クオータ制での大学の一般プログラミング教育の試行

布施　泉 [1)], 平林 義治 [1)], 山本 裕一 [1)]

1) 北海道大学 情報基盤センター

ifuse@iic.hokudai.ac.jp

Trial of a programming-education as a general education of university in academic quarter system

Izumi Fuse [1)], Yoshiharu Hirabayashi [1)], Yuichi Yamamoto [1)]

1) Information Initiative Center, Hokkaido University

概要

例年半期 15 週で行っていた一般教育としてのプログラミングの授業をクオータ制（8 週×2）にて行った。単位数の変更などもあり，動画資料などを準備し学生に自習を促しながら授業を行った。秋学期の履修者が大きく増えた結果，初学者を中心とした教室における学生数が増え，支援が前年度までに比べ手薄になった。その影響もあるためか，学生の理解度は総じて高くなく，逐次確認ミニテスト等を行い，学生が自身の理解の度合いについて把握できるようにすることが必要であることが明らかになった。

1　はじめに

著者らは北海道大学の全学教育（一般教育）において，初学者を念頭においたプログラミングの授業を開講している。初回のガイダンス以外は，PC 教室で授業を行い，履修者は課題に則って実際にプログラムを作り，実行することを通じてプログラミングを学んでいく。

北海道大学の全学教育科目にはいくつかのカテゴリがあるが，著者らの開講している科目は，総合科目として位置付けられるものである。この科目は，2016 年度までは半期 15 週の授業で 1 単位とされていたが，2017 年度からは 2 単位への変更がなされた。加えて，前期・後期の 2 学期制の開講であった授業を可能な範囲で 4 学期制として提供してほしいという要望が全学教育科目を司る高等教育推進機構からなされた。

これらの背景から，著者らがこれまで授業を行っていた後期 15 週（総合科目 1 単位）の授業を，2017 年度は秋学期（8 週 : 1 単位），冬学期（8 週 : 1 単位）のクオータ制として開講をした。本報告ではこのクオータ制での試行の結果を報告する。基本的には，内容はこれまでのものに準じるが，15 週で行うものに比べ，8 週を 2 セットで行う場合には，授業構成等に様々な工夫が必要であることが明らかになった。

以下，2 章では，初学者を念頭においた本授業の基本的な内容，体制，進行等の状況を示す。3 章ではクオータ制として開講する際に想定される課題と工夫を，4 章ではクオータ制で実践した結果を示し，考察を行う。5 章でまとめを行う。

2　初学者を念頭においたプログラミング授業

2.1 2016 年度の後期（半期）の授業の概要と体制

本授業は，後期の選択科目であり，「誰でもプログラミング」と名付けている。1 年生が多いが，上級生も履修可能であり，少数ながら存在する。シラバスにおいて，初学者を念頭に置くことを記載している。プログラミング言語は Ruby を用いている。著者 3 名が担当し，TA2 名が支援する形で行っている。履修者は年度により異なるが，概ね 60 名から 80 名程度である。初回は大きな教室でガイダンスを行い，その際にプログラミング経験等を聞くアンケート調査を行う。調査結果を踏まえ，3 グループに分け，別々のコンピュータ教室にて授業を行う。グループ分けは，履修者のプログラム経験（プログラム言語や程度等も自由記載）に加え，同じグループになりたい人がいればその希望を調査票に記載させており，極力その希望に合わせている。選択科目で主体的なプログラミング演習を行う必要があることから，教え合いができる環境を整える意図もある。

例年，プログラム経験者のみのグループが 1 グ

ループ，初学者が 1-2 グループ（初学者と経験者の混合が 0-1 グループ）程度になっている。なお，ここで初学者とは，必修の一般情報教育科目で行っている Scratch は除いて，プログラミング経験がない者を指している。

初学者のグループには TA を配置する。履修者は，プログラム経験には依らず，基礎課題 11 題と応用課題（多数の応用課題から好きな課題を選択）を行うことを課している。基礎課題の課題一覧を表 1 に示す。

表 1　基礎課題一覧

基礎課題名	内容	学習する知識・必要とする知識
課題 1	整数の演算	プログラム実行に至る一連操作
課題 2	浮動小数点数の演算（限界）	数値の表現
課題 3	変数を用いた計算	代入，出力
課題 4	繰り返しによる演算時間の確認	繰り返し
課題 5	平方根の逐次近似	ループ脱出
課題 6	HTML ファイルの出力	HTML 表示，ファイル出力
課題 7	魔法陣（行列の和を含む）	—
課題 8	メソッドにおける引数の設定	メソッドの作成，引数の対応
課題 9	パスカルの三角形（再帰）	再帰，書式付き出力（再確認）
課題 10	画像データの合成	ビット演算
課題 11	文字の頻度分布	正規表現

初学者は，担当者と一緒に基礎的な文法の確認を行っていく。そのため，特に最初の課題の進度が経験者に比べ遅くなる。基礎課題 7 では，必要知識として，追加の学習事項はないものの，魔法陣のアルゴリズムの理解した上で，縦横のマス目の和の計算を独自に作成し，HTML 形式の表として表示する必要がある。それまでに学習した内容を理解し，組み合わせる必要があり，難易度が高い。初学者では，概ね，15 週中で第 10 週程度あたりから基礎課題を完了する学生が出てきはじめ，その後数週間程度で応用課題に取り組む。一方，経験者はテキストの PDF を確認しながら自律的にプログラムを作成していき，担当者の寄与はさほど高くない。そのため，応用課題に達するタイミングも学生により差が大きくなる状況である。

2.2　2016 年度における応用課題の内容例

応用課題は，様々な例示のなかから各自が興味を持った課題を取り組む。例えば，円周率（モンテカルロ法による計算，計算精度 1000 桁まで求める，ほか），ライフゲーム，スピングラスシミュレーション，公開鍵暗号，迷路の作成と脱出，アミノ酸の頻度分布，文字の頻度分布，金利計算等である。Ruby で計算した結果を，Tk や SVG などで描画する課題が比較的多い。図 1 に円周率をモンテカルロ法により求めたプログラムの実行例を示す。図 1 の上は Tk を用いた結果であり，下が SVG を用いた結果である。

図 1　円周率をモンテカルロ法で求める応用課題の実行例（Tk と SVG での使用例）

応用課題は，単にプログラムを提出するだけではなく，あわせてレポートの提出を課している。レポートには，当該応用課題における作業内容，課題概要，プログラムのポイント，結果や苦労したことや工夫したことなどを適宜まとめるように指示している。これについても，初学者，経験者を問わないで実行させてきた。

3 クオータ制への適用

3.1 クオータ制への対応に向けての意見聴取

クオータ制への適用に際し，2015 年度，2016 年度に履修をした学生から協力者を募り，どのように分割すると良いかの意見を聴取した。特に，利用するシステムやクオータ制における課題内容（どこでどのように分割すべきか）等について確認をした。合計 6 名に，2017 年 5 月に各 2 時間程時間を取って，調査に協力していただいた。結果，以下の意見を得ることができた。

1. 2016 年度は第一著者らが，ウェブで利用可能な学習環境を開発し，一部で試用した[1]。その学習環境を 2017 年度で開始されるクオータ制のプログラミング授業で用いた方がよいかを確認した。初めて当該学習環境を利用させた 4 名を含め 6 名は，使い勝手が良く，使った方がよいとの意見であった。

これらから，秋学期（8 週）では，初学者に対し，図 2 に示すようなプログラミング環境を共通で用いることとした（経験者および冬学期は，強制せず好みに任せた）。プログラムの編集・実行の画面例を図 2 に示す（左半分にプログラム入力画面，右上に教師の提示資料，右下にプログラム実行画面の 3 画面で構成される）。

図 2　プログラミング学習環境の画面例

2. 15 週の授業における基礎課題の中では，魔法陣の課題は難易度が高いように思う。

本意見により，秋学期は魔法陣の基礎課題を抜くことも良しとし，8 週で基礎課題 8～9 のメソッドまでを行う範囲を目安に進度を調整することとした。また，各学期で成績評価が必要であることから，秋学期の最後には，比較的簡単なミニ応用課題を出題することとした。

3.2 クオータ制に向けての学習構成の変更と準備

1 章で述べた通り，2017 年度からは，秋学期 8 週で 1 単位，冬単位 8 単位で 1 単位の授業となる。単位の実質化を念頭に，授業時間外の学修時間をしっかり取らせたいと考え，主に秋学期用には，短時間 5 分程度の予習・復習動画を用意した。

- フローチャート，条件分岐
- フローチャート，繰り返し
- 配列（１）（２）
- メソッド

予習・復習教材は，図 3(a)(b)のようなスライドに音声を付けたものである。これらの資料は担当 TA により作成された。

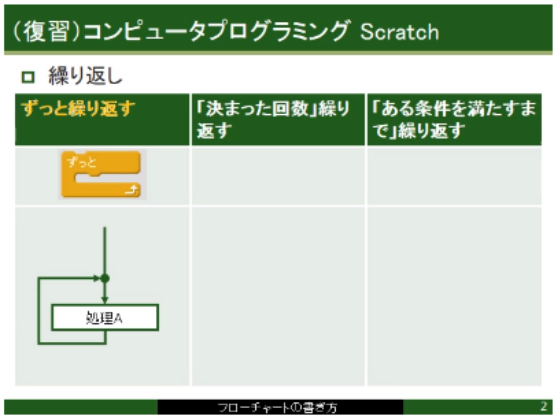

図 3(a)　履修者が前期に学習した Scratch での繰り返し例をアニメーションで提示

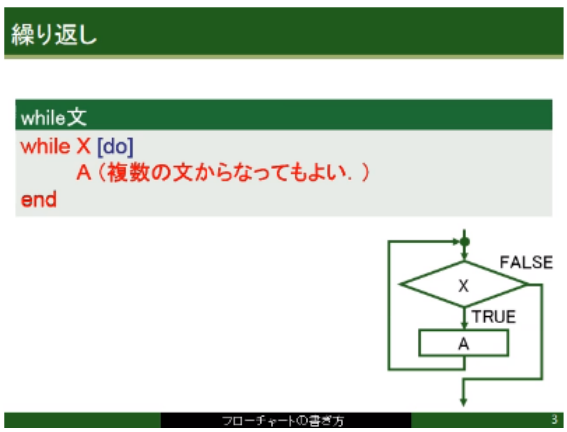

図 2(b)　続いて Ruby での繰り返しの文法をフローチャートと共にアニメーションで提示

3.3 秋学期と冬学期

秋学期，冬学期とも 8 週であるが，秋学期は初回にガイダンスと調査が入るため，プログラミングの授業には実質的には入れない。そのため，秋学期は PC 教室での授業が実質 7 回となる。また，最終回にはミニ応用課題を行う時間が必要であるため，基礎課題 7 を除き，基礎課題 8 もしくは 9 までを第 7 週までに仕上げておく必要がある。

秋学期のみを履修している学生もいるため，冬学期ではグループを再編成する必要がある。2017 年度の冬学期は，クオータ制での試みとして，プログラミング言語を変更し JavaScript とすることも選択できるようにした。つまり，秋学期は Ruby で基礎課題を主に行うことを 3 グループで共通に実施する。冬学期は Ruby で応用課題を行うグループと，JavaScript を行うグループに分かれるものとした（図 4 参照）。ここで，Ruby の応用課題はそれ以前の年度と同様のものとを含めたが，JavaScript に関しては別言語にて課題を行うこととなるため，冬学期の応用課題の一つはオセロに限定して，全員がベースとなるプログラムを共通で取り組むこととした。

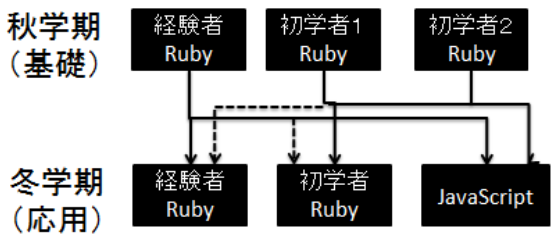

図 4　秋・冬学期のグループ分けと組替の模式図

4 クオータ制の試行と課題・考察

4.1 クオータ制の試行

2016 年度までに比較し，2017 年度の秋学期の履修者は 114 名と多く，教室割り当てと対応が例年に比して大変な状況となった。一方で冬学期を引き続き履修する希望者は 80 名程度となった。秋学期においては，初学者が多人数教室で割り付けらる必要が生じ，TA 等の対応も手薄となる部分があった。

4.2 秋学期

事前の設計と相談のとおり行ったところ，経験者と初学者の差が大きく，初学者はミニ応用課題に到達するまでに時間を要し，経験者はミニ応用課題では足りない学生がある程度いる状況となった。特に初学者に対しては，自習するように促したものの，自習を行っている学生は少数にとどまった。開発した予習用の動画について，音声はいらないといった否定的な意見もあった。全体に自習が足りないと思われる状況であったため，確認ミニテストを複数回取り入れ，学生自身が自身の理解状況を把握できるように努めた。

秋学期最終のミニ応用課題は以下のようなものとした（抜粋でありこれ以外もある）。

・自由課題：自分で問題を設定し，プログラムする
・画像の量子化
・モンテカルロ法で円周率を求める（図の描画無）
・クイズ足し算
・カレンダー（HTML 描画を用いる）

4.3 冬学期

Ruby での授業は例年と同様であったが，初学者は特にグループの組替えが生じたため，どうしても教員が学生の理解状況を把握するまでに時間を要した。

JavaScript のグループに関しては，初学者と経験者の混合グループとなっており，それまでの秋学期の学習項目をきちんと理解していることを前提に授業を進めた。Ruby と JavaScript における条件分岐や繰り返し等を比較させたのちに，HTML の表形式で描画するプログラムを順次提示し，図 4 右のような初期画面と交互に石が打てるところまでを示した。その後，オセロのロジックを組みこむ箇所は各学生に思考させた。

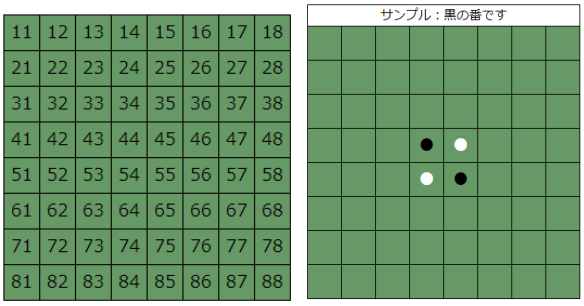

図 5　提示した JavaScript プログラムの実行例

4.4 クオータ制に関わる課題と考察

このようにしてクオータ制を試行したが，以下のようないくつかの課題があった。

・　自習が必要という認識が 2016 年度までの半期 15 週授業の時に比べ全体に低いように思

われた。この理由は明らかではないが，2016年度までで15週で1単位であっても本授業を履修しようとする学生は，プログラム習得の意識が高い可能性がある。また，2017年度は履修者が増えたことから1グループあたりの学生数が明らかに増えており，初学者に対する支援が手薄になったこと。そのため授業時に十分な理解が得られず，興味が持ちにくかった可能性もある。

- 秋学期のミニ応用課題で，モンテカルロ法による円周率を課題として提示したが，描画を伴わないプログラムだけとした。このような場合では，容易にネット検索できてしまい，ネット検索したものをそのまま提出する学生が少なからず生じた。2016年度までは描画を伴う課題としており，このような事例は起きなかった。ミニ応用課題に取り組む時間が限られていることも要因の一つであろう。学生が応用課題で課せられている内容をきちんと把握できていない可能性もある。
- 秋学期では8週しかないため，経験者と初学者との実力差を埋めるのは難しい。そのため，成績評価の基準を調整する必要があった。

4.5 考察

4.4で述べた通り，クオータ制でプログラミングを行うのは半期の授業に比して困難が多いように思われる。特に成績評価に係る部分は大きな課題と考える。半期の授業であれば，15週の間に，努力した初学者は，経験者との実力差が埋められており，初学者であっても成績優秀者が出る。しかし，8週間ではそのようなリカバリーは難しい。現状では，8割の学生は大学に入ってから初めてプログラミングを学ぶ。このような状況での授業であれば，半期でじっくりと授業を構成して行うことが望ましいと思われる。

一方で，RubyとJavaScriptといった異なるプログラミング言語を8週ずつ複数学ぶ授業構成は，学生が得るところも多いように思われた。条件分岐や繰り返しといった処理方法や，問題解決のための考え方が共通していることを理解することは重要であると考える。オセロを作成している際に，学生によっても様々な取り組み方があり，プログラムの考え方が一通りではないことを学ぶこともできる。

これらの課題と知見を活かし，2018年度は，基本的には秋冬学期における学生の組替えをさせず，初回の調査で殆どを調整するようにしたいと考えている。その上で，毎時間の確認テストで理解状況を本人に把握させ，より効果的に授業に取り組めるようにチェック項目を増やしていくことを検討している。

5 まとめ

著者らが行ってきた一般教育としてのプログラミング教育について，2017年度のクオータ制の試行状況を報告し，結果を考察した。現状では，秋冬学期に分けることで，秋学期の履修者が増える傾向にあり，指導が手薄になる可能性がある。秋学期及び冬学期が各1単位となったことから，学生には例年に比べて，授業時間外の学修が必要となっているのであるが，これまでと比較し，自習時間が返って短くなっているいと考えられる。そのため，自身の理解度を確認するためにも，確認ミニテストといったチェックテストを授業時に掲揚しながら進めていくことを検討していきたい。また秋学期のミニ応用課題では，どのような例が良く，またダメなのかを提示し，きちんとしたレポートの提出を促していきたい。

初学者がRubyとJavaScriptといった異なるプログラミング言語を8週ずつ学ぶ学習構成は，学生のプログラミングに関する考え方を広げる可能性があると考える。学生の思考や認識の状態についても引き続き調査を行っていきたい。

参考文献

[1] 布施泉、中原敬広、岡部成玄、プログラムの相互利用と相互評価が可能な初学者用プログラミング授業支援環境の構築、教育システム情報学会誌 35 巻、2 号、221-226 ページ、2018 年.