

海外教育 メディアレポート

アメリカ合衆国マサチューセッツ州

プログラミング教育

カリキュラム大綱によるプログラミング
教育の取組 (アメリカ合衆国マサチューセッツ州)

東京学芸大学名誉教授
篠原 文陽児

マサチューセッツ州の特色

アメリカ合衆国の州の一つマサチューセッツ州は、19世紀には全米における工業の中心地として、20世紀には経済の中心をサービス業に転換し栄えた。そして、21世紀は、ハイテク、高等教育、医療技術、金融業、観光業で、全米のみならず世界をリードする存在になっている。

教育分野に目を転じると、1647年に教育法の成立、1840年には幼稚園から中等学校の公立学校システムを他の州に先駆けて構築し、全米のパイオニアとしての名声を得ている。また、1852年には義務教育法を成立させた全米最初の州でもある。

学校教育制度

州の学校教育制度は、学区によって異なるものの、基本的には、幼稚園(5歳児)1年間、初等教育は6歳児を1年生として、5年生から8年生までのいずれか、そして、中等教育は6年生から12年生(ただし、上記初等教育との組み合わせがある)である。これら幼稚園から中等教育終了

までの13年間はK-12の義務教育である。

カリキュラム大綱 -カリキュラムの標準-

K-12のカリキュラムの標準は、カリキュラム大綱(Curriculum Framework)と称され、州教育省初等中等教育局が公表している。プログラミング教育は、そのうちの一つ、2016年6月公表の「2016年マサチューセッツ州デジタルリテラシーとコンピュータ科学(DLCS; Digital Literacy and Computer Science)」カリキュラム大綱に見出すことができる。そこには、学年と教科のそれぞれに対応した、すべての幼稚園児および児童生徒に期待される学習成果「知ることとできること」が、示されている。幼稚園と学校は、この大綱に示された実技を含む内容を参考に、教育課程を編成する。したがって、教師は、学区と学校の環境および対象とする学習者一人ひとりに質の高い教育を実現するため、web上に用意された副教材や資料等を選択したり、必要に応じて支援を受け独自の教材を開発したりするとともに、他教科との横断的な関連に十分な配慮をしたりして、授業等を実施する。

DLCSカリキュラム大綱とプログラミング教育

大綱の基本理念の一つは、複雑な問題を解くための技術の効果的な活用とその技術を創造する能力、言い換えれば、21世紀に生きる市民に必須の知識と理論および技能を構造化して示すことによって、できるだけ多くの、かつ、できるだけ広範囲を想定する学習者の将来が、革新的で創造的な生き方になるよう、その学習を保証することである。

大綱では、こうした理念を実現するため、学年等段階をK-2、3-5、6-8、9-12の4つに大別し、それぞれに4つの構成要素(Strand)、CAS(コンピュータと社会)、DTC(デジタルツールと協働)、CS(コンピュータシステム)、CT(コンピュータ的思考)を割り当てている。このうちCTがプログラミング教育であり、その内容は、抽象化、アルゴリズム、データ、プログラミングと開発、モデリングとシミュレーションの5項目である。

表は、学年段階3-5のCTの内容等である。

まとめ

わが国のプログラミング教育は、2020年度からの新学習指導要領を踏まえ、特に小学校での試行や実践研究で、「スクラッチ」の活用が多く見られる。「スクラッチ」と聞くたび、筆者は、社会構成主義(Social Constructivism)を背景とするプログラミング言語LOGOと書籍「マインドストーム」を思い出す。そして、この言語の開発者と書籍の著者こそ、マサチューセッツ州にあるマサチューセッツ工科大学(MIT)のS. パパートである。

彼のお膝元であるマサチューセッツ州の、時代を見据え超えた、先進的なプログラミング教育推進の好事例は、わが国の今と近未来のプログラミング教育、および、その先にあるコンピュータ科学教育のあり方を示唆しているように思えてならない。

学年段階3-5: コンピュータ的思考 (CT)

内容	記号	知ることとできること
抽象化 3-5.CT.a	3-5.CT.a.1	数字またはアルファベットを使って、情報を別の形式で表現する(例:暗号、略字)
	3-5.CT.a.2	情報を異なる方法でまとめ、より有益あるいは意味ある表現にする(例:並べ替え、表)
	3-5.CT.a.3	サブプログラムの一覧表を作成する。同時に、より大きな課題を考える
アルゴリズム 3-5.CT.b	3-5.CT.b.1	コンピュータで処理できる指示の順序として、アルゴリズムを書く
	3-5.CT.b.2	同じ問題で異なる解が存在することを確認する
	3-5.CT.b.3	アルゴリズムの結果を予測し、その理由を論理的に考える
	3-5.CT.b.4	一人で、また、協働して、問題解決のアルゴリズムを創る(例:碁盤状に描かれた絵の上で、キャラクター、ロボット、ヒトを動かす)
	3-5.CT.b.5	さまざまなアルゴリズムで、論理的に誤りを見つけ正しくする(例:書き物、地図、実際の動作、数字)
データ 3-5.CT.c	3-5.CT.c.1	日々の生活の中からデータベースの例を記述する(例:図書館のカタログ、電話帳、住所録)
	3-5.CT.c.2	多様なコンピュータ処理の方法(例:並べ替え、合計、平均)を使ってデータを集め操作し、問題に答える。また、ツール(例:スプレッドシート)を使ってデータを構造化、グラフィック化する
プログラミングと開発 3-5.CT.d	3-5.CT.d.1	一人で、また、協働して、図表現でプログラムを創り、テストし、変更する(例:ブロック形式のビジュアルプログラミング言語)
	3-5.CT.d.2	プログラムに演算子、条件、繰り返しを使う
	3-5.CT.d.3	デバッグにより、プログラムの簡単な誤りを見つけ修正する
	3-5.CT.d.4	プログラミングに初期値が必要であることを理解する(例:ゲームで、最初にゼロをセットする)
モデリングとシミュレーション 3-5.CT.e	3-5.CT.e.1	簡単なシステムモデルを一人で、また、協働して創る(例:太陽系)、そして、モデルが示す部分等と示さないそれらを説明する
	3-5.CT.e.2	シミュレーションで表現される概念、特徴、動作を確認する
	3-5.CT.e.3	問題に答えるため、シミュレーションからのデータを一人で、また、協働して使う