DEWS2007 D9-6

チェックポイントマトリクスに基づく 初心者用プログラミング演習システム

長谷　隆介†　　高橋　直久†

† 名古屋工業大学大学院 工学研究科 情報工学専攻 〒 466-8555 愛知県名古屋市昭和区御器所町
E-mail: †hase@moss.elcom.nitech.ac.jp, ††naohisa@nitech.ac.jp

あらまし　本稿では，チェックポイントマトリクスに基づく初心者用プログラミング演習システムを提案し，実現法について述べる．我々は技術要素名と習得スキルによる組をチェックポイントと呼ぶ．これらを行と列とし，チェックポイントに関する演習問題の正解回数と取り組み回数を値としたマトリクスをチェックポイントマトリクスと呼ぶ．チェックポイントマトリクスにより演習達成度を表す．　提案システムの特徴を以下に示す．(特徴 1) 演習問題の解答例プログラムを解析して，その演習問題に関するチェックポイントのリストを求める機能を実現する．これにより，演習問題を解く前に習得すべきチェックポイントが確認可能となる．(特徴 2) 演習問題に対する答案評価結果から受講者毎にチェックポイントマトリクスを計算し，視覚化する機能を実現する．これにより，受講者に演習の行き詰まりに対する原因を判断する手がかりを与える．(特徴 3) 選択されたチェックポイントに対する補強問題を自動生成する機能を実現する．これにより，技術要素を一つずつ易しい問題から段階的に演習を進めることが可能になる．　また，本稿では実現法に基づいてプロトタイプシステムを実装し，提案システムの妥当性を評価した結果について述べる．
キーワード　e-learning, プログラミング学習, 学習支援システム

A programming excercise system for beginners based on a checkpoint matrix

Ryusuke HASE† and Naohisa TAKAHASHI†

† Department of Computer Science and Engineering Graduate School of Engineering,
Nagoya Institute of Technology
Gokiso, Showa, Nagoya, 466-8555 Japan
E-mail: †hase@moss.elcom.nitech.ac.jp, ††naohisa@nitech.ac.jp

Abstract　This paper presents a programming exercise system for beginners based on a checkpoint matrix in which the row and column of the matrix are a basic unit of exercise and a skill respectively. The value of a matrix element, called checkpoint, is a pair of a ratio of correct answers and a number of trials and showes the level of achievement in the corresponding unit and skill. The proposed system has the following three functions: (1) it finds basic units corresponding to an exercise by the analysis of a correct program for the exercise, (2) it computes the values of checkpoints by evaluating learner's answers, or programs for exercises, and (3) it generates new exercises about a basic unit so as to compensate for weak points of the learners who failed in previous exercises corresponding to the unit. This paper also presents the implementation of a prototype of the proposed system and clarifies the effectiveness of the proposed system based on the experimental results.
Key words　e-learning, Programing Study, Study Support System

1. はじめに

我々は，受講者への問題提示から答案プログラムの評価までの一連の流れ (QA サイクル) を半自動的に実行するプログラミング演習システム [1] を開発し，大学で運用している．このシステムは，教師があらかじめ用意した演習問題を受講者に提示し，受講者の答案に誤りがあれば指摘し，誤りを繰り返してしまう受講者には教師が用意したより簡単な演習問題を受講者に提示することができる．しかし，受講者毎に行き詰まりの原因を発見することは難しく，より簡単な問題を受講者一人一人の

ために教師が作成するのは非常に手間がかかってしまう．

本稿では，行き詰まりの原因となっている受講者の不得意な技術要素の発見を支援すること，及び，技術要素１つ１つを重点的に補強できる問題を自動生成することを目標としたチェックポイントマトリクスに基づく初心者用プログラミング演習システムを提案する．我々は技術要素名と習得スキルによる組をチェックポイントと呼ぶ．技術要素名とは，演習で習得すべき主要な文法要素やライブラリ関数などの項目を表す．習得スキルは技術要素を身につけるために必要な次の３つのスキルを表す．

習得スキル１　技術要素の説明ができる

習得スキル２　プログラムの読解ができる

習得スキル３　プログラムの作成ができる

技術要素名と習得スキルを行と列とし，チェックポイントに関する演習問題の正解回数と取り組み回数を値としたマトリクスをチェックポイントマトリクスと呼ぶ．チェックポイントマトリクスにより演習達成度を表す．

提案システムの特徴を次に示す．

特徴１　演習問題の解答例プログラムを解析して，その演習問題に関するチェックポイントのリストを求める機能を実現する．これにより，演習問題を解く前に習得すべきチェックポイントが確認可能となる．

特徴２　演習問題に対する答案評価結果から受講者毎にチェックポイントマトリクスを計算し，視覚化する機能を実現する．これにより，受講者に演習の行き詰まりに対する原因を判断する手がかりを与える．

特徴３　選択されたチェックポイントに対する補強問題を自動生成する機能を実現する．習得スキルに応じて次の 3 種類の補強問題を生成する

習得スキル１の補強問題　技術要素に関する自然言語で記述された説明の一部を穴埋めで答える問題

習得スキル２の補強問題　技術要素を用いたプログラムを読んで出力結果を問う問題

習得スキル３の補強問題　技術要素を用いたプログラムを作成させる問題

これにより，技術要素を 1 つずつ易しい問題から段階的に演習を進めることができる．

2. チェックポイントリストとチェックポイントマトリクスを用いた演習の進め方

本章では，チェックポイントリストとチェックポイントマトリクスを実際のプログラミング演習でどのように適用できるのかについて例を挙げて説明する．ここでは，全 7 回の演習で 1 回の演習ごとに１０問の演習問題が出題されるプログラミング演習授業 A を想定する．この演習授業の演習ごとの技術要素の分布を表 1 に示す．

表 1 の数値（技術要素出現数）は，その回の演習で用いた演習問題の中で各技術要素を含む演習問題の数であり，技術要素は，演習で学ぶ時期の早い順番に上から並べた．

プログラミング演習授業 A において，受講者１は第１回目の演習において 9 問の演習問題を終了し，10 問目の演習問題で行き詰っている．受講者２は第５回目の演習において，2 問の演習問題を終了し，3 問目の演習問題で行き詰っている．

表 1　プログラミング演習授業 A における技術要素の分布

演習番号	1	2	3	4	5	6	7
printf	9	9	8	9	8	9	8
scanf	4	3	4	5	4	3	4
if	10	4	3	3	4	3	3
else	9	2	1	2	2	1	2
else if	5	1	2	2	2	1	1
do while	0	5	2	3	3	3	2
while	0	5	3	4	3	3	3
for	0	0	10	3	2	3	2
配列	0	0	0	10	4	3	4
関数	0	0	0	0	10	3	2
関数形式マクロ	0	0	0	0	2	1	2
列挙体	0	0	0	0	0	10	1
ポインタ	0	0	0	0	0	0	10

表 2　受講者１のチェックポイントマトリクス

技術要素名	習得スキル１	習得スキル２	習得スキル３
	正解数/取組数	正解数/取組数	正解数/取組数
printf	0/0	0/0	5/9
scanf	0/0	1/1	2/4
if	0/0	0/0	5/10
else	1/1	1/2	4/8
else if	0/0	0/0	1/5
do while	0/0	0/0	0/0
while	0/0	0/0	0/0
for	0/0	0/0	0/0
配列	0/0	0/0	0/0
関数	0/0	0/0	0/0
関数形式マクロ	0/0	0/0	0/0
列挙体	0/0	0/0	0/0
ポインタ	0/0	0/0	0/0

受講者１と受講者２の２つの場合についてチェックポイントリストとチェックポイントマトリクスを例示し，演習に行き詰った受講者に対する演習の進め方を示す．

ケース１　　受講者１の場合

受講者 1 のチェックポイントマトリクスを表 2 に示す．受講者１が取り組んでいる演習問題のチェックポイントリストは [(printf，習得スキル３)(scanf，習得スキル３)(if，習得スキル３),(else if，習得スキル３)]

表 2 をみると，チェックポイントリストにあるチェックポイントの中で最も正答率の低いチェックポイントは (else if, 習得スキル３) である．従って，受講者１のケースでは (else if, 習得スキル３) が原因で演習に行き詰っていると推測できる．提案システムは受講者 1 に (else if, 習得スキル３) の学習をさせるために，(else if, 習得スキル３) の補強問題を提示する．この補強問題が難しくて正解できない場合は，習得スキルを下げて，(else if, 習得スキル 2) の補強問題を提示する．習得スキル２の補強問題も難しくて正解できない場合は，(else if, 習得スキル 1) の補強問題を提示する．

表 3　受講者 2 のチェックポイントマトリクス

技術要素名	習得スキル 1	習得スキル 2	習得スキル 3
	正解数/取組数	正解数/取組数	正解数/取組数
printf	0/0	0/0	24/38
scanf	0/0	1/1	10/18
if	0/0	0/0	15/22
else	1/1	1/2	12/15
else if	1/2	1/3	6/10
do while	0/0	0/0	3/10
while	0/0	0/0	5/13
for	1/1	0/2	4/15
配列	0/0	0/0	4/12
関数	0/0	0/0	2/3
関数形式マクロ	0/0	0/0	0/0
列挙体	0/0	0/0	0/0
ポインタ	0/0	0/0	0/0

ケース 1 のように演習があまり進んでいない状態の時は，弱点と推測される技術要素に関する補強問題を難易度を下げながら出題することにより演習に行き詰った受講者を支援する．

ケース 2　　受講者 2 の場合

受講者 2 のチェックポイントマトリクスを表 3 に示す．受講者 2 が取り組んでいる演習問題のチェックポイントリストは [(printf, 習得スキル 3)(scanf, 習得スキル 3),(if, 習得スキル 3),(for, 習得スキル 3),(関数, 習得スキル 3)]　である．

表 3 をみると，チェックポイントリストにあるチェックポイントの中で最も正答率の低いチェックポイントは (for, 習得スキル 3) である．従って，受講者 2 のケースでは (for, 習得スキル 3) が原因で演習に行き詰っていると推測できる．また，受講者 2 は (for, 習得スキル 1),(for, 習得スキル 2) の補強問題に取り組んでいるようだが (for, 習得スキル 2) の正答率が悪い．従って，受講者 2 は for 文について説明はできるが，プログラムの読解と作成ができない受講者であると推測される．

よって受講者 2 の行うべき学習手順は次の通りである．

[受講者 2 の学習手順]

(手順 1) for 文の読解ができるように教科書などを用いて学習する．

(手順 2) (for, 習得スキル 2) の補強問題に正解する．

(手順 3) (for, 習得スキル 3) の補強問題に取り組む．

(手順 4) 手順 3 で取り組んだ問題に正解した場合は，技術要素に for 文が含まれるまだ正解していない演習問題に取り組む．

(手順 5) 手順 4 で取り組んだ問題に正解した場合は，行き詰っていた演習問題に取り組む．

このように演習がある程度進んでいる状態の時は，弱点と推測される技術要素に関する補強問題を出題したあとに，過去の演習問題から弱点と推測される技術要素を含む演習問題を提示することで演習に行き詰った受講者を支援する．

3.　提案システムの概要

本章ではプログラミング演習システム [1] の動作の概要について述べた後に，提案システムの動作の概要について述べる．提案システムとプログラミング演習システム [1] の構成は図 1 の通りである．

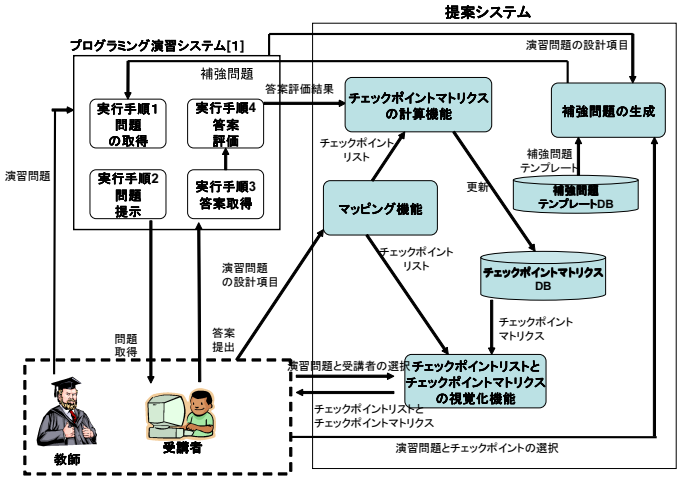

図 1　提案システムの構成

我々は次に示す実行手順を QA サイクルと呼び，プログラミング演習システム [1] は QA サイクルを繰り返すことで演習を進める．

[QA サイクルの実行手順]

(手順 1) 教師が作成した問題を取得する

(手順 2) 受講者に問題を提示する

(手順 3) 受講者の答案プログラムを取得する

(手順 4) 受講者の答案プログラムを評価する．

このような，QA サイクルを繰り返す演習において，演習問題に正解できずに演習に行き詰ってしまった受講者のために提案システムは次の 3 つの動作を行う．

[提案システムの動作]

(動作 1)　チェックポイントマトリクスの計算

プログラミング演習システム [1] の実行手順 4 で生成される答案評価結果を利用して，受講者ごとにチェックポイントマトリクスの計算を行う．チェックポイントマトリクスの計算箇所は，マッピング機能を用いて取得したチェックポイントリストによって決定する．答案評価結果とは，受講者の答案が正解か否かを示すものである．

(動作 2)　チェックポイントリストとチェックポイントマトリクスの視覚化

受講者と演習問題を選択することで，演習問題のチェックポイントリストと受講者のチェックポイントマトリクスの視覚化が行われる．

演習問題のチェックポイントリストは，選択された演習問題に対してマッピング機能を用いて生成する．受講者のチェックポイントマトリクスは，チェックポイントマトリクスデータベースから取得する．

チェックポイントリストとチェックポイントマトリクスを調査分析することで，演習の行き詰まりに対する原因を判断する手がかりを得ることができる．

(動作３)　補強問題の自動生成

選択されたチェックポイントに対する補強問題を自動生成する．補強問題はは，選択されたチェックポイントに対する補強問題テンプレートと現在取り組んでいる演習問題の設計項目（解答例プログラムとテストデータ）から生成する．

生成した補強問題をプログラミング演習システム [1] に登録することで，補強問題を用いて QA サイクルを行うことが可能である．

4.　提案システムの実現方式

4.1　マッピング機能

マッピング機能は，その演習問題に取り組むことで学習できるチェックポイントを解析し，チェックポイントリストを生成する機能である．

[マッピング機能の実行手順]

(手順 1)　　解答例プログラムの解析

演習問題の解答例プログラムを解析し，演習問題に関係する技術要素名を得る．

(手順 2)　　習得スキルの判別

あらかじめ教師が設定した演習問題の種類から習得スキルを自動判断する．

(手順 3)　　チェックポイントリストの生成

技術要素名と習得スキルからチェックポイント決定して，チェックポイントリストを生成する．

4.2　チェックポイントマトリクス計算機能

チェックポイントマトリクスの計算は，受講者の答案の評価結果を元に行う．受講者が演習問題と補強問題に取り組んだ場合の実行手順は次の通りである．

[演習問題に取り組んだ場合のチェックポイントマトリクス計算機能の実行手順（図 2)]

（手順 1）　　チェックポイントリストの取得

マッピング機能によって，答案を作成した演習問題の解答例プログラムからチェックポイントリストを求める．

（手順 2）　　チェックポイントマトリクスデータベースの更新

演習支援システムから受講者の答案評価結果を受け取る．答案が正解であった場合，取得したチェックポイントリストに存在するチェックポイントの正解回数と取り組み回数を１加算する．答案が不正解であった場合は，取り組み回数のみを１加算する．

[補強問題に取り組んだ場合のチェックポイントマトリクス計算機能の実行手順]

(手順 1)　　チェックポイントリストの取得

補強問題の設計項目から対応するチェックポイントを取得する．補強問題は，チェックポイントごとに補強学習を行う問題であるため．補強問題自体にチェックポイントが設定されている．

(手順 2)　　チェックポイントマトリクスデータベースの更新

演習支援システムから答案評価結果を受け取る．受講者の答案の評価結果が正解であった場合，補強問題に設定されたチェックポイントの正解回数と取り組み回数を１加算する．受講者の答案が不正解であった場合は，取り組み回数のみを１加算する．

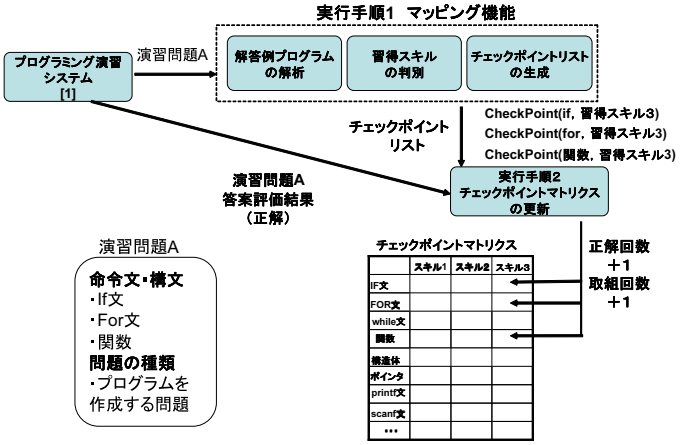

図 2　チェックポイントマトリクスの計算手順

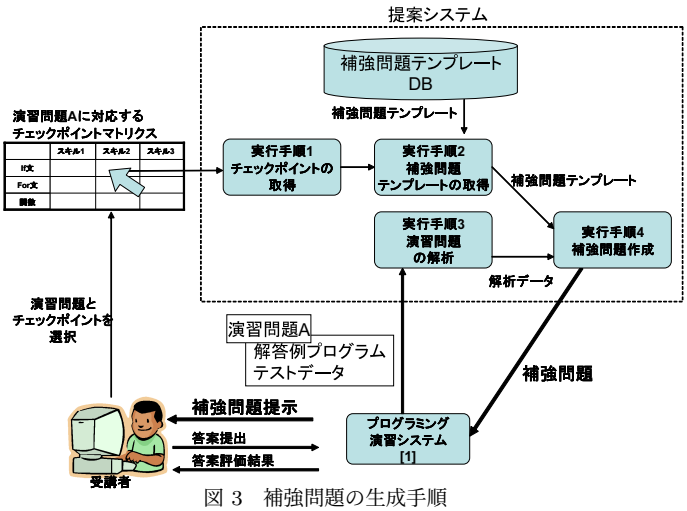

図 3　補強問題の生成手順

4.3　補強問題の生成機能

技術要素に対して定義を説明させる補強問題，プログラムを読解させる補強問題，及び，プログラムの一部を作成させる補強問題の 3 種類の補強問題を生成する．補強問題で学んだことを演習問題に応用しやすいように，補強問題中に演習問題で使用されていた変数，仮引数，及び，制御式等を用いる．補強問題は，受講者が演習問題と補強したいチェックポイントを選択することで生成される．

[補強問題の生成手順 (図 3)]

（手順 1）　　演習問題とチェックポイントの取得

受講者が現在取り組んでいる演習問題と受講者によって選択された補強したいチェックポイントを取得する．

（手順 2）　　補強問題テンプレートの取得

選択されたチェックポイントに対応する補強問題テンプレートを補強問題テンプレートデータベースから取得する．補強問題テンプレートには，問題文，解答，及び，習得できるチェックポイントが設定してある．問題文と解答にタグ（表 4）を記述することで，タグに対応する解析データが挿入される．技術要素によって，補強問題テンプレートで使用可能なタグは異なる．技術要素名と使用可能なタグの関係を表 5 に示した．

（手順 3）　　演習問題の設計項目の解析

演習問題の設計項目には，問題文，解答例プログラム，及び，解答例プログラムを実行するためのテストデータがある．提案システムは，解答例プログラムとテストデータを解析することで，取得した補強問題テンプレートに存在するタグに挿入するデータを生成し取得する．

（手順 4）　補強問題の作成

補強問題の生成手順 2 で取得した補強問題テンプレートのタグに補強問題の生成手順 3 で取得した解析データを挿入することで補強問題を作成する．

表 4　挿入用タグ

ID	タグ	意味
1	<control>	制御式
2	<translate>	制御式を自然言語に翻訳
3	<initialization>	for 文の前処理式
4	<increment>	for 文の後始末式
5	<select>	制御式から生成した選択肢
6	<selectans>	制御式から生成した選択肢の正解番号
7	<vname>	変数名
8	<vtype>	変数の型
9	<funcname>	関数・関数形式マクロ名
10	<functype>	関数の戻り値の型
11	<parameter>	関数・関数形式マクロの仮引数の宣言文
12	<fce>	関数・関数形式マクロ呼び出し式
13	<structurename>	構造体名
14	<member>	構造体のメンバ変数の宣言文
15	<etag>	列挙体のタグ名
16	<econstant>	列挙定数の定義文
17	<declare>	解析データ中の変数の宣言文
18	<assign>	解析データ中の変数の初期化
19	<output>	補強問題中のプログラムの実行結果

表 5　技術要素と使用可能なタグの関係

技術要素名	解析すべきタグの ID
printf	12,17,18,19
scanf	12,17,18,19
puts	12,17,18,19
putchar	12,17,18,19
if	1,2,5,6,17,18,19
else	1,2,5,6,17,18,19
else if	1,2,5,6,17,18,19
swich	1,17,19
do while	1,2,5,6,17,18,19
while	1,2,5,6,17,18,19
for	1,2,3,4,5,6,17,18,19
配列	7,8
ポインタ	7,8
関数	9,10,11,12,17,18,19
関数形式マクロ	9,11,12,17,18,19
列挙体	15,16,19

補強問題は，チェックポイントの習得スキルによって異なる．生成する問題は次の 3 種類である．

補強問題１　技術要素に関する説明が自然言語で記述された定義の一部を穴埋めで答える問題

技術要素の定義を理解（習得スキル１）のチェックポイントが選択された場合に生成する．

補強問題２　技術要素を用いたプログラムを読んで出力結果を問う問題

技術要素を用いたプログラムの読解（習得スキル２）のチェックポイントが選択された場合に生成する．

補強問題３　技術要素を用いたプログラムを作成させる問題

技術要素を用いたプログラムの作成（習得スキル３）のチェックポイントが選択された場合に生成する．

4.3.1 （if, 習得スキル 1）に関する補強問題生成法

チェックポイント（if, 習得スキル１）に関する補強問題を生成する方法について述べる (図 4) .（if, 習得スキル 1）に関する補強問題の生成の手順を次に示す.

[（if, 習得スキル 1）に関する補強問題の生成の手順]

(手順 1)　演習問題とチェックポイントの取得

受講者が選択した演習問題とチェックポイントを取得する.

(手順 2)　補強問題テンプレートの取得

選択されたチェックポイントに対応する補強問題テンプレートを補強問題テンプレートデータベースから取得する.

(手順 3)　解答例プログラムの解析

取得した補強問題テンプレートに control タグがあるので，演習問題の解答例プログラムから制御式を抽出する.

(手順 4)　補強問題の作成

補強問題の生成手順 2 で取得した補強問題テンプレートのタグに補強問題の生成手順 3 で取得した解析データを挿入することで補強問題の問題文と解答を取得する．さらに，テンプレートに設定されたチェックポイントを補強問題のチェックポイントとして設定することで，補強問題に必要な問題文，解答，チェックポイントを得る.

4.3.2 （if, 習得スキル 2）に関する補強問題

（if, 習得スキル２）で表されるチェックポイントに関する補強問題を生成する方法について述べる (図 5).（if, 習得スキル 1）に関する補強問題の生成手順と比べ，手順３が異なる.（if, 習得スキル２）に関する補強問題の生成手順３を次に示す.

- **演習問題設計項目の解析**

取得した補強問題テンプレートにある挿入用タグを解析する．演習問題の解答例プログラムを解析して contol タグに挿入する制御式を取得する．解答例プログラムを解析して制御式に用いられている変数の型を取得し，declare タグに挿入する文字列を生成する．解答例プログラムを実行することで制御式に用いられている変数に入力されている値を取得し，assign タグに挿入する文字列を生成する．program タグで囲まれた部分のタグに，取得・生成した文字列を挿入してプログラムを作成する．作成したプログラムを実行することで，output タグに挿入する出力結果を取得する.

4.3.3 （if, 習得スキル 3）に関する補強問題

（if, 習得スキル３）で表されるチェックポイントに関する補強問題を生成する方法について述べる (図 6).（if, 習得スキル

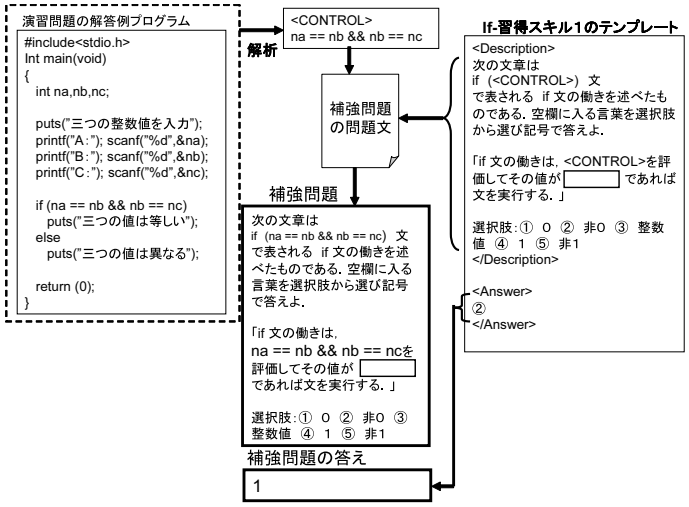

図 4　補強問題生成法（if-定義理解に関する補強問題）

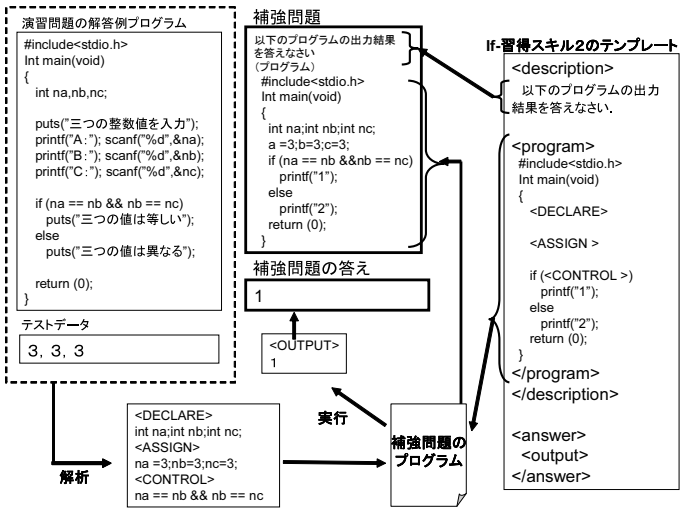

図 5　補強問題生成法（if-読解に関する補強問題）

1）に関する補強問題の生成手順と比べ，手順 3 が異なる．(if, 習得スキル 3）に関する補強問題の生成手順 3 を次に示す．

- **演習問題設計項目の解析**

取得した補強問題テンプレートにある挿入用タグを解析する．演習問題の解答例プログラムを解析して制御式を取得する．取得した制御式を自然言語に翻訳することで，declare タグに挿入する文字列を生成する．取得した制御式の関係演算子や論理演算氏を変えることで，select タグに挿入する選択肢を生成する．同時に，正解の選択肢番号を取得し，selectans タグに挿入するための値とする．

5. プロトタイプシステムの実装

提案システムの有効性を確認するためにプロトタイプシステムを実装した．本プロトタイプシステムは問題生成処理として，補強問題生成機能を実装した．また，演習問題とチェックポイントの対応付け機能としてマッピング機能を実装した．補強問題生成機能は，html 化した Web ページから演習問題とチェックポイントを選択する方法で実現した．マッピング機能は，同様に html 化した Web ページから演習問題を選択する方法で実現した．

5.1 補強問題生成機能のプロトタイプシステム

演習問題の選択は，プログラミング演習支援システム [1] の問題データベースから問題データを取得し，問題名で選択できるように実現した．演習問題の選択を行うと，マッピング機能によってチェックポイントリストが Web ページに表示され，各チェックポイント選択できる．演習問題とチェックポイントを選択することで補強問題が自動生成される．

5.2 マッピング機能のプロトタイプシステム

演習問題の選択は，プログラミング演習支援システム [1] の問題データベースから問題データを取得し，問題名で選択できるように実現した．演習問題を選択すると，技術要素名と習得スキルのリスト（チェックポイントリスト）が表示される．

プロトタイプシステムは Java 言語 [4] を用いて実装した．また，Web ページを作成するために Tomcat [5] を利用した．また，補強問題テンプレートデータベースをリレーショナルデータベース管理システム MySQL [6] を用いて実現した

6. 提案システムの評価と考察

提案システムの妥当性を確認するために，プロトタイプシステムを用いて評価実験を行った．名古屋工業大学情報工学科の C 言語によるプログラミング演習授業で用いられた演習問題にプロトタイプシステムを適用し，その結果を元に考察した．この演習授業は初心者を対象にして，1 年間で演習番号 1 から 1 0 までの演習が行われ，７５問の演習問題が出題された．

6.1 技術要素の分布調査

6.2 実験の目的

提案システムは，技術要素や習得スキルごとに学習の進捗状況を視覚化することで，演習問題を解く前に学習すべき不得意な技術要素を発見することができる．

提案システムを用いない場合，過去に学習した技術要素が後の演習で繰り返し用いられるような演習授業では，現在学習中の技術要素以外の技術要素も学習する必要がある．よって，今まで学習したすべて技術要素の中から不得意な技術要素を発見

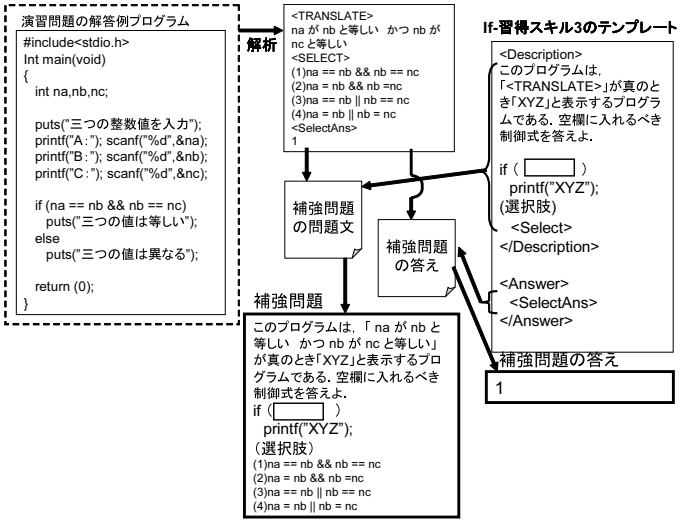

図 6　補強問題生成法（if-作成に関する補強問題）

できなければ，演習問題に正解することができない．

ここでは，実際の演習授業に過去に学習した技術要素が後の演習で繰り返し用いられるような傾向があるのか調査することで，技術要素や習得スキルごとに学習の進捗状況を視覚化することの有効性を評価する．

6.2.1 実験方法

マッピング機能によって出題されたすべての演習問題のチェックポイントリストを取得することで，演習毎に，技術要素の分布を調査した．プロトタイプシステムを用いて以下の調査手順を行い，調査結果を表 6 に示した．

調査手順１ マッピング機能を用いて，演習問題に対するチェックポイントリストを生成する．

調査手順２ 調査手順１で取得したチェックポイントリストからチェックポイントの数を集計する．

調査手順３ 調査手順 1-2 をすべての演習問題に対して行う．

調査手順４ 集計結果を，演習番号毎にまとめる．

6.2.2 実験結果

表 6 の数値（技術要素出現数）は，その回の演習で用いた演習問題の中で各技術要素に関する演習問題の数であり，技術要素は，演習で学ぶ時期の早い順番に上から並べた．

表 6　技術要素の分布

演習番号	1	2	3	4	5	6	7	8	9	10
問題数合計	7	5	10	16	7	8	6	6	5	5
printf	7	5	9	16	7	8	6	6	5	5
scanf	5	5	10	15	2	7	6	4	3	4
puts	3	3	8	3	2	4	2	4	2	0
if	0	0	8	9	0	3	5	2	2	2
else	0	0	5	4	0	1	0	0	1	1
else if	0	0	2	0	0	0	0	0	0	1
swich	0	0	2	0	0	0	0	1	0	0
do while	0	0	0	4	0	0	0	1	0	0
while	0	0	0	6	0	0	6	3	0	3
for	0	0	0	6	7	4	5	1	1	1
putchar	0	0	0	5	1	1	6	0	0	2
配列	0	0	0	0	7	4	0	0	2	5
関数	0	0	0	0	0	8	6	2	3	4
関数形式マクロ	0	0	0	0	0	0	0	2	0	0
列挙体	0	0	0	0	0	0	0	1	0	0
ポインタ	0	0	0	0	0	0	0	0	5	0

6.2.3 考察

表 6 をみると，1 つの演習でしか用いられない技術要素は，関数形式マクロ，列挙体及びポインタの３つであり，ほとんどの技術要素が複数の演習で用いられている．また，演習全体で見たときに，早い時期の演習で出現する技術要素の出現数は多く，出現する時期の遅い技術要素ほど出現数は少なくなる．このことから，実際の演習授業において過去に学習した技術要素が後の演習で繰り返し用いられる傾向があることがわかった．よって，実際の演習授業においても，今まで学習したすべて技術要素の中から不得意な技術要素を発見する必要があるといえる．

このことから，技術要素や習得スキルごとに学習の進捗状況を視覚化することで，演習問題を解く前に学習すべき不得意な技術要素を発見することができる提案システムは有効であるといえる．

6.3 1 つの演習問題に対する技術要素の個数調査

6.3.1 実験の目的

受講者の学習の進捗状況を閲覧する時に，問題毎の正誤結果を閲覧する方法では生徒の不得意な技術要素を特定することはできない．これは，1 問の演習問題には，複数の技術要素に関する知識が必要であることに起因する．そのため，1 問の演習問題に対する技術要素が多ければ多いほど，問題毎の正誤結果を閲覧する方法では不得意な技術要素を特定しにくいといえる．

ここでは，1 問の演習問題に対する技術要素を個数を求めることで，技術要素ごとに進捗状況を視覚化する方式の妥当性を評価する．

6.3.2 実験方法

6.1 で行った実験の結果（表 6）を用いて，1 つの演習問題に対するチェックポイントの個数の平均値を求めた．

6.3.3 実験結果

1 つの演習問題に対するチェックポイントの個数の平均値を表 7 に示す．演習が進むにつれて，演習問題 1 問あたりのチェックポイント数は増加していることがわかった．

6.3.4 考察

演習が進むにつれて，演習問題 1 問あたりのチェックポイント数は増加している．これは，演習の学習効果により，生徒が多くの技術要素を含んだ難しい問題でも作成できると見越して出題されたと推測できる．この結果から，演習が進むにつれて，問題毎の正誤結果を閲覧する方法では，生徒の不得意な技術要素を特定することはできないことがわかった．従って，技術要素ごとに進捗状況を視覚化する方式は妥当であるといえる．

表 7　1 問あたりのチェックポイント数

演習回数	1 問あたりのチェックポイント数
第 1 回	2.1
第 2 回	2.6
第 3 回	4.9
第 4 回	4.3
第 5 回	3.7
第 6 回	5
第 7 回	7
第 8 回	4.5
第 9 回	4.8
第 10 回	5.6

6.4 補強問題の自動生成に関する評価と考察

6.4.1 実験の目的

提案システムでは，補強問題を自動で生成する方法をとっている．これは，手作業ですべての補強問題を作成した場合，教師に非常に大きな負担がかかってしまうためである．ここでは，プロトタイプシステムが補強問題を生成する時間と手作業で問題を作る時間を比較することで提案システムの自動生成法に関する妥当性を示す．

6.4.2 実験方法

補強問題の作成手順は次の３つである．

作成手順１　演習問題の選択

作成手順２　チェックポイントの選択

作成手順３　補強問題の作成

提案システムでは，手順３をシステムが自動で行うため教師の負担にはならない．そこで，手入力で手順３を行うのにかかる時間が，自動生成を行わないことで余分にかかってしまう時間であるといえる．実験は，問題作成に慣れた問題作成者が無作為に選んだ３問の演習問題から無作為に技術要素を選択し，各々の技術要素を持つ３つチェックポイントに関する補強問題（９問）を作成する時間を計測した．作成した補強問題の問題文中には，演習問題の技術要素で用いられた変数あるいは仮引数を用いた．

6.4.3 実験結果

実際に補強問題を作成し時間計測を行った結果を表８に示す．時間計測は，９問の補強問題を習得スキルによって３つ分類した．補強問題作成時間はそれぞれの習得スキルの補強問題を作成するのにかかった時間の平均値である．表６から，この演習授業における技術要素の出現数の合計は 327 である．演習授業においてすべての補強問題を手作業で作成するのにかかる時間を求めるには,「3 種類の補強問題の合計作成時間」に技術要素の出現数をかければよい．したがって，この演習授業では１３４時間かかることになる．

表８　作成時間

補強問題の種類	補強問題作成時間
定義を理解する補強問題	9m25s
プログラムを読解する補強問題	6m20s
プログラムを作成する補強問題	8m25s
3 種類の補強問題作成時間の合計	24m10s

6.4.4 考　　察

1 年間の演習授業において教師が手作業ですべて補強問題を生成するには提案システムが自動生成するのに比べて１３４時間も余分に費やす必要があることがわかった．限られた問題作成時間の中で，このような膨大な時間をかけることはできないため，自動で補強問題を生成する方法は妥当であるといえる．

7. 関連研究

ソースコードを再利用するために，既存のソースコードからプログラミングスライシングにより，ソフトウェア部品を作成する手法 [2] が提案されている．この手法では，再利用可能なソフトウェア部品の切り出しを行っている．同時に，付加情報として切り出した部品の理解や変更を容易にするために，部品を実行するために必要な実行前提条件例と部品がどのように利用されているかを示す利用条件例を生成している．部品本体と付加情報により，必要最低限の記述コードで，実行可能な部品の生成を行える．実行前提条件例は，ソースコード不必要な記述を削除することで生成している．

提案システムでは，単一の技術要素を補強する問題を生成する機能を実現する．補強問題の生成法として，受講者が行き詰っている演習問題の解答例プログラムにあるソースコードの一部を再利用して新たな問題を生成する方法を用いる．しかし，スライシングを用いた部品の切り出し法では部品以外の技術要素が入ってしまう危険性があるため，単一の技術要素を補強する問題の生成には適用できない．提案システムでは，演習問題を実行することで部品が呼び出される直前の変数値を取得する．これにより，スライスの計算を行わずにソースコードの一部を再利用できるようにする．

演習の行き詰まりを検出する方法としては，プログラミング演習の作業履歴の解析による演習の行き詰まり検出とその評価 [3] がある．この研究では，プログラミング演習における作業履歴からエラーの修正に要した時間に着目して行き詰まりを検出している．この手法では，複数の技術要素が絡み合った演習問題においては，何が原因で行き詰っているのかを判断することができない．

提案システムでは，チェックポイントリストとチェックポイントマトリクスを用いることでどの技術要素が理解できていないのかを判断する手がかりを与える．

8. おわりに

本研究では，チェックポイントマトリクスに基づく初心者用プログラミング演習システムの実現法を提案した．得られた成果は次の通りである．

- 演習の行き詰まりに対する原因を判断する手がかりとなるチェックポイントリストとチェックポイントマトリクスの提示が可能となった．
- チェックポイントごとに重点的に補強学習ができる問題の自動生成が可能となった．

今後は，チェックポイントマトリクスの計算・視覚化機能のプロトタイプシステムを実装し実際の演習授業に適用することで，実際に行き詰まりの原因を発見し解消することができるかどうか検証する．

文　　献

[1] 中島秀樹, 高橋直久, 細川宣秀, プログラミング学習のための QA サイクル-受講者の習得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,VOL.J88-D-I,NO.2,pp439-450,2005

[2] 丸山勝久，高橋直久: 区間設定可能なプログラミングスライシングを用いたソフトウェア部品の作成．情報処理学会 論文誌,Vol.37,No.4,pp.520-535,April 1996.

[3] 齋藤吉範: プログラミング演習の作業履歴の解析による演習の行き詰まり検出とその評価．http://www.swlab.ice.uec.ac.jp/grad_paper/98yosinori.pdf, 電気通信大学 情報通信工学科 渡辺成良研究室卒業論文　1998.

[4] JavaTM 2 Platform Standard Edition 1.4.0 API 仕様, http://java.sun.com/j2se/1.4/ja/docs/ja/api/index.html

[5] The Apache Software Foundation,”Apache Tomcat”, http://tomcat.apache.org/

[6] MySQL AB,”MySQL:The world’s most popular open source database”, http://www.mysql.com/