【論文】

プログラミング教育の意義に関する研究

松　　秀樹・難波　宏司

1　はじめに

学習指導要領の改定で、2020 年より小学校でのプログラミング教育が始まることとなった。その関連から、小学生に対するプログラミングの教育に関して、教材の開発やプログラミング教室の開設が盛んになってきた。また、中学校・高等学校でもプログラミング教育のウェートは高まりつつある。更には、社会的にもプログラマの需要は高まっており、社会人を対象としたプログラミング教室も活況である。しかしながら、我が国ではプログラミング教育の教育的意義についての議論は、やや薄いように思える。プログラミング教育の意義の根拠としてよく出されるのが Computational thinking（計算機的思考）である。この概念は最初、教育用言語 LOGO の開発者のシーモア・パパートによって提唱された[1)]が、2006 年、コンピュータサイエンス普及の立場から Wing が新たに提唱した[2)]。彼女の考えを基にカーネギーメロン大学[3)]やロンドン大学[4)]にはその考え方を紹介するサイトがあり教材も提示されている。2016 年 3 月、日本学術会議は「大学教育の分野別質保証のための教育課程編成上の参照基準　情報学分野」[5)]を報告したがその中においても「計算機的思考」の影響を受けた内容となっている。本論では、2006 年の Wing の計算機的思考の考えを基に、小中高等学校向けのプログラミング教育モデルを構築し、試行的に実践したのでそれを報告する。

2　プログラミング技術とプログラミング教育の変遷

1　バベッジの解析機関

元来、コンピュータは計算する道具として発展してきた。計算する道具としてのコンピュータの原型となるのが、19 世紀半ば、イギリスのバベッジが考案した各種の計算を行う「解析機関」である[6)]。この機械は、設計の段階まで進んだが、加工技術の問題から完成には至らなかった。

バベッジは、各種の計算は基本的に、四則演算（加算だけでも可であるが、効率化を考え四則演算とした）と値の比較だけでできると発想し、四則演算を演算し、値を比較する演算装置と計算結果を記憶する記憶装置、演算の指示を与えるプログラムや演算データをパンチカードで機械に伝える入力装置、最終計算結果を印字する出力装置からなる機械を考案した。計算をさせるには、プログラムは、計算式を単純な四則演算に置き換え、計算途中のどの値とどの値を計算する

かを並列処理で指示を出す必要があったので、数学力の優れたバベッジの協力者のエイダが担当し、世界初のプログラマになったと伝えられている。エイダは、プログラミングの効率を図るために、「繰り返し手法」や「副プログラム」を考案している[6]。また、複雑な計算を分解し単純な四則演算に置き換えるために、「集合論」の手法を活用している。「集合論」に基づいて計算機を電子回路的に実現したのがコンピュータである。

2　ノイマン型コンピュータ

現在のコンピュータの基本構成はノイマン型の構成が主流である[7]。ノイマン型は、逐次制御とプログラム内蔵（プログラムを記憶装置に置く）型である特徴を持つ。逐次制御とは、処理を1つずつ順番に行うことであり、プログラム内蔵とは、データとプログラムを同じ記憶装置に格納して実行することである。この方式では、プログラム実行中に自らプログラムを変えてしまうことが可能でプログラム実行の自由度が高まった。ノイマン型が定着したことにより、逐次制御に基づくプログラミング方法の定式化（機器の違いにとらわれないプログラミング手法：アルゴリズムの出現）がある程度行われるようになった。

3　Fortran と Cobol

最初のころのプログラミングは、コンピュータが直接理解できる機械語のプログラムであったが、1957年、科学技術用高級言語である Fortran が出現し[8]、1959年には事務処理用言語の Cobol が出現した[9]。Fortran による数値計算アルゴリズムや Cobol によるソート、ファイル処理などのデータ処理がプログラミングの重要な内容となった。プログラミング教育においては、数値計算のアルゴリズムを理解することは重要な要素となった。その主な内容を以下に示す*[10]。

1. 誤差論　2. 数値補間（ラグランジュ補間・スプライン）　3. 写像　4. 線形計算 （ガウス・ジョルダン法）　5. 方程式解法（ニュートン法・逐次近似）　6. 微積分（台形公式） 7. 微分方程式（ルンゲクッタ法）8. モンテカルロ法　9. シミュレーション（FFT、グラフ理論）

1960年代にコンピュータは社会に普及するとともに、1970年ころより、プログラミング教育が普及しはじめた。1970年には、大学で情報系学科が設置され、同年の学習指導要領改訂では、応用数学（主に職業学科で履修）に計算機と数値計算の項目が導入され、職業学科に情報関連の科目が出現した。また、NHK のコンピュータ講座は1967年現代科学講座に一部としてはじまり、1969年より「コンピュータ講座」として独立した[11]。その構成（目次）を下に示す[12]。

Fortran 講座　講師　森口　繁一	Cobol 講座　講師　大駒　誠一　他
1　四則演算のプログラム	1　コンピュータ入門
2　判定と飛び越しを含むプログラム	2　売上業務
3　配列と繰り返しを含むプログラム	3　電気料金の計算
4　ファイル処理のプログラム	4　仕入業務にみるプログラムの例
5　実数計算のプログラム	5　まとめ
6　シミュレーションのプログラム	

4　構造化プログラミングとデータ構造

1970 年頃、プログラムが普及するに大規模プログラムの需要が高まり、メンテナンスや開発効率の低下を招くようになって「ソフトウェア危機」といわれるようになってきた。こうしたことを緩和する動きとして 3 つ紹介する。1960 年代後半、ダイクストラらはプログラムが込み入って、他人が理解しづらいのが問題であると指摘し、それを改善する方法として、構造化プログラミングを提唱した[13]。その特徴は、以下の 3 つからなり、以後のプログラミング言語の使用に大きな影響を与えた。

1）プログラムの可読可能範囲での副プログラム化（細分化）

2）プログラム構造を「順次」「反復」「分岐」の構造で構成

3）不要なジャンプ（GOTO）の使用停止

次に、ヴェルトは 1975 年「Algorithms＋data　structures＝programs（邦訳アルゴリズム＋データ構造＝プログラミング）」を著し[14]、配列、木構造、リストなどのデータ構造をうまく取り入れることによりプログラミングの手間が軽減されることを教科書的に示した。三番目として、1978 年、C 言語開発者のデニスリッチとカーニハンによる「プログラミング言語 C」[15]が著された。この本では、プログラミングを効率的に学ぶ方法と望ましい（分かりやすい）プログラミングの書き方が強調して示され後のプログラミング教科書構成の原型となった。（プログラミング入門書でよくある「hello　world」の表示はこの本の影響である）。「プログラミング言語 C」の構成を下に示す。

1　やさしい入門（プログラミング言語の概略を説明する）　2　データ型・演算子・式 3　制御の流れ（ループ・分岐）　4　関数とプログラム構造　5　ポインタと配列　6　構造体 7　入出力　8　UNIX システム・インタフェース

5　パソコンの出現

1970 年代後半、パソコンが出現し、コンピュータを個人で持てるようになった。アプリケーションソフトは通常なかったので、使うためには、自分でプログラミング（言語は機械語か BASIC）する必要があった。プログラミングが技術者から一般に普及するようになった。多く開設されたパソコン教室はほとんどがプログラミング講座であった。しかし、当時の機械の性能から、プログラミング記述の厳密性とプログラミングの数学的論理性が要求され、プログラミングは多くの人が理解しにくいものとなっていた。そこで、サンプルプログラムを提示し、それを実行して内容を理解するいわゆる「写経型学習法」[16]が広く行われるようになった。この方法でも、サンプルが学習者にとって不適切であったり、この方法自体が学習者に適していないと、理解不能となったり、理解できても応用できないということになり、「コンピュータ嫌い」を産み出す状況となった。1985 年頃、普通教育分野でも情報教育を推進することが提起された*[17]が、上記のことから、プログラミングを最小限にし、アプリケーションの活用や情報モラルを重視した教育が勧められた。さらに、1997 年の学習指導要領では、中学技術でのプログラミングは選択に

留まり、新設された高等学校普通教科「情報」ではプログラミングそのものは扱わないようになった。

6　オブジェクト指向プログラム

1980 年代以降になると、プログラミングは更に大規模・高度化してきた。それに対応するために、プログラムの再利用が考えられるようになってきた。その考えに基づき生まれてきたのがオブジェクト指向プログラムである[18]。再利用しやすい形で用意されたフレームワークやクラスライブラリを活用して効率的にシステムを構築していく方法である。この手法で求められる技能としては、再利用部品を当て填めて作成していくので、従来とは逆の様々な機能部品を適切に配置するシステム構築力、部品であるフレームワークやクラスライブラリの広範な知識、汎用部品であるクラスを開発する能力、システム開発ツールである統合開発環境やその他のツールを効果的な使いこなす技能等であり、従来とは異なった能力が要求される。現在のところ普通教育においてオブジェクト指向プログラムに関した教育は見当たらない。

3　プログラミング教育の現状と教育的意義

インターネットが世界的に開放された 1990 年以降、情報産業は経済分野が優先され、情報関連の技術者は不足してきている。ACM（Association for Computing Machinery）は CC 91 で指摘し、1993 年高等学校向けのコンピュータサイエンスのカリキュラムを制定しプログラミング入門を提案している[19]。それを発展させて、Wing は普通教育しての「計算機的思考」を提唱した[2]。これに基づき、K 12（初等中等）カリキュラムや最近では K 8（小中）カリキュラムが作成されている[19]。一方 MIT（Massachusetts Institute of Technology）のメディアラボ[20]では、学習理論の構成主義に基づき 1967 年 LOGO を開発したが、それを発展させて、2006 年教育用言語スクラッチを開発した。本研究で実践したレゴの Mind Storm もこの系統のシステムである。また、2000 年には、ニューヨーク大学でコンピュータ制御をデザインする「フィジカルプログラミング」の活動が始まった。こうしたことを受けて、Google、Apple、Micro Soft などの IT 企業も初等中等段階でのプログラミング教育の推進に協力している。イギリスやロシア、エストニア、ニュージランド等も同様に初等中等教育段階でのプログラミング教育に力を入れている[21]。

日本では、一時、普通教育でのプログラミングを避ける傾向があったが、情報技術者が不足していることや世界の流れの中から、2012 年実施の学習指導要領中学校技術・家庭で、「プログラミングによる計測・制御」が必修となった[22]。2013 年政府の成長戦略では「義務教育段階からのプログラミング教育等の IT 教育を推進する」内容が盛り込まれ[23]、2016 年 6 月の「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」[24]で小学校でのプログラミング教育の是非について議論され、2020 年からの新しい学習指導要領で小学校のプログラミング教育を実施することとなった[25]。

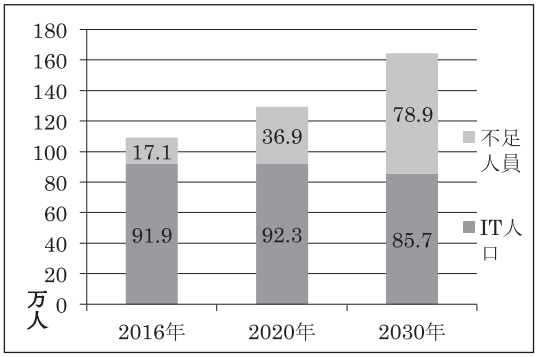

図 1　IT 人材の最新動向と将来推計に関する調査結果（経済産業省　2016 年 6 月）

4　教育モデル

教育内容に関して、大抵の教科・科目においては親学問が存在し、それに基づき生徒の発達段階に応じて内容を系統化し、独自の教育モデルを形成している。そのモデルは教育が発達するにつれ教育効果や形式陶冶的な要素を考慮して、元のものから変質していく。その結果が、例えば中学校数学では、中学生の思考の発達段階に応じた、中学生への教育的意義を考えた内容構成となるので、厳密な意味で、高等学校数学との連続性や関連は薄く、また「数学」という学問とも似て非なるものである。

プログラミング教育と類似した技術的要素を持つ電気回路およびそれに関した電気工学の教育モデルを分析し、教育モデルの必要な要素を記す[26)]。

電気回路は回路網となっており、複雑な結線をすると電圧や電流を求めるのは要素をたどって膨大な作業に思えるが、キルヒホッフの法則によりループ（閉回路）数に応じた多元方程式を立てれば、機械作業的に求めることができる。また、交流回路の計算は本来、微分方程式と三角関数で行い困難を極めるが、周波数が変わらないとすると数学の演算子法を用いて、複素数とベクトルで電圧・電流を表現することができる。こうした、前提となる技術を基に、19 世紀の後半、アメリカのスタインメッツ[26)]らによって電気回路教育モデルが構成された。このモデルでは、キルヒホッフの法則の応用を活用できることと、複素数・ベクトルによる交流回路計算ができることを主眼において、系統性を持たせた教育構成としている。

1　電圧・電流　　2　オームの法則　　3　直並列回路　　4　キルヒホッフの法則
5　キルヒホッフの法則の応用（電圧電流源・分圧・鳳テブナンの定理・電圧降下）
6　交流回路の式（三角関数）　　7　交流回路の素子　　8　複素数表現　　9 ベクトル表現
10　複素数計算の応用（直並列回路・電力計算・共振回路・誘電体損・力率改善）
11　非定常回路（過渡現象・分布定数回路）・・・・　複素数で計算できない回路
電気回路理論教育モデル構成

ここで、キルヒホッフの法則は、実際の回路においては多元方程式となり、実質上解くことが

困難であるので、実務上使用するのは稀である。しかし、この法則を指導する過程で適切な課題を考えさせることにより回路分析の考え方（電圧降下、電圧ポテンシャル、電流の連続性など他で応用できる基本概念）を身につけることができる。また、複素数計算やベクトル計算の応用を体験することにより、回路の特性に応じた解析ツールの適切な使い方がわかったり、効率改善や、回路の発振等の課題解決が初等関数の計算式で分析できるようになる。このように教育モデルは、個々の場所で直接課題解決の方法を学ばせるのではなく、課題解決のための基本的な考え方を身につけさせることを主眼においた内容である。

また、教育モデルにおいては内容の順序性も重要である。電磁気学の分野で、大学や普通高校の物理では電気の後に磁気を学ぶが、工業高校では磁気を先に学ぶ。最終的に両方とも学ぶのでどちらでも同じように思えるが、先に学ぶ方のイメージが強くなるので、両者には電磁気のとらえ方に差が出る。電気を先に学ぶと、電気は単極子（＋、－だけで存在する）であるので、単独点・発散（微分）のイメージができるが、磁気は双極子（単独存在しない）であるので、相互作用・線（積分）のイメージができる。電気を先に学ぶと電波や電気物理現象の理解がスムーズになるのに対し、磁気を先に学ぶと、モーターは発電機、変圧器の理解がスムーズになる。（なお、戦前は普通科・大学でも磁気を先に学んだ）つまり、電気機械や発電を主として学ぶ（電気工学など）には「磁気」を先に押さえ、通信や物性を主として学ぶ（電子工学）には「電気」を先に押さえると学習効果が高くなるということである。

5　プログラミング教育の教育モデル

本節では Wing の「計算論的思考」に基づき小・中・高等学校普通教育におけるプログラミング教育モデルを提案する。

1　Wing の「計算論的思考」

Wing の計算論的思考は、コンピュータサイエンスの教育内容を基に、知識ではなく一般的に通用する思考様式（考え方）を取出したものである。「計算論的思考」は情報的内容を忘れても、思考様式が生活や情報以外の分野でも活用できるようにすることが重要だとしている。コンピュ思考様式として次のようなものを挙げている。

1）抽象化（モデル化）思考

プログラミングは、現実の事象をあらかじめ決められたプログラムの命令やデータに「置き換える」作業である。このように現実の事象を、簡単な言葉や記号、図に置き換える作業が抽象化である。具体的な内容では解決しにくいことが抽象化によって、問題が見通し良くなったり、類推が容易になったりする。

2）ヒューリスティック推論

従来の理論は、理論的構築を図り理論的な解（厳密解）を求める。コンピュータは有限桁の装

置であり、計算時間も有限であるが大量の作業を飽きずにこなしてくれる。これを利用して、あいまいな現象の厳密でないがより解決に近い解を提案してくれる（例えば、線形計画法やモンテカルロ法、ゲーム理論など)。このような思考様式は日常生活で活用できる。

3）再帰的思考

帰納法的思考である。物事を分析する際できるだけ単純な構造を探し、それを繰り返ししようすることにより、事象を単純化して考えることができる。

4）創造性

「論理的思考」について Wing は論理的思考をコンピュータにまかせて（コンピュータは論理演算をする機械であるので）人間として創造力を満たす種々の思考法を育成することが重要であるとしている。一方、他の文献では計算論的思考では「論理性」の育成を含んでいるとしている。その例として、ゲームや問題の解を解くこととしている。問題を解くという行為は、発見という論理からの飛躍を伴う創造的な思考である。他の文献等では、「論理性」とは一貫して説明できる行為（思考）を論理性としていると思えるが混乱を省くため、本論ではこれらの論理性を含めて「創造性」とする。

2　教育目標と獲得すべき資質

従来の教育モデルは、教育内容の設定から付帯的に獲得すべき資質を提示してきたが、「計算論的思考」では逆に獲得すべき資質から逆に内容を考えていく必要がある。教育目標を、前節の思考様式と現状の情報技術、社会的要請から次のようにした。

1）事象を抽象化・モデル化できる能力の育成

2）課題解決に応じた思考パターン（アルゴリズム）を選択できる能力の育成

正解に近い値を効率的に求めるヒューリスティックな推論方法が種々考案されてきた。種々の方法があることを知った上で、実用的側面から正解への近さと効率性を総合的に判断して適切なアルゴリズム（思考パターン）を選択する。

3）部品を組み合わせて、課題解決（創造）する能力の育成

プログラミングは、あらかじめ定められた命令や命令の塊であるパッケージを組み合わせて、要求に応じて新しいものを開発していく行為である。その能力を育成する。

4）コンピュータの動作するしくみを知り、コンピュータの得意・不得意が分かること

5）プログラミングの作成方法が分かり、本格実施に必要な資質・環境が考えられること

3　プログラミング教育モデル案（内容）

教育内容（教育目標に）に応じて、教具（コンピュータシステム）は変更するほうが良い。また教具は実際に実用されているのものではなく、教育効果を高め、抽象化の思考様式を育成するために教育用に特化させたものを利用することが望ましい。

1）プログラミング入門

プログラミングの役割や意義・基本的なプログラミングの操作について直観（見える形）的に学ばせる。内容としては、種々の入力状況から複数の出力を制御し、課題の解答が見える形で現れ、自ら課題を考えられるようにする。使用するシステムとして、ロボットシステムや図 2 のような制御システム（ロボット系教材）を使用する。

図 2　制御システム

ここで行う内容は、

・出力のプログラム　　・命令の書き方理解

・アクチュエータ操作　　・ポート出力の理解　　・逐次実行を理解

・センサ入力　　・条件判断　　・課題の考案　　である。

2）コンピュータの動作原理とプログラミングの意義

・有限桁、逐次制御、メモリという特徴から、コンピュータの得意、不得意を考察する。

・コンピュータ基本演算機能（四則演算、データ移動、データ比較、実行順の変更）を組み合わせて様々なことができることを体験する。

＊ここでは、実際にプログラミングして確かめるのではなく、1）の制御システム若しくは動作シミュレータを作成し、それを使って体験させる。

3）事象を抽象化するための手法

・フローチャート等の図式化技法やデシジョンテーブルを利用して抽象化を体験させる。

＊アンプラグ的にコンピュータを直接使わず、抽象化する技術を取得させる。

4）構造化プログラミング

・3）の内容を踏まえて、構造化の 4 つの必要性を体験的に理解させる。

・構造化の 4 要素で様々の処理ができることを実感させる。（モデル化能力育成）

＊教具はビジュアル言語を使うのが望ましい。

5）データ構造とアルゴリズム

・配列、リスト、リンク、木構造の特徴とそれぞれの検索アルゴリズムを体験する。

・再帰アルゴリズム、繰り返しの数値計算アルゴリズム、コンパイラ技法のアルゴリズムゲーム解法のアルゴリズム、モンテカルロ法などを体験し、種々の思考パターンがあることを理解させる。

課題例

・桁落ち、数値誤差の体験（等差順列加算で小さいほうから加算する場合と、大きいほうか

ら加算する場合で差ができることを体験させる。）

・8 Queen、ハノイの塔　　・分解的平方計算、ニュートン法、台形公式などの数値計算

・4 色問題をテーブル法で検証、木構造でデータ探索等データ構造に関する問題

・モンテカルロ法、線形計画法などヒューリスティックな推論

・スタックを利用したポーランド記法、ゲーム解法等実務的アルゴリズム

＊教具は、アルゴリズムの理解に主にするので、コーディングが容易で見やすいもの。

6）オブジェクト指向プログラム

模擬的パッケージをうまく組み合わせてシステム構築の体験をさせる。

＊教具は実用的な統合開発環境（Integrated Development Environment, IDE）を用いる。

6　実践事例

プログラミング教育モデル　1）プログラミング入門（小学生対象　ロボット系教材を使用）の実践を報告する。

1　使用教材

LEGO 社のマインドストーム EV 3 を使用した。この教材は、レゴブロック（構造部品）を組み立てて作成する。ブロックの自由度が高いので様々なロボットを作成することができる。今回は、プログラミングを主に置いたので、あらかじめ組み立てたセンサ付（超音波センサ、カラーセンサー、回転センサ）自動車型のロボットを使用した。プログラミングは、Lab View で作られたビジュアル言語を用い、タブレットでプログラミングした。

2　教育方法

今回、該当校の状況を考慮して、主体性、協調性の育成を教育目標に加えた。本学学生が指導することにより、児童の発想や想像力、創造性を重視し児童へ思考法の押し付けを行わない指導になると考えられた。大学生が小学生に教えることで、大学生自身の教える力の強化も念頭に置いている。

3　実施報告

尼崎市立立花西小学校の 4 年～6 年生のコンピュータクラブ活動で、2017 年 1 月から 3 月のクラブ活動の時間に計 4 回実施した。対象人数 20 名に対し 5 名の学生が指導した。

（1）学生への事前教育

参加した 5 名の学生への事前教育を行った。学生が児童へ教える際に、自分たちの既知の課題解決方法を押し付けるのではなく、児童の発想や想像力創造性を重視し、児童の考える際のヒントを少しずつ出すように指導した。学生自ら EV 3 の自動車型への組み立てを行わせ、各センサ

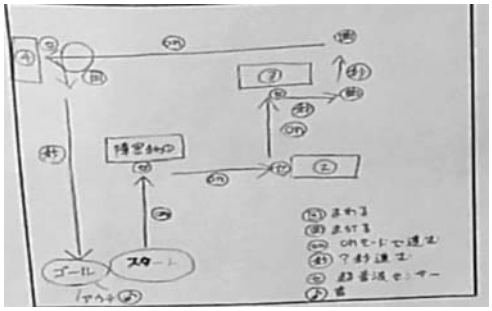

図 3　学生のデモプログラム構成案

やアクチュエータの使い型、条件分岐・ループのプログラム、課題を与えての解決などを学生に指導した（図 3）。

（2）小学校での実施（指導）内容

第 1 回に EV 3 の基本的なスイッチの説明と使い方、タブレットの利用法とプログラミング方法、センサやモーターの解説と接続場所の関係について説明し、実際に児童にモーターを動くプログラムを作らせた。最後に学生がコーストレースのデモンストレーションを行った。機械の台数から 1 グループ児童 4 名で実施した。特定の児童のみが触れる状態にならないよう、交代して使うように留意した。

第 2 回は第 1 回目の復習と、種々のモーターの動かし方、センサの使い方について説明し、児童に体験させた。センサの使い方は、超音波センサの使い方について説明し、センサの感知により動作、音を発生させることを試させた。

第 3 回は第 2 回までの復習を行い、プログラムでループと条件分岐について、それぞれの必要性を体験できる課題（三角形・四角形の動きのトレース、障害物を回避するプログラム）を与え簡易サンプルを提示した後、児童に考えさせた。

第 4 回（最終回）は、学生の発案で、課題のデモンストレーションビデオを作成しておき、それを使って、課題（自分でロボットの動くコースを作成し、そのトレースをするプログラム）を説明し、児童にプログラムを作成させた。児童は各グループでコースを計画し（図 4）、コースを実際に作成しながら計画の修正を行った（図 5）。最後に、各グループで実演・発表させた（図 6）。

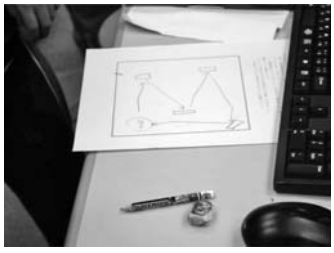

図 4　児童が作ったコース

図 5　児童のコース作成の様子

図 6　発表時のコース説明の様子

4　考察とまとめ

最終的に、課題を達成できたグループは、5 グループ中 1 グループに終わったが、すべてのグループがループやセンサの条件分岐の使い方を理解し、適切に活用していた。課題が達成できなかった原因は、摩擦等の影響で理屈からずれたこと、自分で設定したコースに機械が対応できな

かったり（角度が急であったり、センサの検知からモーターの動作までの時間が間に合わない）等が原因であった。成功したグループは、摩擦等の影響をメジャーで計測し、補正してプログラミングしていた。通常コンピュータ上で行うコードで記述するプログラミングは、記述方法や結果が分かりにくので、初心者がプログラミング出来るまで時間がかかるが、直観できるプログラム環境では短期間でプログラムが理解できる。そうしたことから生まれた、ビジュアル言語と比較したとき、ロボット系教材では、動作させるのには時間がかかり、プログラム以外の要素で正しく動かないことがあるので、ビジュアル言語の方が教育効率に優れているように見える。しかし、プログラミングで獲得すべき資質を考えたとき、コンピュータの得意・不得意なことを理解させたり、ヒューリスティクな推論を初学者の段階で時間できること、科学的な実験、推論等が付帯的にできることなどから、ロボット系教材が優れていると確信できた。

また、指導法として、今回、学生に課題の提示までは指導者（インストラクタ）の役割を担わせたが、児童の活動の場面では消極的なアドバイザの役割を担わせた。余分なアドバイスもせず、児童のモチベーション向上に留意し、正解を言わずに考えさせるアドバイスは効果があったと思われる。指導者人数であるが、実際にプログラミング教育を行う際には、教材そのもので考えられる躓き以外の単純なコンピュータトラブルなども発生する。そのため、クラス単位での授業の中で 1 人の教員がすべての児童をフォローすることは困難であり、教材の用意なども困難を伴う。このことから、複数の指導者による教育が望ましい。

参考文献・注

1）Papert, Seymour（1996）.“An exploration in the space of mathematics educations”. International Journal of Computers for Mathematical Learning.1

2）Wing, Jeanette M.（2006）.“Computational thinking”. Communications of the ACM, 49（3）

3）http : //www.cs.cmu.edu/~CompThink/

4）http : //www.cs4fn.org/computationalthinking/

5）報告「大学教育の分野別質保証のための教育課程編成上の参照基準情報学分野」，平成 28 年（2016）年 3 月 23 日，日本学術会議情報学委員会

6）L. F. Menabrea, Ada Augusta, Sketch of the Analytical Engine, Bibliothèque Universelle de Genève, Number 82, October 1842　URL :　http : //www.fourmilab.ch/babbage/sketch.html

7）Statistics, Economics, Computing, and other research, Michael D. Godfrey, Stanford University　URL : https : //sites.google.com/site/michaeldgodfrey/other#sect_vonN

8）History of FORTRAN and FORTRAN II, Computer History Museum, Plone Foundation　URL :　http : //www.softwarepreservation.org/projects/FORTRAN/

9）今城哲二，横塚大典，床分眞一（2001）「21 世紀も，COBOL だね　－COBOL 生誕 40 周年トピックス－」，Bit April 2001/Vol.33, No.4

＊10）伊理正夫，藤野和建（1985）「数値計算の常識」，山本哲朗（1976）「数値解析入門」，赤坂隆（1967）「数値計算」，などから，中等教育の範囲に合致する内容を抜粋し再構築した。

11）NHK クロニクル　URL : http : //www.nhk.or.jp/archives/chronicle/index.html より検索

12）Web ページ「NHK コンピュータ講座　NHK マイコン入門　テキスト」，CRIMSON Systems　URL : http : //crimson-systems.com/win/book0.htm

13）E. W. ダイクストラ（著），C. A. R. ホーア（著），O. -J. ダール（著），野下浩平（翻訳）（1975），「構

造化プログラミング（サイエンスライブラリ情報電算機 32）」
14）N. ヴィルト（著），Niklaus Wirth（著），浦　昭二（翻訳），国府方久史（翻訳）（1990）「アルゴリズムとデータ構造」＊現在入手できるのはこの題名である。以前「アルゴリズム＋データ構造＝プログラミング」日本コンピュータ協会であったが改訂され名称が変わった。本論では原著に従った。
15）B. W. カーニハン（著），D. M. リッチー（著），石田晴久（翻訳）（1989）「プログラミング言語 C 第 2 版　ANSI 規格準拠」
16）例えば，喜多　一・岡本雅子・藤岡健史・吉川直人（2012），「写経型学習による C 言語プログラミングワークブック」，共立出版
＊17）文部省（1985），「情報化社会に対応する初等中等教育の在り方に関する調査協力者会議第　1 次審議とりまとめ」，1986，臨時教育審議会二次答申など
18）平澤　章（2004），「オブジェクト指向でなぜつくるのか－知っておきたいプログラミング，UML，設計の基礎知識－」，日経 BP 3
19）難波（2012）「コンピュータ・サイエンスのカリキュラム」兵庫県高等学校教育研究会情報部会研究論文集
20）MII Medialab　URL :　https : //www.media.mit.edu/
21）文部科学省報告書（2014）「諸外国におけるプログラミング教育に関する調査研究」
22）文部科学省（2008）中学校学習指導要領解説　技術・家庭編
23）産業競争力会議（2013）成長戦略
24）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（2016）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）関連 URL　http : //www.mext.go.jp/b_menu/shingi/chousa/shotou/122/index.htm
25）関連する公的資料は次のとおり
文部科学省（2014）「プログラミング教育実践ガイド（平成 26 年度）」
総務省（2015）報告書「プログラミング人材育成の在り方に関する調査研究」
総務省（2017）報告書「プログラミング教育事業者等の現状に関する調査研究に係る請負」
総務省（2017）報告書「若年層に対するプログラミング教育の普及促進に向けた調査研究」
26）山崎俊雄・木本忠昭（1976）「電気の技術史」，オーム社

〔まつ　ひでき　有機化学〕
〔なんば　こうじ　教育工学〕