文系大学生のためのオブジェクト指向プログラミング教育実践

石川高行
北海道大学大学院教育学研究科
〒060-0811 北海道札幌市北区北11条西7丁目
e-mail：top@edu.hokudai.ac.jp

これまでのプログラミング教育は、"hello, world" から始まって変数→関数→構造体→クラスと進んで行く伝統的順序に従うことが多かった。この傾向は、C/C++ 以外の言語においても見られる。しかしこの順序では、オブジェクト指向プログラミングに辿り着くまでの道のりが非常に長いものとなってしまう。そこで、この伝統的順序の有効性を再検討し、GUI を利用した新しい指導順序を考え、それに従って学生向け教科書及び web 教材を作成した上でプログラミング実習を実践した。その実践の内容と実践結果について報告し、またオブジェクト指向プログラミング教育のための言語であるドリトルの方向性と重なる点・異なる点について考察する。

1. はじめに

1.1 プログラミングにおける教育内容の順序構造

大学教員は、自分が教わった（学んだ）順序通りに物事を教えようとする傾向がある。プログラミング教育ではこの傾向がとても強い。例えば、「C++ を学ぶなら、その前に C を学んでおくべきだ」という主張はかなり多い[i]。また、C や C++ を教える際の最も典型的な指導順序は、最初に "hello, world" を表示させ、変数→関数→構造体→クラス、という順序でオブジェクト指向プログラミングに辿り着くものである。この順序は、C, C++ 以外の言語においてもおよそこの通りに教えられている。

しかし、このような順序は本当に妥当であろうか。自分が学んできた順序に縛られて、よりよい指導順序を見逃してはいないか。歴史的に C から C++ へと生まれたのだからその順序で学ぶべきだ、という声も聞かれるが、ヨーロッパにおいて 0 よりも前に小数や分数が使われていたことを考えると、歴史的発生の順序が必ずしも指導順序として適しているわけではないことが分かる。また、「旧来のプログラミング言語は適用分野の観点からでなく、コンピュータの観点での思考を強いるものだったので、こうした新しい方法への移行にかえって困難を感じる人がいるかもしれない」[ii]と言われるように、学習する順序によっては、先に学んだ内容のパラダイムの影響で後から学ぶことが納得しにくいこともありうる[iii]。

この実践は、プログラミングにおける従来の指導順序を疑い、最初からオブジェクト指向言語の特徴を利用する順序で教育内容を組み立てたものである。このように、オブジェクト指向言語の特徴を最初から学ぶように作られたものとして、プログラミング言語ドリトル[iv]などもある。ドリトルは本格的なプログラミング言語ではないが、その実践はオブジェクト指向から導入するプログラミング教育の一成功例であると言えるだろう。

1.2 文法とライブラリ

伝統的なプログラミング教育では、標準のライブラリ以外のライブラリを忌避する傾向がある。そのため、C では `printf` や `gets/scanf`、Delphi (Pascal) では `write` や `read`、というように複雑で貧弱な入出力手段を用いざるを得ない。またそのために、入門教育が Character-based User Interface (以下、CUI と略す) 環境に限定され、プログラミング教育において多くの脱落者を出している。

このように難易度を押し上げてまでも CUI プログラミングにこだわる利点はあるだろうか。MS-Windows がこれだけ普及し、その開発環境の殆ど全てが Graphical User Interface (以下、GUI と略す) 環境となっている現在、GUI によって分かりやすいプログラミング教育を行うことこそが求められてい

---

* A Practice of Object-Oriented Programming for Students Specializing in Liberal Arts
ISHIKAWA Takayuki
Graduate School of Education, Hokkaido University

る。

GUI 環境による分かりやすいプログラミング教育が可能かどうかは、プログラミング上重要な概念や技能が GUI 環境上の教育でも得られるかどうか、また GUI 環境上の教育が CUI 環境上の教育に比べてより短い時間で教えられるかどうか、そして脱落者がどれだけ出るか、などによって評価されるべきだろう。

2. 環境

2.1 Delphi の入手方法・使用方法

ここでは、Borland 社が出しているプログラミング統合開発環境 Delphi の入手方法と使用方法について述べる。

Delphi は、Delphi 6 Personal 版であれば無料で入手することが出来る。

http://www.borland.co.jp/delphi/personal/

この無料の Personal 版は、商用・業務用には使用出来ないが、教育に用いることは可能であり、また Borland も教育用途を目的の 1 つとして無料公開している。この論文で報告している実践は、Borland のウェブサイトでも取り上げられている (2001年度分)。

http://www.borland.co.jp/education/

Delphi を起動すると、「フォーム」と呼ばれるウィンドウが出てくる[図 1]。ここに、入力欄やボタンなどのコンポーネント (部品) を配置し[図 2]、プロパティ (属性値) を設定する[図 3]。

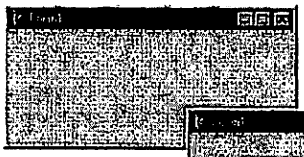


図 1 Delphi 起動時に現れるフォーム

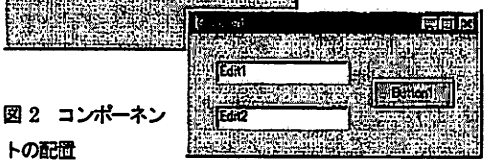


図 2 コンポーネントの配置

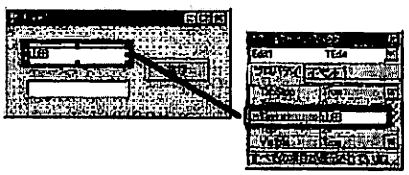


図3 プロパティ (属性値) の設定

ボタンをダブルクリックすると、ボタンが押されたときの動作を書くことが出来る[図 4]。下のプログラム例では、ボタンを押すと下の欄に挨拶が現れる[図 5]。

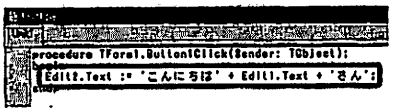


図4 ボタンが押されたときの動作を書く

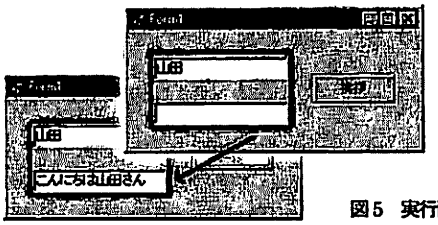


図5 実行画面

2.2 実践の概要・環境など

札幌学院大学社会情報学部の 2 年次学生対象の講義・実習「プログラミング B (2001 年度)」「プログラミング (2002 年度)」では、ビジュアルプログラミング環境である Delphi (Object PASCAL) 及び C++Builder を教えてきた (C++Builder は2002年度のみ)。その実習の実践結果・到達点について報告する。

この社会情報学部は、学部の特徴として「文系、理系を問わないカリキュラムを設置している」[vi]と掲げられていることからわかるように、文系の学生が多数入学してくる[vii]。そのため、理系の学生だけを想定した講義・実習を行うわけにはいかず、必然的に、文系の学生でも充分に理解できる内容を組み立てなければならない。

社会情報学部 2 年次の学生は MS-Windows について既にある程度慣れ親しんでいる。しかし、多くの学生はプログラミングの経験が殆どない。

講義は週に 2 コマ (3 時間)、実習形式で行われた。2001 年度は通年の必修科目 (26 週・8 単位)、2002 年度は半年間の選択科目 (13 週・「プログラミング」「同演習」合わせて 4 単位) であった[viii]。指導員 (院生) や補助員 (3〜4 年生) が履修学生 6〜10 人ずつ担当し、学生の質問に答えたり課題の進捗を確認したりした。

講義・実習内容は全て教科書として学内で印刷・製本されて予め学生に渡され、学生は各自の進度で教科書を読み進み課題を解いていった。「教科書を読めば全てが分かる」という方針で教科書を作ったため、実習は自学自習形態で行われ、学生全体に対する一斉授業は殆ど行われなかった。2001 年度分の目次は下記の通り。

第1章	ビジュアルプログラミング入門
第2章	イベントとイベントハンドラ
第3章	変数と型
第4章	制御命令
第5章	関数と手続き
第6章	ビジュアルプログラミング応用編 ーTimer を活用したプログラム
第7章	CG 入門
第8章	複数のフォームを持つプログラム ー処理の分割ー
第9章	フォームの複製
第10章	継承
第11章	クラス
第A章	ActiveX の取り込み
第B章	文字列操作の基本
第C章	ファイル入出力と XML
第D章	ネットワークプログラミングの初歩
第E章	(データベース・アプリケーション…未完のため欠章)
第F章	画像処理ー色の処理
第G章	(CG 応用編…未完のため欠章)
第H章	C++Builder への移行

第1章から第11章までは、開発環境 (Delphi) の使い方からオブジェクト指向まで、プログラミングの基礎を学ぶ内容である。一方、第A章から第H章までは、『応用編』という別テキストとして配布された。これは、進度が速い学生がより多くのことを学べるように書かれたものである。

教科書の各章はいくつかの節から成り立っており、それぞれの節では説明文の中に基礎課題や応用課題が入っている。基礎課題は、第11章第6節までに全103問が用意され、学生には年度末までに全ての基礎課題を終えることが課されていた。応用課題は、単位取得の条件ではないが、1問解く毎に試験の得点に加算されることとなっていた。第11章7節以降及び『応用編』は、全て応用課題扱いであった。

3. 教科書の特徴

この教科書の最大の特徴は、変数・関数・クラスといったプログラム上の重要な概念を、文字入力欄・ボタン・フォーム (ウィンドウ) というシェーマ[注]によって視覚的に提示している、という点である。また、それ以外にも提示方法を工夫した点がある。以下、それぞれの点を具体的に示す。

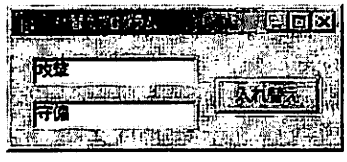
3.1 変数は入力欄で教える

変数を表すシェーマには、文字入力欄 (エディット、テキストボックス) を用いた。文字入力欄を用いることによって、

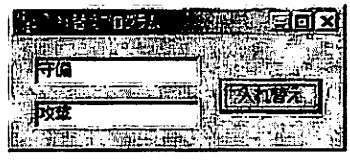
- 文字入力欄自体が入れ物の外見 (長方形に囲まれた領域) を持っているため、「値を保持する」という機能が視覚的にわかりやすい
- read (Pascal) や scanf (C) といった入出力命令・関数を用いなくても、簡単に値を代入することが出来る
- その値をいつでも容易に確認・変更できる

という恩恵が受けられる。

教科書の内容を具体的に紹介していこう。

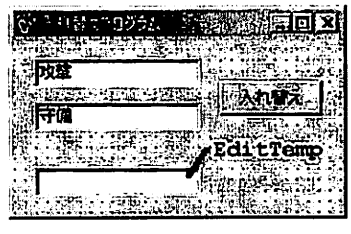


という画面で「入れ替え」ボタンを押すと上のエディットの Text プロパティ (攻撃) と下のエディットの Text プロパティ (守備) が入れ替わる、というプログラムを作りましょう。



〔中略〕

うまく入れ替えを行うためには、Edit2.Text を取り敢えず保存しておく「エディット」が必要です。



〔中略〕

EditTemp の Visible プロパティを False にしても正しく動くでしょうか?

〔中略〕

Visible を False にすると、EditTemp は画面から消えてしまうので、Height や Width, Color などのプロパティにはどんな値を入れても意味がないですね。入れ替えに必要なのは、Text プロパティだけなのです。中継用エディットを使わない、もっとよい方法はないのでしょうか。

ここで、新しい方法を試してみます。まずは、EditTemp を削除してください。EditTemp を削除したので、EditTemp を中継欄に使うことはできなくな

りました。そこで、今消した EditTemp のかわりにもっと単純な中継欄を作ります。procedure と begin の間に、次のように付け加えてください。

```
procedure TForm1.ButtonSwapClick(Sender: TObject);
var
  temp: String;
begin
  EditTemp.Text := Edit1.Text;
  Edit1.Text := Edit2.Text;
  Edit2.Text := EditTemp.Text;
end;
```

EditTemp はなくなりましたが、これで temp という入れ物ができ、temp が EditTemp.Text のかわりをします。EditTemp.Text は String 型だったので、そのかわりとなる temp も String 型でなくてはなりません。そのため「temp: String;」と書きます。

このように、文字入力欄（この場合は EditTemp）を退化させて変数の概念を導入している。こうした導入によって、学生が変数（又は文字入力欄）を宣言（用意）できずに困る場面は見受けられなくなった。理解が遅い学生でも、変数の代わりにとりあえず文字入力欄を配置しておけばプログラミングが進むからである。必要に応じて文字入力欄を単なる変数に退化するよう指導してやれば、次第にわざわざ文字入力欄を配置してから退化させる手間を惜しんで最初から変数を宣言するようになる。

この文字入力欄は、文字列型変数を表すシェーマである。同様に、整数型変数は数値入力欄（TSpinEdit）、ブール型変数はチェックボックスをシェーマとして用いた。

3.2 関数はボタンで教える

関数を表すシェーマには、ボタンを用いた。クリックすると何らかの処理が行われる、という点が関数を表すのに適していると考えたからである。

文字入力欄が変数を表しているのと同様にボタンが関数を表しているので、学生は必要な関数の数だけボタンを配置するようになる。また、最終的にはボタンを単なる関数に退化させて見せるので、わざわざボタンを配置してから退化させるという過程を面倒に感じた学生は最初から単なる関数を作るようになっていった。

3.3 クラスはフォームで教える

クラスは、簡単に言えばフィールド（メンバ変数）とメソッド（メンバ関数）から成り立っている。そのため、この教科書では、文字入力欄（フィールドにあたる）とボタン（メソッドにあたる）を自由に配置できるものとしてフォームを用い、クラスの概念を獲得させるようにした。具体的には、次のようにフォーム上に文字入力欄やボタンなどを自由に配置させ、最終的にはこれを（フォームではない）単なるクラスへと退化させていった[図6]。

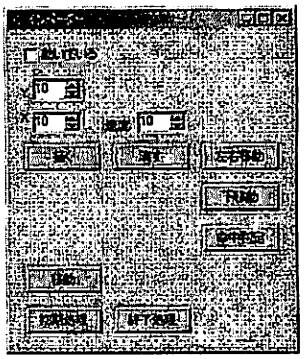


図6 クラスのシェーマであるフォーム

また、フォームは、変数のように複数のインスタンス（オブジェクト）を宣言してやれば、実行時にはそのフォームが複数出現する。こうして、「クラスは型である」という認識が学生に形成される。

さらに、継承の本質である、基底クラスの機能を引き継ぎつつ新しい機能を追加する差分プログラミングという点も、このやり方によって視覚的に分かりやすく提示することが出来る。

3.4 全ての要素を含む典型から退化

プログラミングの概念の視覚的提示とは別の点であるが、if 文など、決まった形がある構文は、より典型的な形（全ての要素を含む形）を最初に提示することにした[図7]。

```
procedure TForm1.ButtonDia
begin
  if ((SpinEditSum.Value >
  then
    begin
      CheckBoxHigh.Check
      EditMessage.Text :
    end
  else
    begin
      CheckBoxLow.Checke
      EditMessage.Text :
    end
  ;
end;
```

図7 典型的な if 文

この例は、

- else を含む（この例を先に提示し、その後「else の部分は省略可」と教えた）
- then の処理も else の処理も begin と end で括られている（then の処理も else の処理も一般に複数行であり、begin と end が省略できるのは特殊な場合に過ぎない、と教えた）

という点で if 文の典型である。

3.5 水道方式の応用

ここまでに述べた「典型を退化させる」という順序は、数学教育で一定の成果を挙げた水道方式[x]の考え方を元にしている。

例えば、変数のシェーマとして用いた文字入力欄は、実際には変数よりもはるかに複雑なオブジェクトである。変数と同様に「値を保持する」という機能を持つ他にも、色や大きさを設定できたり、フォーカスを受け取ったりし、単なる変数よりも扱いやすい。これを変数の典型として最初に導入し後から変数に退化させることは、ただ変数の概念を学ぶだけではなく、オブジェクトに関する漠然とした認識も同時に形成し、後にオブジェクトを学ぶ際の敷居を低くすることになる。

また、if 文で全ての要素 (else, begin と end) を含む典型を最初に見せることも、学生の学習負担の軽減に寄与する。

4. 実践結果

4.1 学生の到達度

4.1.1 記録したもの

実習では、各学生について

- 何時間出席・遅刻・早退したか (毎週)
- 何時間自習してきたか (毎週)
- どの課題をどの週に終えたか

という記録を残した。この記録から、この教科書の必修部分 (第 11 章 6 節まで、課題数 103) を終えるまでに各学生が自習時間を含め何時間を要したか、を算出することが出来る。

4.1.2 所要時間・遅い学生

この 2001 年度「プログラミング B」は 43 人の学生が履修し 38 人の学生が単位を取得した。単位を取得できなかった 5 人の学生は、途中で退学したか、1 年間の実習で 0〜3 回しか出席しなかった学生である。即ち、単位を取得する意思がある学生は例外なく単位を取得した、ということである。

単位の取得には、「基礎課題」と呼ばれる必修課題を全て終わらせることが必須である。38 人の単位取得者中 36 人は、実習時間内 (最後の実習の終了時刻まで) に必修課題を終えた。その平均所要時間 (実習参加時間と自習時間の合計) は 59.0 時間であり、その標準偏差は 11.2 時間である。最も短い (速い) 学生は 28 時間で必修課題を終え、最も時間がかかった学生は 87 時間である。これは、この実習の時間が 3 時間×26 週=78 時間であることを考えると充分に短い時間である。

半年間の実習となった 2002 年度「プログラミング」では、203 人の学生が履修し、184 人の学生が単位を取得した。教科書の内容は 2001 年度とほぼ同じのまま、第 6 章までが必須とされた。自習を含む平均所要時間は 36.8 時間、標準偏差は 9.2 時間であった[xi]。最も速い学生は 6 時間で必修の範囲を終え[xii]、最も時間がかかった学生は 62.5 時間であった。

4.1.3 速い学生

2001 年度は、必修課題を早く終えた学生のために、応用編である第 A 章〜第 H 章には様々な応用課題を用意した。

- 簡易ウェブブラウザ (第 A 章) を作った学生…13 名
- 暗号化・復号プログラム (第 B 章) を作った学生…6 名
- XML による名簿検索プログラム (第 C 章) を作った学生…2 名
- POP3 メーラ (第 D 章) を作った学生…1 名
- 任意の 2 色を任意の比率で混ぜるプログラム (第 F 章) を作った学生…2 名

実用的なプログラムを自分で作る力も一部の学生には備わったものと思われる。

2002 年度は、必修課題を早く終えた学生は第 7 章以降へと進んでいった。

5. 議論と考察

5.1 ドリトルと比較して

ここでは、ドリトルの設計思想[xiii]とこの実践の方向性とを比較・考察する。

BASIC や LOGO などの古い言語を用いてプログラミングを体験しても現代のソフトウェアシステムとはかけ離れている、という指摘[xiv]はその通りである。オブジェクト指向言語を用いてプログラミング教育を行うことは、すでに時代の要請であると言えるだろう。

しかし、ドリトルを用いて実用プログラムを作ることは難しく、学生・生徒が日常接しているソフトウェア製品とはまだ隔たりがあるだろう。だがこれは、初中等教育での利用を念頭において簡明な言語を設計するために犠牲になった部分であると言え、大学生を対

象とする本実践で用いた Delphi とは差異があることは必然である。

またドリトルは、敷居の低さのためにクラス方式ではなくプロトタイプ方式としたことを特徴として挙げている。確かに、従来のオブジェクト指向言語の指導ではクラス方式の諸概念は獲得しにくいものであるが、この実践では

- フォームがクラスを、入力欄が変数・プロパティを、ボタンがメソッドを表す
- 設計画面がクラスに、実行画面がインスタンスに対応

という視覚的提示によってクラス方式でも充分に分かりやすい指導が可能であることを示したと考えている。

5.2 実践の評価基準

実践が成功したかどうかは、

- 履修する気がある学生すべてが一定の地点まで到達したかどうか
- 進度が速い学生を退屈させずにより多くのことを学んだかどうか

によって判断されるべきであろう。

5.3 講義形態の問題点

この実習では、「全ての必修課題を終えたら残りの実習を欠席しても出席扱いにする」ものとした。この取り決めは、「さっさと必修課題を終わらせてしまおう」という動機を強く呼び起こすこととなった。その一方で、必修課題を全て終えてしまった学生にとっては以後の授業は出席しなくてもよいものとなり、1年間の終わり頃には出席率が低下した。このような取り決めは、学習動機を強める動機にも実習を休む理由にもなる諸刃の剣である。

しかし、別の見方をすれば、このような取り決めがあるにも関わらず応用編へ突入した学生が全体の 1/3 以上いたことは、このような応用的な分野への興味を持っている学生が少なくないことを示していると言えるだろう。

6. おわりに

「大学生だからこれくらい分かるだろう」という理由で今まで軽視されてきたこれまでの大学でのプログラミング教育において、オブジェクト指向をしっかり教えるための実践を行った。今後は、その先の専門領域との結びつきをさらに考察していきたい。

このショートペーパーの作成にあたって、金沢工業大学の江見圭司氏に多大なる助言を戴いた。ここに感謝の意を表明する。

---

[i] 「C++ 言語は、C 言語と互換性を保ちながら拡張された言語で、オブジェクト指向プログラミングに必要な機能を追加したものなのです。したがって、C++ 言語を学習するためには、C 言語の知識は欠かせません。まず、C 言語から習得しなければならないのです。」蒲地輝尚『はじめて読む C 言語』(ASCII, 1991 年)

[ii] J.ランボー/M.ブラハ/W.プレメラニ/F.エディ/W.ローレンセン著、羽生田栄一監訳『オブジェクト指向方法論 OMT モデル化と設計』p.iii(トッパン, 1992 年) 原著: James Rumbaugh "Object-Oriented Modeling and Design" Prentice Hall, Inc., 1991

[iii] オブジェクト指向言語が生まれる前と後では、プログラマーの思考も大きく異なる。"Before object-oriented languages appeared, programming languages (such as FORTRAN, Pascal, Basic and C) were focussed on actions (verbs) rather than on things or objects (nouns)." H. M. Deitel & P. J. Deitel "C++ HOW TO PROGRAM" 4th edition, Pearson Education, Inc., Upper Saddle River, New Jersey, 2003, p.14

[iv] http://www.logob.com/dolittle/

[v] 一時期は INPRISE という社名だったこともある。

[vi] http://jupiter.sgu.ac.jp/soc/gakubu/tokutyou.html

[vii] 一般入試の教科も、社会か数学から 1 つ選択、国語、英語、というように文系型である。

[viii] 選択科目であっても、大部分の学生が履修した。それまで「プログラミング A」「プログラミング B」と 2 つに分かれていた実習が 2002 年度は「プログラミング」に統一された。履修者は 203 人である。

[ix] ここでは、概念の本質を表す具体物をシェーマと呼んでいる。ピアジェが用いた「シェーマ」という言葉とは無関係である。

[x] 遠山啓を中心にして 1958 年に生まれた、筆算における指導順序の体系。遠山啓, 銀林浩編『水道方式入門 整数編』(国土社, 1992 年) などに詳しい。

[xi] 単位取得者 184 人中 8 人分は担当指導員の記録不足により正確な数値が分からなかったため、残りの 176 人から平均と標準偏差を求めた。

[xii] この学生は現職プログラマでもあったため、僅か 6 時間で課題を終えた。この学生の次に速い学生の所要時間は 11.5 時間であった。

[xiii] 兼宗進, 中谷多哉子, 御手洗理英, 福井眞吾, 久野靖「学校教育用オブジェクト指向言語『ドリトル』の設計と実装」情報処理学会論文誌, Vol. 42, No. SIG11(PRO12), 2001 より。

[xiv] 同上、pp.78-79 より。