Vol.2016-CE-136 No.10
2016/10/15

プログラミング入門教育と受講者モデル
ー抽象化の教育ー

大岩 元[†1]

プログラミング入門教育は、逐次処理、繰り返し処理、条件分岐処理の実例を実行することから始まる。しかし、その活動は指示された通り実行したら、予定された結果が出るという経験だけでは不十分で、そこで実行したプログラムの意味を受講者がどれだけ理解したかが問題となる。さらに、理解した簡単なプログラムをもとに、複雑なプログラムを作成できるようになるための教育が必要である。これらについて抽象化概念の教育の視点から考察する。

How to Execute Introduction to Programming and Modeling a Learner
-From the Viewpoint of Abstraction-

Hajime OHIWA[†1]

Introduction to programming should not be done as just an experience of running some simple examples. It should include the explanation of the meaning of codes to be learned so that the learner can make a coding for any similar programs in future. Notion of abstraction should be stressed in learning process.

1. はじめに

次期学習指導要領は 2020 年に小学校、2021 年に中学校、2022 年に高等学校で実施される。情報化の進展をうけて、これまでの学校教育の内容が時代遅れとならないようにするために現在可能な教育を、中教審が模索した結果がいよいよ実施されることになる。

発表された次期指導要領で重要なことは、「何を学ぶか」だけでなく「どのように学ぶか」を重視することである。これが、これまでの指導要領と大きく異なる。このための学習方法として「アクティブ・ラーニング」が推奨されており、プログラミング教育はそのための格好の教材の一つになるはずである。

アクティブ・ラーニングでは、「主体的な学び」「対話的な学び」「深い学び」が求められている。プログラミングはゲームのような児童が興味を持つ題材をとりあげることが容易であり、作り出す過程で協同作業が行ない易く、対話的な学びが可能である。また、こうした教育過程の中で指導者が適切に方向づけすれば、computer science の深い概念を伝えることも可能である。

しかし、現行のプログラミング教育はほとんど全てが、特定のプログラムを作り出す過程を体験させるだけで、深い学びにつながるような教育を行っていない。このことは専門家の教育に至るまで続いており、その結果として少数の例外を除いて、他国とは違う低レベルのプログラミング技術者しか日本には存在せず、高い人件費のもと、高コストの情報投資を続けてきた。おそらく、日本の経済力の低下の原因は情報技術をうまく使いこなせていないことに起因するものであろう。

新しい学習指導要領の求める、主体的・対話的な深い学びをプログラミング教育で実現するには、従来の「プログラミング教育」、即ち、プログラミング言語教育から決別することが必要である。

2. プログラミングの教師に求められるもの

一般人を対象としたプログラミング入門教育においては、受講者は計算機について何も知らないことを前提に行なうことになる。そこでは、プログラミングに関する前提知識と能力は何もないと仮定しなければならない。受講者の母語を読み書きする能力だけが出発時の能力ということになる。次に必要となるのは、文字列をファイル上で編集する能力である。入門教育はここから始めることになるが、アプリケーションの利用を経験した受講者には必要なくなる。

プログラミングは、情報を操作する手順を書き下すことである。さらに、それがコンピュータの進歩にともなって大規模化したことから、複雑な手順の記述を人間が誤りなく作り出すための抽象化の概念を身につける必要がある。この抽象化については本来数学で教育されるはずであったが、そのようには行なわれておらず、種々の問題に対する解法を修得するという方法で行なわれているため、プログラミング教育と同様、学習者には数学の深い概念が身についていかない。

学習者に抽象化をどこまで学習させるかについては議論すら始まっていないが、明らかなことは、教師の側にその

†1 慶應義塾大学
Keio University

概念の理解と問題解決においてその概念を運用する能力が求められることである。しかし、現状は専門学科に至るまで、そのような教育は行なわれずに、ひたすら現れる新しい技術を教育の中にどうとり入れるかという、専門学校的な発想で教育が行なわれている。例えば、状態と状態間の遷移は大変応用範囲が広い概念であり、状態遷移図と正規表現の等価性とその拡張としてのチョムスキーの階層関係は全ての情報技術者と教師がわきまえておくべき事項であろう。

3. 抽象化を体験するアクティブ・ラーニングの具体例

アクティブ・ラーニングも、最初の段階では何らかの知識やスキルを教えこむ必要がある。プログラミングの場合、教育に用いる環境で使える命令について学習者が理解する必要がある。制御構造を教えるタートルグラフィックスの場合、亀を前後に動かす命令と、向きを変える命令を理解して、これを使いこなしてその効果を体験することが学習の第１歩となる。この命令を使って家を描く場合を例にして、そこで抽象化の概念をどう把握させていくかを考えてみよう。

最初の描画：四角形

最初に四角形を描かせてみるのが最初のプログラミング体験になる。これは、100 歩前進、90 度右回転という命令を書き下して実行することで、結果が確認できる。

次の描画：三角形

次に、三角形を描かせる課題を与えると、普通は問題が生じる、三角形の学習を済ませた学習者には、三角形と内角の 60 度という関係が刷り込まれているために、100 歩前進、60 度右回転という命令を３回書いて失敗する。教え込む教育であると、ここで答の 120 度を教えてしまう。これでは答を覚え込むことで三角形は描けるようになるが、同種の問題への解答能力が育たない。

三角形を手で描かせて、内角が 60 度であることの意味を再確認させると同時に、三角形を描かせるために亀にどんな命令を与えるべきかを考えさせると、三角形に関する理解も深まる。このような「アンプラグド」と呼ばれる方法も当然併用しなければならない。

繰り返し機能の導入

次に行なうのは、繰り返し機能の導入である。自分の書いたプログラムを読み返してみれば、同じ事が繰り返されていることが分る。そこで１連の命令を繰り返す方法を教えて、その形式で同じ結果を得ることができることを確認させる。

組み合せて家を描く

四角形の上に３角形をのせると、簡単な家の絵が描ける。三角形は屋根、四角形な本体の壁を描いたことになる。

ここで、三角形と四角形の関係が問題となる。屋根から描いて次に壁を描くことにすると、最初に上を向いている亀の向きを 30 度右に回す前処理が必要となる。三角形を描き終ると、亀の向きを 60 度回す前処理をしてから、四角形を描くと家の絵が完成する。

ｎ角形を描く関数を作って利用する関数の利用

ｎ角形を描く関数を定義すると、家は３角形と四角形の描く関数を呼び出して実行することになる。そこで問題となるのは、三角形を描いた後に行なう 60 度の回転の意味づけである。関数を使うことになると、三角形を描いた後は、亀を標準状態に戻す方がよい。この命令は、屋根を描くための後処理となる。続いて、壁を描く前処理として 90 度右回転する命令を書くことになる。壁を描き終ると亀は水平方向を向いているので、90 度左回転する後処理命令を書いて、プログラムの記述を終る。

最後の関数の利用の所で説明したように、各命令ないし命令群をどこで分けるか、何を命令群としてまとめるか、ということが問題になる。また繰り返しの内容はどのような命令群を使うかも同様に検討する必要がある。これらは抽象化の作業であり、このように具体的に抽象化を考えることで、抽象化という概念を具体的に理解することができるようになる。また、このように抽象化された一群の命令は関数としてまとめることができる。

このような単純な例で小さな抽象化を体験しておけば、２重ループのような繰り返しも、容易に組み立てられるようになる。そこで強調すべきことは、逐次処理、繰り返し処理、条件分岐処理はいずれも入口一つ出口１つの構造であることから、これらの複雑な構造の文を単純な文と同じように文として複雑な構造の文を構成する文として埋め込むことができるという一般的な原理である。

4. 教師と学習者のモデル

入門教育は、学習者に対して一定の知識、スキルレベルを求める必要があるが、教えたい内容、今の場合プログラミングについては何も持っていないという前提で教育を計画する必要がある。従って、説明は全て母語である日本語で行なう必要があり、上記の作図の例では、まず前進命令と回転命令を日本語で説明する。その上で、作図すべきものを伝えれば、プログラムが書けるはずである。不用意に、専門用語を使ってはいけない。上記のように徐々に複雑な作図を行なうことで、与えられた命令を使って目的の達成されることが体験できる。

このように単純なことをプログラミングすることで、抽象化の概念を体得させることによって、必要な命令と作りたいプログラムの仕様が決まれば、それを実現できる能力が獲得されるようになる。その際、抽象化を意識することが重要であり、その結果の命令群に対して適切な名前をつ

け､またコメントとして残す習慣をつけなければならない。

実際の専門家によるプログラミング作業の場合も、作りたいプログラムの仕様に従って用意された機能を実現する命令を組み合せてそれを実現することが必要となる。現在はソフトウェア技術の進歩によって、用意される機能が複雑化したが、日本の情報技術者の多くは覚えたプログラミング言語の命令しか使えないので、いわゆるスクラッチ開発しか出来ない。今のようにソフトウェア部品が潤沢に用意された環境でプログラミング作業を行なうことができない技術者が多く、ソフトウェア生産が非効率で高コスト化している。専門技術者の教育も、単にプログラムが書けるようになるだけでなく、抽象化を意識した教育が必要になる。

従来の、写経型教育に代表されるような教え込み型の教育では、このようなシステマティックな知識体系の構築が学習者によって行なわれることはないので、学習者は自分が覚えこんだプログラムの小修正の範囲でしか、新しいプログラムを作る能力が育たない。

教師に求められることは、学習者の能力範囲を常に意識して、その範囲で適切な課題設定を伝え、つまずいた時に解答を教えるのではなく、自分で考えるために役立つ最小の示唆を与えることである。これには、教えているプログラミング技術に関する深い理解とともに、学習者の知識レベルの理解とそれに応じた日本語の説明能力が要求される。

5. 日本語プログラミングの意義

プログラミング教育で問題になるのは、どの言語を使うかである。しかし、この問題は入門教育に関しては学習者の母語が決定版と言ってよいだろう。言語教育が全く不要になるからである 1)。

自然言語は曖昧だから、プログラミング言語にはなり得ないという考えが一般的であるが、自然言語をそのままプログラムとして使えという主張をしているのではない。プログラミング言語としての仕様を決めた上で、それに従って書かれたプログラムが自然言語として読むことができるようにするのが、母語プログラミングである。

機械翻訳が研究され、現在ではかなりの程度実用になっている。機械翻訳の困難は、人間が使う言語の解釈に使う人の暗黙の知識が必要で､それが特定できないからである。しかし、プログラミング言語の場合、書かれた言語の意味が確定できるようなものを作るのであるから、機械翻訳の困難は最初から無い。

現行のプログラミング言語は計算機が非力な時代に使い始めたために、英語等の言語表現を、簡略化している。その方が書く手間が少なくて済むという利点もある。

問題は、簡略化されていても、元の言語の構造は反映されていることである。我々は英語でプログラムを読み書きしているのである。これは特に、初心者の場合には大きな障害になる。数学も同じ問題を抱えている。数式の記法を変えることは殆ど不可能と言ってよいが、プログラミング言語の場合はそうでない。

日本ではなく米国で、FORTH や APL のように日本語の語順の言語が開発されたが普及しなかった。これは、ヨーロッパ言語の人達には負担になったからであろう。しかし、彼らがわざわざ開発する程、日本語の語順は計算機処理に適しているのである。日本人がこれを使わない手はない。

日本語でプログラムを書いても外国人が読めない、という批判がある。これも、各国語のブログラミング言語が制定されれば、簡単に翻訳できるので、問題は生じない。

現実のプログラミング言語は自然言語ではないので、入門教育が終った先はどうするのかという問題がある。しかし、これも我々の経験では、プログラミングが習得できなかった学生に対して最初に日本語プログラミングで基本事項を修得させることで、同じ授業時間で全員に挿入ソート程度のプログラムを書けるようにすることができた 2)。プログラミングの概念を学べば、その後に言語を学ぶことはずっと容易になる。

さらに、組込みシステムの開発現場などで、日本語プログラミングに興味を持つ人もいる。入門時に一番効果が現われるが、そこに限むものではなく、実用的にも意味がある。例えば、金融分野ではコボルのプログラムをまず日本語で表現して、それを「プログラマー」がコボルに翻訳するという作業形態が行なわれてきた。現在なら、この翻訳は人間でなく、計算機が行なう仕事である。しかし、金融の現場の問題を一番深く理解している人間がプログラムを書くことには大きな意味がある。日本語プログラミングは長い歴史を持つのである。

6. おわりに

抽象化を中心にした入門プログラミング教育の例を具体的に示した。このような教育は、入門時に限らずソフトウェア部品を使って開発を行なう情報産業の現場の教育としても必要であろう。そこで重要になるのは、開発に使える部品はどのようなものかを深く理解し、それを使って意味のあるプログラムを作ることである。そこで教師に要求される能力は、日本語で対象を正確に記述し、伝える能力である。プログラム自体も、日本語で記述することが、入門教育を効率的に行なえるようにするだけでなく、情報産業における現場においても有効であることが判明した。

参考文献

1) 大岩 元, “識字教育としてのプログラミング”, 情報処理学会論文誌教育とコンピュータ（TCE）Vol.1, No.2(2015)
2) 杉浦 学, 松澤芳昭, 岡田 健, 大岩 元. アルゴリズ構築能力育成の導入教育：実作業による概念理解に基づくアルゴリズム構築体験とその効果. 情報処理学会論文誌, Vol.49, No.10, 2008