研究ノート

プログラミング教育の手法

Methodology for Programming Education

中　村　有　一*
Yuichi NAKAMURA

キーワード：プログラミング教育、プログラミング言語、Java
Keywords：programming education, programming language, Java

1. はじめに

これまで大学においてプログラミング教育に携わってきた経験を踏まえて、多摩大学開学当初（1989 年）から現在（2022 年）までの、プログラミングを取り巻く環境について、簡単にまとめておきたい。またその情報から、プログラミング教育の手法とその問題点、今後の在り方について考察していきたい。

2. プログラミング環境の変化

まず約 33 年間にわたるプログラミング環境の変化を、関連する事項とともに概観する。そのために全期間を下のように大きく 5 つに切り分けることにする。

① 第 1 期　1989 年から　PC-9801　MS-DOS の時代

最初の 5 年間は、NEC のパソコン PC-9801 シリーズのデスクトップ機を数十台導入し、その後 EPSON のラップトップ機を数十台追加購入してコンピュータ環境を整えた。これは貸出できるようにラックに収納する方式である。

② 第 2 期　1995 年から　MacOS の時代 「情報基礎」の導入

その後、Macintosh のデスクトップ機に更新するとともに、ネットワーク環境、インターネット環境を整える。この当時まだ Windows 95 は発売されていなくて、実用的に使える環境は、MacOS のほうが有利であった。このころ PC 互換機（DOS/V マシン）は、まだ普及していなかったが、将来のことを考え、10 台程度導入して、Windows 95 を使えるようにしたが、授業で主流として使うわけではなかった。まだこの当時はパソコンを持ち歩けるような時代ではなかったので、放課後 8 時までコンピュータ教室を開放し、自習できるようにすることも並行して行った。その後 iMac に更新した。

③ 第 3 期　2001 年から　ノート PC　Windows OS の時代

* 多摩大学経営情報学部　School of Management and Information Sciences, Tama University

次に大きな変化となるのは、ノートPCを学生に配布して、一人1台いつでも使えるようにするという段階である。まだ結構重量があり、持ち歩くには不便であったが、本当の意味でパソコンになったという意味で大きな前進であったと思われる。OSはWindows系のOSに限定していた。

その後、PCは高性能化、軽量化が進んでいく。すべて同じ機種を配布したほうが、保守の手間とかが軽減されるなどのメリットはある。しかし人によっては機能的に不満が出てくるという点が問題になってくる。これに対応し、だいたいのスペックは指定するが、個人の指向に合わせて各自が購入する方式に変更される。ただMacOSには十分対応できないという問題も出てきた。この時期にインターネット化が進行し、クラウドを使ったシステムが一般化していく。例えば、Gmailなどである。

④ 第4期　2020年から　コロナ禍の時代　リモート化・デジタル化

そして2020年、大きな変革期が訪れる。コロナウィルスの流行である。これによってコンピュータの使い方に大きな変化が起こることになる。特に大きな変化は、リモート授業など、遠隔地とインターネットで繋ぎ、会議や授業などを行うことが一般化したことである。この傾向は、コロナ禍が収束に向かいつつある現在でも、変わることなくかなり定着するものと考えられる。この時期はクラウド化が加速する時期とも重なる。紙による手続きが少なくなり、レストランの注文にもタブレット端末が多く使われるようになる。スーパーのレジにも無人レジが導入されるなど、情報化が大きく進展した時期でもある。

⑤ 第5期　2023年から　アフターコロナの時代

そして来年（2023年）からは、アフターコロナの時代になり、まだ予想の段階であるが、コロナ前の時代に戻ることはなく、一層デジタル化が進むものと思われる。

3. プログラミング言語の移り変わり

上で述べたPC環境の変化に伴い、プログラミング教育も大きく変化してきた。これを上の第1期から第5期についてみてみよう。

① 第1期

この時代は、MS-DOS上のTurbo Pascalが使われていた[(1)]。まだGUI以前の時代なので、プログラミングは単純であり、試行錯誤の段階と言えるだろう。まだ世間一般にプログラミングに対する認知度が低く、学生の感じるギャップも大きかったように思われる。

② 第2期

「情報基礎」が1年次に必修科目として導入される。またこれに伴い、大学の主流となるPCがMacintoshに置き換わることになる。プログラミングの授業は、1年次の入門にあたる部分と2年次以降の本格的な部分に分離される。

当時入門の部分は、MacOSの基本的なGUIプログラミング環境であるHyperCardを利用することにした。このプログラミング言語はHyper Talkと呼ばれる[(2)]。その後REAL basicという汎用のビジュアルなプログラミング言語に置き換えた[(3)]。

2年次以降の本格的なプログラミングには、MacOS上で動くCodeWarrior（Metro Works社）を利用する。プログラミング言語としては、C言語を採用した[(4)]。このころからC言語の教科書作成に取り掛かる[(5)]。これはMacOS、Windows OS、UNIXに対応したもので、CUIの範

囲をカバーするものである。

③ 第3期

Windows OS上で動くC言語を中心に、プログラミングの授業を行う。言語としてはMicrosoft社のVisual Studioの中のC++を使用した。ただしC言語の範囲でしか使用していない。この時期の途中で主要なプログラミング言語をC言語からJavaに移行した。JavaはC言語を拡張したC++の問題点を整理した言語で、基本的にはC言語と同様である。教科書を新たに作ることはやめ、PowerPointで授業資料を作成した。内容的には、ほぼ以前のものを踏襲している。

④ 第4期

コロナ禍によりリモート授業への対応などが大変だったが、デジタル化、ペーパーレス化などが大きく進んだ。

⑤ 第5期

2023年度からのカリキュラム改定に伴い、プログラミング言語の変更を考えている。「システム開発系」と「データ分析系」に分野を分けて、主要な言語も変更する。具体的には以下のようになる。

システム開発系・・・C＃（C言語系、Javaと同等な言語）

データ分析系・・・・Python

Pythonは2020年頃から特に流行してきた言語で、ビッグデータ、人工知能などの分野でよくつかわれている。

4. 主流のプログラミング言語以外の言語

主流となるプログラミング言語としては、Pascalから始まり、C言語、Javaと変遷してきた。ここではこれ以外に授業で使われたプログラミング言語についても取り上げてみたい。

まず2001年～2005年ごろに、Visual Basicを使って「プログラミング基礎」という科目を実施した。この科目は最初にプログラミングに触れる初心者を対象に、基本的な内容について興味を持てるような形で提供することが目標である。このためGUIプログラムや、グラフィクス、イベント処理などの要素も一部取り上げている。レポートの書き方や流れ図の描き方まで含んでいたので、少し欲張りすぎていたかもしれない。Visual Basicを採用したのは、比較的簡単にビジュアルなプログラムを作ることができ、また必要ない細部がかくされており、初心者向けである点を配慮したためである。

また単に簡単な例題を実行するという単調なものから、原型となるゲームのプログラムを与えて、それを動かしながら改造するという課題にも取り組んだ。取り上げたゲームはマインスイーパーというゲームを参考に独自に作ったもので、Visual Basicで記述されたプログラムを、実際に動かすことから出発し、考察を深めながら、最終的には改造したゲームに仕上げることにした。またグループ作業の経験をしてもらうことも目的の一つである[(6)、(7)、(8)]。

上と同じ時期に始めたもう一つの試みは、「プログラミング言語III」という上級者向けの科目を始めたことである。これは「入門者→中級者→上級者」というながれの第3段階にあたるものである。プログラミング言語は、「Visual Basic → C言語 → Java」と推移していく。このように設定した理由は、オブジェクト指向をちゃんとやろうとしたからである。C++とい

うC言語を拡張した言語があり、これを使うのが当時は一般的であったが、Java のほうが、煩雑化した C++ よりも文法が整理されており、オブジェクト指向の学習には向いていると判断したためである。教科書のみでは難しいと思われたので、オブジェクト指向に関する解説資料を別に用意した[(9)、(10)]。

オブジェクト指向の考え方の理解に重点が置かれたが、実際にはかなり難しかったようである。この経験があったので、数年後にC言語に代わって、Java を採用したときに、あまりオブジェクト指向にはこだわらず、C言語を改良したものとして Java を使うことにした。

5. C言語から Java への移行

2016 年以降、それまで主要な言語として位置付けていたC言語を、Java に置き換えることにした。低レベルな（ハードウェアに近い）プログラムにおいては、依然としてC言語が主流であるが、エンドユーザに近い分野では、Java のほうが優勢であること、需要も高いことなどを配慮してのことである。例えばスマートフォンのアプリの開発などでは Java が主流である。

内容は前述のようにオブジェクト指向にはこだわらずC言語の代替案の一つとして採用した。またペーパーレス化を目指して、講義資料は PowerPoint の資料の形で作ってある[(11)]。それまでは演習課題など、まだ紙での提出を行っていたが、コロナ禍の起こった 2020 年以降、ほぼ完全なペーパーレス化を実現した。これにより紙のプリントを管理して、授業に出なかった人に後日配布するなどの手間がなくなると同時に、スマホなどでも読めるため紙の資料を持ち歩く手間がなくなった。ただし試験については、やはり対面でペーパーテストを行う必要性があることも認識された。

またアクティブラーニングの一つとして、クイズ形式の簡単な問題を講義の途中に入れることにより、授業が単調にならないように工夫した。クイズは Google Form を使って、即時に答えが集計できるため、説明したことがどの程度理解できているか、把握する効果も期待できる。

6. 情報系科目でのプログラミング言語の利用

これまで取り上げてきたプログラミング系の科目以外でのプログラミング言語の利用についてまとめておこう。

①「システムデザイン」2013 年から

これは以前担当していた科目で、現在はなくなっている。これはシステム設計の基本を概論的に学ぶものである[(12)]。この中でオブジェクト指向の設計手法を取り上げたが、かなり抽象的で難解だったと思われる。そのときに Java を使って説明したが、実際にプログラミングを行い、動かすところまではいかなかった。この経験から、あまり中途半端に取り上げるべきではないことを痛感した。

②「コンピュータサイエンス」2013 年から

アルゴリズムの原理を理解することが主眼となる科目である。学生が自分でプログラミングをする必要はないが、プログラムを読んで理解することが求められる。現在はなくなってい

るが、新カリキュラムでは、上級者向けのプログラミングコースとして復活させる予定である。この科目では最初C言語を主に使っていたが、途中でJavascriptに切り替えることにした。これは実際に動作させるだけであれば、Javascriptのほうが簡単だからである。

③「コンピュータ概論」2016年から

主にコンピュータの基本原理などを扱う初心者向けの科目である。ハードウェアが中心であったが、興味を持ってもらうために、数年前からプログラミングの初歩を取り上げるようになった。プログラミング言語としては、最初は小学校とかでも使われているScratchという開発環境を使って、ゲームを改造する課題に挑戦した。Scratchは、従来のプログラミング言語とは違い、おもちゃのLEGOブロックのようにプログラムの部品を視覚的に組み立てていく仕組みになっている。これによりプログラムの自由度が制約されるため初心者でも間違いが少なくなる効果がある。その後この目的には、Javascriptを使うことにした。Scratchは他の科目でも使っていること、Javascriptは、開発環境を準備するのが容易であることなどが理由である。

7. まとめと展望

これまで大学での授業で取り組んできたプログラミング言語について述べてきたが、実際にやってみて明らかになった問題点や課題をいくつかの視点から取り上げてみたい。

① プログラミング環境の複雑化

GUIが一般化したことによりプログラミング環境が複雑化し、初歩的な段階で作れるプログラムと実用的に使われているプログラムの間に大きなギャップができてしまった。このためプログラミングの演習が魅力のないものに感じられるという問題が出てきた。この対策として、初心者向けのコースでは、Visual Basicを使ったゲーム作成を課題とし、できるだけ簡単に面白いものが作れるという体験をすることを目指した。ある程度この体験が生かされれば、単調とも思われる本来のプログラミングの学習にも効果が期待できるかもしれない。

またこれに関連して開発環境のインストールおよび環境設定が難しくなり、最初の大きな障壁となってきた。オープンソースのツールを組み合わせることも多くなり、バージョンの違いでうまく動かないなどの問題も出てくる。これを避けるために、クラウド環境を利用してプログラミングを行う手法も一般的になりつつある。これは初心者には向いているが、ある程度の段階で限界になるものと思われる。インストールの仕方を学ぶことも重要なスキルと考えて、オフラインでの開発環境構築を目指すことも重要であろう。

② 積み上げ問題

次にプログラミングの学習においては、どうしても積み上げが必要となる。ただ基礎的なことを延々とやっていては、到達点が見えなくなり飽きてしまうという問題が危惧される。このため積み上げの段数を減らすことも必要になってきた。それまで3段階あった主要なプログラミング系の授業は、2段階あるいは1段階に整理され、見通しを良くしたのである。第1段階は「プログラミング基礎」の授業で、Visual Basicを使って、ビジュアルなプログラムを体験してもらうことに重点を置いた。その後、第2段階でC言語による汎用的なプログラミングを学習し、第3段階においては、Javaによるオブジェクト指向プログラミングを目指した。

またプログラミングの学習においては、どうしても個人差が出てきてしまうので、足踏みを

して先に進めない人が出てこないように、上級者向けの課題を出す一方で、躓いている人をフォローするという 2 正面作戦が必要となる。

③ オブジェクト指向プログラミングへの取り組み

実際にやってみた印象では、オブジェクト指向の考え方は理解できない、必要性がわからないなどの意見が多く、ある意味そこまでする必要性がないのではないかという感想を持った。プログラミングの世界は世の中の流行に左右されることが多い。オブジェクト指向もその流行の一つであったのかもしれない。流行からある程度さめて冷静に考えると、必ずしもすべての領域においてオブジェクト指向が有効であるとは限らないということである。

現段階では、どちらかというとオブジェクト指向でゼロから独自に設計し、プログラミングも行うという方式は主流ではなく、フレームワークを使ったひな型をカスタマイズして目的のプログラムに仕上げていくという方法がよく使われる。実際にインターネットの普及に伴って、実用的に使われるプログラムは、似たような構造をしており、ひな型を改造したほうが、一から作るよりも簡単で、安定した製品が作れるということが実感されるようになった。この契機となったのが、Ruby on Rails の出現である[13]。これは Ruby という言語をベースにしたフレームワークで、ひな型となるプログラムを作るシステムである。Ruby 自体は以前からあり、主にインターネットのサーバ上で動くプログラム（CGI など）をつくるために利用されていた。

④ スマホアプリの開発

最近の動向としてスマホのアプリ開発の需要は無視できないものとなっている。それほど簡単ではないが、需要が高いので、重要な目標として意識しておく必要があるだろう。このため 2021 年あたりから、スマホアプリ開発を、プログラミング教育のひとつの着地点として定めることにした。具体的には「プログラミング入門 II」という授業で行っている内容を、こちらの方向に軌道修正することにした。それまでは特にその後の目標については示していなかったが、せっかく Java を学んでいるのだから、それがさらに発展的に生かせるような方向を模索したのである。

それまで NetBeans という統合開発環境を使っていた。NetBeans は、Java 本体の開発グループが手掛けた Java で書かれた標準的な開発環境である。また比較的初心者向けで癖のないシステムであるために採用した。2021 年度からは、この統合開発環境を IntelliJ IDEA に置き換えることにした。

Eclipse は Java 用に開発されているものではなく、他の言語にも対応した統合開発環境である。かなり高度な機能をもっているが、オープンソースで開発されており、混乱が残っているなどの弱点もある。IntelliJ IDEA は、最近注目されている開発環境で、ハンガリーの JetBrains 社が開発している。会社組織で開発を行っているので、統制がとれているなどの利点がある。また Google に買収され、Android アプリの標準開発環境として採用されるなど、近年利用者が急増している。

NetBeans から IntelliJ IDEA に軌道修正することにより Android スマホのアプリ開発への距離が短くなった。Android アプリの開発環境は、Android Studio が標準であるが、これは IntelliJ IDEA と同じ会社が作っており、操作性が共通している。一般向けの授業ではスマホアプリの開発にまでは進めないが、ホームゼミとかでは、さらに発展させることが可能であろう。

⑤ 開発環境の統合性

NetBeans においては、GUI プログラミングを行う際、画面の設計とプログラミングの分離

が不十分であり、煩雑であった。この点でIntelliJ IDEAはできるだけ不要なプログラムはかくして重要な部分だけを書き換えるようになっている。さらにこの画面設計の部分が発展して、画面設計の部分とそれを動作させるプログラムが絡み合ったような状態をすっきりさせることができるようになった。JavaFXとScene Builderという画面設計ツールである。ただこのように別々のツールを組み合わせて連動させることは、統合性の面では不利になる。

2023年度以降のカリキュラムで、主要言語のJavaからC＃への移行が計画されている。C＃はMicrosoft社により作られた言語であるが、標準化されており、必ずしも閉鎖的な言語ではない。C言語がベースになっていることは、Javaと同様であるが、C++、Javaなどの先行する言語の問題点を改良し、より使いやすくしている。またVisual BasicからのGUIライブラリを引き継ぎ、いろんなツールを組み合わせなくても、GUI環境がシームレスに実現できるという利点がある。

⑥ グループ作業環境の課題

最近のコロナ禍のために中止せざるを得なかったプログラミング手法について述べておきたい。ペアプログラミングとモブプログラミングという手法である。

「ペアプログラミング」

2人一組で一緒のプログラムに取り組むという方法である。これは実際に意識しないでやっていることが多い。プログラミングにおいては行き詰ったとき一人で悩むことが多いが、2人でやっていると、意外なところで突破口が開かれる経験をしたことがあるかも知れない。このようなやり方を、意識的行いプログラミングの効率を高める手法である。

「モブプログラミング」

ペアプログラミングの考え方をさらに発展させたものが、モブプログラミングである。3人以上のグループで、同一のプログラムに取り組む手法である。メンバーには以下のような役割分担が決まっている。

・ドライバ（タイピスト：交代で務める）

・ナビゲータ（コードをみて意見を言う）

IntelliJ IDEAにはこれらの手法に対応する機能がある。以上の手法はアクティブラーニングの一環として試してみようとしていたが、運悪くコロナ禍の時期と重なってしまい実際にはまだ試していない。今後の課題である。

8. おわりに

これまでの経験をもとに大学におけるプログラミング教育の問題点と課題を整理してきた。その時々の状況に振り回されることが多い分野であるが、基本的なことをしっかり押さえて、右往左往しないことが望まれる。

参考文献

(1) 小林侔史「Turbo Pascal Ver 5.5 トレーニングマニュアル」JICC出版局、1990年4月（教科書）

(2) 「Hyper Talkによるプログラミング入門」1995年～1997年（自作教科書）

(3) 「REAL basic プログラミング入門」2000年5月（自作教科書）

(4)　椋田實「(改訂第 2 版) はじめての C」技術評論社、1988 年 (教科書、現在「改訂第 5 版」まで出ている)
(5)　「C 言語プログラミング入門」2000 年 4 月 (自作教科書)
(6)　「プログラミング基礎 – Visual Basic によるプログラミング入門 --」2001 年～ 2022 年 (講義資料)
(7)　「Visual Basic.NET プログラミング入門」2004 年～ 2005 年 (自作教科書)
(8)　「宝探しゲームを作ろう」2007 年 7 月 (講義資料)
(9)　神吉達郎「Java プログラミング入門」オーム社、1996 年 8 月 (教科書)
(10)「Java によるオブジェクト指向プログラミング入門」1999 年～ 2001 年 (講義資料)
(11)「Java プログラミング入門」(講義資料 PowerPoint 形式)
(12) 松永俊雄、中村太一、亀田弘之「コンピュータシステム開発入門」オーム社、2008 年 9 月 (教科書)
(13) Bruce A. Tate 著、角谷信太郎訳「Java から Ruby へ:マネージャのための実践移行ガイド」オライリー・ジャパン、2007 年 4 月