社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＦＩ－ 76　(21)
2004－ＮＬ－163　(21)
2004／9／17

プログラミング教育における合理的評価法

武内亮[†]　佐藤匡正[††]

[†]島根大学大学院総合理工学研究科数理・情報システム学専攻　〒690-8504 島根県松江市西川津町 1060

[††]島根大学総合理工学部数理・情報システム学科　〒690-8504 島根県松江市西川津町 1060

E-mail: [†]ryo@cis.shimane-u.ac.jp [††]satou@cis.shimane-u.ac.jp

概要　大学等で一般に行われているプログラミング教育は，受講側にとっても指導側にとっても十分なものとは言い難い．この原因として，両者の教育の目的が乖離していること，プログラミング教育の技能と学問の二面性，そしてプログラミング教育の学問としての歴史が浅いために教育法が確立されていないことが挙げられる．昨今，日本の大学では FD 活動の重要性が指摘され，実施されている．指導側が授業方式の改善を試みていこうという活動である．本研究では，その流れの中で，上述した問題点を解消してプログラミング教育法を改善することを目的として，指導側と受講側の両方の立場をほぼ同時期に並行させていた経験からプログラミング教育における合理的評価法を提案し，試行結果について論ずる．

A rational method to evaluate programming education

Takeuchi Ryo[†]　Satou Tadamasa[††]

[†]Faculty of Science and Engineering, Graduate School of Shimane University

[††]Faculty of Science and Engineering, Shimane University

E-mail: [†]ryo@cis.shimane-u.ac.jp [††]satou@cis.shimane-u.ac.jp

Abstruct　A programming education course generally performed at a university is insufficient for the teaching and learning. Three reasons can be guessed that the purposes are different each other, programming has two aspects of learning computer science and acquiring Technique, and programming teaching methods are not established due to its short history. Now, in universities, Faculty Development activities for improvements on teaching methods are being performed. To conform them, this paper suggests a rational education method in programming education course based on learning purposes, and discusses the results.

１．　序論

大学で行われているプログラミング教育は，FD（Faculty Development：大学教員の指導能力開発）などの指摘に見られるように，受講側にとっても指導側にとっても改善の余地がある．この原因には，プログラミング教育が大学教育の一環として行われていることからくるものと，プログラミング教育固有の特性による原因とがある．

前者による問題点として，指導者と受講生の授業に対する考え方の乖離が挙げられる．乖離の中でも最大のものは，指導者は受講生が自主学習するのが当然だと考えているのに対して，受講生は課題でも出されない限り授業時間以外に自主学習する必要が無いと考えていることである．これは，自主学習をしなくても最低限の単位は認定されるのが普通であることに起因すると推定される．

プログラミング教育において最も重要な「できていない箇所をできるまでやり直す」という行為，つまり復習が受講生に軽視されているのは致命的である．これは，受講生からすると，一度提出して評価された課題の内容を復習す

ることが, 評価面でのプラスに必ずしもつながらないことによる. 受講生は単位認定を動機として行動するので, 復習することによって評価が向上することを示せば, 十分学習させることが可能であると考える.

プログラミング教育固有の特性による原因は, プログラミング教育を含む計算機科学の学問としての歴史が浅く, 教育法が確立していないことや, プログラミングの「技能」としての性質のためと考えられる. 教育法が確立していないことによって, 指導者自身もどう指導すれば受講生の能力を伸ばすことができるか把握できないという問題点が生じる.

さらに, プログラムを組む際に絶対となるべき「仕様」の概念が完全に無視されている点を指摘する. 「仕様を満たさないプログラムは無価値である」というプログラミング技能に関する前提条件が, 「授業内容の約６割の理解があれば単位を認定する」という評価システムとうまく噛み合っていない. 仕様を「ある程度」満たすことで単位が認定されるので, それ以上を望まない受講生には最も重要な前提条件が身に付かないという現象が起こる. これは, 十分なプログラミング能力が身に付いていることを前提として授業を進める, 他の指導者にとっての大問題となる.

昨今, 日本の大学では FD 活動の重要性が指摘され, 実施されている. 指導側が授業方式の改善を試みていこうという活動である. 関連して, 文部科学省の大学改革案として, 適正で厳格な学生評価や教育の充実策を考えることを, 各大学は求められている[1][2]. 本論文は, その流れの中で, 上述した問題点を解消してプログラミング教育法を改善することを目的として, プログラミング教育に関する授業方式の改善案や, 適正で厳格な学生評価の尺度を, 指導側と受講側の両方の立場をほぼ同時期に並行させていた経験から示す. そして, それらを実際に適用した結果とその効果を示す.

２． 教育目的

２．１ 類型化

受講生と指導者の教育目的を整理するため, これまでの学生としての経験によって, 大学に所属する学生を次の２種類に大別する.

表１ 類型的な２種類の学生

	能力重視型	単位重視型
受講目的	勉強するため	卒業するため
授業への取り組み	積極的	消極的
重要度	評価＜能力	能力＜評価
割合	約２０％	約８０％

割合は, FD 活動の一環として本学で行われた学生アンケート調査[3][4]の結果, 「予習・復習をしているか」という設問に対して肯定的な回答をした学生を能力重視型, 否定的な回答をした学生を単位重視型として数えたものである. 能力重視型学生を「積極的な学生」と位置付けていることから, 「どちらでもない」との回答は, 単位重視型とカウントしている. この比率は, 他大学での FD 活動の調査結果ともほぼ等しいものなので, 一般性をもつと考え, 以下のように定義する.

定義：
学生は, 能力重視型学生と単位重視型学生に大別することができる

定義：
大半の学生は単位重視型である

ほとんどの大学教員の考え方は, 能力重視型学生のそれの延長であると推定される. それは, 大学教員のほとんどが研究者の延長であることからすれば自明である. このことと学生のタイプの比率から, 「大学では能力重視型の指導者が単位重視型の受講生を教育しようとしている」ということができる. 指導者と受講生の授業に関する考え方に乖離が生じるのは, 現状では当然であると言える.

定義：
教員の考え方は, 能力重視型学生の延長上にある

２．２ 受講生の目的

とにかく単位が認定されればいいという単位重視型学生は明快な立場にある. 能力重視型の学生は必ずしも「卒業したくない」「単位を認定してほしくない」と考えているわけではない. むしろ自身の能力に自信をもって「単位認定されて当然」と考えているので, 万が一低い評価を受けた場合は強く不満をもつことになる. 逆に, どちらの学生も自分が思っていた以上に高い評価を受けて不満をもつことは無い.

以上のことから, 受講生の目的は授業を終えた後に自身の能力以上の評価を受けることであると言える.

定義：
受講生の目的は, 授業を終えた後に自身の能力以上の評価を受けることである

評価を得ることが目的である受講生が理想とする授業は, 授業に沿う活動をしていれば自然に能力が身に付いているものである. この論拠を学生の類型から考えてみる. 能力重視型の学生と単位重視型の学生では異なる. 前者は「単位は後からついてくる」と考えるため, 厳しい授業を理想とすると推定できる.

しかし, 後者の場合は同じ単位を認定しても

らうにしてもできるだけ楽をする姿勢に問題がある．学科試験を嫌がり，レポートの提出を嫌がり，出席を取られるのを嫌がる．だが，単位を最優先に考えるということは，単位を取るための行動なら嫌々ながらもこなすということになる．上で「能力重視型学生と言っても単位を認定して欲しくないわけではない」と述べたが，同様に単位重視型学生と言っても能力が身に付いて欲しくないわけではない．したがって，単位を認定してもらうための活動に能力向上が伴えば理想である．

定義：
受講生の理想は，高評価を得るための行動が能力向上につながる授業である

２．３　指導者の目的

前節で整理した受講生の目的をみると，受講生に対する評価を下すことが大学における授業の本質になってしまっているように見える．評価を行わない授業は存在せず，逆に実際には何もしていなかったとしても，評価を与えれば少なくとも事務手続き上はそこに授業があったことになるためである．しかし，本来授業の目的は評価にあるのではない．ここで，それをはっきりさせるためにも「評価とは何か」について述べる．

評価とは，当該授業について受講生がどの程度の理解をしているかを示すものである．基本的には，指導者が「授業内容を１００％理解した受講生」の像を描き，各受講生が理想の能力のどれくらいの割合を満たしているかを評価して受講者に示したものである．ここから逆に，指導者の目的は，受講生に基準以上の力を付けさせることであると推定できる．そして，受講生から見れば，授業とは受講生を指導者の理想像に近づけるための啓蒙活動であるといえる．

定義：
評価とは，指導者の理想とする能力をどの程度満たしているかを示したものである

定義：
指導者の目的は，受講生に基準以上の力を付けさせることである

２．４　両者の目的の乖離

本論文では，「評価」を「指導者の理想とする能力をどの程度満たしているかを示したもの」と定義した．「受講生に基準以上の力を付けさせること」と定義された指導者の目的と併せると，指導者は点数が高い受講生に高評価を与えるのではなく，自分の理想とする受講生が高評価を取れるような採点基準を設定していると考えられる．

事務手続きとして評価を示す際には「優・良・可・不可」の形がとられる．これらは８割・７割・６割・６割未満という点数に対して付けられる評価である．つまり，最終的な評価を付ける際には，受講生の能力を正確に数値化する必要がある．受講生の立場で考えるとこの数値化の方法は不透明なものである．指導者の立場で考えるとどのような規則に従って数値化するのが適当かに悩まされる．

ここでは，プログラミング授業における適正な評価基準を探る目的で，大学における評価として公開されている評価基準を参考とする．本学シラバスに評価基準として掲載されているものから評価対象と評価項目を表にまとめる．これは，ｗｅｂ上に公開されている全国の大学のシラバスと全く同じ傾向にあるので，一般性を持つといえる．

表２　一般的な評価基準

評価対象	評価項目
能力	レポート（報告書）・演習
隔離状況での能力	学科試験・口頭試問
人格	出席・受講態度

評価は受講生にとっては最も影響の大きい要素である．指導者は，極端に言えば評点を餌として利用できる立場にある．高い評価を得るためという動機として示して受講生にできるだけ力を付けさせるのが目的であり，評価すること自体が目的というわけではない．教育という点では，むしろ一度不可という評価を下した受講生の力を引き上げることこそが本当の指導と言えるかもしれない．しかし，理想とは裏腹に一度低評価を与えた受講生の力を引き上げることは不可能に近い．

たとえば一つのレポート課題を考える．この推移を図１に示す．

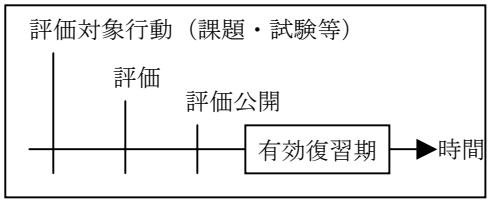

図１　評価のタイミングと有効復習期間

受講生がレポート課題提出という評価対象行動を取ると，指導者はそれを受けて内容を評価，公開（点数を付けて返却）する．復習を行うことで力を身に付けることができるのは，自分ができていなかった部分を知り得る評価公開後である．この有効復習期間にやることをやっていれば力が付くはずである．しかし，現状では復習を行う学生は稀である．仮に復習して内容を自分のものとしても，一度下された評価

が覆ることは無いからである. 評価が下された時点で「出来が良ければ高評価が貰える」という動機が消えてしまっていることになる.

3． プログラミング教育

3．1 教育内容の定義

議論の途中で用語として使用するために, まずはプログラミング教育内で指導すべき内容を定義する. 本研究では, プログラミング教育で指導すべき内容は6階層の技能部分と2つの根本思想からなると考える.

(1)プログラミング知識
プログラミング言語に因らない, 汎用的に用いることができる知識をプログラミング知識と呼ぶ.「プログラムとは何か」という根本的なところから変数の意味, 選択や繰り返しといった制御構造の仕組み,「オブジェクト指向とは」というような話までも含む.
(2)プログラミング言語知識
C, JAVA, Pascal といった実際にコンパイル・動作可能なプログラミング言語に関する知識をプログラミング言語知識と呼ぶ. 変数・関数の定義方法や組み込み型関数の用途などについてなど,「プログラミング授業」と聞いて自然に頭に思い浮かぶであろう部分にあたる.
(3)設計能力
仕様を読み解いて, 自分が何をすべきかを把握する能力を設計能力とする.「○○を実現するプログラムを作れ」といった課題において最初に要求され, 成長させることができる.
(4)開発環境知識
開発環境に関する知識を開発環境知識と呼ぶ. ライブラリの使用法から最近かなり多くなってきている統合開発環境型のアプリケーションの使用法などをいう.
(5)コーディング能力
設計を実際のソースコードに変換する能力をコーディング能力と呼ぶ. 現実的には試行錯誤することが当然の部分であり, デバッグ能力もここに含む.
(6)拡張機能知識
言語とは関係無く, 別のアプリケーションを操作したりする機能に関する知識を拡張機能知識と呼ぶ. 本論文ではプログラミング「基礎」教育に焦点を当てるので,「十分な力がついて初めて手を出せる領域」とする. プログラミング応用授業で学ぶのがこの部分である.
(7)プログラミング倫理
求められた結果を出さないプログラムには何の意味も無いという思想をプログラミング倫理と呼ぶ. 仕様の絶対性に関わる部分であり, そもそもの大前提であるはずの考え方である. これを徹底しようとすることでデバッグ能力が成長するが, 逆に軽んじるとその部分が全く成長しない.
(8)プログラミング書法
各種命名規則から関数化やファイル分割のタイミング, どのような場合にどのようなデータ構造を使うか, どのような場合にどのような関数を使うかといった実際にプログラムを組む際の思考の道筋をいう. プログラミングスタイルとも呼ばれる. 放っておけば最も各受講者の個性が出る場所である.

(1)から(6)までは登場順に前の段階の知識を土台とする階層構造をとっている. (7)(8)は技能部分とは独立で, 指導者側からすれば「言われなくても自分で身に付けて当然」と考えがちな部分である. しかし, 2．1で述べたように, 指導者と受講生の教育に対する意識の違いから, 指導者が自明であると考えて明言しないことが最も受講生に伝わり難い部分となってしまっている.
以上の関係を図2にまとめる.

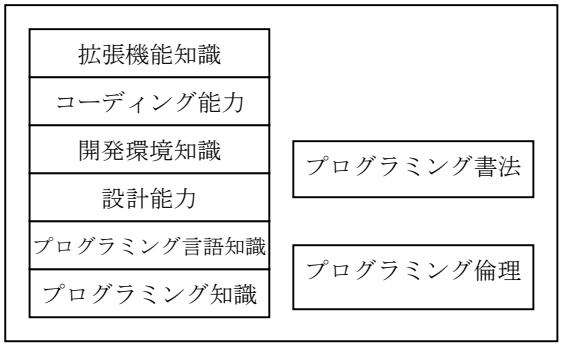

図2 プログラミング教育指導内容

3．2 情報教育内での位置付け

大学の情報教育の目的は二面性をもつ. 一面は研究者育成であり, もう一面は総合的な力をもつシステム技術者の育成と考えられる. この視点から, プログラミング教育もやはり同様の二面性をもつといえる.
一般に, プログラミング教育は計算機に関する基本的な知識を土台として行われ, 情報分野の他授業の土台として用いられる. この関係を階層モデルとして次に示す. それぞれ, 下層を基礎層, 上層を応用層とする.

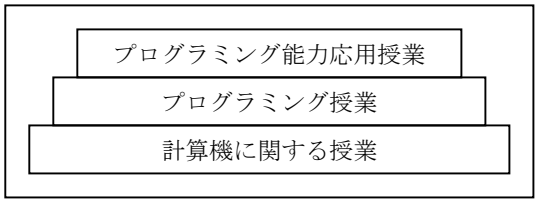

図3 情報分野授業階層モデル

もちろんプログラミング能力自体を伸ばすことを目的とした授業もあるが, 授業で解説し

た「知識」が計算機上でどう実装されるかを確認するための手段として使われることがほとんどである. その際に繰り返してプログラミングの知識について説明されることはなく,「これまでに身に付けているはずの知識を駆使してこの課題を完成させよ」という形式で使用される. これは他の授業についても同様で, この「科学が『体系』の上に成り立っている」という本質的な部分が中学・高校での教育で欠けている点である.

情報教育内でのプログラミング教育の位置付けからすると, その目的は「最低限ツールとして使用可能なプログラミング能力を身に付けさせること」と断ずることができる. 理由を以下に述べる.

プログラミング教育の上位層にあたる授業において, 主役はプログラミングではない. 作成されたプログラムは授業内容の確認等のために用いられる. したがって, 下層にあたるプログラミング教育に求められるのは, プログラミング作成能力の涵養に止まらない, プログラミング能力を応用する力である.

以上のことから, 情報教育内でのプログラミング教育の目的は, ツールとして使用可能なプログラミング能力を身に付けさせることと言える. なお, これはプログラミング教育を研究者育成教育の一環としてみた場合のことである. 技能教育の視点で見ると, それだけではとても満足することができない. よって, ここで定義する目的は目標点ではなく最低限のラインである.

定義：
プログラミング教育の目的は, 最低限ツールとして使用可能な能力を身に付けさせることである

４． 授業方式

４．１ 一般的な授業方式

日本国内の大学で公開されているシラバスを調査したところ, 現行プログラミング教育における授業法の大半は以下のようなものであった.

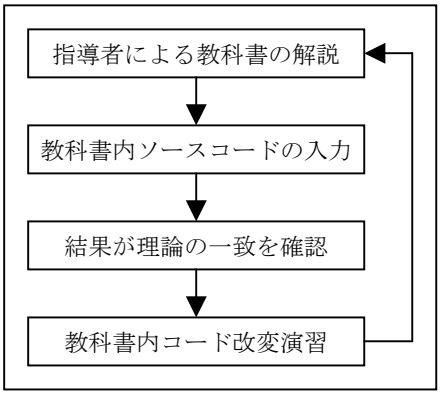

図３ 一般的なプログラミング授業の流れ

教科書は自作の資料であったり市販のプログラミング入門書であったりするが, その本質には変わりが無い. 教科書を読み進めながらループを繰り返し, 規定授業回数に達したときにその一つの単位のプログラミング授業は終了する. 数回のループ終了時か授業終了時などに課題が出されることがあるが, その課題もまた教科書に載せられているものである. 似たような問題を反復演習させてその内容を身に付けさせることを目的として作成された問題なので, 例題プログラムの改変によって解けることがほとんどである.

授業内での演習以外にも, レポートとしてプログラミング課題が出されることが多い. 一回のプログラミング課題に着目すると, 指導者が出したプログラミング課題に対して受講生がレポートを提出し, 評価した結果を受講生に公開するという一連の流れで完結している. この演習モデルを図４に示す.

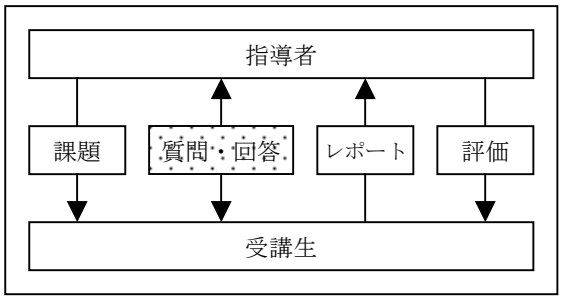

図４ 基本演習モデル（１：１）

このモデルは, プログラミング教育に関わらず, 様々な授業で用いられている. 質問・回答の網掛け部分は, ループとなっていることを示す. 以後, この部分を質問ループと呼ぶ.「(１：１)」というのは, このモデルが一人の指導者と一人の受講生の間のモデルであることを示す. 相手が一人なら質問ループも評価も一人分で済むのだが, 通常, 授業に参加するのは２０や３０といった数である. 受講人数が増えるほど指導者の負担が大きくなっていくのは明白である.また,図４では一度評価を下されることによって一回の演習が完結していることも示されている. 復習が評価面での意味をなさないということである.

４．２ 一般的な授業方式の問題点

一般的なプログラム授業における最大の問題点は, 受講生にプログラミング倫理が身に付きにくいことである. ここでは, 2 章で指摘した指導者と受講生の教育目的の乖離と「評価」の定義を論拠として述べていく.

プログラミング授業の指導者やプログラミング入門書の著者は, プログラムとは求められた結果を出すのが当然であるというプログラミング倫理を自然なものとして受け入れてい

る. 複数のテストデータを使って動作試験を行うと明言された場合, 最低限そのテストデータについては求められる結果を出すプログラムを作るのが常識である.

しかし, 指導者にとって大前提であることはわざわざ明言されることが少なく, その結果, 大前提が受講生に伝わらないということが頻繁に起こる. プログラミング倫理に関しては, 指導者が明言しないことだけでなく, 指導者が仕様を満たしていないプログラムをも評価対象に含んでしまっていることも主要な原因となっている. 指導者が結果を出さないプログラムをどう採点するかについては個人差があると思われるが, 仕様を絶対と考えない受講生の意識は次のようなものである.

○求められた結果を出せば１００点である
○６０点以上なら単位が認定される

まとめると「卒業するため, 単位を取るという目的のためには仕様の６割も満たせば十分である」となる. 実際の授業でどう評価されているかは採点基準が公開されていないので知ることができない. だが, 「仕様を満たして当然だと考えない受講生がプログラミング基礎授業を習得済みとして扱われている」という結果から逆算すれば, それは自ずと明らかになる.

５. プログラミング演習モデルの試行

５. １ 演習モデルの提案

これまでの問題点の指摘事項に対して, それらをほぼ解消する新たな演習モデルを提案する.

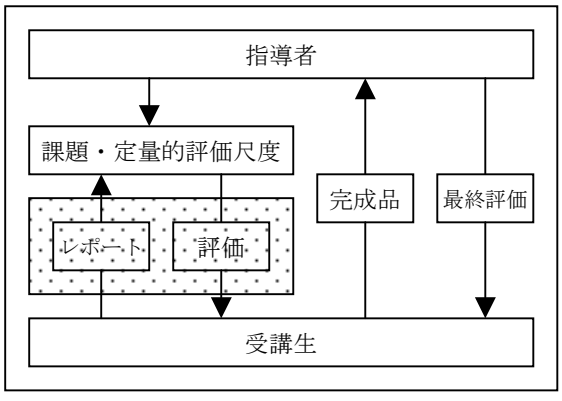

図５ 定量的評価尺度公開演習モデル（１：１）

指導者が課題と同時に定量的評価尺度を示すことで, 受講生がいつでも自分の評価をチェックできるようにしたものである. このモデルを試行した結果, 既に２年間のプログラミング基礎教育を受けてきた学生達に, 現行プログラミング教育では身に付き難い部分を植え付けることに成功した. 試行結果と考察については次節に譲り, ここでは上のモデルの妥当性について述べる.

受講生にとって, 授業内で何を評価されるかという点は最高の関心事である. これまでに出てきた評価に関する問題点を一つずつ挙げていき, その原因について表３にまとめる.

表３ 評価に関する問題点とその原因

問題点	原因
自己評価が通用しない	受講生が評価尺度を知らない
復習がプラスにならない	評価公開日までに開きがある
	一度下された評価は覆らない
指導者の負担が大きい	指導者が評価を行う

ここで, 原因を一つずつ裏返していくことで問題点の解決を図る. 上で提案したモデルは, ４章で示された演習モデルに以下の修正点を加えて再構成したものである.

表４ 評価に関する問題点の原因を裏返した結果

問題点の原因を反転	問題点解消後の状態
受講生に評価尺度を伝える	自己評価通りの結果が出る
リアルタイム評価を行う	復習がプラスになる
受講生が評価を行う	指導者の負担が小さい

実際に何を行うかについて述べる. 受講生は課題の仕様を満たすようなプログラムを作成し, 指導者から与えられた定量的評価尺度を用いてリアルタイムで自らを評価する. 評価を高めるための修正の際に自力でわからないことがあれば指導者に質問する. 評価が満足の行くものとなったときにそれを完成品として指導者に提出する. 指導者は, 受講生に公開したのと同じ評価尺度をもって最終的な評価を下し, それを成績評価に使用する.

指導者と受講生が同じ評価尺度を持つことになるので, 自己評価と実評価の間に差異が生じることは無くなる. もし何かの間違いで狂いがあった場合は, 根拠をもって指導者に意見することができる.「評価尺度自体に問題がある」という疑いがあるのなら, 他の誰かにそれを見せて判断を仰ぐことも可能である.

５. ２ 指導法の試行

プログラミング基礎教育済みの学生に, プログラミング能力の向上を狙って演習を行った. 演習全体の仕様を一般的なものと比較し, 表５に示す.

演習モデルとして, 前節で示した定量的評価尺度公開モデルを使用する. その際に用いる評価尺度は, 指定した４つの入力から「解答」と

一致する出力を得られた場合のみ加点，それができていなければ０点という極端なものとした．これは，プログラミング倫理を育てることを狙ってのことである．

表５　一般的な演習法との比較

	一般的演習法	提案演習法
前提知識	講義で解説	書類で解説
出力仕様	曖昧	一意に定まる
評価基準	非公開	公開

結果は以下のようなものだった．

試験不良率：27%（70/259）
関数個数：3.4 個
main 関数規模：75.6 ステップ
平均関数規模：80.4 ステップ
平均複雑度(サイクロマティック数[5])：22.5

図６　第１回演習結果

仕様を満たしていない場合は自分でそうとわかる仕組みになっているにも関わらず，仕様を満たすことなく提出されたプログラムが約３割存在している．演習中に寄せられた質問は０件だった．これらの結果から，プログラムを組む必要に迫られた際に，他人に頼ってでも完成させることよりも未完成であることを選ぶ受講者が少なからず存在することがわかった．ここで聞き取り調査を行った結果が，４．２で挙げた「全体の６割程度ができているので問題無いかと思った」というものである．現行プログラミング教育ではプログラミング倫理が育たない可能性があるという証左といえる．

５．３　指導法の再試行

演習１の反省を活かして演習２を行った．評価尺度を「全ての問題で仕様を満たして初めて６０点ラインである」とし，関数化やコメント付けの妥当性によってそれ以上の加点を行う．さらに，かなり酷い状態だったプログラミング書法を身に付けさせるため，指導者がプログラムを組む際の手順をゼロから記した資料を用意した．これは思考の道筋の一例を示したものであり，命名規則や関数化のタイミングまで話が及んでいる．教科書のプログラムが最初から完成形として示されているのに対し，本資料は主に過程を示している．

試験不良率：0%（0/147）
関数個数：12.9 個
main 関数規模：9.42 ステップ
平均関数規模：20.1 ステップ
平均複雑度：5.6

図７　第２回演習結果

全員が全課題に対して試験を完遂した結果を報告した．プログラミング倫理を身に付けさせるという目的は達成できたといえる．なお，プログラミング書法についての分析は次の通りである．

演習１と比べると，関数の個数が４倍になり，平均関数規模が４分の１になった．これは機能分割が進んだ結果と考えられる．最大関数規模が小さくなっていることから，各関数の見通しが良くなっていると推定される．これは main 関数で顕著である．これらの傾向は，全て演習２になって配布した資料で順を追って作成されたプログラムに等しい．つまり，指導者の理想通りの展開である．以下に，配布資料内プログラムの関数化状況と演習１から２への変化と配布資料との関係を示す．

表６　関数化能力の成長と資料の関係

	個数	main	規模	複雑度
第１回演習	3.4	75.6	80.4	22.5
第２回演習	12.9	9.4	20.1	5.6
指導者理想	23	7	12	4.7

演習１から２への変化は，明らかに解説資料に近付く方向で起こっている．思考の道筋を示すタイプの資料は，受講生を指導者の理想の形に誘導していくために有効であると言える．

６．　結論

本論文では，全国の大学で行われている FD 活動の流れの中で，プログラミング教育の改善を目的として現行教育の問題点を洗い出した．大学教育の一環として行われるが故の問題点と，プログラミング教育特有の問題点について分析した．それに基づき，これら問題点を解消可能な演習モデルを提案し，試行して結果と考察を示した．その結果，目的としていたプログラミング倫理の徹底が実現できたので，提案された演習モデルは妥当であると言える．

文献

[1] 大学審議会：高等教育の一層の改善について（答申）　（平成９年１２月１８日）
[2] 大学審議会：２１世紀の大学像と今後の改革方策について-競争的環境の中で個性が輝く大学-（答申）　（平成１０年１０月２６日）
[3]島根大学自己評価委員会：教育方法改善企画報告書　（平成１４年９月３０日）
[4]島根大学：ファカルティ・デベロップメント（FD）研修会報告書　（２００１年３月）
[5]TJ McCabe:A Comprexity Measure, IEEE Transactions on Software Engineering, Vol.SE-2, No.4(1976)