2007－CE－92 (18)
2007／12／8

日本語プログラミングを用いた論理思考とプログラミングの教育

岡田 健[†1] 杉浦 学[†1] 松澤芳昭[†1] 大岩 元[†2]
[†1]慶應義塾大学政策・メディア研究科 [†2]慶應義塾大学環境情報学部

概要

日本語プログラミング言語「ことだま on Squeak」を用いて、プログラミング入門教育を行なうことにより、論理的に考える訓練を行った後、Java で同じ内容を行ない、更にそれを発展させる「論理思考とプログラミング」という新しい授業科目を実施した。その設計思想と実施結果について報告する。

１．はじめに

慶応大学湘南藤沢キャンバス（ＳＦＣ）では、1990 年の開学以来プログラミングを全学生に必習として課してきた。これは、プログラミングこそがコンピュータの本質であり、それを理解することで、コンピュータの可能性と限界が理解できると考えたからである。

しかし、学生のプログラミング能力は年々低下してきて、最近では自分でプログラムを作成できる学生は、もともとそうした体験を持って入学した学生に限られるようになってきた。コマンド操作が必要であった Unix 環境で教育していた間はまだ何とかなったものの、具体的に作業ができる Windows 環境で情報教育を行なうようになると、その傾向は一段と加速されたように思われる。

こうした状況を改善するために、日本語でプログラムを書くことから始めて、アルゴリズム概念を理解してから、本格的にプログラムを作るという授業を 2007 年度から始めた。その授業設計と結果について報告する。

２．UNESCO の提案と論理思考

UNESCO は、IFIP の協力のもとに ICT 教育に関する勧告を 2002 年に出版している[1]。その報告書では、ICT 教育の内容として次の４つの内容を議論している。

ICT Curriculum for Secondary Students:
- ICT Literacy
- Application of ICT in Subject Areas
- Infusing ICT across the Curriculum
- ICT Specialization

最初の ICT Literacy は、情報技術の活用を教育を行なうことであり、日本の教育制度のもとでも、中学と高校で必習として行なわれるようになった。次の Application of ICT in Subject Area は、全ての教科の中で ICT を活用することであるが、これは日本では実験的に行なわれているものの、進んだシンガポール等と比較すると遅れていると言わざるを得ない。Infusing ICT across the Curriculum は、複数の教科にまたがる内容の実社会の問題をプロジェクトとして解決に導く活動などにおいて、ICT を活用することである。日本の学校では総合的学習の時間に、このような教育が試行されている。最後の ICT Specialization は次のような学生を対象とした教育である。

For those who plan to go into professionals that uses ICT
For those who plan to go into higher education

このうち、中等教育を終えてプロフェッショナルになる人達のための教育については、日本では専門教科「情報」が設けられており、制度としては確立している。これに対して、高等教育に進学する人のための教育については、その必要性すら認識されていない。

ICT Specialization の内容は次のようなものである。

ICT Specialization:
・Specialization Preparation Modules
SP1 Introduction to Programming
SP2 Top-Down Program Design
・General Specialization Modules
GS1 Foundation of Programming and Software development
GS2 Advanced Elements of Programming
・Vocational Specialization Modules
VS1 Business Information Systems
VS2 Process Control Systems
VS3 Project Management

これらの教育目標は、「実社会の問題をalgorithmicに解くことができるようになる」ことである。しかし日本の教育では、時間の制約もあって、実際の問題とは無関係に、情報技術を教えている場合がほとんどである。情報技術の使い方だけを教わっても、それがどのように自分の問題に関係があるかが分らないために、受講学生はせっかく教わったことをじきに忘れてしまう。この結果、情報教育を行っても社会に影響を与えるような効果が上がっていないのが、日本の実情である。

ICT Specialization の最初の内容である SP1 Introduction to Programming は次のようなものである。

Introduction to Programming
・Design a Task-Oriented Algorithm
Describe and specify the task to be realized; develop an effective and efficient algorithm that realizes the identified task, applying a simple, given standard method.
・Translating the Design into a Program
Transform their simple algorithms into computer programs using a (Procedural) language; Produce a readable, understandable and user-interactive program.
・Bringing the Program to Life
Use a give programming environment to enter, edit, compile, debug, update, and run programs they construct; Give a meaningful and useful written description of the internal and external behavior of their program.

ここで注目すべきなのは、Task-Oriented Algorithm という概念である。まず解くべき問題における課題を具体的に書き下して確定することから始めることになる。

プログラミング教育を行なう多くの教師は、自分のためにプログラムを書いた経験はあっても、他人のために自分が知らない問題を解決するプログラムを書いた経験がない。情報産業では、この問題が実は開発の失敗の大きな原因となっている。一方、教わる生徒の方も、初めて聞く問題を解くのであるから、その問題について明確な理解を持っていない。まず課題を自分のことばで書き下すことは、問題理解の重要なステップとなる。続いて、知っている単純な方法を用いて課題を解くアルゴリズムを作り

出す。プログラムを作る前に、まずアルゴリズムの設計を行なうのである。

次のステップは、作り出したアルゴリズムを実行可能な手続き型のプログラムに書き直す。その際、プログラムは作成者以外が読んで直ちに理解できるようであり、それを使う人にとって使い易いような配慮が成されていなければならない。

最後に、作ったプログラムを実際にコンピュータに入力し、デバッグを行って利用可能な状態にまでもって行く。さらに、そのプログラムに関して何が実現されたのかが分る外部記述文書と、それがどのように実現されたかが分る内部記述文書にまとめる。

以上の考えをもとに、「問題から解くべき課題を設定して，合理的な方法で課題を解決し，それによって問題が解決できたかを検証できるようになる」ことと、「自分でアルゴリズム（仕事の手順）を考案し，それを他人やコンピュータに伝わるように正確に，完全に記述できるようになる」らの二つを教育する授業として「論理思考とプログラミング」という科目を作った。

3．日本語プログラミング言語「言霊」

現在広く利用されている Java 言語は、実行する際に仮想計算機（JavaVM）を用いる。Java プログラムを記述すると、コンパイラは JavaVM が解釈できる機械語に変換する。その機械語を JavaVM が解釈して演算する事で Java プログラムは実行される。例えば図１のような Java 言語のコードがあったとする。Java コンパイラはこれを機械語に変換するが、その機械語を Jasmin[2] と呼ばれるアセンブラのニモニックで表現したものが図２である。実行時には図２のニモニックで表された機械語命令に従って演算が行われる。

「言霊」は、Jasmin 形式のニモニックを日本語化し、かつその表現を拡張する方向で設計された。例えば図１・図２を「言霊」で記述すると、図３のような機械語レベルの日本語表現になる。だが図３のような記述は低級表現であり表現が複雑になるので、図４のような高級表現も可能にしている。

4．教育環境「ことだま on Squeak」

"言霊"は日本語表現を使ったプログラムが記述出来るが，全ての日本語表現を解釈できるわけではない．コンピュータを使っている以上は解釈可能な表現には限りがある．

"言霊"におけるプログラムの書きにくさは，日常言語における表現の幅は広いのに対して，

```
if( x==0 )
  y=1;
else
  y=2;
```

図１ Java 言語の記述

```
  iload_1
  ifne Label_0
  iconst_1
  istore_2
  goto Label_1
 Label_0:
  iconst_2
  istore_2
 Label_1:
```

図２ アセンブラ Jasmin による記述

```
  ローカル変数１番目から整数を積む。
  ０以外ならば、ラベル０にジャンプする。
  整数１を積む。
  ローカル変数２番目に整数を格納する。
  ラベル１にジャンプする。
ラベル０：
  整数２を積む。
  ローカル変数２番目に整数を格納する。
ラベル１：
```

図３　日本語による低級表現

```
ｘが０ならば、１をｙに代入する。
そうでなければ、２をｙに代入する。
```

図４　日本語による高級表現

"言霊"の表現の幅が狭いことに起因する.例えば関数名として「現れる」という動詞が定義されているとき,代わりに「あらわれる」という平仮名表現を使ったり,「現われる」と異なる送り仮名を使ったりする.このように慣れ親しんできた日本語が使えると,ついつい日常的な癖で様々な表現を使ってしまう.

多くの初心者は様々な日本語表現を用いてプログラムを記述し,その結果コンパイルエラーを引き起こしてしまう.コンパイルエラーを解決するにはプログラム言語の文法の知識が必要になり,これが受講者にとってアルゴリズムを考えることの障壁となる.

構造エディタを持ち,かつプログラミング教育に適した環境として近年注目を集めているのが,Squeak[3]である.SqueakとはAlan Kay博士が開発したSmalltalk環境のひとつであり,オブジェクト指向プログラミング環境である.Squeakでは描画ツールで描いた絵をオブジェクトとしてプログラムから操作することが出来る.

図5　描画ツールによる車の作成

Squeakは開発者向けSmalltalk言語とは別に,非開発者向けのeToyと呼ばれるビジュアルプログラミング環境を持つ.eToyはタイルスクリプティング環境とも呼ばれ,パネル上のパーツ(タイル)をドラッグ&ドロップすることでプログラミングが可能である.図6は,図5で作成した車オブジェクトに対して送ることが出来る命令のタイルの一部を表示したときの画面である.

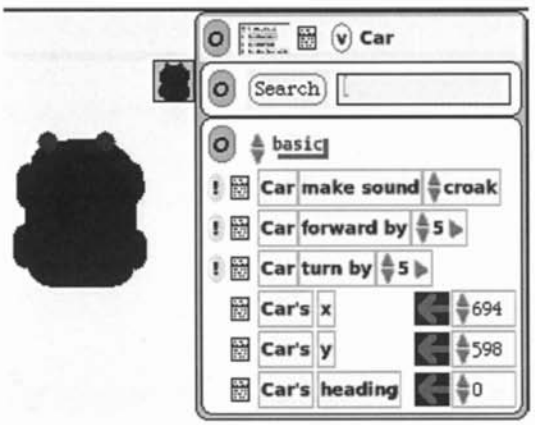

図6　車が実行可能な命令タイル

eToyが持つコーディング環境は,構造エディタを用いている.ユーザはコンパイルエラーに悩まされることなくアルゴリズムの記述に集中することが出来る.eToyはプログラミング未経験者やキーボードに不慣れなユーザでも容易に操作できることから,プログラミングを通した情報処理教育に適している.

Squeakを英語圏以外のユーザも利用できるようにするために,大島らによって多言語化Squeakが開発された[4].多言語化されたことで,Squeak上で平仮名や漢字などの日本語の使用が可能になった.

多言語化Squeakをベースに開発されたのが,日本語Squeak[5]である.図7は日本語Squeakの画面である.日本語Squeakでは,Squeakの

インターフェースで用いられている英語表現を日本語化している．eToy で使用されるタイルの日本語化も行われた．

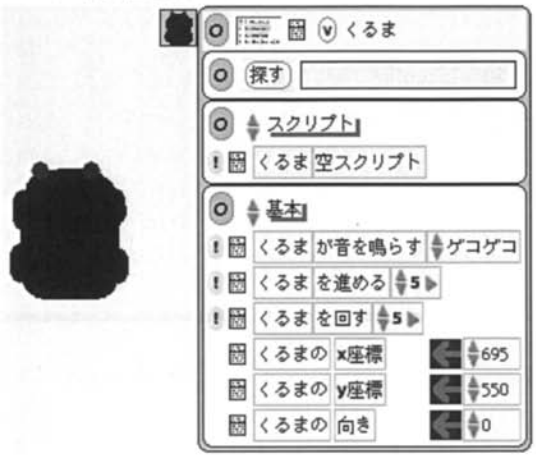

図７　日本語 Squeak

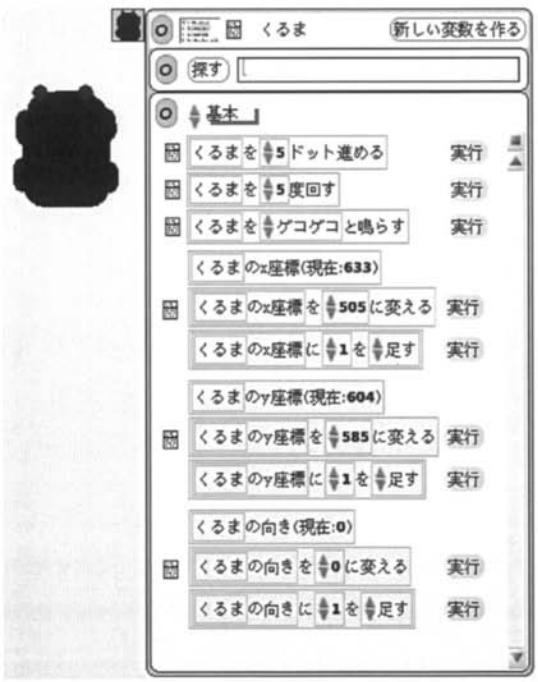

図８　“ことだま on Squeak”

日本語 Squeak の開発で行われた日本語化は，我々の視点からは不十分なものだった．語句は日本語に置き換えたものの構文は英語のままであったため，タイルの表現から意味をくみ取るには推測が必要となり，日本語プログラミング言語の利点は得られないからである．

我々は日本語 Squeak のインターフェースを改造することで，eToy のタイルが日本語として読解でき，３章で述べた日本語プログラミング言語の利点が得られる環境“ことだま on Squeak”を開発した．即ち、日本語 Squeak 環境において，Smalltalk 言語を用いて，日本語 Squeak を改造する形で行われた．図８は“ことだま on Squeak”の画面である．図７と比べるとタイルの表現が日本語として自然に読解できる表現に変わっていることが分かる．

５．「論理思考とプログラミング」の授業

慶応大学湘南藤沢キャンバス（ＳＦＣ）では、2007 年度に行なわれたカリキュラム改訂にともなって、新しく「論理思考とプログラミング」という授業が設けられた。この授業の目的は、

１．論理思考を身につける

２．プログラムが作れるようになる

の二つである。前者は、「問題から解くべき課題を設定して，合理的な方法で課題を解決し，それによって問題が解決できたかを検証できるようになる」こと、後者は、「自分でアルゴリズム（仕事の手順）を考案し，それを他人やコンピュータに伝わるように正確に，完全に記述できるようになる」ことを意味している。

従来のプログラミング教育は、プログラミング言語自体を教えることまでで時間が尽きてしまい、自分の問題を解決するためにプログラムを作り出すことまで扱うことができない状況がほとんどである。このため、一部の才能に恵まれ、プログラム作りに時間をさくことができた学生だけが、自分でプログラムを作るようになるだけで、ほとんどの学生は授業の終了と

ともに、その内容を忘れてしまう状況が続いてきた。

この授業では、前半で”ことだま on Squeak”を用いて制御構造を使いこなす所まで体験させた後、後半では同じことをJavaで行ない、さらにmethodによる抽象化まで行って、複雑なプログラムの構成原理までを体得させる。

前半の教育の最後では、整列問題をあつかう。まず、実際のカードを与えて、選択法による整列を実行させる。その後、実際のカード操作と同じ操作が可能となるデータ構造を用意して、この上で自分のやった作業をアルゴリズムとして記述させる。以上を90分の授業2回分で行なう。さらに、次の2回の授業で、挿入法によるカードの整列作業をビデオで見せた後、これをプログラムとして完成させる。以上の準備のもと、整列の応用として絵辞書を手本として見せてから学生の能力に応じてこれを改造して以下のようなゲームを作らせる。

・タイピングゲーム（難易度：低）
・歴史クイズ（難易度：中）
・記憶君（難易度：高）

ここで絵辞書とは、検索キーワードを入力し，検索ボタンを押すと，絵の入った入れ物を検索して該当する絵を表示する機能を持つものである。また、記憶君はキーワードとそれに対応する内容を登録する機能まで実装する必要があるので、難易度が高い。

このように、単に整列アルゴリズムを理解させるだけでなく、学んだ直後にそれを応用する体験を持たせることで、理解を深め、応用能力までつけることが可能となる。また、日本語プログラミングで基本概念を学んだ後で、実用言語であるJavaを学ぶと、理解が格段に早く進むことが確認できた。

5．おわりに

Squeak環境を用いることで、今までのテキスト形式のプログラミングから具体的に「もの」を動かす環境で教育を行うことになった結果、クラスの雰囲気が単位をとるためだけのために苦行を行っている状況から、クラス全体が楽しい雰囲気に変った。また、配列のような現実にない抽象的な構造ではなく、実際のカード操作と同じ環境を用意することで、「作業を正確に、完全に表現する」というプログラミングの考え方が容易に理解されることが分った。このような授業は、普通高校の「情報B」の時間でも行いうるものである。

参考文献

[1] Jonathan Anderson & Tom van Weert(Ed) "Infromation and Communication Technology in Education - A Curriculum for Schools and Programme of Teacher Development -" UNESCO 2002
http://unesdoc.unesco.org/images/0012/001295/129538e.pdf

[2] http://jasmin.sourceforge.net/

[3] Dan Ingalls, Ted Kaehlei, John Maloney, Scott wallace, and Alan Kay 1997, Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself, Proc. of ACM OOPSLA '97, pp.318
http://www.squeak.org

[4] Yoshiki Ohshima and Kazuhiro Abe 2003, The Design and Implementation of Multilingualized Squeak. Conference on Creating, Connecting and Collaborating through Computing, pp.30-36

[5] http://squeakland.jp/