コンピュータと教育　58－4
（２０００．１２．１５）

プログラミングの個別指導のための演習問題サーバ

竹田　尚彦、　川口　清志、　浅見　美紀、　佐合　尚子

（愛知教育大学　情報科学選修）

プログラミング教育におけるプログラミング演習は総合的なソフトウェア開発能力の養成するのに欠かせない重要な要素の一つである。しかし、大学教育においては時間的制約や個別対応の難しさから、効果的な演習を実施することが難しいとされている。本研究では、学習者の到達レベルに応じた演習問題を出題することで、効果的なプログラミング演習を行うことを目的としている。そこで、プログラミングカリキュラムを単元に分け、そこで学ぶべき習得項目群と演習問題群を定めた。そして、学習者の達成度に応じた演習問題を出題する演習問題サーバを開発した。本報告では、演習問題サーバの基本的概念と試作について述べる。本サーバを実際の講義で試用したところ、どのレベルの学習者も比較的容易にプログラミング能力が身につくことが分った。

Exercise Server for Progrming Practice

Naohiko Takeda,　Kiyoshi Kawaguchi,　Miki Asami,　Naoko Sagou

(Aichi University of Education　Information and Computer Science)

Programming Practice is one of the most important element of programming education to get the ability of programming. But, it is so difficult that teachers carry on effective training course of programming on their university. So, we discuss the effective training method using the Exercise Server. The Exercise Server give learners a suitable exercise for learners skill. To designing this server, we decided a unit of the training course (Tangen) and training items. And also we selected 2000 exercises and more from twenty textbooks. This paper describe the basic concept of the Exercise Server and its protetype. When we used this server at our programming training course, we made learners to get the ability of programming without non-achiever.

1．はじめに

プログラミング教育では、言語の文法教育と習得した文法事項を適用して実用的なプログラムを作成できるようなプログラミング能力を養成することが重要である。また、同時にアルゴリズムやユーザインターフェースの設計などの能力もプログラミングに必要な能力である。このため通常、プログラミング教育では、言語の文法教育を講義で、実際のプログラミングを演習で行うという形式をとることが多い。

プログラミング演習は、学習者の総合的なソフトウェア開発能力を高めるの重要な役割を果たす。なぜならば、前述したようにプログラミングという行為には、ソフトウェア開発にまつわる全ての要素が少しずつ含まれているからである。

したがって、プログラミング演習を効果的に行うことがプログラミング教育では重要になる。しかし、実際の大学教育で実用的なレベルの教育を行なうことは、困難であると考えられている。その原因は、文法事項を理解することと、プログラムを理解すること、さらに自らプログラムを作成することの間のギャップにあり、それを埋めるような適切な演習方法をとっていないからだと筆者は考えている。

文法事項を理解することは、さほど学習者にとって難しいことではない。しかし、学習者が与えられた文法事項を、どのような場面で使えばいいのか、または使わなくていいのか、などといった実践的な側面は文法事項の例だけでは理解することは難しい。

例題プログラムや他人のプログラムを読むことは、さらに学習者にとって難しい内容になる。プログラムはすべて How（何をすべきか）という内容を中心として記述されるという性質を持っている。つまり、学習者は一文一文ごとの文法的な内容は理解できるが、一連の文がどのような意味を持っているのか、すなわち What（なんの目的で）を理解することが難しいからだ。

さらに与えられた問題を解釈し、解決方法をプログラムという形で示すことは、一層困難な知的作業となる。なぜならば、自分の解決したい問題を解釈し、解決できるようなアルゴリズムを作成し、適切な文法事項を選択し、さらに誤りなくプログラムとして記述しなければならないからである。これができるためには、文法事項、例題、実際のプログラムでの適用の間の知的ギャップが埋められている必要がある。

このように考えると、文法の事項を確認できればよいという簡単なプログラム（いわゆる "toy program"）を作成する程度演習では、総合的なプログラミング能力の養成という観点からすると不十分であることが分る。

工学部、特に情報系学科では、いわゆる工学実験の枠内に十分に時間をかけて大きなプログラムに取り組ませることができため、演習を通じて体験的にプログラミング能力を得ることができる。

一方、プログラミングを二次的な授業科目として扱っている学科や学際的に幅広い内容を取り扱っている学科では、プログラミングは単に動作確認程度であり、十分な内容であると言えない。このため、プログラミング関連授業では実践的な能力が身につかず、プログラミングの教育は不要だと言った議論も起こっている。

しかし、筆者らはプログラミング関連授業を実施することは重要であり、効果が上がっていないとすれば、プログラミング演習の実施方法にこそ、問題があるのだと考えている。

プログラミング演習を通じて、問題分析、設計、文書作成、プログラミング、テストなどシステムエンジニア的技能を養うことも可能である[1]。そのためには、教えるべき文法事項を絞り、Step by Step のカリキュラムを実施し、その上で学習者が能動的にプログラミング演習に取り組む必要がある。

筆者の所属する愛知教育大学総合科学課程

情報科学選修（平成12年4月に情報教育課程に改組）では、この考え方に基づいてシステムエンジニア的な技能の養成も含めたプログラミング教育を行っている[2]。

ここで行っているプログラミング教育では、1)小さなプログラムでも実用性を重視する、2)必ず仕様や設計書を作成させる、3)ユーザインタフェースや例外処理を考慮して作成する、4)プログラム書法を考慮し第三者に分りやすいプログラムを作成するなどの点を重視している。最終的には2年生終了までに、300行程度の実用的なプログラムを設計・制作できるようになることを目標としており、一応の成果をあげている。

一方、このようなプログラミング演習を重視した教育を実施するためには、学習者の個々の理解度や進度も異なるため、演習内容を高度に個別化し学習者の理解度に適合させる必要が生じてくる。しかし、教師が全ての学習者にきめ細かく対応し、必要な演習課題を出題することは容易なことではない。

そこで、本研究では、学習者の技量に合せて自動的に演習問題を出題する演習問題サーバを作成した。本報告では、その考え方と設計方針、試用結果について報告する。

2．プログラミング演習実施上の問題点

プログラミングの講義では、1回の講義が終了すると、演習問題を出題する。通常は学習者全員に共通の問題を出題することが多い。本節では、このような演習の形式における実施上の問題点について述べる。

1)学習者間の進度の差

プログラミングは、いわゆる「実技」（音楽科でいえばピアノ演奏のようなもの）である。いくら他人のプログラムが読めても、自分でプログラミングができなければ意味がない上に、自らプログラミングをしてみないと前述した文法とプログラムの間のギャップを埋められない。また、「実技」であるため、能力レベルによる個人差を無視できない。講義のレベルが学習者のレベルよりも高いと「落ちこぼれ」が発生しやすくなる。逆に講義レベルを下げすぎると、技術の高い学習者が興味を失い「噴きこぼれ」という現象を生む[3]。

2)演習問題の難易度の妥当性

学習者相互の能力差により、ある学習者にとっては簡単すぎる問題や逆に難しすぎる問題が出題されてしまっており、個々の学習者に進度に適応していないことがある。なぜならば、演習問題は単元に関連した、中程度の学習者レベルを想定した問題を出題することが多いからである。したがって、レベルの高い学習者にとっては、容易に解け、かつ面白みのない問題になり、不十分な理解している学習者にとっては、とても難しい問題が出題されることになる。

また、教師にとっても「中程度の学習者レベル」を把握することは難しく、妥当な出題がされているとは限らない。

3)学習者の自力解決が行われない

出題された演習問題は自力でプログラムを作成できない学習者は、できた人のプログラムをコピーしたり、改変したりして自分のプログラムとして提出するようになる。コピーをした学生は多くの場合、プログラムの内容も理解していないことが多い。一方、この風潮が蔓延すると、真面目に演習に取り組んでいる学習者の意欲を低下させる。

なお、自力解決ができないのは、学生ばかりの責任ではない。現代の学生は、受験勉強の影響からか「問に対する答えは唯一」と考える傾向があり、「問に対する解法は様々」であるが、「結果は唯一」であるようなプログラム的な思考とうまく合わない傾向がある。このことについては、また機会を改めて論じたい。

3．演習問題サーバ

3.1.演習問題サーバの発想

前節で述べたようなプログラミング演習上の問題を解決するためには、「学習者のその時点での能力に適合した演習問題を学習者別にに出題」すればよい。

このことにより、1)不必要に難しかったり、逆に簡単すぎる問題が出題されることがない、2)そのため、「落ちこぼれ」、「噴きこぼれ」が生じない、3)個別に問題が出題されるため、不正なコピーができない、4)よく似たプログラムを参考にする場合でも、他人のプログラムを解釈し、自分のプログラムへの応用を考えなければならない等の効果が期待できる。

学習者の能力に適合する演習問題を出題するためには、プログラミング能力を表す客観的な指標を定め、学習者の能力をデータベース化しておく必要がある。一方、また個別に演習問題を出題するためには、多数の演習問題を体系づけて管理するよう、演習問題もデータベース化しなければならない。

本研究では、上記の２つの機能を持ったサーバ「演習問題サーバ」の開発を行った。

このサーバを授業で使用すれば、学習者の力量に適合した問題が出題されるため学習者は無理なく学習を進めていくことができ、カリキュラムを授業進行に合わせて消化していくことができる。この結果、学習者の各々に高度に個別化したプログラミングの学習を進めることができる。また学習者には余分な負担を強いることがないので諦めることなく最後まで学習できる。

3.2.演習問題の収集

筆者ら学習者の能力に合った演習問題を与えかつ、学習者全員に別々の問題が出題されるようにするためには、あらかじめ多くの演習問題を用意しておく必要がある。

そこで、まず市販されており、よく使われているプログラミングの教科書[4]から 2000 問余りの演習問題を収集した。

教科書は特に C または C++に限らず、Fortran や Pascal の教科書からも集めた。これは、Pascal の教科書の方が C/C++の教科書よりも、系統的なプログラミング教育を意図しているものが多く、演習問題もよく考えられたものが厳選されているためである。また Fortran の教科書には、数値計算の問題が豊富に掲載されていたからである。

3.3.学習状況の指標（習得項目）

各学習者の学習状況を客観的に表すために**習得項目**という指標を導入した。

習得項目とは、講義を通じて教えられるプログラミングに関する技術的な事項である。

習得項目は、プログラムの文法事項、プログラミング書法などのについて、できるだけ細かな内容について定められている。

例えば、一番最初の講義で扱う"Hello World!" を表示するプログラムでは、表１にあげたように 20 個の習得項目が設定されている。

表の中で「ヘッダファイル iostream.h を取り込む」「main 関数の中に return 0」を書くなどは C++言語の文法に関する習得項目である。「プログラムのファイル名をコメントにする」「main 関数内のインデントが適切である」などはプログラム書法に関する習得項目である。

習得項目は、「合格」「不合格」「保留」の３レベルの度合で表し、「合格」は、すでにこの項目は習得されたことを表し、「不合格」は未習得であることを表す。理解度が不明確か、該当項目を使わないで、プログラミングした場合には「保留」となる。

// を使ってコメントを書く
/* */ を使ってコメントを書く
一行でコメントを書く
複数行でコメントを書く
プログラムのファイル名をコメントに書く
プログラムの説明をコメントする
プログラムの作成者氏名をコメントする
プログラムの作成日をコメントする
#include 文を適切な位置に書く
ヘッダファイル iostream.h を取り込む
main 関数を書く
main 関数の{ }の位置が適切である
main 関数内のインデントが適切である
main 関数の中に return 0 を書く
cout を使って文字列リテラルを出力する
ストリーム演算子<<１文に１つ使う
文字列リテラルを書く
改行文字¥n を使う
１文が適切な長さで書かれている
タイトルコメント、プリプロセッサ、関数が適切に分かれている

表１　最初の講義での習得項目一覧
（題材は Hello, World）

習得項目は、１回の講義あたり約 20 個程度（少ないところでは数個）ずつ増えていく。学習者がプログラムしたものを採点して、「不合格」「保留」になった習得項目は、未習得項目として、次の回の講義に持ち越される。

習得項目数の多寡が、その学習者の能力として、演習問題サーバの出題に反映される。

3.4.演習問題の分類（学習単元）

3.2.節で述べた 2000 問あまりの演習問題の中から、学習者の能力に適合した問題を選ぶためには、各演習問題とそれを解くために必要な習得項目を関連づけておけばよい。しかし、プログラム課題には、多くの別解が存在する、技術的には平易でも大規模な問題がある、また同じ内容の問題であっても作り方によっては難しくも容易にもなる（例えばソートを含んだような問題）など、一意に習得項目を定めることは難しい。さらに演習問題数が多いため、手作業では困難で、自動化するには規則が曖昧過ぎる。そこで現段階では、各回の講義に対する演習問題として適切なもの数 10 個手作業で選び、まとめておくことにした。

各回の講義のことを「**単元**」と呼ぶ。

演習問題サーバでは、各単元ごとに対応する演習問題を数 10 問保持しており、その中から学習者に適合する内容のものを選んで出題する。

なお、単元は講義の後半になると、２回分の講義をまとめて１単元とする。これは、内容が難しくなり、演習問題を解くのに時間がかかるようになるためである。

表２に関数の利用までの単元と習得項目数を示す。

No.	単元の内容	習得項目数
1	最も簡単なプログラム	20
2	変数と計算	23
3	入力－計算－出力	6
4	条件分岐 if	15
5	if - else	3
6	if - else if	8
7	switch	7
8	while	14
9	for	18
１０	配列と繰返し	11
１１	続・配列と繰返し	1
１２	関数入門	24
１３	関数	4
１４	文字と文字列	17

表２　学習単元と習得項目数

3.5.プログラミング分野以外への応用

演習問題サーバは、カリキュラムを単元ごとに分け、それに対応する習得項目群、単元に対応する演習問題群が定めることができる教育内容（例えば数学の問題集など）であれば、プログラミングに限らず汎用の問題サーバとして用いることができる。

4．演習問題サーバの概要

4.1.講義の流れ

演習問題サーバを用いた講義の流れは以下のようになる。

①教師が講義をおこなう

②課題を出題する

（演習問題サーバ→学習者）

③レポートを提出する

（学習者→教師）

④レポートを採点し、結果を入力する

（教師→演習問題サーバ）

⑤採点結果を学生に返却する

なお、ここで扱うレポートは全て紙に印刷されたものである。

4.2.サーバの構成

演習問題サーバの構成を図１に示す。

演習問題サーバは、サーバ本体と学習者が演習問題のサービスを受けるクライアントの２つの部分からなる。教師はサーバを用いて、演習問題の入力と学習者プログラムの採点結果を入力する。一つの単元の講義を終えると、学習者の以前の成績に基づきその単元に関連した演習問題が個別に出題される。

演習問題サーバのデータ

サーバでは、以下のデータを管理している

- 学習単元　…　各講義で講義する内容。
- 演習問題　…　各単元に対応する演習問題
- 習得項目　…　各単元の習得項目
- 学習記録　…　個々の学習者に出題された演習問題、採点結果、習得項目

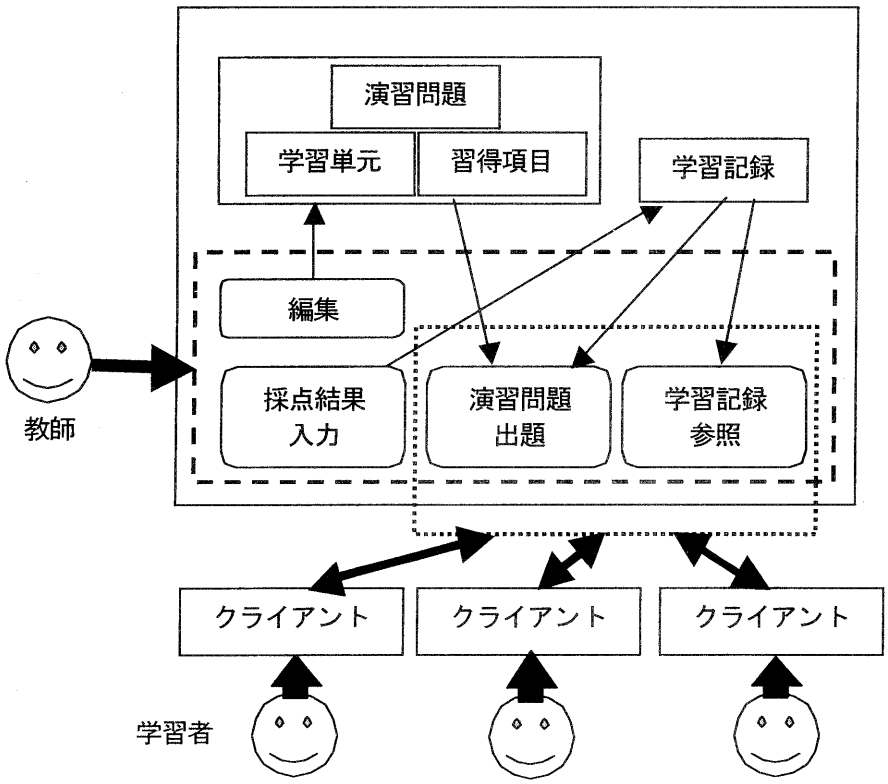

図１　演習問題サーバの構成

教師用インターフェース

教師は、演習問題サーバに対して

1)演習問題の入力・編集

2)学習単元の入力・編集

3)習得項目の入力・編集

4)採点結果の入力・編集

5)学習者登録、学習記録の閲覧

などを行うことができる。

学習者用クライアント

学習者は、学習者用クライアントを用いて、演習問題の出題依頼、過去に出題された問題とその採点記録を参照することができる。

4.2.採点結果の入力

演習問題を出題するには、各学習者の学習記録を参照し、その習得項目の習得状況から、出題する演習問題を決定する。このため、あらかじめ採点結果をサーバに入力しておく必要がある。なお、最初の単元の演習問題は、学習者のレベルが未知であるので、ランダムに出題される。

図２に採点結果の入力画面を示す。

習得項目（入力画面中では「採点項目」と表示）は、教師がレポートを見て採点し、単純に「合格」、「不合格」、「保留」という３段階で評価する。各々の項目を選択し、それぞれのボタンを押せば採点できる。

4.3.演習問題の出題

演習問題サーバでは、採点された習得項目から達成度を計算し、これに基づいて演習問題を出題する。達成度は、全ての出題された習得項目中、「合格」判定された習得項目を割合を表したもので、なおかつ直近に出題された演習問題の結果が反映されるような計算式を用いている。ここでは、経験的に求めた計算式を用いて算出している。

算出された達成度と各演習問題の難易度（それぞれ0～100の値を持つ）からその時点で学習者の達成度に近い演習問題が出題する。この時、既に同じ問題が別の学習者に出題されていたら、別の問題を出題する。

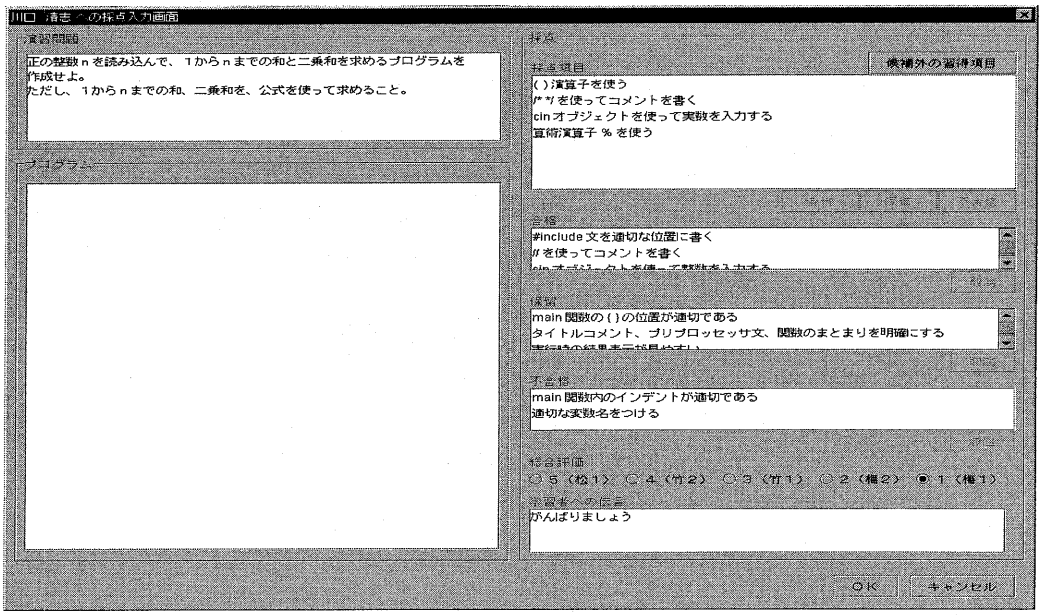

図２　採点入力画面

5．サーバの運用・評価

本学情報科学選修の1・2年生の学生、それぞれ40人に対して1998年度と1999年度の2度にわたり、演習問題サーバを利用したプログラミング教育を実施した。プログラミング言語はC言語で、1年生の学習範囲は入門から制御構造と関数の入門まで、2年生は配列の導入、関数（値引数と参照引数）、アドレスとポインタまでの範囲である。この結果、次のようなことが分った。

1)プログラム書法の定着

習得項目中にプログラム書法や使いやすさを重視したインタフェースなどに関する項目を充実させたところ、定着の度合が高かった。書法や使いやすさに関する議論は、実際の問題に依存して妥当な解が変化するため、単に人のプログラムをコピーするだけでは身に付かない。個別問題が出題されると自分の問題に即して考えないといけないため、定着度が高くなると考えられる。

2)落ちこぼれの回避

従来の単一問題出題方法では、一度理解できなくなると、クラス全体の学習進度に追い付くことが困難であった。演習問題サーバを利用すると、未習得な習得項目は再び演習問題の中に現れ、比較的平易な問題が出題されるため単元の最後まで進めない学生が減少した。

3)教師の採点の手間

本サーバを用いると教師は学習者の提出したプログラムを採点し、所定の習得項目がクリアされているかどうかを全てチェックしていかなければならない。また、サーバの問題提出の精度を向上させるためには、細かな習得項目をも定義しておかなければいけない。このため、紙に手書きでコメントを書き入れる従来の方法と比較すると作業が増加する。

6．まとめ

高度な個別学習を可能とする演習問題サーバについて、考え方とその概要を述べた。学習者側についてのみ見れば、定性的には学習効果が上がっている。但し、定量的な効果の測定は、今後の課題である。特に、このサーバを用いることにより、実際に総合的なソフトウェア開発能力が高まったかどうかの検証は難しい。ただ、高度に個別化することで、学習者の習得項目を参照して、教師が適切な指導方針がたてられるようになり、結果としてきめ細かな指導が可能となる。これによって、結果として効果的なプログラミング演習が可能となるであろう。

一方、達成度の信頼性を高めるためには、多くの習得項目について採点をしなければならならず、教師の負担が増えることになる。そのため、現状の作業手順を変えずに採点の簡略化をはかるための採点ツール[5]を開発中である。

また、演習問題のDB化と習得項目および単元との自動的な関連づけを行なうことにより、演習問題の動的選択を可能にすること、そしてプログラミング以外の教育にも応用できるような一般化についても考えて行きたい。

【参考文献】

[1]竹田、大岩「プログラム開発体験の基づくソフトウェア技術者養成カリキュラム」、情報処理学会論文誌 Vol.33-7 p.p.944～954

[2] 竹田他「情報科学コースにおけるプログラミングカリキュラムの改良」、平成7年度情報処理教育研究集会講演論文集 p.p.470～473

[3]広澤他「習熟度の推移における個人差の影響」に関するメモ

[4]例えば、ダイテル「C 言語プログラミング」、プレンティスホールなど約20冊

[5]浅見、佐合他「手書きのインターフェイスによるコンピュータ上でのプログラム採点」情報処理学会第58回コンピュータと教育研究会報告集