複数の正方形の書き方を発表してもらい，黒板に残した。そして長方形や平行四辺形など様々な四角形に挑戦するよう促し，児童がしばらく取り組んだ後，指導者が「長方形を描くにはどうしたらいいかな」と聞いた。すると児童が黒板の前に出てきて，正方形を描くプログラムをもとに数字を変えて説明した。それが以下である。

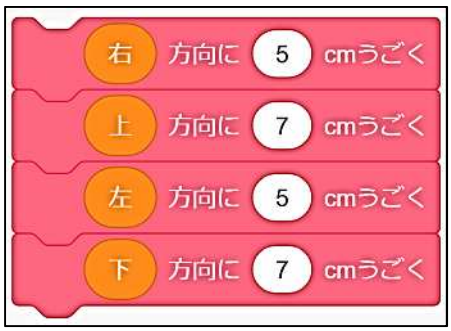


図 3-3　長方形を描くプログラム

このプログラムに従って実際に長方形を描いたあと，「ほんのちょっと変えるだけで別のものが描けるんだね。」と指導者が取り上げている。

この体験の後に行った振り返りの記述を示す。

・少しの工夫などで(できることが)かわるのがおもしろかったです。
・どこかをちょっと変えれば違う形になることがわかった。

あるパターンを少し変えて，違うものが作れるという体験を通して，パターンを見つけ出す一般化のよさに触れることができたと考える。

第３節　児童へ可視化，意識化して

（１）合同な図形を描く手順

５年生「合同な図形」の単元において，児童が考えた合同な三角形を描く手順を指導者がステップチャートに整理して黒板に示した。その手順は以下である。

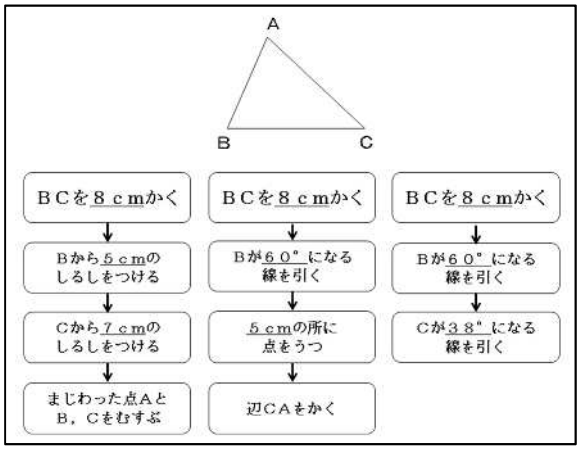


図 3-4　合同な三角形を描く三つの手順

合同な三角形を描く場合，3辺3角の6要素のうち3要素を調べればよく，その調べ方は「3辺」「2辺と間の角」「1辺とその両端の角」の3通りがある。このことを学習する授業においては，これら3通りの描き方を考えるだけでなく，「三角形は3点が定まれば形と大きさが決まること」「1辺を定めて，残り1点を定めれば決まること」などを関連付け，3要素を調べれば3点が決まることに気付かせたい。また，「三角形を決めるのは3点，ならば四角形はどうなるか」などと発展的に考えるための準備段階とも捉えられる。

授業の導入において，指導者が「3点写すことで合同な三角形が描けた」体験を想起させており，児童は「紙に写さず3点決めるにはどうすればいい

か」というねらいで学習に臨んでいる。また,「最初に辺ＢＣをひく＝2点は決めてしまう」ことは全員共通とし,残り1点を決めることに児童の思考を限定した。

図3-4の考え方が出た後，指導者が「三つの方法の違いや共通点は何か。」と発問し，グループで考えさせた。プログラミング的思考としては，具体的な辺BCや5ｃｍと表現されていたものを，「3辺」「3要素」などと抽象化させるねらいである。児童から出た意見は以下である。

- どこを使うかは違う
- 辺の長さと角度を調べるのは共通する。
- 三つの場所を測っている。
- 全部の角や辺を調べる必要はないとわかった。

自分たちの考えが整理されて可視化されたことおよび思考を促す発問がされたことで，児童が話し合いの中で「3要素を調べれば三角形が描ける」ことに気づけたことがわかる。

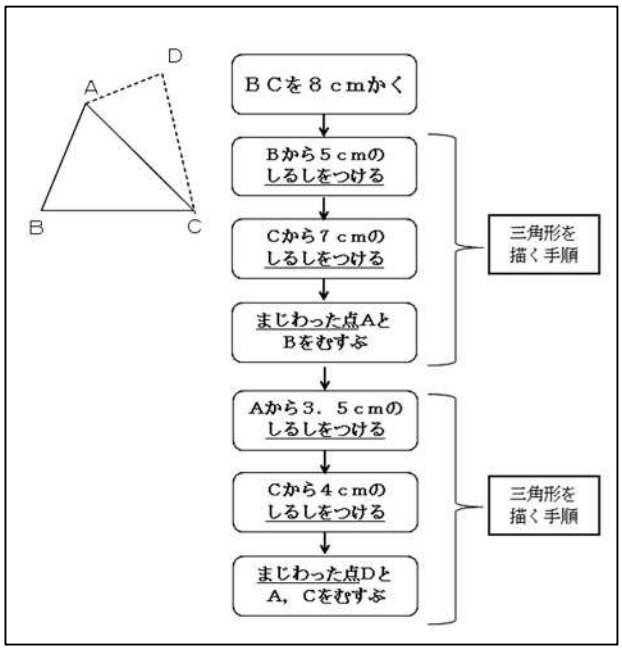


図 3-5　合同な四角形を描く手順
（児童のノートを筆者が整理したもの。下線は，筆者）

次時は合同な四角形の描き方を考える時間だが，四角形を「三角形に点を一つ加えたもの」と捉えることで,1点を決めるためには合同な三角形を描く方法が使えるとの推測ができる。

実際，ステップチャートにしてみると合同な四角形を描く手順は，合同な三角形を描く手順の組合せで成り立っていることがわかる。ある方法に含まれるパターンが他の場合にも使える「一般化」を，児童にわかりやすく示すことができた。

また，授業後に指導者へインタビューを行ったところ，以下のような効果が確認された。

- 普段きまりを見つけることが苦手な児童が，積極的に発言していた。
- ステップチャートに描くことでわかりやすかったと言っていた。
- 次の時間（合同な四角形の描き方を考える時間）は，（指導者が言っていないのに）自分たちでステップチャートに書いていた。そして，共通点がないか考えていた。
- 自主学習で，別のことのステップチャートを描いていた。

これらのことから，思考を可視化して問題解決に至る取り組みを繰り返すことで，抽象化したり一般化したりといったプログラミング的思考の有効性を実感させ，転移を促すことができると考えられる。

（２）筆算の手順

第3章2節2項において述べたように，２年生「足し算と引き算のひっ算②」では，筆算の手順をステップチャートに整理して示す活動を行った。ここでは，問題や事象をいくつかの部分に理解や解決できるように「分解」することや，明瞭手順を組む「アルゴリズム的思考」を可視化することを意図している。

第6時，2回繰り上がる足し算の筆算や三つの数を足す筆算について学習した後，「135－72」の筆算のやり方を考える場面である。多くの児童はステップチャートを描くことに慣れ,図3-6のように描いていた。

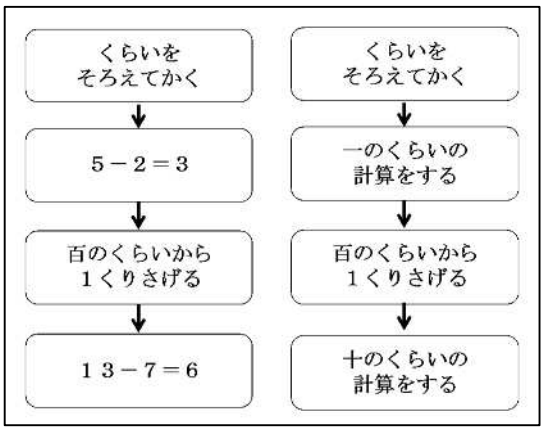


図 3-6　135-72 のステップチャート

筆算のやり方について話し合っていき，このやり方でよいという意見にまとまったころ，指導者が「先生はこういう手順もあると思うんやけど。」と「数をくらべて，ひけない」と書かれたひし形

のブロックを示した。手順の中には，こういった判断も含まれることを示す意図がある。

ひし形のブロックは，ステップチャートでは条件分岐を示すものだが，２年生という発達段階をふまえ，分岐して複線にはせず，どういう時に繰り下げるのかという判断基準を示すブロックとして扱った。児童の話し合いの結果，完成した図を以下に示す。

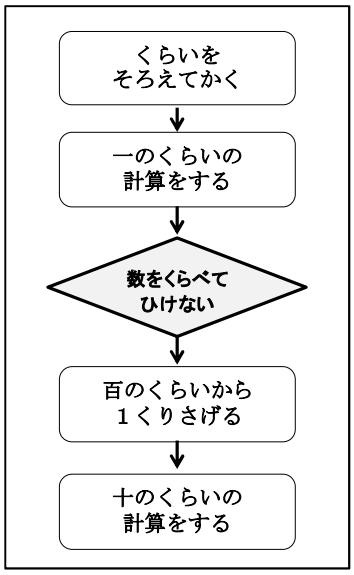


図 3-7　どんな時に繰り下げるかを示す図

この一つのブロックをどこに入れるかを児童は考え，それぞれの思う位置に挿入した。その際，とりあえず入れてみて，しばらく考えた後，他の場所に入れ替えるという評価の思考を働かせている様子がうかがえた。また，友達と考えが異なると，そこに自然と対話が生まれていた。その際「まず」「次に」と順番を意識したり，「ひけなかったから，百の位から借りてくる。」と原因と結果を意識したりしながら，短い文に区切って説明している様子がうかがえた。ステップチャートにすることで，順番をより意識しながら考えることができたのではないだろうか。

第４節　思考とプログラミングをつなげる

ステップチャートを使ったアンプラグドでの学習を通して児童は，分解，アルゴリズム的思考，一般化などのプログラミング的思考を意識化してきた。しかし，コンピュータを用いたプログラミング体験を経ていない状態では，それらの思考と「コンピュータに意図した処理を行わせること」とは結び付いていない。自分たちが学習したり発揮したりしてきた思考が，コンピュータに意図した処理を行わせることに生かされていると気付けるよう，それまでの学習とプログラミング体験とを関連付けることを意識して実践を行った。

（１）５年生「整数」

先述した通り(p. 12)，児童が段階的にScratchに慣れることも意図して，「整数」の単元において「ネコが数を1から数えて，倍数や公倍数の時に特別なリアクションをするプログラム」を考える活動をした。

なお，算数科としてのめあては，倍数について「ある数の倍数とは，ある数で割り切れる数」とも捉えられるようになることである。

授業の導入では，数を1から順に数えていき「3の倍数で手を叩く」「5の倍数でジャンプする」というゲームを行った。3と5の公倍数，すなわち15の倍数ではジャンプしながら手を叩くことになる。このゲームによって，これから作ることになるプログラムのイメージをもてるようにした。

次いで，チャートを使って設計を行った。児童が作成したチャートを図3-8に示す(30)。

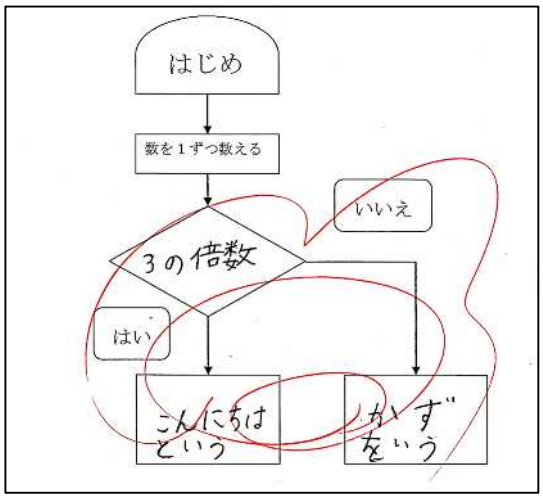


図 3-8　３の倍数でこんにちは！というプログラムのチャート

このチャート作成により，以前に書いたステップチャートと今回のScratchでのプログラミング体験が関連付けられることになる。

そしてScratchでプログラムを作る活動に移ったが，児童のICTスキルの状況，児童にScratchのブロックの意味を理解させる必要があること，初めての児童にとってはプログラムが難しいことをふまえて，指導者の指示に従って一斉にプログラムを作成していった。最初に作ったプログラムを図3-9に示す。

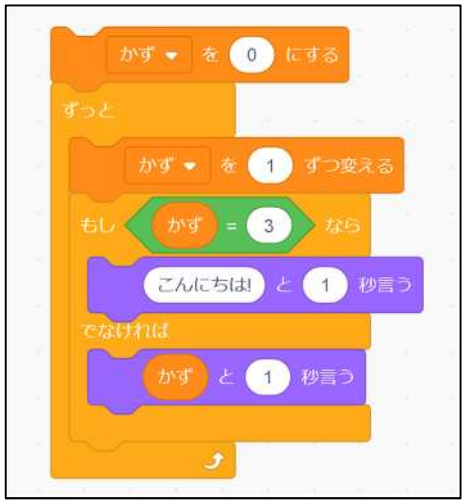


図 3-9　児童が最初に作ったプログラム

このプログラムを実行し，どこかがおかしいと児童が感じたことで，プログラミング教育として重要な活動に進んだ。自分が構築したプログラムを評価し，そのアルゴリズムを見直していく活動である。その際の指導者と児童とのやり取りを以下に示す。

C「このままだと，【もし数が 3 のとき】だと 3 のときしかこんにちはと言わないので，3 のあと，ずっと数しか言わない。」
T「かず＝3 ってどういうことかな。」
C「かずが 3 の時。」
T「かずが 3 の時しか，ってことだね。やってみて。」
C「できたできた。3 の時しか言わへん。」
T「では，どこかをかえて，倍数のプログラムにしたい。3 のときっていう指示をしているプログラムはどこかな。ここかな。」
C（【かず＝3】でうなずく。）
T「そこを変えて作り変えてください。使えそうなブロックは用意してあります。」
C「どうやってかえればいい。」
C「3 の倍数の時ってあればいいのに。」
T「あるかな。」
C「ない。」
C「つくればいい。」
C「とりあえずいれてみよう。」
T「なるほど，①何回失敗しても大丈夫だから，とりあえず入れてやってみるのもありだね。」

下線①は，間違えてもよい雰囲気を作り出すために重要である。プログラミング教育においては，自分が作ったプログラムを目的に合わせて評価し，失敗の原因を探して改善していくことが大切だからだ。失敗を恐れて前に進めないのでは，評価してまた構築し直すという活動になりえない。もちろん行き当たりばったりでは論理的な思考は身につかないので，偶然うまくいった場合でも「このプログラムで成功したのはなぜだろう」と帰納的に考えさせることが必要になる。

この後，児童は【かず＝3】の演算部分を変更してなんとか3の倍数を表現しようと試みた。その際使えるブロックは，指導者があらかじめ提示している。それを図3-10に示す。

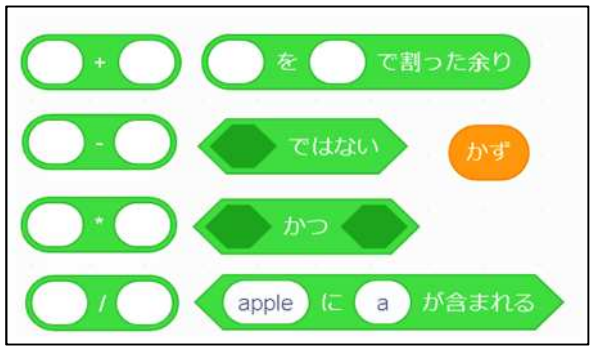


図 3-10　演算部分のブロック

児童はこれまでの学習で，3の倍数を探すときは，3，6，9…と3の段のかけ算をして探してきた。教科書に書いてある定義も「3に整数をかけてできる数を，3の倍数といいます」となっており，多くの児童が「倍数はかけ算」とイメージしていたと思われる。しかし，Scratchにある部品ではそれがうまくできない。そこで，「たし算でできないかな」「ひき算ではどうだろう」などと「倍数」の性質の様々な部分に着目していた。「分解」の思考が働いていると捉えられる。そうしてできたプログラムを図3-11に示す。

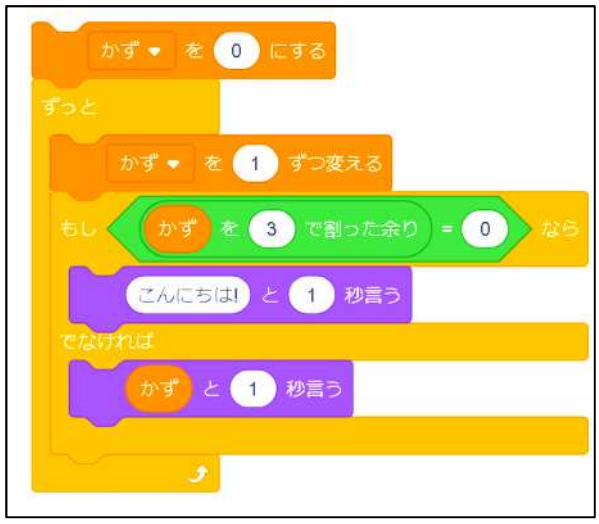


図 3-11　3 の倍数でこんにちは！というプログラム例

このプログラムでは，ネコが1から順に数を数えていって，「条件：かずを3で割った余り＝0」の時に「『こんにちは！』と1秒言う」ようになっている。倍数の性質のうち，3に整数をかけていって探せるという部分を捨象し，3の倍数は3で割り切れるという部分を抽象することで，問題解決に至ることができた。この後，同じプログラムを使い，「5の倍数でこんにちは！というプログラム」「3と5の公倍数でこんにちは！というプログラム」へと応用，発展させる「一般化」を行っている。

授業の終末に行った児童の振り返りでは，算数に関する記述と，プログラミングに関する記述が見られた。算数に関する記述は，「3の倍数を3で割って余りが0だと考えられることがわかった」「倍数や公倍数は割り算でも考えられるとわかった」などである。プログラミングに関する記述は，「組み立てることができるととっても達成感があった」「いろいろなものを組み合わせるのは難しいと思った」などである。どちらの記述も偏った人数ではなく，ほぼ半数ずつであった。また，両方の内容を含む記述をしている児童もいた。

以上のことから，授業の中でプログラミング的思考が働いており，その活動が教科の学びをより確実なものにしていることが読み取れる。

さらに「むずかしかったけれど，楽しい」「いっぱい考えて楽しかった」「達成感があった」などの記述も多く見られ，難しい課題にあきらめずに取り組んだことに楽しさを見出していることがわかった。

（２）５年生「プログラミング体験」

この単元は，「円と正多角形」の学習を行う前に児童の成長を見取るために行った単元であるが，これまで行った実践とプログラミング的思考を関連付けることを意識して行っている。なお，算数科としては「図形の特徴を根拠にしながら，プログラムの意図を説明すること」をめあてとしており，知識を活用する時間として扱っている。

授業の導入は，二十二角形の模様とそのプログラムを見せることから行った。自分で書こうと思うと大変な模様を倍数のプログラムよりも少ないほんの数行のブロックで書けることを知った子供たちは，「プログラムはすごい」「やってみたい」と思えたようだった。

そしてまず簡単な図形，正方形から描くために，「分解」「抽象化」「アルゴリズム的思考」を働かせるように促した。以下，そのやり取りである。

T「(【わける】のカードを示しながら)正方形ってどんな特徴がありますか。」
C「辺の長さがすべて等しい。」
C「角がすべて直角。」
C「辺が４本。」
C「角が四つある。」
T「なるほど。①ロボットの動きを考えた時も，どんな動きが必要か，最初に分けて考えましたね。では，キャラクターに正方形を描かせるとき，どの特徴を使って描きますか。②倍数の時はかけ算で考えるっていう特徴は使えなくて，割り算で考える方法を選びましたね。(【えらぶ】のカードを示す)」
C「辺が４本あること。」
T「それをつかってどんな命令をしますか。」
C「５ｃｍとかを４回言う」
C「(直角なので)90°曲がるように言う。」
T「なるほど。そんな風に必要な動きを選んで，正しい手順を作ってください。(【ならべる】のカードを示す)」

下線①②は過去の体験と本時の体験，そしてプログラミング的思考を関連付ける言葉である。また，ある思考にはどんな名前がついているのか，カードを用いることで明示的にしている。カードを用いた板書は，図3-12に示しておく。

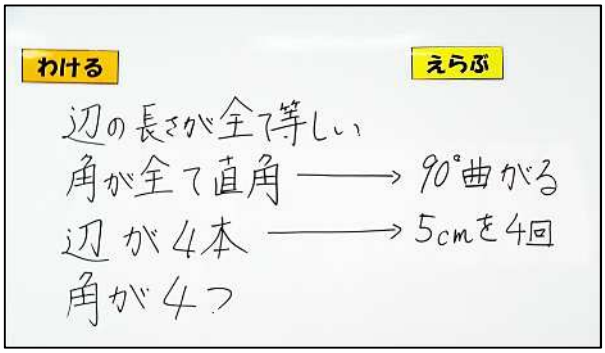


図 3-12　明示的にするカードの使用例

授業の終末に行った振り返りでは，以下のような記述がみられた。分類して示す。

【プログラムの多様性】
・プログラムはあっていれば，様々な作り方があるのだとわかった。
【一般化】
・少しの工夫で違うものに変わるのが面白い。
・ちょっと変えれば違うものになるのだとわかった。

【分解】
・特徴にわけて考えるとできるとわかった。
・ほかの図形も特徴を考えて作ってみたい。
【アルゴリズム的思考】
・必要な条件や数字があっていても，順番が違っているとうまく作れないことにも気づいた。
【評価】
・動作してみながら，少しずつ正しいプログラムを見つけられた。
【あきらめずに取り組む姿勢】
・頭をすごく使って楽しかった。
・最初はできなかったけれど，だんだんできるようになってくると楽しかった。
・できなくてもあきらめずに挑戦できるようになった。

【一般化】や【分解】【アルゴリズム的思考】などプログラミング的思考に関わる記述があり，児童のなかに意識化されていることが読み取れる。また【プログラムの多様性】のように，こちらが特に意図していなかったが児童が気付いたものもあった。さらに，「整数」での実践同様に，難しい課題にあきらめずに取り組むことを楽しいと感じていることが読み取れた。

（３）２年「三角形と四角形」

２年生は四角形を描くプログラムの作成をした。この単元では，三角形や四角形を辺の長さや本数や角の形に注目してとらえることがめあてになる。あるキャラクターに四角形を描かせるプログラムを作る活動を通して，プログラムと四角形の定義を関連付けることで，より学びを確かにすることが算数科としてのねらいだ。

パソコンに初めて触れる児童が多いクラスだったので，機器操作習得の時間も含めて 60 分の授業を設定した。パソコンは，台数の関係から２人ペアに１台となってしまったが，理想は１人１台である。

まずは５年生と同様，【わける】のカードを示しながら長方形や正方形の特徴を確認した。そして，ステップチャートの枠を描き，どんな命令をするかを尋ねた。以下がそのやり取りである。

T「長方形ってどんな形かな？」
C「辺が4本あります。」
C「角が四つあります。」
C「角は直角です。」
C「上と下の辺の長さが同じです。」
C「右と左も同じで，むかいあうのが同じです。」
T「いろいろなきまりに分けて考えられたね。じゃあ，こんなきまりの図形をえんぴつくんに描いてもらうためにどんな命令をするかな？」
C「①まず鉛筆を持ちます。」
T「どうして？」
C「鉛筆を持たないとかけないからです。」
T「確かに。鉛筆持ったらどうしたらいいですか。」
C「②右に線を引くってしたらいいと思う。」
T「このコンピュータでは【～に～ｃｍ動く】って命令したら，その通りに動いてくれるからね。」

下線①は分解とアルゴリズム的思考が働いている。このクラスは，パソコンに触れることさえ初めてという児童が多く，当然 Scratch に触れたことはない。【鉛筆をもつ】というブロックがあることを知るわけがないのである。誰かが図形を描くにはどうすればいいかという行動を正確に思い浮かべ，分解しないと浮かんでこない回答である。また下線①も②も言葉が短いことにも注目したい。２年生といえば，説明する際に「～して，～して」と長い文章になることもしばしば見受けられる発達段階である。筆算の授業でたくさんステップチャートを書き，端的に順序だてて表現することに慣れてきた結果ではないだろうか。

この会話の中で指導者は【えんぴつをもつ】→【～に～ｃｍうごく】とステップチャートに記入すると，その後は児童が自由に考える時間とした。

図 3-13　長方形を描かせるプログラム

図 3-13 は児童が作成したプログラムである。辺の長さや縦横どちらから描かせるかは児童が決めている。児童が思考する時間を十分に与えた後，

集団解決を行った。以下が，そのやり取りの一部である。

T「なぜここととここが同じ数字なの。」
C「①むかいあう辺の長さが等しいから。」
C「長さが違うと，線がくっつかない。②囲まれていないから四角形じゃない。」
T「下に９ｃｍ進むでもかけるかな。」
C「むり，画面からはみ出る。」
T「これって③順番変えてもうまくいくかな。」
C「部品は一緒だから大丈夫じゃない。」
C「④無理です。右に行ってから左に行ったら線が重なっちゃうからです。」

下線①②のように，プログラムの意味を説明することを通して，自然と図形のきまりに関する知識が表現されている。また，下線③④では，「問題の解決には必要な手順があることに気付くこと」というプログラミング教育のねらいが達成されていることがわかる。

(29)この手順は，筆算のアルゴリズムを示した正しいステップチャートとは言えない。しかし，正しいステップチャートの書き方を習得することが目的ではないので，児童の実態に合わせてこのような形式にした。

(30)このステップチャートは，図 3-11 のように反復するものになっていない。しかし，本時では分岐のイメージをもたせることを重点としたため，この図を採用した。

第４章　実践の成果と課題

第１節　プログラミング教育の効果

（１）可視化，意識化による効果

実践を行う前に，研究協力校の児童にプログラミング的思考に関する事前アンケートを行った。対象は５年生1学級24名，２年生2学級55名である。プログラミング的思考の5要素および「あきらめずに取り組む姿勢」について，「どういうものかわかっているか，有効性を実感しているか」「実践しているか」を問うたものである。

「あきらめずに取り組む姿勢」とは，問題解決を支える姿勢である。仮に様々な思考を働かせて解決手順を構築したとして，その評価を行った結果，失敗と判断しあきらめてしまったら問題は解決しない。成功するまで何度も何度も構築と評価を繰り返し，より目的や意図に合うものを作り出そうとしていく，そのような「あきらめずに取り組む姿勢」がプログラミング的思考を支える姿勢として重要だと考えている。そのような姿勢が，プログラミング教育によってさらに育まれる可能性があるとも考え，設問に含むことにした。

有効性の実感についての回答には「そう思う」「大体そう思う」「あまりそう思わない」「そう思わない」「わからない」の5項目を用意した。実践しているかについての回答には「よくできている」「だいたいできている」「あまりできていない」「できていない」の4項目を用意した。2019年7月～9月に行った。以下がその内容である(32)。

表 4-1　プログラミング的思考および関係する姿勢に関するアンケート

分解	問題やモノをいくつかの部分に分解すると，その構成や全体を理解しやすくなると思う。
抽象化	いくつかのものの共通点を見つけると，そのグループの特徴や決まりが見つけやすくなると思う。
アルゴリズム的思考	問題を解決する時に，やるべきことの手順を考えることで，解決しやすくなったり，他の人に伝えやすくなったりすると思う。
評価	問題を解決する時は，間違いを直したり，他のより良い方法を考えたりしながら，解決方法を考えるようにすることが大切だと思う。
一般化	問題を解決する時は，同じような種類の問題におきかえたり，その解決方法を参考にしたりすると，考えがうかびやすいと思う。
あきらめずに取り組む姿勢	問題の解決方法を考えても，うまくいかないこともあるから，あきらめずに何度も挑戦して修正することが大切だと思う。

これは５年生用のアンケートだが，２年生については簡易な言い方に直したもので実施している。結果は以下の表4-2の通りである。

表 4-2　事前アンケート結果 A 校５年生（単位は%）

A校５年生 (n=24)	有効性の実感		実現度
	そう思う ＋大体そう思う	わからない	良くできている ＋大体できている
分解	13.0	87.0	4.3
抽象化	39.1	34.8	21.7
アルゴリズム的思考	73.9	17.4	47.8
評価	90.9	0.0	54.6
あきらめずに取り組む姿勢	96.5	0.0	65.2

分解や抽象化の有効性の実感について「そう思う」「大体そう思う」と答えた児童の割合が，他の3項目に比べて低い数値となった。一方，「わからない」と答えた児童の割合は高い数値となった。多くの児童が，それらの思考がどのように考える

ことなのかイメージできていなかったということであろう。一方，手順を考えるアルゴリズム的思考やその手順を目的や意図に応じて確認する評価については，イメージがしやすいようである。同様に，あきらめずに取り組む姿勢についても高い数値となった。他の教科や日常の学級経営，あるいはこれまでの生活体験の中で意識されたり，育まれてきたりしているということであろう。

このような傾向は，実践を行った他の2クラスにもうかがえた。以下，表4-3および表4-4に示す。

表 4-3　事前アンケート結果 A 校２年生（単位は%）

A校２年生（ｎ＝25）	有効性の実感		実現度
	そう思う＋大体そう思う	わからない	良くできている＋大体できている
分解	20.8	50.0	32.0
抽象化	52.0	20.0	40.0
アルゴリズム的思考	66.7	20.8	56.0
評価	87.5	0.0	70.9
あきらめずに取り組む姿勢	60.0	8.0	66.6

表 4-4　事前アンケート結果 B 校２年生（単位は%）

B校２年生（n=30）	有効性の実感		実現度
	そう思う＋大体そう思う	わからない	良くできている＋大体できている
分解	27.6	58.6	41.3
抽象化	34.5	48.3	17.2
アルゴリズム的思考	69.0	20.7	58.6
評価	68.9	10.3	58.6
あきらめずに取り組む姿勢	75.9	6.9	51.7

これらの事前アンケート結果から以下のような示唆を得た。それはプログラミング的思考には，児童のこれまでの経験でイメージしやすいものとそうでないものがあるということである。

先に述べたように,プログラミング的思考が様々な問題解決にあたって応用されるような力，すなわちメタ認知的活動に組み込まれる方法となるには，それがどういう方略なのか子どもたちが理解していなければならない。このことを踏まえた時に，プログラミング的思考の育成においては分解や抽象化の思考をより意識化させることが重要であると言える。

また，２年生の分解については，有効性は実感できず，よくわかっていないにもかかわらず実現できているという結果になっている。イメージがわからなかったことで正しく判断できなかったのだと考えられる。それを除けば，おおむねどの項目もその実現度は有効性の実感の数値に比べて低い数値となっている。

では，実践によって児童がどのように変容したかを述べる。事後アンケートは，授業実践終了直後2019年12月に行ったものである。

まずは有効性の実感についての変化を述べる。例として,A校５年生の事前と事後での回答の比較を表4-5に示す。

表 4-5　A 校５年生における有効性の実感についての事前事後比較(単位は%)

A校５年生（ｎ＝20）	そう思う＋大体そう思う		わからない	
	事前	事後	事前	事後
分解	13.0	85.0	87.0	10.0
抽象化	39.1	100.0	34.8	0.0
アルゴリズム的思考	73.9	100.0	17.4	0.0
評価	90.9	95.0	0.0	0.0
あきらめずに取り組む姿勢	96.5	100.0	0.0	0.0

分解の思考について有効性を実感していると回答した児童の割合は，13.0%から 85.0%に 72 ポイント増加した。分解同様，事前アンケートではイメージされにくかった抽象化についても,39.1%から 100.0%へと 60.9 ポイント増加している。わからないと回答した児童の割合もそれぞれ減少している。また，分解と抽象化以外の 3 項目についても，有効性を実感していると回答した児童の割合が増加していることがわかる。

表 4-6　A 校２年生における有効性の実感についての事前事後比較(単位は%)

A校２年生（ｎ＝25）	そう思う＋大体そう思う		わからない	
	事前	事後	事前	事後
分解	20.8	92.0	50.0	4.0
抽象化	52.0	72.0	20.0	28.0
アルゴリズム的思考	66.7	88.0	20.8	4.0
評価	87.5	96.0	0.0	0.0
あきらめずに取り組む姿勢	60.0	100.0	8.0	0.0

表 4-6 は，A 校２年生の有効性の実感に関する変化について示したものである。分解について有効性を実感していると答えた児童の割合は,20.8%から 92.0%に増加した。さらに「わからない」と答えた児童の割合が 50%から 4%に減っていることもわかる。他の項目についても，有効性を実感

している児童の割合が増加していることが見て取れる。ただし，抽象化については「わからない」が高いままであった。これは，抽象化について意識化する機会に恵まれなかったことに起因すると推察される。

次いで，プログラミング的思考を実際に行っているか，実現度についての変化を述べる。A 校の２年生と５年生に行ったアンケートから，実現度について「そう思う」「大体そう思う」と回答した児童の割合の比較を表 4-7 に示す。

表 4-7　A 校２年生と５年生における実現度についての事前事後比較（単位は%）

	２年生(n=25)		５年生(n=24)	
	事前	事後	事前	事後
分解	32.0	88.0	4.3	75.0
抽象化	40.0	60.0	21.7	80.0
アルゴリズム的思考	56.0	88.0	47.8	70.0
評価	70.9	96.0	54.6	85.0
あきらめずに取り組む姿勢	66.6	100.0	65.2	95.0

すべての項目について，「そう思う」「大体そう思う」と回答した児童の割合が増加していることがわかる。

これらの効果は，プログラミング的思考を可視化し，明示的な指導や価値付けによって意識化し，さらに有効性を実感できるような声かけを行ってきたからといえる。しかし，数値が上昇したことですなわち「プログラミング的思考は育まれた」とは言い切れない。上記アンケートはあくまで自己評価だからである。またアンケートに回答したのが授業実践直後であったことから，授業での楽しさや達成感がアンケート結果に影響をおよぼしていることを否定できない。本当にプログラミング的思考が育ち転移したかどうかは，他教科や日常生活での児童の姿を観察し，そこにプログラミング的思考が発揮されているかを見て取らなければならない。あるいは，ある問題場面を想定してどのように解決するかを問うようなテストを行う必要がある。上記アンケートで言い切れることは，児童がプログラミング的思考の５要素についてイメージができるようになったことと，「できている」と回答できるくらいに，各授業の中にそれらを発揮する場面があったということであろう。今回のように可視化し意識化する実践を繰り返すことで，転移するようになるのかもしれないという可能性を感じることはできた。また，転移したかどうか評価する方法については例えばルーブリック評価が考えられるが，次年度での課題としたい。

（２）あきらめずに取り組む姿

再び表 4-5，4-6，4-7 のアンケート結果に目を通していただきたい。実は２年生においても５年生においても，そして有効性の実感においても実現度においても一番高い数値を示したのは，あきらめずに取り組む姿勢であった。これはこの研究を通して明らかになったプログラミング教育の成果である。第３章４節(2)で紹介した振り返りの内容からも，児童が何度も失敗，修正を繰り返して課題をクリアしたこと，そしてそのような体験を肯定的に受け止め，楽しいと感じていることが読み取れる。

実践後，５年生の児童にプログラミング教育の授業に関して，それぞれ楽しかったかどうかを尋ねたところ，ほぼ全員がどの授業も楽しかったと答えていた。一方，難易度について問うと過半数が「難しかった」と答え，「簡単だった」と答えた児童は，「整数」の授業について２名のみであった。多くの児童が難しかったと感じながらも楽しかったと感じていることがわかる。そのうち３名の児童を抽出し，楽しかった理由について聞き取り調査を行ったところ以下のような回答を得られた。

- うまくいかないのを考えていったことが楽しかった。
- 難しかったけれど，自分で考えて，できたらどんどんわかっていく。
- 次はこうしようと，いろいろ目的を自分で作れるところ。すごく考えたことが自分のためになった。

難しかったのに楽しかったのは，児童が自分自身の力で考えてどう解決するかを決めることができ，さらにあきらめずに取り組んだことで課題を解決することができたからだとわかる。予想困難な時代にあっても自ら主体的に問題を解決する姿，学習指導要領で目指されている姿を感じることができるのではないだろうか。実践後の研究協力員へのインタビューでも，以下に示すような回答が得られた。

Q. プログラミング教育の成果はなんでしょう。
A. 自分の力で何とかしようっていう姿勢がみえた。例えば今日の授業(三角形と四角形)でも，えんぴつくんが消えたとか，トラブルの解決についての質問はあったけど，四角形をどうやって書けばいいかという質問はなかった。

プログラミング教育の授業においては，ぜひこういった児童の姿が大切にされてほしいと願う。

なお，構築したものを評価，さらに修正をしながらあきらめずに取り組む姿勢というのはアンプラグドよりもコンピュータを使ったプログラミング体験のほうが現れやすい。それは，正確な即時評価が起きるからである。例えば，筆算の手順をステップチャートに書いたとして，それが正しいかどうか客観的な判断はできない。思い込みで正しいと判断することもある。しかしコンピュータは，正しいプログラムでないかぎり，意図した通りに動かない。思い込みの入る余地なく，客観的にだれが見ても明らかに失敗だとわかる。だからこそ，すぐ修正せざるを得ない。こうして，「即時評価→修正」というサイクルが何回も繰り返されることになる。そのサイクルの中で思考した量が，達成感にもつながるだろう。コンピュータを使ったプログラミング体験の重要性を確認することができた。

第２節　可視化することが支援に

本研究の授業実践では，思考ツールのうちステップチャートを用いた。ステップチャートを用いて思考や手順を可視化することは，物事の構成要素の違いや共通点を見つけることの支援になる。共通点を見つけることは，それらを関連付けて捉えることにつながる。違いを見つけることは，個々の特徴を導き出すことにつながる。本研究においては，合同な図形を描く手順を可視化することにより，「辺や角度を合わせて三つを調べればよい」という共通点を導いたり，「合同な四角形を描く手順は合同な三角形を描く手順の繰り返しである」ことを気づかせたりすることにつながった。また筆算の手順を可視化することで，繰り下がりが増えていることや，繰り下がりが増えても手順は同じことの繰り返しであることなどに，視覚的に気づかせることができた。

また，見えることで話すきっかけになっている様子もうかがえた。例えば筆算の手順にしても，頭の中で考えて筆算をノートに書いているだけではプロセスは見えない。一続きになった長い文章の説明では読む気にならない。しかしステップチャートになっていると自分の手順と隣の席の友達の手順が異なることが一目でわかる。そこで自然と「なんでそうなったの。」「なんでかというと。」と対話が生まれる。相手を説得しようとするため，既習事項をもとにした根拠を示すことにもなる。ステップチャートを使い思考や手順を可視化することは，児童が主体的に対話的に学ぶきっかけを生み出すといえるかもしれない。

さらには，表現の支援となった場面もあった。2年生では筆算の手順をステップチャートに整理する活動を繰り返し行ったが，多くの児童がステップチャートを描き，それをもとに説明できるようになっていく中，なかなか描けない児童がいた。しかし単元の終盤，2 回繰り下がるひき算の筆算のステップチャートを自力で描き，友達に説明する時も自信をもって説明していた。今日の授業が楽しかったというので理由をきくと，「最初できなかったのに説明できたから。」と言っていた。なんとなくわかっているが，説明できないという児童は多い。ステップチャートを描くことは，それ自体が説明することの練習になり，順番や構成要素が明確になることで説明しやすくなるのではないだろうか。

本研究で確認されたステップチャートの効果を整理すると，

・違いや共通点を見つけることの支援になる。またはそういった思考を促す。
・そのことにより，対話が生まれる。
・表現することの支援になる。

となった。これらの効果を生かすことが，プログラミング的思考を働かせるだけでなく，教科の学びを確かなものにする一助になるのではないだろうか。

第３節　さらなる充実に向けて

（１）プログラミング体験の重要性

本研究においては，アンプラグドでの学習，教科でのアンプラグドの学習，プログラミング体験という順番で行った。そして実践した授業時間数はアンプラグドの方が多い。先にプログラミング的思考がどんな思考かを学習し，それがプログラミング体験で生かされるという展開になっている。これは，筆者に「教材作成や準備の負担の少ないであろうアンプラグドでの実践を増やしたい」「児童の ICT スキルが十分でない現状にあう実践をしたい」という思いがあったからである。

しかしこの学習展開だと，「正しく命令しないと動かない」というコンピュータの特性や，その特性を実感することで生まれる正しい手順や論理の必要性への気づきなしに，児童は学習を進めるこ

とになる。プログラミング体験を先に行い，そこで行った「どんな命令を使うかを考えること」や「命令をどんな順番に並べるか考えること」がプログラミング的思考だと児童に伝え，ステップチャートでも示してみるといった流れの方が有効だったかもしれない。二つのやり方を比較検証してみないことにはどちらが有効であると言い切ることはできないが，今後プログラミング体験を先にする学習計画も検証したい。

（２）情報教育という視点

何年生から学習するか，どのような形で学習を始めるかも考えなくてはならない。現時点での仮説は，１年生からプログラミング体験を行うことが望ましいのではないだろうかというものである。

理由の一つは先述したように，プログラミング体験を先に行うほうが有効かもしれないからである。プログラミング的思考が発揮される場面は，１年生や２年生の学習にもある。それらをアンプラグドで行うのであれば，その前にプログラミング体験をしておいた方がいいということになる。

もう一つの理由は，プログラミング教育にはICTスキルが必要だからである。例えば，Scratch を使うだけでも，マウスによるドラッグ，データの保存，キーボードでの入力，半角入力など必要なスキルがある。それらのスキルが十分でない児童がプログラミング体験を行った場合，1 時間の授業の中に「スキルの習得」と「教科の目標」の二つのねらいが混在してしまい，教員にも児童にも負担のかかる授業となってしまう。本研究における実践でも，コンピュータの操作に手間取ったり，トラブルの対処方法がわからなかったりして予想外に時間がかかってしまう場面は多々あった。そうならないためにも，１年時ではマウスを使ったドラッグ＆ドロップ操作，２年時では数字の半角入力，４年時ではローマ字入力というように，各学年で少しずつICTスキルを育むことが望ましい。

なお，タブレットでもScratchの操作は可能だが，画面の大きさに合わせてブロックや入力スペースが小さくなることや，中学校や高等学校でのプログラミング教育への接続を考えると，PC でのマウスとキーボード操作に慣れておいたほうがよいと考える。

さらに付け加えると，低学年からのプログラミング教育により，児童にとって ICT がより身近なものになるだろう。そのため，ICT の危険性に関する知識やマナーも同時に身につける必要がある。児童の発達段階に合わせて情報モラルの学習も系統的に進める必要がある。

つまり，プログラミング教育はそれ単体で行うのではなく，情報教育の一環であるという前提に基づき，ICT スキルや情報モラルと共に低学年から学習を進めることが大切だと言える。

情報教育の重要性は何も新しい学習指導要領から述べられているものではない。しかしながら，十分に行われてきていないのが現状だと感じる。プログラミング教育をきっかけに，もう一度，情報教育の視点でカリキュラムを構築することが必要ではないだろうか。

(32)事前アンケートを取った後に本研究における一般化の定義を変更している。実践は本研究の定義に従って行われているため，アンケートでの一般化の項目に影響を与えないとし，結果では割愛している。なお，抽象化については事前アンケートの後にその定義の範囲を広げている。実践はアンケート実施時点の定義も含んだものになるため，抽象化については掲載した。

おわりに

本研究を進めている間にも，ICT 技術は着々と前進した。AI 技術で業務改善，AI が○○をするサービスが登場と，我々がもうAI と共に歩み始めていることを実感せざるを得ないニュースがどんどん飛び込んでくる。

先日ある学校で，AI や 5G の技術がどういう社会をもたらすかについての動画を見て，技術の功罪について児童が考える授業を参観した。AI やAI を搭載したロボットが様々な仕事をしている様子を見た児童は，「便利」「すごい」と思った一方「AI が間違ったらどうするんだろう。」や「自分がなりたい仕事がなくなるかも。」と不安も感じていた。その不安を的中させるも，回避するも，これからを生きる彼ら次第である。そんな彼らを育てる教育活動の支えに，少しでもこの研究が役立てば幸いである。

最後に，日々の教育活動が大変忙しいにもかかわらず，本研究の趣旨を理解し協力してくださった京都市立砂川小学校と京都市立柊野小学校の校長先生をはじめ，研究協力員の先生方，温かく迎えてくださった両校の教職員の皆様，そしていつも全力で授業に参加してくれた子どもたちに心から感謝の意を表したい。