実行テスト系列を取り入れた小コンテスト形式の初級 C プログラミング演習における段階的実装を誘導する得点ルール

富永浩之[†1]　太田翔也[†1]

概要：実行テスト系列を取り入れた小コンテスト形式の初級 C プログラミング演習を提案し，数年にわたって授業実践を行っている．ローカル PC で作成したプログラムのソースコードを大会運営サーバに提出し，時間調整点も含めて得点を算出し，順位を公開する．誤答を減点する最終テストの前に，部分的な仕様を満たすだけでよい幾つか予備テストを設けて，段階的な実装を誘導する．予備テストの構成には，限定的な入力に対して正しい出力を求める包含系と，部分的な出力のみを求める構成系がある．前者では，予備テストを飛ばしての解答も認められるが，本論では，予備テストに対する得点方法を変更し，実行テスト系列に従った実装を誘導することを提案する．

キーワード：初級 C プログラミング演習，小コンテスト形式，実行テスト系列，得点方法

Scoring Rule for Stepwise Implementation in an Introductory C Programming Exercise with Contest Style using Execution Test Series

Hiroyuki TOMINAGA[†1]　Shoya OTA[†1]

Abstract: To improve introductory C language lesson in computer engineering college, we have proposed programming exercises with a small contest style adjusted to beginners in classroom. It raise students' motivation and activity by competitive learning approach. We have also developed tProgrEss, the contest management Web server. The server judges an uploaded student's program by an execution test with input and output data. We offer several preparation tests, which are for stepwise sub-goals as partial specification. They give clues and guidelines for solving the final goal. We revised the scoring rule of execution test series with time adjustment. The purpose of execution test series and the scoring rule is to monitoring students' progress situation in more detail.

Keyword: introductory C programming, contest style exercise, execution test series, marking and scoring method

1. はじめに

大学の情報系学科の多くは，初年次に C 言語の入門的な科目を必修とし，演習を重視している．授業の前半で文法や例題を解説し，後半に類題を解く時間を設ける形態が多いが，学生の理解度や受講態度に差があり，全ての学生を集中させて演習に取り組ませることは難しい．

そのためには，自分のペースで取り組ませ，各学生の演習中の取組み状況をできるだけ把握できるようにし，必要に応じて適切な指導や助言を迅速に行うことが望ましい．選択方式や穴埋め方式で設問を増やす方法も考えられるが，やはりコードを実際に書かせる訓練が重要である．したがって，プログラムを段階的に作成していき，その過程を追従できるような手法が求められる．これは，教員による進捗状況の把握だけでなく，自分で進捗状況を確認できるという面も重要である．

一方，プログラミング演習にも，コンテストなど競争型学習を導入することが注目されている．コンテストには，自由なテーマで作品としてのプログラムを作成する審査系と，与えられた仕様を満たすプログラムを作成する競技系とがある．

本論では，特に，後者に着目する．オープンな競技系のコンテストとしては，大学対抗の国際プログラミングコンテスト ACM-ICPC [1]が有名である．会津大学の AOJ [2]や北京大学の POJ [3]は，元々はその練習のためのシステムとして開発され，今でも広く利用されている．また，AtCoder 社の AtCoder [4]は，学生の運試しとしてオープンな利用が可能なコンテストから，企業が契約して就職試験などで実施するコンテストも開催している．

ゲーム感覚のイベントとして，コンテスト形式を採用すれば，その場で採点され，良い成績を得るという目的意識が明確になる．また，全体の順位や特定の相手との勝敗などで，学生同士の対抗意識や挑戦意欲が，学習効果を高めることが期待される．しかし，ここで挙げた一般的なコンテストの方法では，初心者への敷居が高く，入門科目で授業時間内に組み込みにくい．ほとんど着手できない，解けずに得点が得られない時間が長ければ，むしろ挑戦的な意識が減退してしまう．上位陣においても，順位の逆転できる可能性が最後まで残されているような仕組みが望ましい．

†1 香川大学
Kagawa University

2. 小コンテスト形式のプログラミング演習

本研究では，初級 C 授業と初心者に適応した，競技系の小コンテスト形式のプログラミング演習を提案している[5][6][7][8][9][10]. 図 1 のように，コンテストにおける出題とプログラムの提出，採点と結果の公開を目的とする大会運営サーバ tProgrEss を開発している．特徴として，テスト駆動を意識した段階的な実行テスト系列を用意している．テスト駆動では，求められた入出力サンプルのテストケースに対し，それを満たす最低限のコーディングを行うようにする．したがって，各段階のテストケースに応じて，部分的な仕様を満たす中間的な解答にも得点を与える．ただし，スケルトンコードのテンプレートファイルを与え，初心者でも解法の糸口が得られやすくする．これにより，早期からの頻繁な提出を促し，評価の粒度を下げている．すなわち，問題数の割に，提出機会が増え，進捗状況の把握がしやすくなっている．

2006 年度に試行的に導入し，2009 年度から毎年，実際の毎回の授業で運用している[11][12][13]．対象は，大学情報系の 1 年次および 2 年次である．前者では，基本データ型と四則演算，論理演算と制御構造，配列処理，数学関数が学習項目である．後者では，複合データ型，関数定義，再帰法，データ型定義，変数の有効範囲，文字列，ポインタ，構造体，ファイル入出力である．ただし，担当教員の関係で，主として後者での利用が中心となっている．なお，本システムは，ファイル入出力の問題には未対応である．

本論では，段階的な実装を誘導するため，コンテストの実施形態と利用場面を整理する．部分加点や時間調整点の方法を再検討する．

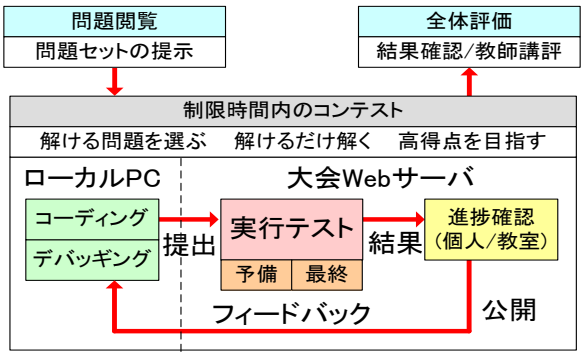


図 1 小コンテスト形式のプログラミング演習

3. コンテストの実施形態と利用場面

コンテストの実施形態の区分として，開始同期には一斉式と個別式，時間単位には分計と日計を設ける．これにより，一斉式で分計の教室型/試験型，日計の宿題型，個別式で分計の自習型，日計の課題型を用意する．図 2 のように，利用場面で使い分ける．これらの型は，異なる機能を提供というより，コンテストの要項や問題編成の指針を示すテンプレートである．

教室型は，毎回の授業の最後に，総まとめ的な位置付けとして，教室内で一斉に実施する．基本的な文法事項で，例題の類題とする．15～30 分で解ける問題を，4～6 問の出題で，60～120 分のコンテストとする．時間経過は，分単位とする．教室型の変種として，自習型と試験型を用意する．自習型は，オープンな出題とし，受験者が解答を開始した時点から一定の間を試験期間とする．いつでもできる自習用として利用する．主に予習に用いる．試験型は，教室型より長い時間で，解答に 30 分程度かかる，難易度の高い問題を出題する．制限時間を厳格にし，成績に関与するような授業で利用する．

宿題型は，指定された開始時期から 1～2 週間の期間で，学外からのアクセスも許容して行う．情報数学，数値計算，統計処理などを題材に，応用的な事項の組合せとする．60～90 分かかる問題を，6～8 問の出題とする．時間経過は，日単位とする．課題型は，その変種として，オープンな出題とし，受験者が解答を開始した時点から一定の間を試験期間とする．長期間の休暇の課題で利用する．出題は発展的な内容も含む．

小コンテスト形式の演習の進行は，問題文に沿ったプログラムをローカル PC で作成し，大会運営サーバにアップロードして正誤の自動判定を行う．制限時間内に早く多く解いた方が高得点となる．進捗状況は Web 上に公開され，受講者が相互に確認できる．受講者は，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，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．

このような演習により，学生は自分のペースで解答を進められる．さらに，学生の提出状況を閲覧する教師監視ページも用意する．自動判定では見逃される不十分なコードや理解不足の点に注意を与える．これにより，解答が進まない学生を早期に発見し，個別に指導する．

演習中は，受講者からの質問の答える補助者の存在も必要不可欠である．彼らにも受講者の進捗状況を容易に参照できる手段があるとよい．そこで，教師だけでなく，補助者に対し，コンテストの進捗状況と連動する出席管理と補助学生による巡回指導の支援ツール vRoundEd も開発している[14]．60 名程度の受講者に対し，10 名前後の補助者がタブレット PC を携帯して利用している．補助者には，副担当の教員，技術職員，TA 学生が含まれる．彼らへのアンケートにより，本研究の小コンテスト形式による演習が，受講者への質疑を促進し，補助者の支援にも繋がることが確認された[15][16].

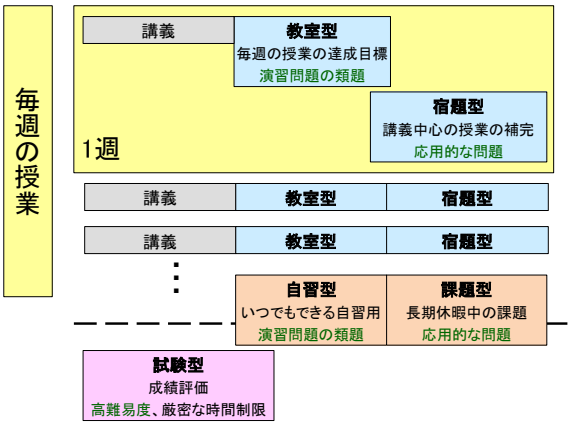


図 2 コンテストの実施形態と利用場面

4. プログラムの正誤判定

学生の作成したプログラムを評価する際，幾つかの段階が考えられる．1 つ目は，問題の仕様を満たし，想定される入力に対して，常に正しい出力結果が得られることである．2 つ目は，適切な算法に基づいて，効率的な処理が実現されていることである．3 つ目は，人間が読むソースコードとして，処理の流れが明確で，誰が見ても理解しやすいことである．1 つ目の評価では，ブラックボックステスト，ホワイトボックステストを用い，ほぼ客観的に判定できる．2 つ目の評価は，大きなデータを与えて計算時間を測定したり，オーバーフローとなる限界を調べることで，ある程度は推測できる．3 つ目の評価は，基準が曖昧で主観的な部分があり，出題意図を理解した教師が自分で採点する必要がある．

本研究では，演習における判定結果の即時通知を重視するため，1 つ目の評価を採用する．すなわち，事前に用意された入出力サンプルによる実行テストを用いて，プログラムの正誤判定を行う．これをサーバ側に提出されたソースコードに対して行う．まず，アップロードされたソースコードを，システムがコンパイルする．生成された実行バイナリに対し，入力サンプルを与えて実行する．その実行結果と，正解となる出力サンプルとで照合を行う．出力サンプルは，問題文の仕様を満たした模範プログラムによる実行結果である．

照合に成功し，正答と判定されれば，規定のルールに従って計算された得点を与える．ただし，実行結果のみに基づいて照合を行うため，ソースコードの内部については問わない．必要があれば教師が後から自分で内容を吟味する．また，不正コピーなどを防ぐため，ソースコードの類似度を判定するモジュールを組み込む．

ソースコード提出の判定結果は，6 段階で行う(図 3)．一般的なオンラインジャッジシステムでは，これら以外に，メモリ使用量など，より具体的な判定段階が設けているものもある．しかし，本研究では，対象が入門的な演習のため，最低限のものに留めている．

(1) 不正提出　提出したコードのファイルサイズや拡張子に不備があることを示す．誤って実行バイナリを提出した場合などである．また，プロセス制御文のような，字面で分かる危険コードを排除する．

(2) 静的エラー　コンパイルに失敗し，実行バイナリが生成されなかったことを示す．タイプミスや文法的な間違いである．

(3) 実行時エラー　プログラムの実行がエラーとなり，異常終了したことを示す．主に，配列の範囲外参照やポインタ関連のミスである．

(4) 実行打切　指定時間内にプログラムの実行が終了しなかった，または，出力結果が指定サイズを超えたことを示す．無限ループが発生して暴走している，仕様にないデータを要求して入力待ちの状態になっている，などである．

(5) 誤答　実行は正常に終了したが，実行結果が出力サンプルとの照合に失敗したことを示す．アルゴリズムや出力書式のミスである．

(6) 正答　正しく実行が終了し，実行結果が出力サンプルとの照合に成功したことを示す．

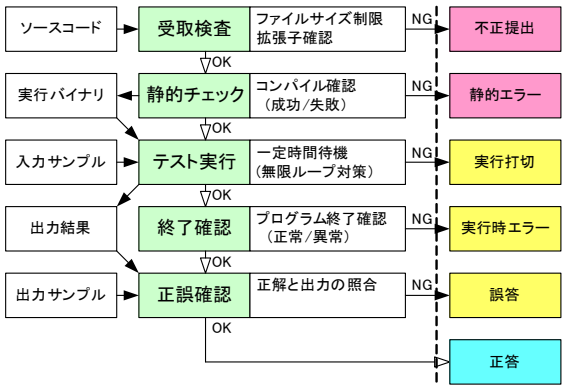


図 3 正誤判定の段階と手順

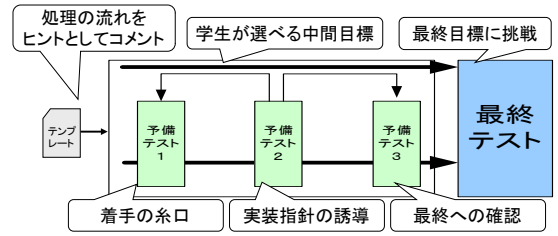
5. 実行テスト系列の予備テストと最終テスト

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，複数の予備テストおよび最終テストからなる実行テスト系列を提示する(図 4)．最終テストは，題意を完全に満たす解答を要求するが，予備テストでは部分的な仕様を満たせばよい．

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．コーディングの進め方を誘導したり，プログラムの作成指針を示す．予備テストの正答には，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．正答には大きな得点を与えるが，失敗すれば配点への減点となる．一般的なコンテストでは，この最終テストに該当するテストが用意されている．最終テストで入出力サンプルを提示しない理由として，具体的な処理を実装せず，示したサンプルにのみ対応したようなコードを防ぐためである．また，予備テストを利用して慎重にテストを繰り返すことを促す効果も狙っている．

学生は，自身の理解度に応じ，最終テストの正答を目指して自由に実行テストを利用できる．プログラミングが不得意な学生には，完全な解答への着手の糸口となる．中間目標として，段階的な実装を誘導する．最終テストへの正答が難しいと判断した学生も，部分点を狙って，諦めずに挑める．プログラミングが得意な学生は，一気に最終テストの正答を狙ってもよい．ただし，最終テストのみに誤答減点を設けることで，最後の一歩前の確認として，誤答減点のない予備テストも利用させ，慎重さを促す．



	予備テスト	最終テスト
入出力サンプル	部分的な仕様　データを明示	完全な仕様　データを隠蔽
正答の場合	解ける所まで取り組める部分点	解答時間を差し引いた得点
誤答の場合	減点なしで何度でも提出可能	回数による減点で慎重さを促す

図 4 実行テスト系列の予備テストと最終テスト

6. 包含系と構成系の予備テスト

中間目標としての予備テストは，部分仕様の与え方と得点ルールで包含系と構成系の 2 種類を用意する[17]．これらの実行テスト系列を学習項目や教育目的に応じて使い分け，学生のプログラム作成を適切に誘導する．

包含系は，問題の仕様を緩めて，徐々に完成に近づけさせる．予備テストの各段階を最終テストに至る IS-A 構造で捉えている．入出力サンプルとしては，パスすべきデータの難易度を段階的に高めていく．副次的な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．本研究でのコンテストでは，包含系を中心に予備テストの作成を行う．

構成系は，問題の仕様を分割して，別々の小さな問題として実装させ，最後に統合させる．最終テストに対する個々の予備テストの関係を HAS-A 構造で捉えている．機能ごとに部分的な動作を別の入出力サンプルを通して確認する．関数ごとの実装を確認するような実行テスト系列では，予備テストは単体テスト，最終テストは結合テストに相当する．例えば，番兵式入力に関するコード，加算と除算に関するコードなどを別々に作成させ，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．

本研究のコンテストでは，主に包含系の予備テストを採用する．これは，構成系を実際にコンテストで利用する場合は，基本的な処理は理解できているものとして出題するため，ある程度難易度が高く，規模の大きな問題に適用することになる．そのため，短時間で行う授業中のコンテストには，余り適していない．また，包含系では，最終テストの入出力サンプルを，そのまま予備テストにも流用できるのに対し，構成系では完全に別の入出力サンプルが必要となり，運用面のコストが大きくなる．以上の 2 つの点から，包含系の予備テストを利用する．

7. 入力範囲の限定と出力形式の許容

包含系の予備テストでは，入力範囲の限定と出力形式の許容の 2 通りを考える．それらを組み合わせて，予備テストを構成する．

入力限定では，仕様が要求する入力範囲内の一部でのみ正しく実行できればよい．テストケースとしては，例外処理となるデータ，継続条件や分岐条件での境界値を避ける．また，予備テストで用いる入出力サンプル以外では，正しく計算できなくてもよい．本来の計算方法とは異なり，その入力でしか通用しない実装でも構わない．例えば，二次方程式の解法で判別式が正の場合のみ，文字列処理で全てが英大文字の場合のみ，といったケースである．

出力許容では，照合において，一部の出力データを無視して採点する．具体的には，仕様で要求される出力の個数，書式や精度において，照合の柔軟さを設定する．例えば，最終テストがデータ列の平均と最大値を求める問題で，予備テストでは平均だけの出力が合えばよいとする．このとき，最大値の出力がなかったり，誤った値が出力されていても，その予備テストは正答とする．書式や精度においては，printf 文の出力書式による空白の個数や小数点表示などの相違を無視し，データとして一致すればよいとする採点を行う．例えば，平均値の問題ならば，出力桁数を指定されていたり，計算途中の小さな誤差を見逃すなどである．

包含系の予備テストは，以下のようなガイドラインに沿って構成し，問題内容に即した処理の確認を行う．予備テスト 1 では，入力値をデータとして正しく扱えるかを主な確認項目とする．まずは，問題に取り組むきっかけとなる部分として，確実に解ける内容で実行テストを行う．予備テスト 2 では，主要な反復処理や配列操作を確認項目とす

る．問題の中心となる処理において，一部の機能のみを確認したり，例外を含まない簡単な例で実行したりする．予備テスト 3 では，最終テストに至る最後の確認として，整形出力や例外処理の確認を行う．最終テストとほぼ同様の仕様とすることで，予備テスト 3 を最終テスト前の確認として利用することができる．

8. 誤答減点と時間調整点による得点ルール

プログラムの判定結果には，実行テストの種別による標準的な配点，および時間調整点と誤答減点を導入した得点ルールを適用する．時間調整点は，最終テストにおいてのみ，提出時間によって，通常の配点に加減点を付与する(図 5)．提出時期を，事前，早期，通常，延長，事後，無効の 6 つに区分する．事前期間の提出は，コンテストの開始前なので無効とし，得点は 0 である．早期期間の提出は，積極性を評価して標準的な配点すなわち通常点に加点する．通常期間の提出には，通常点に加点も減点もしない．延長期間の提出は，締切の延長という扱いであるため，通常点から減点する．

事後提出は，延長された最終締切後で，追試的な措置としての提出である．本来の得点の 2 分の 1 以下を付与する．無効提出は，授業閉講後にシステムの正誤判定の機能のみを使用する際の提出である．得点は 0 である．早期と延長における加減点は，提出時期で傾斜させる．これらの時間による得点調整を設けることで，学生の早期解答を促し，演習の活性化を狙う．いずれにしろ，努力が無駄になるような得点にはならないようにしている．

一方，誤答減点は，最終テストの誤答に減点のペナルティを与えるものである．ただし，実際に減点を与えるのは，後で最終テストに正答した場合にのみ，配点からの減点である．正答しないままの場合には，減点だけが行われることはない．最終テストでは，例外的な入力を含む，網羅的な入出力サンプルを用意している．誤答減点により，完答への慎重な確認を行わせる．最終テストに正答できなかった場合でも，予備テストに正答していれば，部分点のみを与え，減点は行わない．なお，時間や誤答による減点は，ある程度で打ち切り，最低でも基準点を与える．

例えば，配点 150 点の問題で，最終テストに対して，1 回の誤答(-10 点)を経て，開始後 30 分(+30)に正答すれば，150+30-10=170 点になる．逆に，3 回の誤答を経て，締切後 40 分(-10 点)に正答すれば，150-10-30＝110 点となる．予備テストには，誤答減点も時間調整点は適用されない．そのため，減点のある状態で最終テストに正答しても，予備テストの得点を下回る状況が起こり得る．減点に対する下限を設けているのは，それを防ぐためである．

これらの得点ルールにより，素早く解答させるコンテスト，じっくり時間をかけて取り組ませるコンテストなど，コンテストの性格を調整することができる．

9. 予備テストを重視する得点ルールへの変更

現状の実行テスト系列の問題点の 1 つとして，予備テストが軽視されがちであることが挙げられる．これまでの数年における実施状況においても，受講者の性向や包含系の出題の傾向によっては，予備テストが十分に機能していない側面もあった．これらは，tProgrEss の教師側の実施監視ページにおけるモニタリング機能で，幾つかのケースが確認できた[18][19][20][21][22]．最終テストの得点には，予備テストへの解答の有無は影響がない．そのため，最終テストだけを解答しても，得点上は問題がない．時間調整点の影響もあり，特に上位の受講生は，予備テストに取り組まず，最終テストのみ解答を行いがちである．しかし．予備テストは，段階的な実装を体験させるために，できるだけ取り組んでもらうことが望ましい．そのため，予備テストの解答状況を最終テストに反映させるように得点ルールを変更する．すなわち，予備テストの正答状況に応じて，最終テストの点数に加点を行う．具体的には，予備テスト 1 つ毎に加点幅を設定し，予備テストに正答していれば，最終テストの得点に加算する．

例えば，緩和系で予備テスト 3 つ，それぞれが 50，100，150 点，最終の点数が 200 点とする．予備テストの加点幅は，全てて 0 点とする．このとき，予備テストに解答せず，最終テストだけに正答した場合は 200 点となる．予備テストに 1 つ正答していた場合，200+20=220 点，全ての予備テストに正答していた場合，200+20+20+20=260 点となる．なお，最終テストへの解答後に予備テストに正答した場合は，段階的実装としての意味が無いため，加点しない．

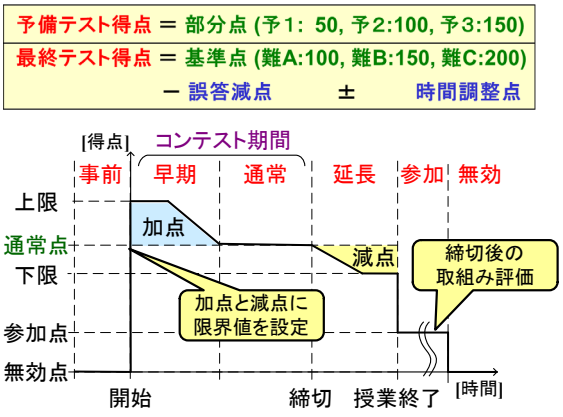


図 5 誤答減点と時間調整点による得点ルール

10. おわりに

大学情報系学科の入門的 C 授業において，競技系の小コンテスト形式を採用した初心者向けのプログラミング演習

を提案し，大会運営サーバ tProgrEss を開発している．tProgrEss は，サーバ側に提出された解答コードを自動採点し，コンテストの進捗状況を即時に表示する．中間目標として，最終テストと予備テストから構成される実行テスト系列を導入する．また，誤答減点と時間調整点による得点ルールを設けている．これらにより，段階的実装を誘導し，頻繁な提出を促進し，受講者の進捗状況を細かい粒度で把握する．

ただし，これまでの数年における実施状況により，受講者の性向や包含系の出題の傾向によっては，予備テストが十分に機能していない側面もあった．そこで，予備テストの利用を促進するように，得点ルールを再検討した．

今後，新しい得点ルールを実装し，tProgrEss のモニタリング機能を充実させる．その上で，コンテストの提出履歴と得点推移を詳細に分析する．また，補助者の机間巡視と個別指導のための支援ツールにおいて，操作ログの分析を進めている[23][24]．その結果とも合わせ，教育効果の向上を目指す．

参考文献

[1] ACM ICPC, ACM ICPC, https://icpc.baylor.edu/.

[2] The University of Aizu, Aizu Online Judge, http://judge.u-aizu.ac.jp/onlinejudge/.

[3] Peking University, PKU JudgeOnline, http://poj.org/.

[4] AtCoder Inc., AtCoder, https://atcoder.jp/.

[5] 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 －コンテストの運営方法と実行テストの評価方法の提案－, 信学技報, Vol.105, No.632, pp.129-134 (2006).

[6] 倉田英和, 富永浩之, 林敏浩, 垂水浩幸, 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, 情処研報, Vol.2006, No.108, pp.9-16 (2006).

[7] 倉田英和, 富永浩之, 林敏浩, 山崎敏範: 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).

[8] 倉田英和, 富永浩之, 林敏浩, 垂水浩幸: 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, 情処研報, Vol.2007, No.101, pp.11-18 (2007).

[9] 倉田英和, 富永浩之, 林敏浩, 垂水浩幸: 初級 C プログラミングの演習支援サーバ tProgrEss によるコード判定と授業実践, 教育システム情報学会 研究報告, Vol.22, No.5, pp.17-24 (2008).

[10] 富永浩之, 倉田英和: コンテスト形式による初級 C プログラミングの演習支援, 情処研報, Vol.2008, No.42, pp. 49-56 (2008).

[11] 川崎慎一郎, 富永浩之: 競争型学習を取り入れた入門的 C プログラミング演習 －演習支援サーバ tProgrEss の出題解答と採点結果のページ表示の改良－, 信学技報, Vol.109, No.335, pp.187-192 (2009).

[12] 富永浩之, 川崎慎一郎: 競争型学習を取り入れた入門的 C プログラミング演習 －運用実験での実行テスト系列の利用状況－, 情処研報, Vol.2010-CE-104, No.4, pp.1-12 (2010).

[13] 川崎慎一郎, 西村智治, 富永浩之: 競争型学習を取り入れた入門的 C プログラミング演習－授業実践における解答コードの提出履歴の詳細分析－, 情処研報, Vol.2010-CE-107, No.10, pp.1-10 (2010).

[14] 太田翔也, 花川直己, 中矢誠, 富永浩之: 実行テスト系列を取り入れた小コンテスト形式の初級 C 演習における教師支援 －コンテストの進捗状況と連動する出席管理ツールと補助学生による巡回指導の支援機能－, 教育システム情報学会研究会, Vol.30, No.5, pp.7-12 (2016).

[15] 太田翔也, 富永浩之: 実行テスト系列を取り入れた小コンテスト形式の初級 C 演習における巡回指導の支援ツール －補助者によるタブレット PC でのシステム利用とアンケート分析－", 信学技報, Vol.116, No.126, pp.1-6 (2016).

[16] 富永浩之, 太田翔也: プログラミング演習における補助者の机間巡視と個別指導のためのタブレット PC 上の支援ツール －小コンテスト形式の初級 C 演習での実践におけるアンケートの分析－, 教育システム情報学会 研究会, Vol.32, No.1, pp.1-6 (2017).

[17] 川崎慎一郎, 富永浩之: 競争型学習を取り入れた入門的 C プログラミング演習 －実行テスト系列による部分採点のための柔軟な照合機能－, 情処研報, Vol.2009-CE-103, No.10, pp.1-8 (2010).

[18] 川崎慎一郎, 西村智治, 富永浩之: 競争型学習を取り入れた入門的 C プログラミング演習 －解答状況による学生の振舞いパターンの分析－,信学技報, Vol.111, No.141, pp.53-58 (2011).

[19] 西村智治, 川崎慎一郎, 富永浩之: 小コンテスト形式の初級 C 演習における教師支援 －解答状況の時系列表示によるモニタリング機能の試用実験－, 情処研報, Vol.2011, No.10, pp.1-8 (2011).

[20] 西村智治, 川崎慎一郎, 富永浩之: 競争型学習を取り入れた初級 C プログラミング演習における教師支援 －モニタリング機能としての解答状況と提出状況の視覚化－, 信学技報, Vol.110, No.453, pp.163-168 (2011).

[21] 富永浩之, 西村智治: 実行テスト系列を取り入れた小コンテスト形式の初級 C 演習 －学生の得点状況の時系列分析による活性度の推定－, 情処研報, Vol.2012-CE-116, No.15, pp.1-8 (2012).

[22] 西村智治, 青木辰徳, 富永浩之: 小コンテスト形式の初級 C 演習における教師支援 －解答プログラムの提出状況と得点推移によるモニタリング機能－, 情処研報, Vol.2012-CE-119, No.19, pp.1-8 (2013).

[23] 太田翔也, 富永浩之: プログラミング演習における補助者の巡回指導のためのタブレット PC 上の支援ツール －小コンテスト形式の初級 C 演習での実践におけるツールの操作ログの分析－, 情処研報, Vo.2016-CE-137, pp.1-8 (2016).

[24] 太田翔也, 富永浩之, プログラミング演習における補助者の机間巡視と個別指導のためのタブレット PC 上の支援ツール －小コンテスト形式の初級 C 演習での実践における機能改善の効果と操作ログの分析－, 信学技報, Vol.116, No.517, pp.95-100 (2017).