ビジュアル型言語からテキスト型言語への橋渡し教材の開発

中島　敏*

（2019 年 1 月 7 日受理）

1. はじめに

本校では、本年度より、小・中学校の生徒さんたちに群馬高専のポテンシャルを広く知ってもらい、生徒さんたちが将来を考える際の選択肢に群馬高専を入れてもらうようにするという「入試 PR の一環」としての位置付けで「出前授業」を展開することとなった。従来から行っている「出前セミナー」との大きな違いは、出前授業が、小・中学校からの依頼を受け、ゲスト講師として小・中学校の授業を行うものであるという点にある。このため、出前授業の各テーマにおいては、現行および改訂予定の小・中学校における学習指導要領などに基づいた授業内容を計画に盛り込むこととなっている。

特に、小学校では 2020 年度よりプログラミング教育が全面的に実施されることとなった。このような状況を受け、小学生高学年から中学校低学年を対象とし、「プログラミング的思考」を育むことを目的とした出前授業に用いる教材を開発した。

2. 背景

2-1. 新学習指導要領における情報処理教育について

文部科学省の新学習指導要領では、本年度を含めた移行期間を経て、小学校においては2020年度よりプログラミング教育が全面的に実施されることとなった[1)]。文部科学省の中学校学習指導要領解説「技術・家庭」[2)]の中で、「今回の改訂で小学校では、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よいのか、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ことのできる力であるプログラミング的思考等の育成を目指した学習活動を（中略）計画的に実施することが求められている」と述べている。

中学校においても、小学校から 1 年遅れの 2021 年度より新学習指導要領の全面実施に伴い、技術・家庭の科目の中で、「情報の技術」についてより踏み込んで学習することとなる。文部科学省の中学校学習指導要領解説「技術・家庭」[2)]の中では、「技術分野としては、小学校において育成された資質・能力を土台に、生活や社会の中からプログラムに関わる問題を見いだして課題を設定する力、プログラミング的思考等を発揮して解決策を構想する力、処理の流れを図などに表し試行等を通じて解決策を具体化する力などの育成や、順次、分岐、反復といったプログラムの構造を支える要素等の理解を目指すために、従前はソフトウェアを用いて学習することの多かった「ディジタル作品の設計と制作」に関する内容について、プログラミングを通して学ぶこととした」などと述べている。

このように新学習指導要領においては、小・中ともに、プログラミング教育の比重が高くなり、小学校ではその導入を行い、そして中学校以降ではプログラミング的な思考により課題解決に応用できる力へと育てていくことになる。

森田秀一氏のまとめ[3)]によると、小学校における「学習指導要領におけるプログラミング教育の目的は『情報活用能力の育成』。狙いは『コンピュータに意図した処理を行うように指示することができる、ということを体験させる』こと」であり、そのための教科は新設せず、従来の科目の中で対応することとなっている。

「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」は、「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）」[4)]において、「第３章 各教科等の目標・内容を踏まえた指導の考え方」の中で、「小学校段階のプログラミングに関する学習活動の分類（例）」として、以下の 6 項目を挙げている。

A 学習指導要領に例示されている単元等で実施するもの
B 学習指導要領に例示されてはいないが、学習指導要領に示される各教科等の内容を指導する中で実施するもの
C 各学校の裁量により実施するもの（A,B,D 以外で、教育課程内で実施するもの）
D クラブ活動など、特定の児童を対象として、教育課程内で実施するもの
E 学校を会場とするが、教育課程外のもの
F 学校外でのプログラミングの学習機会

* 物質工学科

Aの例としては、数学において、正多角形の図形を描くプログラミングであったり、理科において、コンデンサに蓄えた電力を夜間照明として使用する際のセンサからの応答に対しての制御をするプログラミングであったりする。Bに対しては、音楽の時間内に、リズムのパターンを予め用意しておいて、その組み合わせによりまとまりのある音楽をプログラムするという課題などが例示された。

ここで全てを挙げなおすことが目的ではないので部分の引用にとどめるが、小学校においてはどの教科の中で行うのかについての選択肢が広い。群馬高専で提供する小学校プログラミング教育についての出前授業は、上記分類のAからDのいずれかで活用してもらえるような受け皿を提供することに意義があると考えられる。

また、同「議論の取りまとめ」[4]において、この時期のプログラミング教育について、「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」などを育むことであり、コーディングを覚えることが目的ではない」と述べている。また、同とりまとめにおいて、プログラミング的思考という言葉を「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」と述べている。

「小学校プログラミング教育の手引（第一版）」[5]では、子供たちにコンピュータに意図した処理を行うよう指示することができるということを体験させることが「コンピュータに意図した処理を行うよう指示をする活動を通して、コンピュータはプログラムで動いていること、プログラムは人が作成していること、また、コンピュータには得意なこととなかなかできないこととがあることを、体験を通して気付かせること」であると言い換えている。

2-2. 構造化定理

コンピュータに意図した処理を行うように指示する際、現時点では、人間を相手にするような曖昧な表現では目的を達することが難しい。しかしながら、非常に複雑に見える問題であっても、その論理構造を紐解いていくと、最終的には単純な処理の組み合わせでできていると考えることができる。

いくら複雑に見えても、全ての処理は「順次、反復、分岐」の3種類の要素の組み合わせで表現できるという考え方を、構造化定理と呼ぶ。この3つの要素は、中学校学習指導要領解説「技術・家庭」[2]の中でプログラムの構造を支える要素として例示されているものである。この構造化を意識するとき、プログラミングの書き手は、ひとつずつの処理について入口と出口が一つずつになるようにする。

「順次」（順接、などと呼ばれることもある）では、記述された順に処理が行われることとなる。もっとも単純な構造でもある。この処理のみで構成された手順では、物語を初めから順に読むように処理が進められるので、あらかじめ決められた処理を決められた通りに行うことになる。

「反復」は、同じ種類の処理を何度も並べる代わりに、その並べられた処理が同じ種類であるということに気づき、一般化、抽象化する作業と言える。たとえば、矩形の部屋の中で、南東の頂点の位置から壁に沿って動くことで対角線上の頂点の位置に移動することを考える。部屋の大きさが分かっていれば、たとえば北へ進む処理を5回記述し、西へ進む処理を8回記述する。この計13行の処理の記述に代わり、北へ進む処理を5回繰り返し（北へ5歩進み）、次いで西へ進む処理を8回繰り返す（西へ8歩進む）と記述するとき、はじめの記述よりも抽象化、一般化が進んでおり、より見通しがよくなっている。

更に抽象化、一般化を進めるためには、「壁にぶつかるまで北に進み、ついで向きを変え、更に壁にぶつかるまで西に進む」としてやれば、矩形の部屋の大きさに依存しない。「壁にぶつかったかどうか」を判断して行動を変えるためには、次の「分岐」処理が必要となる。

条件により異なる処理を行わせる「分岐」（選択、などと呼ばれることもある）は、センサなど外部からの応答に応じて行動を変えるような処理の記述には不可欠である。あらかじめ地図が分かっていれば、その地図上の動きを事前に決定してプログラミングすることもできるが、この条件分岐を用いることで、より抽象化し、地図が変化した場合でも対応できるような一般化したアルゴリズムの記述をすることが可能となる。その例は、後述するが、迷路の脱出のアルゴリズムの一つである「右手法」について、教材の中で取り上げる。

2-3. アルゴリズムの図的表現

一般に、アルゴリズムの構造を図的に表現することが可能である。事実上の標準としてフローチャートが使用されているが、これに代わる構造化チャートとして開発されてきたものが多数ある。

PAD（Problem Analysis Diagram）[6,7]は、日立製作所で発明され、1979年に公表された。要素の組み合わせでプログラム構造を表現するための方法である。この方法を用いると、横軸方向に構造化のレベルの深さを表し、縦軸方向に処理の大きさを表すことになる。ここで、構造化のレベルの深さとは、要素となる一つずつの処理

（ここでは、モジュール、セグメント、サブルーチンなどと呼ばれるようなもの）が、更にどのよう小さな処理の集合でできているのかを分解していく作業の一段に相当するものである。

また、NS チャート(Nassi-Shneiderman diagram)[8]は、1 枚のチャートの中で構造化のレベルを直接的に表すようには設計されていないが、要素となる一つずつの処理を矩形の「ブロック」で表し、隙間なく平面内を埋め尽くすことで一つの大きな処理を記述する。すなわち、縦軸方向には処理の時間軸があり、横軸方向には条件分岐で生じたパラレルワールドが、その数だけ並ぶイメージの二次元図を与える。そのため、直感的に理解しやすい。また、処理の間をつなぐ矢印線は全く使用せず、隙間なく処理を並べていく点で、後述するビジュアル型プログラミング言語である Scratch や Google Blockly との間に共通点がみられる。

2-4. ビジュアル型プログラミング言語の例

Scratch[9] は、マサチューセッツ工科大学（MIT）で開発された環境であり、プロトタイプは 2002 年に発表され、2006 年には一般公開された。ソースコードは GPLv2 ライセンスと Scratch Source Code License の下で公開されている[10]。ウェブブラウザ上でも利用できるほか、シングルボードコンピュータである Raspberry Pi の OS である Raspbian に「Scratch Pi」として搭載されているため、電子工作に応用したり、Raspberry Pi 版の Minecraft「Minecraft Pi」と連動させることも可能である[11]。

Google Blockly[12] は、オープンソースソフトウェアとして 2012 年に公開された JavaScript のライブラリであり、ウェブブラウザで動作する。ユーザインターフェイスは Scratch によく似ているが、Blockly 上で作成した実行プログラムについて、JavaScript、Python、PHP また、Dart などのコードを生成することが可能である[13]。

Scratch、Google Blockly ともに、ほぼ同じ GUI を備え、「ブロック」として用意された命令を並べてプログラミングするよう設計されている。これは設計思想として、コーディングにおける文法などを意識せず、小さな子供でも、コンピュータ上で図形などを意図通りに動かしたり、試行錯誤するなかでプログラミング的思考を会得させようとするためであると思われる。

これは、ひとつずつの「ブロック」として用意された処理や命令については、背景でどのような処理が行われているのかを、利用者が全く意識する必要がないということを意味し、より直感的でわかりやすい環境を与えることができる。

また、ブロックをマウスによる操作で Blockly の用語でいうところの「ワークスペース」内に配列するだけでプログラミングができるので、文法エラーを生じる余地が無い点も初心者に優しいと言える。

これらのビジュアル型プログラミング言語において定義済みの「ブロック」を並べていく作業は、分岐後の処理を if 文における then 区画と else 区画を順に並べるのと同じように縦に並べる点を除けば、上述した NS チャートを作成する作業と類似している。また、定義済みのブロックを一つの処理として、別の処理を記述したり、更にはそのように記述した一連の処理を一つの処理として別の処理の記述に使用するので、必然的に構造化を意識したプログラムができあがるように思われる。

また、これらはいずれもウェブアプリとして動作するので、汎用性が高いこともあり、プログラミング教育の導入場面として使われることが多い。Code.org[14]が世界的に主唱するプログラミング教育活動である「Hour Of Code」において、Scratch や Blockly を利用したチュートリアルプログラムが多数用意されており[15]、だれでも無料で利用することができる[16]。

図 1 に、ビジュアル型プログラミング言語（Blockly）で記述した（画面上に表示されたロボットの）操作のプログラム例を示す。

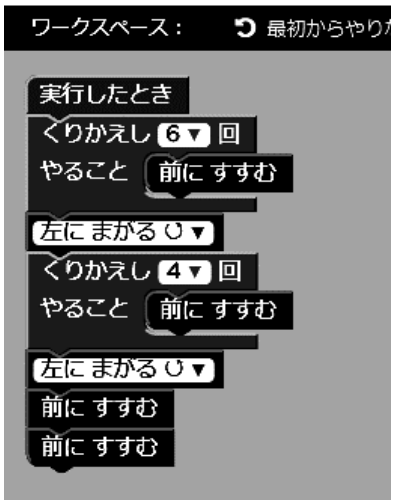

図 1 ビジュアル型言語によるプログラムの例

2-5. テキスト型プログラミング言語への橋渡しの意義

Scratch や Google Blockly のようなビジュアル型プログラミング言語は、初心者にとってプログラミング的思考を学修するために適切な言語であるといえる。

しかし、そもそも「小学校プログラミング教育の手引（第一版）」[5] の中で、「子供たちがコンピュータを用いて情報を活用したり発信したりする機会が一層増えてきている一方で、その仕組みがいわゆる「ブラックボックス化」してい」ることを背景として挙げ、「コンピュ

ータに意図した処理を行うよう指示をする活動を通して、コンピュータはプログラムで動いていること、プログラムは人が作成していること、また、コンピュータには得意なこととなかなかできないこととがあることを、体験を通して気付かせること」について学ばせることを提言しているのに、導入教育の場面では仕方ないとはいえ、解決策として「ブラックボックス化」された処理ブロックを並べるようなビジュアル型プログラミング言語に頼るというメタ的な構図が出来上がっている。

またこれらのビジュアル型プログラミング言語は、仕様上、機能に制限もあり、万能ではない。そのため、目的に応じて、いずれかの段階で、従来からあるテキスト型（ソースコードが文字列であるようなものを指す）のプログラミング言語に移行することが求められる。

同手引き[5]の中でも、「プログラミング言語や教材選定の観点」として、プログラミング言語を学ぶことが目的ではないとしながら、「文字により記述する言語（テキスト型プログラミング言語）にも様々なものがあります。キーボード操作が多く、それぞれの言語の文法の理解も必要となりますが、英数字だけでなく日本語で記述できるものや、文法的な誤りがあった場合には間違いを指摘してくれるものなど、児童でも比較的取り組みやすい言語もあります。ある程度の授業時数を確保して取り組む場合や、プログラミングに強い興味・関心を示す児童については、こうした言語を活用することも考えられます」と述べており、テキスト型言語が、発展的な学習としての位置づけとなり得ることを示している。

2-6. テキスト型プログラミング言語の選択

テキスト型プログラミング言語としては、構造化に対応した記述が可能なものとして、確立されたものが多く存在し、選択の幅は広い。ここでは、「十進 BASIC」[17]を選択した。

これは、十進 BASIC が、Windows、Mac、Linux、Raspberry Pi など、プラットフォームを選ばずに無料で使用できるソフトウェアであること、国際規格の JIS Full BASIC に準拠しており、行番号や無条件分岐（GOTO 文）は原則として廃止されており、また、変数にスコープが導入されているなど、構造化プログラミングに対応していること、また、コンピュータを計算の道具として用いることを目的として設計されており、数値計算などを行うためにも十分な機能と精度を備えていること、などを理由とする。ただし、十進 BASIC は、使用規定の中に「十進 BASIC は教育研究を目的として作成されたプログラムです。本 BASIC を利用して得られた研究結果は必ず公開してください。」の文があるため、商用の非公開のプログラムの設計には向いていない。

また、BASIC（Beginner's All-purpose Symbolic Instruction Code）という言語は、初学者用に設計された高水準語であり、C 言語などの低水準語にくらべて導入がやさしいと思われる。なお、旧課程（2012 年度まで）の数学 B「数値計算とコンピュータ」で、BASIC を言語として利用しており、センター試験でも BASIC の文法に則った問題が出題されていたことなどから、学校教員においても慣れている方が多い可能性がある。

BASIC は英語ベースの命令文から成り立っているものの、英語そのものについては、小学校高学年において、2011 年に「外国語活動」として必修化されている[18]ほか、2020 年度からは学習時間の増加が諮られるなどの背景をうけ、一部の命令文に用いられているような内容の限られた英単語であれば、導入に際し大きな困難を伴わないと想像されることから、発展的な学習の範囲であれば差し支えないと判断した。

3. 教材の内容と学習の狙い

今回の出前授業用の教材として、対象を小学校の高学年および中学 1 年生を対象とし、次のように計画した。ただし、次の第 4 段階以降は、発展的な学習としての位置づけとした。

1) ビジュアル型プログラミング言語を用いて、簡単なプログラミングを体験する。ここで処理のブロックを順序に従って並べることを学ぶ。
2) 目的を達成するためのプログラムを作成したり、そのプログラムを実行して、結果に応じてプログラムを直すなど、試行錯誤することを体験する。目的通りに動かない場合に工夫することを学ぶ。
3) プログラムの抽象化、一般化について学び、簡単な繰り返しなどを記述できるようにする。
4) ビジュアル型プログラミング言語で行っていた処理を、テキスト型のプログラミング言語で記述し、ビジュアル型プログラミング言語の「ブロック」を、テキスト型プログラミング言語のサブルーチンに対応づけするとともに、どのような処理を行っているのかが記述されていることを示す。併せて、変数やパラメータといった抽象概念をパソコン画面上のオブジェクトの制御に結びつける。
5) あらかじめ地図がわからないような場合にも対処できるよう、条件分岐を含む、やや複雑だがより一般化されたアルゴリズムとして、右手法を例に挙げ、これを記述するための考え方の過程を示す。

ここで、第 3 段階までについては、Hour of Code に用意されたチュートリアル[15]より、「画面上に表示されたロボット」（以下、単純にロボット）を目的地に移動させるという教材を選んだ。命令の「ブロック」として

予め用意されているのは、「前に進む」「右に曲がる」「左に曲がる」「～を指定回数繰り返す」などである。

第4段階として、十進BASICを用いて、図1のビジュアル型言語によるプログラムおよび動作を再現したものを教材に印刷して示した。その具体的なプログラムソースと解説は、プログラム例1として次章に示す。

このプログラム例1では、メインルーチン内には、ロボットを動かすためのサブルーチンを置くように設計し、ビジュアル型言語での記述に倣い、このためのスペースを"Work Space"として確保した。またこの"Work Space"に引き続き、STOPを置き、メインルーチンの終了を明示的に示したが、制御的に実質の意味合いはない。

ビジュアル型言語ではマウス操作によるドラッグでブロックを置く作業に代わり、テキスト型のプログラミング言語では、サブルーチンを呼び出す命令を書く(例: Call MoveForward)という対応を含めて、視覚的にもほぼ完全に相似であると言える。ビジュアル型言語において、ブロックの構造として繰り返し区画を挟み込んで表示する(図1参照)のに対し、テキスト型のスクリプトではDo文とLoop文、または、For文とNext文によりこの区画を挟み込むことになる。また、この相似性はNS図との間にも保たれている。

テキスト型のプログラミング言語では、綴り上の間違いなどによる文法エラーが生じやすい。しかしながら、予め準備されたサブルーチン名をマニュアル等に列挙しておけば、あとはそのサブルーチン名をコピー、ペーストして並べるだけで、ビジュアル型プログラムと同等の操作性を担保できると考えられる。

また、上述したように、ビジュアル型のプログラミング言語では、それぞれのブロックの命令が、裏で何をしていることによって画面上のオブジェクトが動いたりするのかが見えにくい。言い換えれば、テキスト型のプログラミング言語に比べて、ビジュアル型のプログラム言語の方が、高水準化の度合いが高いともいえる。

とはいえ、初学者にとっては煩雑で、はじめ包括的な理解には不要であるように見えるこの部分の理解が、プログラミングをより一般化していく過程では、必要になると考えられる。

テキスト型のプログラミング言語では、そのサブルーチンの記述が同じ階層内で見ることができ、更には必要に応じて自分で書き換えたりしながら操作できるという点がある。

たとえば、ロボットの移動で考えると、プログラム例1内では、ロボット(オブジェクト)に、位置情報(座標x, y)および向き(x軸方向を基準として反時計周りの角度 t、0度から90度刻みで270度まで)というプロパティが付与されており、これに基づいて画面上に表示をしている。そのため、この位置情報を書き換えることがロボットの移動に相当している。プログラム例1では、表示上の工夫として、位置の変更の前後で、連続的に位置をずらしながら表示を繰り返す移動のアニメーションも実装し、位置の移動をイメージしやすくした。

向きの情報 t と x, y 方向の移動量 Δx, Δy の関係は、三角関数を用いて、Δx = cos(t), Δy = sin(t) で表すことができるが、高校または高専で学習する内容として紹介し、以下のような表で結果を示すにとどめた。

表1　ロボットの向きと角度 t、三角関数の値の対応

向き	t	Δx = cos(t)	Δy = sin(t)
東	0	1	0
北	90	0	1
西	180	-1	0
南	270	0	-1

今回教材として用いるHour of Codeに用意されたチュートリアル[15]では、ロボットが動ける範囲は、縦横10×10マスのフィールド内である。そのフィールド内に、川や橋、ゲート、ロボットが踏むとゲートが開くスイッチなどが表示されている(「スイッチをロボットが踏む」=「ロボットの座標がスイッチの座標と一致する」状態であるときに「ゲートが開く」という因果関係は、プログラミングの中では、条件分岐で記述する)。

実在するロボットを制御する場合でも、適切なセンサ等を用いて、壁の存在や床面に描かれた線などを検知し、これら周囲の状況によって行動パターンを変えるようなプログラムを書く場合が多い。

このように「周囲の条件によって行動を変える」という考え方自体が、構造化定理の示す3つの要素の中のひとつ、分岐の処理に相当し、プログラミング的思考について学ぶ際に、避けては通れないものである。

単純化のために、プログラム例1では周囲の状況の情報を得るような部分は実装していないが、このプログラム例1をもとに、更に地図情報を付加したり、周囲の状況を調べたり、行動を分岐させるなど、種々の機能を追加した十進BASICによるプログラムを準備した。そのプログラム「RoboSteering.bas」[19]の一部のみを、プログラム例2(部分)として示す。ソースコードが長くなるため、プログラム自体の解説は授業の範疇外とし、行わないものとしたが、興味をもった生徒がゲームとして遊ぶことができるような工夫を盛り込み、教員ウェブサイトに公開した。ソースコードは、たとえば、関係個所のみを引用したように、プログラム例1の中では12行で記述したサブルーチン「MoveForward」だけでも、約4倍の長さとなった。これは、ロボットを置くフィールドの地図情報によって動作を変えたり、移動時にロボッ

トの位置の再表示のみに終わらず、フィールドのロボットの移動した後の位置に、地図情報に応じた再表示させたりする必要が生じているためである。

今回用いる Hour of Code に用意されたチュートリアル[15]に状況は近いが、例えば、地図中に目的地が存在するとき、そこに至る経路を考えるものとする。最初の段階で地図に沿った動きを計画し、これをプログラミングすることができるようになったとする。さらに、問題を一般化、抽象化すると、あらかじめ地図がわからない場合であっても目的地に達することができるような処理とすることもできる。

第 5 段階では、はじめに、壁があったときの回避として、単純に「前方に進めるなら前へ一歩進め。前方に進めないなら右を向け」という命令を、分岐を用いて構築するところから始める。これは、If 文により、then 区間と else 区間に異なる動作に対応する処理を書けばよい。ついで、「右手を壁につけて、それを離さないようにして進め」という迷路解法の有名なアルゴリズムである「右手法」を紹介し、これをプログラミングするための過程の説明を行う。この際、「右手を壁につけて離さない」という、人間にとってはわかりやすい（と思われる）指示を、コンピュータにとって理解できる「具体的、かつ、単純な」命令に置き直す必要がある。そこで、指針として、図を描いてこれを参照しながら、現在の位置、向きと、周囲の壁の有無に応じて、場合分けして考えてみる必要があると説明する。また、この際、90 度回転させて重なる図形は、相対的に同じであるものとして考えると場合の数が少なくなることを示す。

結論として、右手法は「右側に壁がないなら、右を向いて前へ一歩進め。右側に壁があれば、周囲を調べ、前、左、後ろの優先順で進める方向に一歩進め。」と等価であることを説明する。

このように組んだアルゴリズムは、RoboSteering.bas の ″WorkSpace″ 内に記述して、組み込まれた迷路において動作を確認することができる。

4. 十進 BASIC のプログラムソースと解説

教材配布資料として配布し、説明に用いるプログラムソースを以下「プログラム例 1」に示す。プログラム例 1 にさまざまな機能を追加して、それぞれが遊びながら学習に使用できるように公開する別のプログラムも用意した。ソースが長くなるので、資料としての配布は行わないが、この「RoboSteering.bas」は、筆者の教員ウェブサイトに公開した。このソースの一部のみ抜粋して「プログラム例 2」に示す。

これらのプログラムでは、メインルーチンとして、″Work Space″ 内に、ロボットの移動などに関するサブルーチンの呼び出し命令を並べて実行させることができる。また、必要なサブルーチンは、すべてプログラム後半に列挙した。また、″Work Space″ には図 1 に示すビジュアル型プログラミング言語によるプログラム例と同じ処理を記述した。

以下、ソースコード中に、※1 ～ ※5 を付記した点について述べる。これらは、プログラム実行時の動作が綺麗に見えたりするための工夫であり、プログラミングの本質部分とは無関係である。

※1 ： 十進 BASIC では、画像にレイヤーは準備されていない。そのため、アニメーションにおいてはレイヤーの移動や消去ではなく、画像位置への背景色での上書きを用いる。簡易的に、ロボットは色付きの文字で丸を表示しているため、サブルーチン ClrRobo では、白色で同じ文字を上書きする。この際、同じサイズのままだとわずかに輪郭が残ってしまうことがあり、画面が汚くなるので、少し大きめの指定を行っている。

※2、3 ： ロボットの移動のアニメーションとして、サブルーチン ClrRobo の呼び出しによる画面上のロボットの消去と、サブルーチン SetRobo の呼び出しによる新しい座標上での表示を行う。これを滑らかに連続して動いているように見せるため、20 分の 1 マスずつ移動させている。

※4 ： Draw Mode について、Hidden は、グラフィックスの特殊処理であり、内部にあるビットマップメモリにのみ描画するモードに移行する命令である。これにより描画に伴うばたつきを抑えることができる。このビットマップメモリの情報を画像として反映させるためには、Draw Mode を Explicit に戻す。

※5 ： ロボットの向きを表す角度 t は、0 ～ 360 以外の任意の値をとっても問題ない。たとえば、表 1 のような計算過程において、t = 90 と t = 450 は同じ結果を与える。また実質的には、このプログラムの実施に当たって t の値が ±1e16 に達して数値演算の桁あふれが起きることはないだろう。そのため、Mod をとる必要性は低い。ここでは、論理的な美しさと表 1 との整合性を保つために、t の範囲を [0, 360) に限ることとした。

また、プログラム例 2 の、行末の「&」と、続く行頭の「&」は、本来，一行に書くべき文を複数行に分けて書くための行継続の記号である。

```
! 十進 BASIC プログラム例 1
CALL Init
pause       ! 一時停止して OK を待つ
! When Run =====  ″Work Space″ ここから  =====
! ″Work Space″ 内は、自由に書き換えることができます。
! マニュアルを参考にして、この ″Work Space″ 内に
! ロボットを操作する命令を並べてから「実行」して下さい。
! 以下、図 1 に示したロボット操作と同じ操作の例
```

```
FOR k = 1 TO 6
   CALL MoveForward
NEXT k

CALL TurnLeft

FOR k = 1 TO 4
   CALL MoveForward
NEXT k

CALL TurnLeft

CALL MoveForward

CALL MoveForward


! ======= "Work Space" この上まで ============
STOP
! プログラム本体ここまで
!
! ============ マニュアル ====================
! ● サブルーチンを Call で呼び出せます。
!   コピーペーストする場合は、その行をペースト後、
!   先頭の ! を外してください。
!
! 例： Call MoveForward
!
! MoveForward   ! ロボットを１歩前へ進めます。
! TurnRight     ! ロボットの向きを 90 度右へ回転します
! TurnLeft      ! ロボットの向きを 90 度左へ回転します
!
! ● 同じ処理を繰り返すために、計数ループを使用できます。
!     ※ 単純ループ Do ～ Loop も使用できます。
!
! 例：
! For k = 1 To 3        ! この例では 3 回繰り返します。
!    Call MoveForward
! Next k                ! ← Next の後ろには、計数の
!               ための変数名（ここでは k）を書きます。
!
! ==========================================
! 以下、サブルーチンの記述

SUB Init      ! 初期画面の表示
   SET WINDOW 0.5, 10.5, 0.5, 10.5
   DRAW grid
   SET TEXT HEIGHT 0.8
   SET TEXT JUSTIFY "center", "half"
   OPTION ANGLE DEGREES
   ! ロボットの初期位置と向きの情報
   LET x = 7      ! ロボットの座標 x
   LET y = 1      ! ロボットの座標 y
   LET t = 90     ! 向き、x 軸方向から反時計周り
   CALL SetRobo
END SUB

SUB SetRobo  ! ロボットを表示
   SET TEXT COLOR 2          ! 青色
   PLOT TEXT ,AT x,y : "●"
   CALL ShowDirection
END SUB

SUB ShowDirection  ! ロボットの向きを矢印で表示
   SET TEXT COLOR 6   ! 黄色
   SELECT CASE t
   CASE 0
      LET Direction$ = "→"
   CASE 90
      LET Direction$ = "↑"
   CASE 180
      LET Direction$ = "←"
   CASE 270
      LET Direction$ = "↓"
   CASE ELSE
      LET Direction$ = ""
   END SELECT
   PLOT TEXT ,AT x,y : Direction$
END SUB

SUB ClrRobo  ! ロボットを非表示
   SET TEXT COLOR 0          ! 白色
   SET TEXT HEIGHT 1.0       ! ※1
   PLOT TEXT ,AT x,y : "●"
   SET TEXT HEIGHT 0.8       ! 元に戻す
   DRAW grid
END SUB

SUB MoveForward  ! ロボット移動のアニメーション
   FOR FootSteps = 1 TO 20    ! ※2
      SET DRAW MODE HIDDEN    ! ※4
      CALL ClrRobo            ! ※3
      LET x = x + COS(t)/20
      LET y = y + SIN(t)/20
      CALL SetRobo            ! ※3
      SET DRAW MODE EXPLICIT  ! ※4
      WAIT DELAY 0.05
   NEXT FootSteps
   WAIT DELAY 0.3
END SUB

SUB TurnRight  ! ロボット進行方向右へ 90 度回転
   LET t = MOD(t-90, 360)     ! ※5
   CALL SetRobo
END SUB

SUB TurnLeft   ! ロボット進行方向左へ 90 度回転
   LET t = MOD(t+90, 360)
   CALL SetRobo
END SUB

END

! 十進 BASIC によるプログラム例 2（部分）
SUB MoveForward  ! ロボット移動のアニメーション
   CALL SearchFront
   IF Destination$ = "empty" THEN
      DO
         LET LPx = x
         LET LPy = y
         FOR FootSteps = 1 TO  20
            SET DRAW MODE HIDDEN
```

```
            Call ClrRobo
            CALL LastPoint
            LET x = x + COS(t)/20
            LET y = y + SIN(t)/20
            CALL SetRobo
            SET DRAW MODE EXPLICIT
            WAIT DELAY 0.05
         NEXT FootSteps
         CALL LastPoint
         CALL SearchFront
      LOOP WHILE map(x,y) = 8 AND ( SerchResult = 0 &
&             OR SerchResult = 6 OR SerchResult = 7 &
&             OR SerchResult = 8)
   ELSE
      WAIT DELAY 1
   END IF
   WAIT DELAY 0.3
END SUB

SUB LastPoint   ! 移動後にもとのマップ色を置く。
   IF map(LPx, LPy) = 6 THEN
      SET TEXT COLOR 6
      PLOT TEXT ,AT LPx, LPy : "★"
   END IF
   IF map(LPx, LPy) = 7 THEN
      SET TEXT COLOR 7
      PLOT TEXT ,AT LPx, LPy : "■"
   END IF
   IF map(LPx, LPy) = 8 THEN
      SET TEXT COLOR 8
      PLOT TEXT ,AT LPx, LPy : "■"
   END IF
END SUB

SUB SearchFront     ! 前方のマップデータを読み取る。
   LET SerchResult = map(x + COS(t), y + SIN(t))
   IF SerchResult < 1 OR SerchResult > 5 THEN
      LET Destination$ = "empty"
   ELSE
      LET Destination$ = "occupied"
   END IF
END SUB
```

5. まとめ

小学校に導入されるプログラミング教育では、課題を解決するための論理的な思考の組み立てにより解決策を構想する力、処理の流れを図などに表し試行等を通じて解決策を具体化する力などの育成や、順次、分岐、反復といったプログラムの構造を支える要素の理解を目指している。とはいえ、そのため、具体的なコーディングの技術を求めるのではなく、アルゴリズムを構造的に捉えることが求められている。

そのような導入学習に最適なのは、広く普及している Scratch や Google Blockly に代表されるようなビジュアル型のプログラミング言語である。しかしながら、学習者全員に必要であるわけではないが、関心の高い生徒たちが、より複雑な処理を行うプログラムを作成しようとしたとき、いずれかの段階でテキスト型のプログラミング言語に移行することが必要となる。一部のビジュアル型の言語では、自分で組み上げたソースを、文字列によるコードとして吐き出す機能をもつものもある。とはいえ、それぞれのブロックに相当するサブルーチンの中で、どのような内部処理が行われているのかまでを理解させるようなところまでは、対応していないように見受けられる。

そこで今回、テキスト型のプログラミング言語として十進 BASIC を選択し、ビジュアル型のプログラミングの形態を模写する形で、ビジュアル型の言語からテキスト型の言語への橋渡しとなる教材の開発を試みた。同時に生徒の興味を引くために、出前授業などにおいて配布資料として用いることのできる短いコードの他に、ゲーム性を盛り込んだコード「RoboSteering.bas」も開発し、筆者の教員ウェブサイト上で公開した。

6. 参照

1) 文部科学省、初等中等教育局教育課程課「改訂のスケジュール」平成 29 年 4 月登録、http://www.mext.go.jp/a_menu/shotou/new-cs/1384662.htm よりリンク（最終閲覧日 2018.08.06）.
2) 文部科学省、初等中等教育局教育課程課「中学校学習指導要領解説」平成 29 年 6 月登録、http://www.mext.go.jp/a_menu/shotou/new-cs/1387016.htm よりリンク（最終閲覧日 2018.08.06）.
3) 森田 秀一、「小学校でのプログラミング必修化、どの教科でどう教えるかは学校・教員の裁量、細部はこれから議論」2017 年 4 月 26 日 https://internet.watch.impress.co.jp/docs/news/1056922.html, （最終閲覧日 2018.08.06）.
4) 文部科学省、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)」平成 28 年 6 月 16 日 http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm, （最終閲覧日 2018.08.06）.
5) 文部科学省、生涯学習政策局情報教育課「小学校プログラミング教育の手引（第一版)」平成 30 年 3 月登録、http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1403162.htm よりリンク（最終閲覧日 2018.08.06）.
6) 二村良彦, 「PAD の開発」日立評論 vol.68 No.5 1986, pp 7-11.

7) 日立製作所、COBOL85　言語文法書より「2.3　プログラム構造表記法（PAD)」、http://itdoc.hitachi.co.jp/manuals/3020/3020378270/LANG0027.HTM,　(最終閲覧日 2018.08.06)．
8) Wikipedia（英語)、「Nassi-Shneiderman diagram」（最終更新 2018.04.18)、https://en.wikipedia.org/wiki/Nassi-Shneiderman_diagram,　(最終閲覧日 2018.08.06)．
9) M. Resnick et al., "Scratch: Programming for All", *Communications of the ACM*, vol. 52, no. 11, pp. 60-67 (Nov. 2009).
10) Wikipedia（英語)、「Scratch (programming language)」(最終更新日 2018.07.29)、https://en.wikipedia.org/wiki/Scratch_(programming_language),　(最終閲覧日 2018.08.06)．
11) 薬師寺国安、「プログラミングを学習する意義、Scratch の基本的な使い方超入門（1/3)」 2016 年 3 月 21 日公開、http://www.atmarkit.co.jp/ait/articles/1603/21/news012.html,　(最終閲覧日 2018.08.06)．
12) 「Google for Education > Blockly」、https://developers.google.com/blockly/,　(最終閲覧日 2018.08.06)．
13) Wikipedia（英語)、「Blockly」(最終更新 2018.01.27)、https://en.wikipedia.org/wiki/Blockly,（最終閲覧日 2018.08.06)．
14) Code.org 公式ウェブサイト, https://code.org/,（最終閲覧日 2018.05.18)．
15) Code.org,　「Hour of Code アクティビティ」, https://hourofcode.com/jp/learn,　(最終閲覧日 2018.05.18)．
16) Ascii 倶楽部,　「Hour of Code で学ぶプログラミング基礎の基礎 ― 第 1 回」,（最終更新 2016.08.13)、http://ascii.jp/limit/group/ida/elem/000/001/194/1194772/,　(最終閲覧日 2018.05.18)．
17) "十進 BASIC のホームページ", http://hp.vector.co.jp/authors/VA008683/,（最終閲覧日 2018.03.14).
18) 文部科学省、初等中等教育局国際教育課、「小学校外国語活動について」（登録 平成 21 年以前) http://www.mext.go.jp/a_menu/shotou/gaikokugo/index.htm,　(最終閲覧日 2018.08.11)．
19) プログラミング導入学習　ロボット操縦ゲーム RoboSteering.txt 2018.05.05 登録, http://www.gunma-ct.ac.jp/staff/nakajima/Lecture/Joho/Programs/RoboSteering.txt,（最終閲覧日 2018.12.25)

Development of Teaching Materials for Mediating from Visual Programming Language to Text-type Programming Language

Satoshi NAKAJIMA

For the introduction to learn algorithm structurally, a visual programming language represented by Scratch or Google Blockly spreading over widely seems to be suitable.　It may not be all the learners, but a certain student would need the knowledge of text-type programming language, when an enthusiastic student is going to make a more complex program.　Therefore, I tried the mediation from a visual language to a text-type language by copying the form of the blocks in a visual language working as a subroutine, with a text-type program language.　Other than the short source code that I could use as a distribution document in a catered lecture, I made long program "RoboSteering.bas" which included game characteristics to attract the interest of the student at the same time.