プログラミング教育における反復学習を採り入れた授業方式

多田　知正・丸田　寛之*

The Course Design with Repetitive Lessons for Education of Programming

Harumasa TADA, Hiroyuki MARUTA*

Accepted January 20, 2010

抄録 : 近年プログラミング教育の重要性は高まっているが，現在行われているプログラミングの授業には，プログラミング実習が有効に機能していないという問題がある。この原因は，授業で解説する内容と実習で行う課題の間に難易度のギャップが存在するためであると考えられる。本研究では，このギャップを埋めるために反復学習を取り入れた授業方式を提案し，これを実際の授業に適用して従来の授業方式との比較を行った。その結果，反復学習により，プログラミング実習の有効性が改善することが確認された。

索引語 : 情報教育，プログラミング実習，常套句，プログラミング言語

Abstract : Recently, education of programming is getting more important. In most of current programming classes, however, programming practices do not work well. We think that this is because there is a gap in difficulty between lectures and practices performed in a class. In order to fill the gap, we proposed the introduction of repetitive lessons into a course of programming. We performed repetitive lessons in an actual programming class and compared it with the traditional class. The results showed that the repetitive lessons can improve the effectiveness of programming practices.

Key Words : information education, programming practice, common phrase, programming language

* 株式会社ピクチャレスク（平成 21 年度京都教育大学大学院修了生）

1　はじめに

プログラミング能力を持つ人材の重要性は近年高まりつつある。社会の情報化に伴い，実際にプログラムを作成するプログラマが多数求められているのはもちろんのこと，システムエンジニアやプロジェクトマネージャといった情報技術者にとっても，システムの設計，保守などを行う上で，プログラムを読み書きする能力は必須である。

またそれに伴い，プログラミング教育の重要性も高まっている。かつてプログラマを目指す人の大半はもともと趣味でプログラミングを行っていた人たちであり，自ら多くのプログラムを読み書きする経験を通じてすでにプログラミングの能力をある程度身に付けていた人が多かった。しかし現在，パソコンユーザがプログラミングを行う機会は非常に減少している。現在では有償，無償のものを含め膨大なソフトウェアが存在しているため，ほとんどのパソコンユーザは自分ではソフトウェアを書かなくなっている。今後，プログラミング能力に対する需要が高まるにつれ，こういった人たちがプログラミングを学ぶ機会が増加すると考えられる。このため，今後，プログラミングの経験のない人に対して，プログラミングの能力を効率よく高めることがよりいっそう求められることが予想される。

しかしながら，プログラミング教育における授業方式については，これまで大きな進歩は見られない。実際のプログラミング教育の現場の大半において，依然としてプログラミング教育が開始された当初からほとんど変わらない形で今なお授業が行われているのが現状である。

プログラミング教育をテーマとした既存の研究としては以下のようなものがある。プログラミング学習を支援するためのシステムがこれまでにいくつか開発されている[1) 2) 3) 4) 5)]。また，すでにある程度プログラミングが出来る人を対象として，より高度な技術を習得させる手法についても検討されている[6) 7)]。実習における成績評価の方法を改善することにより受講生の動機を高めることができ，プログラミング能力の向上につながったとの報告もある[8)]。さらに，プログラミング経験のない生徒を対象として，プログラミングの基本的概念を理解させる試み[9)]なども行われている。しかしながら，プログラミング教育の初期段階における授業方式に着目した研究は，著者らの知る限り見られない。

そこで本研究では主にプログラミング経験の無い受講生を対象としたプログラミング教育における効率的な授業方式について考える。

従来のプログラミング教育における授業方式の問題点として，プログラミング実習が有効に機能していないことがあげられる。現在，大学や専門学校において行われているプログラミングの授業は，まず講義形式で対象となるプログラミング言語の文法についての解説を行ったあと，ただちに実習として受講生に実際にプログラムを書かせるという形で行われることが一般的である。しかしこの際，実習時間中に何もできずに時間を浪費してしまう受講生が多い。彼らの多くは授業で習った文法知識については理解しているにもかかわらず，実際のプログラムを書く段階になると手が止まってしまうといった状況に陥っている。その結果，課題を提出できなかったり，場合によっては他の受講生が書いたプログラムをそのまま書き写して提出したりしており，実習の効果がほとんど出ていないのが実状である。

このように，現在プログラミングの授業において多くの時間が実習に割かれているにもかかわらず，実習がその本来の目的を果たせていないことは大きな問題である。また，さらに深刻な問題は，授業の説明を理解しているにもかかわらず，実習においてプログラムを書くことができないという経験をした受講生がプログラミングを実際以上に難しいと感じ，その後プログラミングを敬遠するように

なってしまうことである。もしもそう言った受講生の中に，もともとプログラミングの適性がある人が含まれていれば全く逆効果であり，情報化社会にとって大きな損失である。

そこで本研究では，このような状況を改善するための方法として，反復学習を採り入れた授業方式を提案する。この方式は，プログラム中に頻出するパターンを反復学習により効率的に記憶させることにより，実習におけるプログラミングを容易にするものである。この授業方式を専門学校における実際のプログラミング言語の授業に採り入れ，その教育的効果について評価を行った。授業実践では，2 つのクラスの一方に反復学習を適用し，もう一方を従来の方式で授業を行い，それぞれのクラスで課題として提出されたプログラムの内容を比較した。実践の結果，一部の課題において反復学習を適用したクラスの方が良い結果が得られ，反復学習の有効性が認められた。

以下，2 節で反復学習を採り入れた授業方式について説明し，3 節で授業実践による提案方式の評価およびその結果について述べ，4 節でまとめを行う。

2　反復学習を採り入れた授業方式

(1) 従来の手法の問題点

大学や専門学校におけるプログラミング言語教育科目の一般的な授業の形式は，「プログラム言語の文法の説明」と「プログラミング実習」で構成されるものである。全国の 12 の大学，専門学校におけるプログラミング言語教育科目のシラバス（合計 89）を調査した結果，約 72% の科目において，文法の説明のあとただちにプログラミングの実習を行う形式をとっていることがわかった。

こういった形式の授業において，多くの受講生が実習の際に時間を浪費してしまい，実習が有効に機能していないという問題がある。著者らがこれまでに担当したプログラミングの実習における一般的な状況は以下のようなものである。実習を開始してしばらくの間，もともとプログラミング経験のない受講生の多くは全くプログラムを書きはじめることができず，何もしないまま時間が経過していく。一方で，すでにプログラミング経験のある受講生は早々にプログラムを完成させる。プログラムを完成させた受講生は，手が止まっている他の受講生に自分の書いたプログラムを具体的に説明し，場合によってはプログラムの一部をそのまま書き写させることもある。これにより，他の受講生はやっとプログラムを書き始めることができる。このように大半の受講生は実習の時間を有効に利用できておらず，結果的に課題を提出できなかったり，他人のプログラムを写したものを提出したりしている。このような受講生は，実習後も自分でプログラムを完成させたという実感はなく，実際に課題のプログラムを自力で書く能力も身についていないのが実状である。

こう言ったプログラムを書けない受講生の大半は，教科書に例として挙げられているようなごく単純なプログラムであれば書くことができる。すなわち，彼らは，授業に全くついていけないというわけではなく，授業で説明された文法事項については知識として理解しているが，その知識を使ってプログラムを書くことができないという状態にある。このことは，「文法的な知識を習得する」ということと「実際にプログラムを書く」ということの間に大きな難易度のギャップがあることを示していると考えられる。

(2) 授業方式の検討

プログラミングの授業において実習が有効に機能していないという状況を改善するためには，授業方式を変更する必要がある。

そもそもプログラミングとは，プログラミング言語という人工言語を用いて，計算機に行わせる仕事の内容を記述する行為に他ならない。そこでプログラミングの習得を我々が日常的に使用する自然言語の習得に当てはめて考えてみる。我々は，母国語については，特に意識して学習しなくても，幼少期に大量の言葉を聞いたり，話したりすることで自然に言語の規則や利用方法を習得していく。これをプログラミングに当てはめれば，大量のプログラムを読み書きすることで，自然にプログラムの書き方を習得するということになる，実際に現在プログラミング能力を持つ人の大半は，実質的にはこのような方法でプログラミングを習得しており，これが「プログラミングは習うより慣れろ」と一般に言われるゆえんでもある。しかし，限られた授業時間の中で，大量のプログラムを読み書きするということは実際には困難である。

そこで本研究では，言語教育として長い実績のある英語教育における授業方式に着目した。一般的な英語の授業においては，文法の説明の後でいきなり英作文の練習をさせるのではなく，さまざまな練習問題を通じて，実際の英文の中で文法が適用されるパターンを身につけさせた上で，英作文に取り組ませるという段階的なアプローチがとられる。一方，従来のプログラミング教育の授業では，文法の説明の後，いきなり作文に相当するプログラム作成を求められるため，ここに難易度のギャップが存在していると考えられる。そこで，文法の説明とプログラミング実習の間に何らかの練習課題を採り入れることにより，難易度のギャップを埋めることができる可能性がある。

(3) 練習課題の検討

難易度のギャップを埋めるための練習課題の内容を考えるにあたり，本研究ではプログラムに含まれるパターンに注目した。

プログラム中には，特定の状況において頻繁に登場するパターンがある。これはアルゴリズムのように洗練されたものではなく，単に決まり文句として書かれるプログラムの断片である。本稿ではこのようなパターンのことを常套句と呼ぶことにする。常套句の例を表 1 に示す。

表 1　常套句の例

・配列変数の参照や要素の入れ替え
・リスト，木などの動的データ構造の生成，参照，削除
・再帰手続きにおけるデータの受け渡し
・コマンドラインからオプション抽出
・入力データの処理
・計算誤差（丸め誤差，切捨て誤差）に対する対処

常套句は構造的には複雑なものではなく，実際のプログラムを見ればそのふるまいを理解することは容易であるが，プログラムを読み書きした経験の浅い初学者にとってそれらを自ら考案するのは困難であるという特徴がある。

常套句は実際のプログラムにおいて頻繁に用いられるため，職業として多数のプログラムを書くプログラミングの熟達者は，頻繁に常套句を書くことになる。その結果として，熟達者は多数の常套句

とそれを適用すべき状況を記憶しており，必要な状況においてただちにこれらを適用できるため，プログラム本来の仕事を記述することに集中できる。一方初学者は常套句に相当する処理手順を自ら考案しようとして，プログラム本来の仕事を記述する以前に時間を浪費してしまう。これが熟達者と初学者のプログラミングにかかる時間の違いの大きな要因であると考えられる。このことは，たとえプログラミングの初学者であっても，常套句を記憶し，適切に利用できるよう訓練することで，プログラミングにおいて「手が止まってしまう時間」を解消し，プログラミングの実習の効率を上げる可能性があることを意味している。

プログラミングの熟達者が常套句を習得するのは，実際に多数のプログラムを読み書きする経験を通じてである。このことから，常套句を効率よく習得させるための練習課題としては，反復学習が最も有効であると考えた。反復学習とは，漢字や計算のドリル学習と同様に，同じような問題を何度も解くことによって，知識の定着をはかる学習方法である。実際に多数のプログラムを読み書きする代わりに，常套句を習得させることを意図して作成した問題に集中的に取り組むことにより，より効率的に常套句とその使い方を記憶することが可能であると考えられる。

(4) 提案する授業方式

本研究で提案する授業方式の特徴は，講義と実習の間に反復学習を導入することである。この授業方式における 1 つの単元の流れを図 1 に示す。まず，講義の中でその単元で扱うプログラミング言語の文法や，プログラミング上の概念の説明を行う。次に十分な量の反復学習を通じて，その単元で扱う内容に関連した常套句を記憶させ，その後実習で実際のプログラムを書かせることになる。この時点では受講生は常套句が頭に入っており，比較的容易にプログラムを書きはじめることができるため，実習時間を有効に利用できることが期待される。

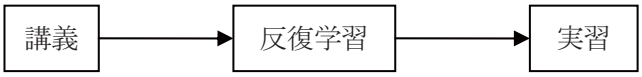

図 1　提案する授業方式における単元の流れ

反復学習の効果は，課題の内容によって大きく影響されると考えられる。このため，課題の作成にあたっては，実際にプログラミング経験のある人材が関わることが望ましい。また，実践によるフィードバックを通じて良問を蓄積していくことも重要である。そのために，作成した課題を WWW を通じて共有するといったことも有効であると思われる。

これまでに提案されているプログラミング学習支援システムの中にも，反復学習の機能を持つものが存在する [2) 4)]。しかしこれらのシステムは反復学習の課題として比較的単純なものを自動生成する形を取っているため，その効果は限定的であると考えられる。

反復学習の課題は，通常のプログラミング実習の課題とは異なり，以下のような点を考慮して作成する必要がある。

・ 難易度が低い

一般にプログラミング実習の課題は成績評価の対象となるが，反復学習の課題は全員がすべての

課題を終了することを前提としている。すなわち，各課題は授業内容が理解できていれば無理なく解くことができる程度の難易度に設定する。

・実際に手を動かす作業を重視する

通常プログラムを作成する上で，プログラムの一部を別の場所にコピーし，必要な箇所にのみ修正を加えるといったことはよく行われる。しかし学習においては，自分で実際に手を動かすことが記憶する上で重要な役割を果たす。このためサンプルプログラムは紙に印刷して配布するなど，受講生自身が実際に手を動かす作業を促すような工夫をする。

・プログラムを読む練習を含む

プログラミング能力の向上には，プログラムを書くだけでなく，読むことも有用である。そこで，常套句を含むプログラムを読んで，動作の様子を追跡してみることで，構造をより良く理解した上で常套句を記憶するような課題を含む。

このような条件を踏まえ，具体的には表 2 に示すような課題が考えられる。これらの課題は，著者らがこれまでに経験したプログラミングの過程で行った作業の中で，プログラミング能力の向上に有益であったと思われるものを基に考案したものである。

表 2　反復学習の課題

・プログラムの一部が空欄となっており，そこに当てはまる適切な文を答えさせる。
・プログラムと入力値を与え，実行した場合の出力結果を答えさせる。
・プログラムを印刷して配布し，それを自分で見ながら入力し，実行させる。
・バグ（プログラムの間違い）を意図的に混入したプログラムを与え，それをデバッグ（修正）させる。
・サンプルプログラムを与え，少しだけふるまいの異なるプログラムを作成させる。

3　授業実践による評価

(1) 目的

本研究で提案する反復学習の効果を確認するために，実際のプログラミング言語教育の授業において実践を行った。本授業実践の目的は，反復学習を採り入れることによってプログラミング実習の効率がどの程度上がるのかを調査し，反復学習の有効性を確認することである。

(2) 対象とする授業科目

本実践は京都コンピュータ学院における 2009 年前期のプログラミングの科目の中で行った。この科目は週 1 回，90 分の講義および演習が 2 時限連続で全 12 回行われるものである。科目の内容は，プログラミング言語 C# の基礎を学ぶものであるが，前半は特定の言語に依存しない基本的なプログラミングの解説に当てられており，後半で言語 C# の特有の機能について解説を行う。本実践は前半の部分にあたる第 2 回から第 7 回の授業に対して行った。対象授業の第 1 回から第 7 回の内容を表 3 に示す。

表 3　対象授業の内容

第 1 回　導入，コマンドプロンプト入門，C# の基礎事項
第 2 回　変数の概念，予約語，演算，簡単な入力操作
第 3 回　分岐構造（単分岐，双分岐，ネスト，多分岐）
第 4 回　ループ構造（while，do while，for）
第 5 回　配列の仕組み（配列，多次元配列，配列の配列）
第 6 回　メソッドの仕組み
第 7 回　再帰メソッド，クラスの基礎

同一授業を受講する 2 つのクラスに対して，一方のクラスには反復学習を含む授業方式を適用し，もう一方のクラスには従来どおりの方式で授業を行った。その上で，両方のクラスのプログラミング実習で作成されたプログラムを比較することで評価を行った。以下では本研究で提案する授業方式を適用したクラスを適用クラス。従来どおりの授業方式を適用したクラスを非適用クラスという。

(3) 反復学習の実施方法

反復学習は同じような問題に繰り返し取り組むものであり，実施には時間が必要である。しかし反復学習を通常の授業の中で行うと，時間的制約のために十分な量の課題を行うことができない。そこで本実践では，反復学習を宿題として課すこととした。ただし自宅にパソコンがない受講生がいることを考慮して，反復学習は紙ベースで行うこととし，課題をプリントとして配布し，記入して提出するように指示した。これにより，受講生はプログラムを繰り返し手で紙に書くことになり，常套句の形を記憶する上でも有効であると考えられる。しかしながらこの制約により，本実践では実施できなかった反復学習（例えば，プログラムのデバッグなど）も存在する。なお，両方のクラスで学習時間の差が大きくならないように非適用クラスの方にも宿題を課した。非適用クラスに対する宿題は既存の教科書や参考書に掲載されている練習問題から抜粋したものを用いた。

(4) 実習テスト

本実践では，プログラミング実習を実習テストと呼ぶ。本実践における実習テストは通常のプログラミング実習と同様に，与えられた仕様を満たすプログラムを作成する課題であるが，反復学習の有効性を評価するために，以下のような制約を設けて実施した。

- 授業中の決められた時間内にプログラムを作成すること。もし時間内にできなかった場合はその時点のプログラムを提出すること。
- 実習テストの間，他の受講生との相談および教員への質問等は認めない。
- インターネット（WWW）を自分で調査することは認める。ただしその場合は参考にした WWW サイトの URL を記載すること。

(5) 単元の流れ

本実践において，1 つの単元の流れを図 2 に示す。

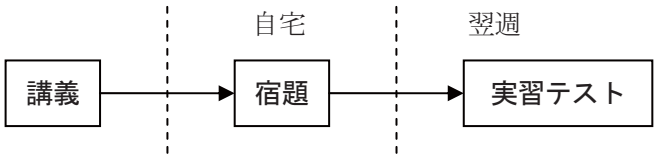

図 2　本実践における単元の流れ

講義終了時に，両方のクラスにそれぞれ宿題を課し，翌週までにやってくるように指示する。適用クラスの宿題は反復学習である。翌週の授業の最初に宿題を提出させ，その後，その単元の実習テストを行ったのち，次の単元の講義を行う。

1 回の講義（90 分× 2）のスケジュールは以下の通りである。

・前回の単元の実習テスト：60 分

・講義：120 分

(6) プログラムの評価

実習テストにおいて，制限時間終了時にプログラムを提出させ，その内容によって採点する。一回の実習テストはそれぞれ複数のプログラム課題からなり，それぞれのプログラムを，授業内容が理解できているかどうかという観点から，0 点から 4 点の 5 段階で評価する。プログラムの評価基準を表 4 に示す。実習テスト全体の得点は各プログラムの評価点の合計とする。

表 4　プログラムの評価基準

4 点：内容が理解できており，正常に動作する
3 点：内容は理解できているが，正常に動作しない（ケアレスミス）
2 点：内容はおおむね理解できている。課題に指定した方法を使っていない。 プログラムが未完成であるなど。
1 点：内容が理解できているとはいえない。方法が大きく誤っている。 プログラムの一部しか書かれていない
0 点：白紙あるいはそれに近いプログラム

(7) 結果および考察

紙面の都合上，授業実践の初期，中期，後期の代表的な単元（第 2 回，第 5 回，第 7 回）をとりあげ，それぞれの単元における実習テストの結果を示す。クラスによって受講生の数が異なるため，適用クラス，非適用クラスからそれぞれ 20 名を無作為抽出して実習テストの得点を比較する。はじめに，両クラスの得点の平均を Welch の t 検定を用いて検定した。結果を表 5 に示す。5% の有意水準で検定したところ，第 7 回についてのみ，統計的に有意であるとの結果が得られた。

表 5　実習テストの得点の平均

単元	第 2 回		第 5 回		第 7 回	
クラス	適用	非適用	適用	非適用	適用	非適用
平均	9.500	8.750	11.850	11.650	10.150	6.800
分散	14.053	19.882	11.503	24.45	28.976	41.011
自由度	37		34		37	
t	0.576		0.149		1.791	
P（T<=t）片側	0.284		0.441		0.041	
t 境界値 片側	1.687		1.691		1.687	

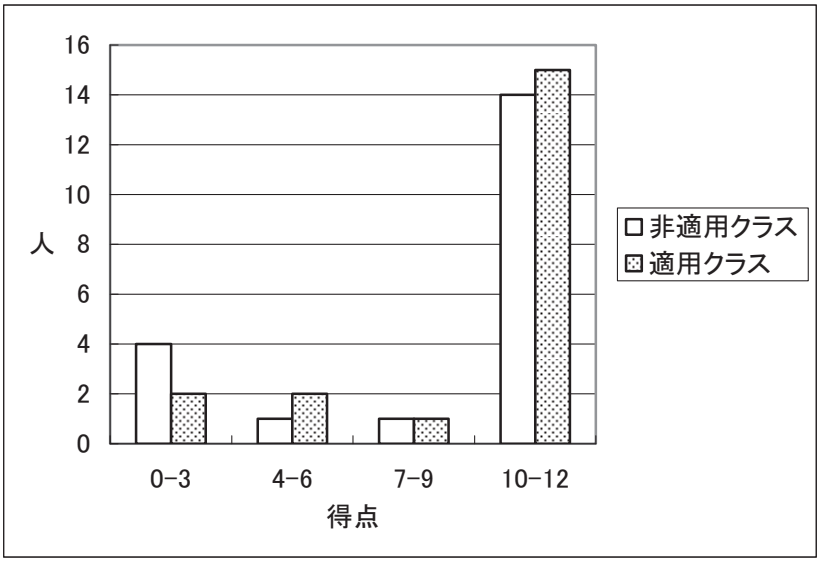

図 3　実習テストの得点分布（第 2 回）

図 3 に授業実践初期（第 2 回）の実習テストにおける得点分布を示す。この回の実習テストは 3 つの課題からなるため，最高得点は 12 点である。この回は入出力文や変数など，プログラムの基本的な書き方を説明する単元であり，常套句に相当するようなものは登場しない。実習テストの課題も授業内容が理解できていれば問題なくできるものとなっている。反復学習では変数の型や基本的な構文などを定着させるための練習を行ったが，実習テスト自体が容易な課題であったため，両方のクラスの結果に大きな差は見られなかった。ほとんどの受講生がすべての課題を完成させているが，どちらのクラスにも課題を完成できなかった受講生が含まれており，その分布はどちらのクラスも似通っている。この結果から，この時点での両クラスの能力にはそれほど大きな差はないと考えられる。

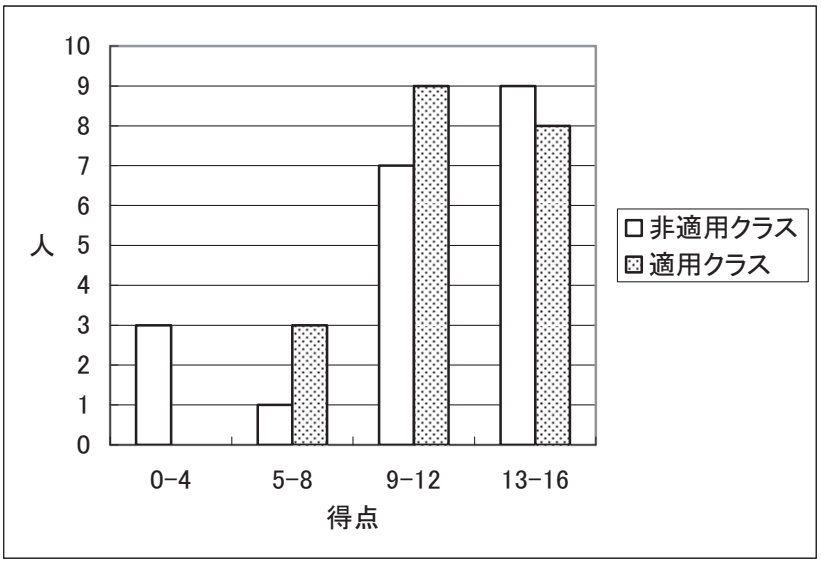

図 4　実習テストの得点分布（第 5 回）

図 4 に授業実践中期（第 5 回）の実習テストにおける得点分布を示す。この回の実習テストは 4 つの課題からなるため，最高得点は 16 点である。この回は配列変数の使い方を扱う単元であり，配列の参照の仕方や，要素の入れ替えなどの常套句が登場する。得点分布を見ると，適用クラス，非適用クラスともに全体的に高い得点であり，ここで登場する常套句は多くの受講生にとって自ら考案することはそれほど難しくなかったと思われる。また，実習テストの課題として取り上げた選択ソートは，プログラミングにおける代表的な問題であって WWW 上に解説のサイトがいくつか存在するため，それを参照した受講生も見られた。その中で，非適用クラスの方には 4 点以下の生徒が 3 人おり，ほとんどプログラムに手を付けられなかったことを示している。その一方で，適用クラスの方には 4 点以下の生徒は一人もいなかった。このことから適用クラスでは反復学習によって常套句を習得した結果，全員がプログラムを書きはじめることができたと考えられ，反復学習の効果が一部現れていると言える。

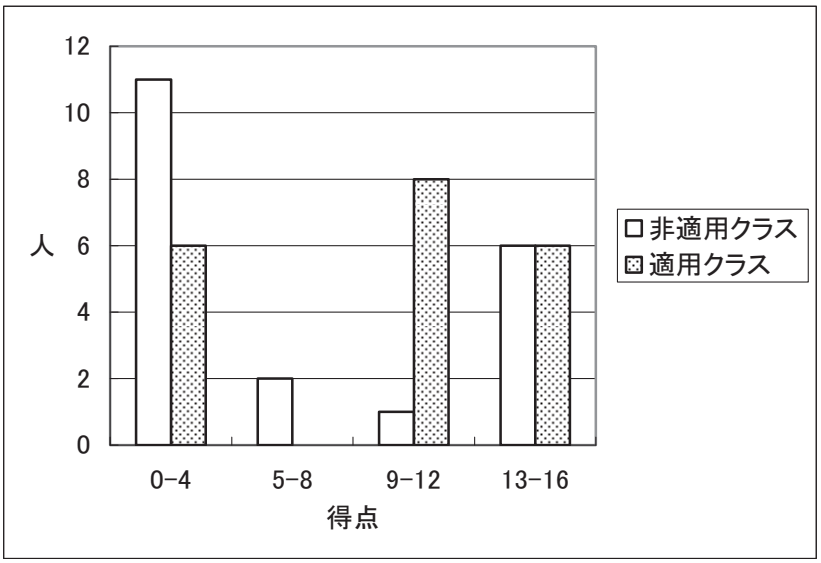

図 5　実習テストの得点分布（第 7 回）

図 5 に授業実践後期（第 7 回）の実習テストにおける得点分布を示す。この回の実習テストは 4 つの課題からなるため，最高得点は 16 点である。この回は再帰呼び出しを扱う単元であり，データの受け渡し方法などの常套句が登場する。得点分布を見ると，どちらのクラスの成績も良いほうと悪いほうにはっきりと二分化していることがわかる。再帰呼び出しはプログラミングの概念の中でも特に理解が困難であることで知られており，この概念を理解できた受講生は高い得点を得る反面，理解できなかった受講生は全く課題に手をつけることができず，低い得点にとどまるという傾向を示している。その中で，適用クラスにおいて，4 点以下の受講生の数は非適用クラスの約半分であり，その分 9 ～ 12 点の受講生が多くなっている。この結果は適用クラスでは反復学習によって再帰呼び出しを習得できた受講生が多数いることを反映していると考えられ，反復学習の有効性を示していると言える。

4 おわりに

本研究では，プログラミング教育において，反復学習を採り入れた授業方式を提案した。授業実践の結果，一部の課題において，反復学習を適用したクラスのプログラミング課題の成績が従来手法を適用したクラスの成績と比べて高いことが確認できた。この結果から，本研究で提案する反復学習は，比較的単純なプログラムを書く能力を身につける上で一定の効果があり，プログラミング実習の効率の改善に寄与すると結論できる。特にプログラミング教育の初期においては，プログラミングの能力を身につけさせると同時に，受講生のプログラミングそのものに対する興味を引き出すことがより重要である。その際，実習において自分が作成したプログラムが意図通りに動作するという経験をさせることには大きな意義があり，その意味からも反復学習は有効であると考えられる。

また，反復学習には，従来プログラミング能力を向上する上で必要不可欠と考えられてきた「慣れ」の部分を擬似的に再現することによって，将来的にはより高度なプログラミング能力の向上に適用できる可能性がある。このためには，反復学習の課題としてより技術的に高度なものが求められるため，プログラミングの熟達者が課題の作成に携わることが望まれる。

引用文献

1）竹田尚彦，川口清志，浅見美紀，佐合尚子「プログラミングの個別指導のための演習問題サーバ」『コンピュータと教育』58-4，2000，pp.21-28

2）中島秀樹，高端直久，細川宜秀「プログラミング学習のための QA サイクル -- 受講者の習熟度に応じた問題自動提示メカニズム --」『電子情報通信学会論文誌 D-I』Vol.J88-D-I No.2，2005，pp.439-450

3）知見邦彦，櫨山淳雄，宮寺庸造「失敗知識を利用したプログラミング学習環境の構築」『電子情報通信学会論文誌 D-I』Vol.J88-D-I No.1，2005，pp.66-75

4）内田保雄：「初級プログラミング学習のための自動作問システム」『情報処理学会研究報告 . コンピュータと教育研究会報告』2007（123），2007，pp.109-113

5）田口浩，糸賀裕弥，山本哲男，高田秀志，島川博光「プログラミング演習評価と講義反応を連携させた理解の契機の抽出」『電子情報通信学会論文誌 D』Vol.J91-D No.2，2008，pp.345-357

6）松澤芳昭，青山希，杉浦学，川村昌弘，大岩元「「目的の表現」に着目したオブジェクト指向プログラミング教育とその評価」『情報処理学会研究報告 . コンピュータと教育研究会報告』2003（123），2003，pp.77-84
7）駒谷昇一「新入社員に対するレビュを中心としたプログラミングの教育方法」『情報処理学会研究報告 コンピュータと教育研究会報告』95（111），1995，pp.67-72
8）武内亮，佐藤匡正「プログラミング教育における合理的評価法」『情報処理学会研究報告 . 自然言語処理研究会報告』2004（93），2004，pp.153-159
9）兼宗進，中谷多哉子，御手洗理英，福井眞吾，久野靖「初中等教育におけるオブジェクト指向プログラミングの実践と評価」『情報処理学会論文誌　プログラミング』SIG 13 PRO 18，2003，pp.58-71