2007－CE－91 (3)
2007／10／5

実行テストによるプログラム判定を用いた初級Cプログラミング演習支援と授業実践

倉田 英和, 富永 浩之, 林 敏浩, 垂水 浩幸

香川大学工学部 〒761-0396 香川県高松市林町 2217-20
E-mail: s06g462@stmail.eng.kagawa-u.ac.jp

あらまし 大学情報系学科の入門的Cプログラミング授業において，演習を活性化させる初心者向けのコンテスト形式を提案する．その支援環境として，実行テストによるプログラム判定をサーバ側で行うシステムを開発した．実行テストでは，部分的な解答も許容する複数の予備テストを導入する．段階的な予備テストを中間目標として与え，解法のヒントや実装手順を示すことで，苦手な学生にも学習意欲を持たせる．実行テストの具体例や得点ルール実際の授業での運用についても報告する．

キーワード C言語演習，プログラミングコンテスト，実行テスト，大会運営サーバ

Introductory C Programming Exercise with Program Judgment System by Execution Test Series

Hidekazu KURATA, Hiroyuki TOMINAGA, Toshihiro HAYASHI, Hiroyuki TARUMI

Faculty of Engineering, Kagawa University 2217-20 Hayashi, Takamatsu, Kagawa, 761-0396 Japan
E-mail: s06g462@stmail.eng.kagawa-u.ac.jp

Abstract C language is the necessary subject for computer engineering education in college. For effective and enjoyable exercise in daily lesson, we propose a contest style exercise in classroom. We develop tProgrEss, the contest management Web server. After uploading student's source code, the server judges the program by execution test with input and output data. To adjust a contest for beginners, we offer two stages of the test by given some samples. The several preparation tests are for step-by-step subgoals. They suggest a guideline of coding with program structure. They make students have custom of testing. The current situation of the tests is exhibited in the Web page to raise competition volition. The achievement test is the final check to give adequate points concerned with answering time and trial frequency. The purpose of our research is to promote their learning motivation with team competition. And it is to give sense of satisfaction and achievement with solving problems. We prepared some problems and performed practical exercises using tProgrEss.

Keyword C language exercise, programming contest, execution test series, support server

1. はじめに

大学の情報系学科の多くは，C言語教育を重視し，初年時の必修科目にしている．C言語には説明すべき文法事項が多く，最も重要な演習に十分な授業時間を確保しにくい．また，理解度が異なる多様な学生が受講しており，進捗状況や演習態度に大きな差が生じやすい．例えば，プログラミングの得意な学生が退屈したり，分かったつもりで後回しにしたり，苦手な学生が途中で解答を諦めてしまう．演習の効果を高めるには，授業の最後まで集中して取り組ませる工夫が必要である．特に，簡単な問題でも実際

に解いたという体験を積み重ね，繰り返し達成感を与えることが重要である．

一方，学生の多くは，コンパイル成功で安心してしまい，簡単な実行確認だけで正しいプログラムが完成したと考えがちである．実際には，多くの入力サンプルで出力データを十分に吟味すること，実行結果からデバッグすべき点を見つけることを意識させる必要がある．また，プログラムの書法や構造を把握ができず，演習が進まない学生も多い．書法としては，二重配列の走査や番兵による入力打切など，一連の文法事項の組合せをパターンとして応用する．構造としては，初期化，入力，計算，例外処理，出力など，プログラムの大きな流れを把握して段階的に作成する．これらを習得させるには，コードの単なる穴埋めではなく，仕様や手順をコメントの形で与えたテンプレートなどで，プログラム作成の指針を示すことが重要である．

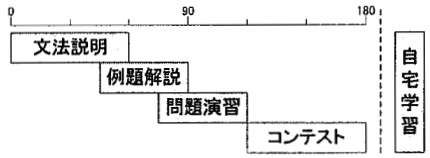

図 1 授業進行と総まとめのコンテスト

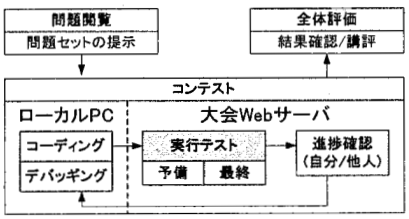

図 2 コンテスト形式の演習の進行

2. コンテスト形式の入門的 C 言語演習

2.1. コンテスト形式の演習の概要

入門的なプログラミング授業では，標準的な形態は，以下のようになる．まず，最初に文法事項などを講義で説明し，幾つかの例題を解説した後，基本的な問題の演習に入る．演習問題は，その日の例題の発展や類題である．授業後には，応用的な課題について宿題を出す．本研究では，コンテスト形式を取り入れたCプログラミング演習を提案する[1]．授業最後の総まとめ的な位置付けとして(図1)，演習を活性化させ，自宅学習に繋げる．また，段階的なプログラム作成を実行テスト系列で支援する大会サーバtProgrEssを開発している[2]．本論では，実行テストの具体的な構成方法を論じ，プログラムの自動判定と演習の進捗状況を公開するサーバの機能を述べる．実際の授業での運用についても報告する．

2.2. 初心者向けのプログラミングコンテスト

近年，プログラミング教育において，競争学習(competitive learning)が注目を浴びている[3]．特に，一定の時間内にプログラミング問題を解く競技系のコンテストを授業に取り入れることが考えられる．コンテストでは，ゲーム感覚のイベントとして，勝つという目的意識や，学生同士の対抗意識が，学習意欲を高める．また，入力に対する正しい出力という明確な判定基準が設けられており，即時的な評価で結果が分かり，学生の不公平感も生じにくい．オープンなコンテストとしては，大学生向けでチーム対抗の ACM-ICPC[4]などがある．

しかし，これらは，レベルの高い参加者を対象としており，初心者への敷居が高く，毎回の授業で気軽に取り組めるものではない．入門的な授業にコンテストを取り入れるには，初心者向けに裾野を広げる適応が必要である．各自の実力に応じた参加を促進するため，幅広い題材から，難易度と配点の異なる複数の問題を用意する．簡単な問題を多く解くか，少ない難問に挑戦するか選択でき，どちらの戦略でも得点のアンバランスが生じないようにする．また，ゲームやパズルの題材を取り入れる，親しみやすい問題設定を行う，進捗状況を短いサイクルで

把握して意欲を持続させる，ある程度の失敗も許容する，最後まで解けなくても何らかの評価を与える，などの工夫も必要である．

2.3. 演習の進行

コンテスト形式の演習の進行は，以下の 3 つのフェーズで捉える(図 2).

(1) 問題閲覧フェーズ

複数の問題で構成された問題セットが提示される．学生は問題文や必要となる文法事項などから，自分の実力に応じた問題を選択する．

(2) コンテストフェーズ

コンテスト中の学生は，各自のローカル PC を利用し，選択した問題のコーディングを行う．解答が完成したと考えれば，サーバ側にソースコードを提出する．サーバ側では，用意された実行テストを自動で行い，プログラムの正誤を判定する．学生は，判定結果を受け，自分と他人の進捗結果を確認する．プログラムに誤りがある場合は，判定結果を元にして，デバッグを行う．以上を繰り返し，制限時間中に問題を多く解き，高得点を目指す．

(3) 全体評価フェーズ

コンテストの終了後，結果の確認と教師からの講評を行い，その日の演習が終了する．限られた演習の中で，問題を解いたという実感や，解けなくて悔しい，などの印象を学生に持たせ，演習後の課題や復習への動機付けを行う[5].

2.4. 学習段階とコンテストの親和性

我々の大学の情報系学科では，初級 C 言語授業は，半期 15 回で実施されている（表 1）．我々は，それを大きく三段階に分けて捉えている(図 3)．特に，第二段階において，コンテストが有効であると考えている．

(1) 第一段階

コーディングとコンパイルの作業を体験する．また，基本的な文法事項を理解することが中心である．ここでは，コンテスト問題は，簡単でそれほど面白いものとはならない．

(2) 第二段階

配列や関数を扱い，二重反復や局所脱出などを用いて，様々なデータを処理する．番兵による入力終了や，入力データのチェック，配列の走査など，制御構造の語法が重要となる．また，題材として，統計や行列，盤面ゲーム，表計算のような多彩なものが扱えるようになる．使える文法や題材が増えるため，プログラミング問題が作りやすくなる．この段階では，題意を正しく読み取り，複数の入力データで検証を行えるようになることが重要となる．

(3) 第三段階

ポインタや構造体などを扱う．入出力だけでなく，コード内でどのようなデータ構造を用いているか，どのような設計を行っているか，といった点も調べなければ，適切な判定が難しい．機械的な実行テストだけでなく，教師によるソースコードの精査が必要である．

表 1　授業計画と学習内容

授業計画		学習内容
入門編	第01回	DOS環境、C言語の概要、言語処理系
	第02回	変数とデータ、演算子と式、入出力、コメント
	第03回	選択構文、論理式、数学関数、定数定義マクロ
	第04回	反復構文、構文要素と書式、基本制御構造
	第05回	多重反復、二分法、局所脱出、基本データ型、文字
初級編	第06回	関数の定義と呼出、基本的な数値算法、時間と乱数
	第07回	配列の宣言と参照、配列の基本操作、データ型名定義
	第08回	多次元配列、行列と整式の算法、ソースの分割
	第09回	単純整列算法、集合演算の算法、ビット処理
	第10回	再帰的処理、変数と宣言、関数形式マクロ
中級編	第11回	文字列、書式付入出力、文字種判定、端末制御
	第12回	文字列とポインタ、文字列処理、文字列検索
	第13回	構造体、共用体、列挙体、データの抽象化
	第14回	配列とポインタ、構造体とポインタ、メモリ管理
	第15回	ファイル入出力、実行時引数、テキスト加工

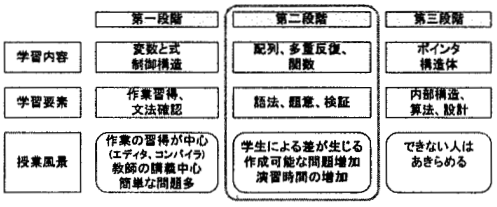

図 3 初級 C 言語の学習段階

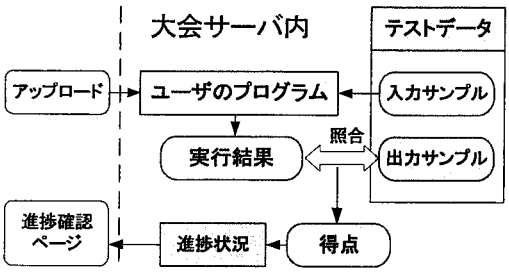

図 4 プログラム判定の手順

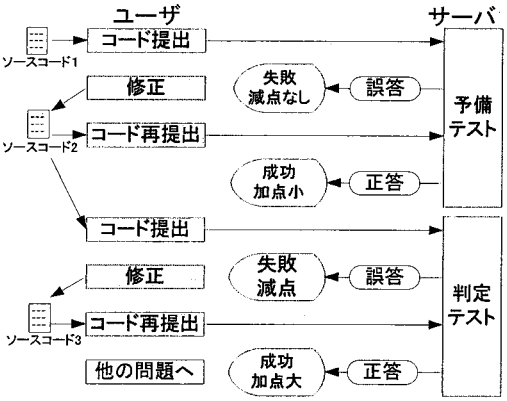

図 5 予備テストと最終テストによる判定

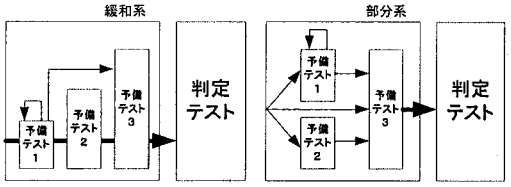

図 6 予備テスト系列

表 2 入出力サンプル例と確認項目

不定個の整数値を入力し、平均値を出力する。負数を番兵とし、小数点以下2桁を出力せよ。		
番兵式による入力の確認		予備テスト1
5 -1	5.000000	
平均値計算の確認		予備テスト2
2 3 4 5 -1	3.500000	
出力書式の確認		予備テスト3
1 2 3 4 5 6 -1	3.50	
十分な量の入出力サンプルを用い、プログラムの正誤を判定する。		最終テスト

3. 実行テスト系列によるプログラム判定

3.1. 段階的な実装を促す実行テスト系列

ソースコードの正誤判定を行う実行テストは，以下の手順で行う(図 4)．まず，学生がサーバ側にソースコードをアップロードし，自動的にコンパイルを行う．コンパイルで生成された実行バイナリを用い，入力サンプルを与えて実行する．出力として得られた実行結果と，正解としての出力サンプルとで照合を行う．照合結果が正答の場合は，得点を与える．正誤判定の結果や得点は，学生個人や全体に公開される．

初心者の演習では，必ずしも全員が問題を完答できるとは限らない．そこで，1 つの問題につき，複数の予備テストと最終テストから実行テストを構成する(図 5)．予備テストでは，最終テストへ至る前の段階的な確認を行う．予備テストの正解には，部分的な得点を与える．何度でも挑戦できるよう，失敗でも減点しない．入出力のサンプルは公開されており，エラー情報も提示するため，積極的にテストに取り組み，その結果を元にデバッギングを進められる．最終テストは，十分な量の入出力サンプルを用いるが，学生には提示されず，結果のみを通知するブラックボックスとして行う．正解には大きな得点を与えるが，失敗すれば減点となる．両方の実行テストを総合して評価を行う．

3.2. 問題例と実行テストの入出力サンプル

コンテストで出題する問題は，二重反復，配列，関数，文字列などの応用が中心となる．ただし，個々の文法事項の理解ではなく，番兵の扱い，例外による脱出など，書法として活用できることが重要である．題材としては，データの整列や統計処理など教科書的な内容だけでなく，パズルなども扱う．例えば，「不定個入力の平均と最大最小」,「年末までの日数計算」,「科目の合計点による順位表」,「再帰的定義による階乗冪計算」,「盤面上のパターン照合」な

どがある．関数や構造体の利用など，実装方法を限定する場合は，テンプレートを用意して，コードの一部のみを作成させる．また，実装方法の概略をコメントとして与え，ヒントとする．

各問題は，学習要素に基づく確認ポイントを幾つか有する．予備テストでは，それらを段階的に判定するような入出力サンプルを用意する．例えば，「不定個入力のデータの平均」を取り上げる．予備テストとして，表2の3組の入出力サンプルを提示する．テスト1は，番兵による入力の打切りと，番兵自体はデータに含めないことを確認する．このテストでは，整数値の入力で平均が整数値となるサンプルのため，実数値での平均を正しく求める処理は求めない．テスト2では，実数への型変換を正しく行っているかを確認する．テスト3は，小数点以下の指定など，出力書式の確認を行っている．より複雑な問題では，「偶数のみ加算」や「100を超える場合は100で計算」など，場合分けや例外処理が正しく行えているかを確認する．このような実行テスト系列で，最終テストに至る中間目標として，初心者の段階的な解答を助ける．また，プログラムの構造やコードパターンを意識させる．さらに，テストファースト的なプログラミングでテストを習慣化させる．

3.3. 緩和系と部分系の予備テスト系列

予備テストの方法には，図6のような緩和系と部分系の2種類が考えられる．緩和系では，表2中の入出力サンプルのように，パスすべきデータの難易度を段階的に高めていく．例外処理を除々に追加し，完成度を高めていくような問題に適している．学生に段階的なプログラム作成を意識させることができる．部分系では，機能ごとに部分的な動作を別の問題を通して実現させる．例えば，番兵式入力に関するコード，和算と商算に関するコードなど，別々に作成させ，最終的に1つのコードに必要な処理を統合する．これは，処理をモジュール単位に分割しやすい問題に適している．別段階で作成したコードを，最終的に1つに統合し，プログラムが機能の集まりで構成されていることを意識させることができる．一方，最終テストでは，プログラムの正誤を適切に検証できるよう，幅広い入出力サンプルを用意しておく．

3.4. 時間と誤答による減点を反映した得点ルール

プログラムの判定結果の得点は，ACM-ICPCのルールをベースに，経過時間と誤答による減点方法を採用する．得点ルールには，解けば解くほど得点の増える単調性，参加する学生が納得できる公平性，得点計算が容易な明瞭性が必要である．学生は，得点ルールに従って，自分の実力に応じた，最適な解答の戦略を練る．

得点ルールのパラメタは，各問題の満点，基準点，標準時間，時間減点，誤答減点，予備テストごとの部分配点である．満点は，標準時間内に誤答なしに最終テストにパスした場合の得点である．この満点から，誤答回数と経過時間で減点する．経過時間に関しては，標準時間を超過した時間を5分単位で減点とする．ただし，最終目標への意欲を最後まで失わせないため，減点は無制限としない．コンテストの制限時間内に正答すれば，最低でも基準点を与える．例えば，満点100，基準点70，時間減点5，誤答減点5，標準時間30分の場合，最終テストにパスした時点で，図7のような得点となる．

最終テストにパスしなかった場合，最終テストに関する減点は行われず，予備テストの部分配点が適用される．部分配点の与え方は，緩和系と部分系により異なる．緩和系は，難易度により徐々に得点が増加し，最終的に解いた最大難易度のテストの得点が反映される．部分系は，解いた予備テストの得点を合計する．これらの各種の値を調整することで，解答方針や実行テストへの取組み方を誘導する．

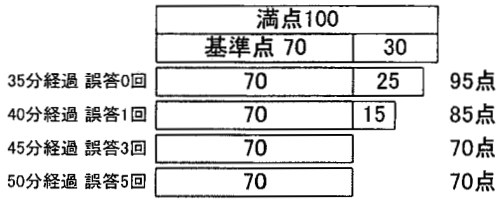

図 7 最終テスト誤答後の得点

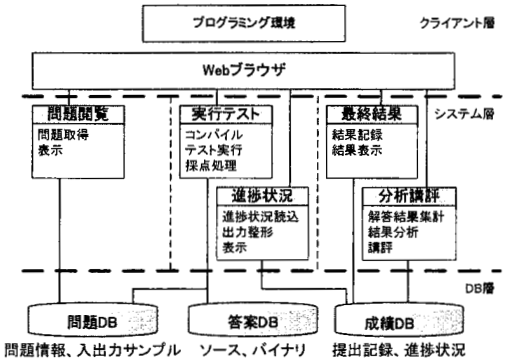

図 8 tProgrEss システム全体構成

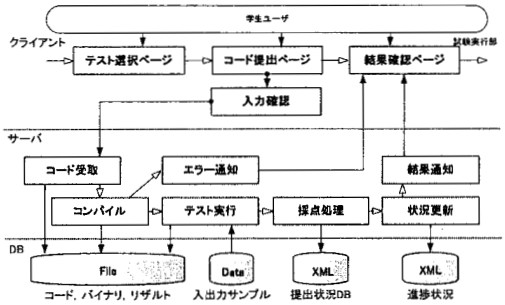

図 9 実行テストの処理モジュール

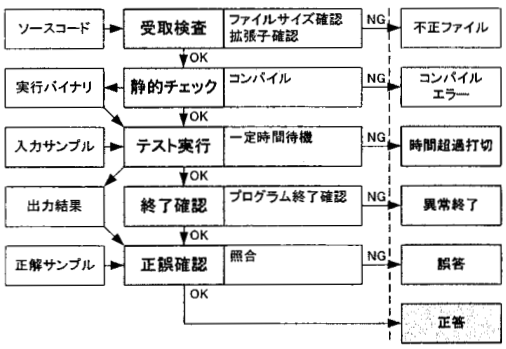

図 10　実行テストによる 6 通りの判定

4. 大会運営サーバ tProgrEss

4.1. 大会運営サーバの全体構成

以上のようなコンテストを円滑に進めるため, 大会運営サーバ tProgrEss を開発する(図 8). システムの全体構成は, 問題閲覧, 実行テスト, 進捗状況, 教師監視, 分析講評のモジュールからなる. まず, 問題閲覧ページで提示された問題セットから解くべき問題を選択する. ここで, 予備テストの内容も明示される. 作成したプログラムに対して, 実行テストを選択し, コード提出ページから, ソースコードをアップロードする. 実行テストの判定結果, およびコンパイルと実行時のメッセージを, 判定確認ページで学生に提示する. 学生は, これらの情報を基に, 解答の修正や, 次のテストに取り組む. また, 進捗状況ページで, 各自の判定結果は教室全体にも公開される. 他人の進捗状況を見て, 競争意識を高め, 意欲を持続させる[6]. 教師監視ページでは, 学生の提出したソースコードも閲覧でき, 自動判定では見逃される欠点や理解が不十分な部分に注意を与える.

4.2. 実行テストの処理手順

tProgrEss の中心的な機能である実行テストの処理手順は, 図 9 のモジュールで実現する. 実行テストの結果は, 図 10 の手順により, 6 段階で判定し, 結果確認ページに提示する.

受取検査では, 提出されたソースコードに対し, ファイルサイズや拡張子の異常を検査し, 不正ファイルを除外する. 静的チェックでは, コンパイルを行い, 実行バイナリを生成する. 静的エラーとなれば, コンパイル結果として, エラー内容を表示する. テスト実行では, 実行バイナリに入力サンプルを与えて実行し, 出力結果を保存する. 一定時間(3 秒程度)を経過して終了しなければ, 打切終了とみなし, 判定コメントとして, 無限ループの可能性を指摘する. また, 実行時エラーの場合, 異常終了と判定し,

エラー情報を表示する．これらの場合でも，途中までの出力データがあれば，出力結果として表示し，デバッグの参考にする．正常終了の場合，出力サンプルを用いた正誤確認を行う．出力結果との照合に成功すれば正答，失敗すれば誤答と判定する．

4.3. 学生の競争意欲を促進する進捗状況ページ

コンテスト中の実行テストの結果は，進捗状況ページで，Web 上で公開される．進捗状況の表示方法には，個人進捗，全体進捗，教師監視の 3 通りの Web ページがある(図 11)．

個人進捗ページでは，各学生が自分の提出履歴や解答状況を表示する．コンテスト中に，次に解答する問題を選択する参考にしたり，過去に提出したソースコードの内容を確認し，コーディングの参考にすることができる．

全体進捗ページでは，教室全体の進捗状況が表示される．他の学生の解答状況や得点が確認でき，全体の中での自分の位置付けを把握して，学生間の競争意欲を促進させる．ただし，他人のソースコードを閲覧することはできない．

教師監視ページでは，教師が教室全体を見渡すために利用する．問題ごとの取組み度合い，各自の時系列的な振舞いなどを確認することができ，進捗状況に応じたリアルタイムの個別指導を支援する．例えば，実行テストを行わず，演習に積極的に参加していない学生に注意したり，予備テストで何度も失敗して，壁に付き当たっている学生にヒントを与えたりする．

4.4. 大会運営サーバの試行運用と予備実験

tProgrEss の試作版を用いたコンテストの予備実験として，C 言語の演習授業で試験的な運用を行った．再履修の 2 年生 23 人を対象とし，半期 14 回のうち，最後の 6 回で実施した．学習範囲は，再帰法，文字列処理，ポインタ，構造体である．二重反復や配列処理も，既習事項として出題内容に含めた．2 時限 180 分の授業の最後の 60 分で，6〜9 問の簡単な問題セットを出題した．1 問 15〜20 分を解答時間の目安とし，4 問程度を解答させた．難易度に応じ，予備テストを 1 または 2 個とした．ただし，試行運用ということで，明確な制限時間を設けず，得点による順位付けなど，結果による総合評価は行わなかった．

実施結果では，実行テストの自動判定は正しく動作し，授業後の感想でも即時的な正誤判定は好評であった．ただし，プロンプト表示など余分な出力を行った解答が誤答と判定され，一部の学生が混乱した．問題文で条件を明確に指示するか，文字列としてのマッチングを緩める必要がある[7]．実行テストの利用状況は，平均 11 回程度で，約 5 分に 1 回であった．進捗状況ページを教卓のスクリーンに投影し，随時更新を行ったため，演習の活性化に繋がった．また，制限時間後に残った問題を宿題として，自宅からもプログラム判定できるようにしたが，頻繁に利用する学生もいた．

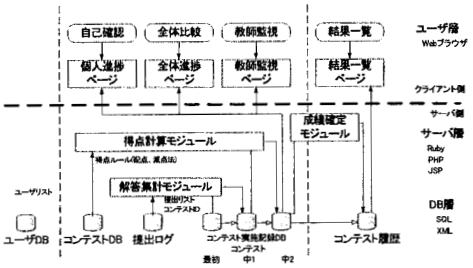

図 11 進捗状況の表示モジュール

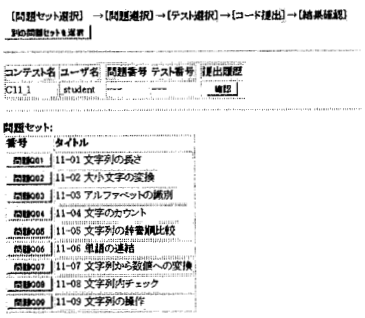

図 12 問題選択ページ

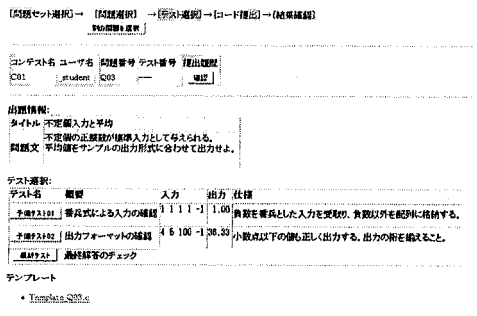

図 13 テスト選択ページ

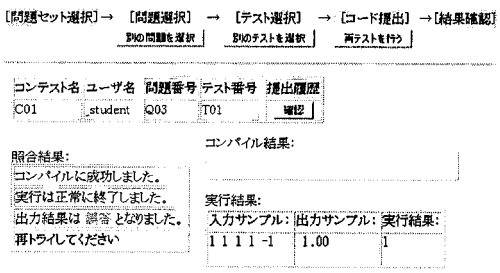

図 14 判定結果ページ

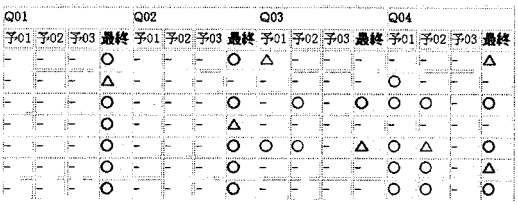

図 15 進捗状況ページ

図 16 教師監視ページ

5. おわりに

大学情報系学科の入門的 C 言語授業において，初心者を対象とするコンテスト形式のプログラミング演習を提案した．実行テストの構成方法や得点ルールを検討し，問題例を挙げた．プログラムの正誤判定には，入出力サンプルによる実行テストを用いる．最終テストに至る中間目標として，複数の予備テストを用意し，実行確認によるデバッグの重要性，書法や構造を意識した段階的なコーディングを認識させる．また，大会運営サーバ tProgrEss を開発し，予備実験として，実際の授業で試行運用した．実行テストによるプログラム判定は正しく動作し，好評を得た．進捗状況ページでは，解答過程を教師や学生が即時的に確認でき，演習の活性化に繋がった．

今後の課題として，得点ルールに基づいた総合評価や順位表示を実装する．問題作成のガイドラインを検討し，授業全体に亘る実践を目指す．コンテストとしての運用実験を行い，教育効果を検証する．

謝辞 本研究は，日本学術振興会科学研究費補助金基盤研究(B)(課題番号 17300269)の助成による．

文　献

[1] 倉田英和，富永浩之，林敏浩，山崎敏範，“実行テストを用いたコンテスト形式の入門的 C プログラミング演習の大会運営サーバの開発”，情報処理学会研究報告，CE86，pp.9-16，(2006).

[2] 倉田英和，富永浩之，“グループコンテスト形式のCプログラミング演習支援環境 tProgrEss -出題構造に基づいた入出力サンプルでの実行テスト-”，信学技報，vol.106, no.507, pp. 87-92, (2007).

[3] IPSJ，第 69 回情報処理学会全国大会 シンポジウム，” Competitive Learning（競合学習）を進めよう”，http://secure1.gakkai-web.net/gakkai/ipsj/program/html/event/

[4] ACM-ICPC, http://icpc.baylor.edu/icpc/

[5] 村井万寿夫，“学習意欲を高めるための手立てについて”，日本教育工学会研究報告集，JET03-4, pp.31-36, (2003).

[6] 安留誠吾，内藤広志，“プログラミング演習における進捗状況表示システム”，教育システム情報学会 第 29 回全国大会，pp.171-172, (2004).

[7] 田上恒大，阿部公輝，“比較的大きなプログラミング課題のための指導採点システム”，情報処理学会研究報告，CE83, pp.135-140, (2006).