2008－CE－94 (8)
2008／5／17

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

富永 浩之, 倉田 英和, 林 敏浩, 安藤 一秋, 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: tominaga@eng.kagawa-u.ac.jp

あらまし 大学情報系学科の入門的C授業において，初心者向けの小コンテスト形式でのプログラミング演習を提案する．また，大会運営を支援するサーバtProgrEssを開発する．本提案のコンテストでは，授業中に複数の問題を提示し，学生がローカルPCで作成したソースコードをサーバ側にアップロードさせる．サーバは，入出力サンプルによる実行テストで，プログラムの正誤判定を行う．出題においては，学習項目となる書法を整理して，中間目標となる複数の予備テストを導入し，解法のヒントや実装手順を示す．これにより，テストの重要性を認識させ，仕様に沿ったプログラムの完成までを段階的に誘導し，取組みへの手掛かりとさせる．時間と誤答による得点ルールを採用し，結果をランキングとして公開する．これにより，競争意識を促進し，演習を活性化させる．問題作成のガイドラインを示し，授業実践として，実際にコンテスト形式の演習を行った．アンケート評価や学習状況の履歴を分析し，本研究の教育効果を検証した．

キーワード 初級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

Support for Introductory C Programming Exercise with Contest Style

Hiroyuki TOMINAGA, Hidekazu KURATA, Toshihiro HAYASHI, Kazuaki ANDO,

Hiroyuki TARUMI

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

Abstract To improve introductory C language lesson for computer engineering education in college, we propose a contest style exercise for beginners in classroom. And we developed tProgrEss, the contest management Web server. In our contest, a teacher shows a set of problems in classroom to solve them on students' local PC. After uploading students' 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 as partial specification. They suggest a hint and coding strategy with several program idioms. Students must understand importance of testing to satisfy the total requirement. 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. And it is to give sense of satisfaction and achievement with solving problems. We prepared some problems based on authoring guideline and performed practical exercises using tProgrEss. Analyzing questionnaires and logs of learning situation, we examined the educational effect of our research.

Keyword introductory 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．入門的なプログラミング授業の標準的な進行は，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．

しかし，授業中のプログラミング演習の時間に，解答の全てをその場で確認することは難しい．また，理解度の異なる学生の進度に応じて，適切に指導することも難しい．一方，学生の多くは，コンパイル成功で安心し，十分な実行確認を行わずに，正しいプログラムが完成したと考えがちである．また，プログラムの語法や構造を把握できず，取掛りの糸口を掴めない学生もいる．得意な学生でも，何らかのフィードバックがないと，意欲を失うことになる．演習の効果を高めるには，解答時間を制限して，その場でプログラムの正誤を確認させるなど，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．また，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．

近年，プログラミング教育において，競争型学習(competitive learning)が注目を浴びている[1]．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は，授業中の演習形式として期待される．オープンなものとしては，大学生向けでチーム対抗のACM-ICPC[2]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を考える必要がある．

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

本研究で提案するコンテストは，毎回の授業の最後に，総まとめ的な位置付けとして実施する[3][4]．コンテスト形式の演習の進行は，図1の3つのフェーズで捉える．授業内容に関連する複数の問題を与え，何問かを選択し解答する．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実行テストで採点する．実行テストによるプログラムの判定は，サンプルとして与えられた入力データに対し，正しい出力が得られたかどうかを文字列のマッチングで行う (図2)．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，学生が相互に確認できる．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．また，解答が進まない学生を早期に発見し，個別に指導することもできる．

初心者のコンテスト形式の演習では，必ずしも全員が問題を完答できるとは限らない．そこで，実行テストは，1つの問題につき，複数の予備テストと最終テストから構成する(図3)．予備テストでは，部分仕様を満たす少数のサンプルを用い，最終テストへ至る前の段階的なチェックを行う．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し，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，学生に内容は提示せず，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．最終的には，両方の実行テストを総合して評価を行う．

2.2. 実行テスト系列の構成と中間目標

中間目標としての予備テストは，仕様を緩めて徐々に完成に近付けさせる緩和系と，仕様を分割して別々の小さな問題として実装させる分解系の 2 種類に分ける(図 4)．これらの予備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．緩和系の予備テストは，入力制限と出力許容に分けられる．入力制限では，仕様で想定される範囲内の一部のデータでのみ正しく実行できればよく，例外処理を必要とするもの，継続条件や分岐条件で境界値となるものを避けて実行テストを行う．出力許容では，仕様で要求される出力の個数，書式や精度において，照合基準を緩和し，データ列の平均と最大値を求める問題で，平均だけの出力が合えばよいなどとする．予備テスト系列は，3 つ程度の中間目標として構成し，プログラムの作成指針を提示する(図 5)．例えば，図 6 のような入出力サンプルを用いる．予備テスト 1 では，番兵による入力の打切りや，データ型などの，入力段階を確認する．予備テスト 2 では，問題の主要な処理や機能を確認する．予備テスト 3 では，出力の整形や吟味，例外的なデータ処理を確認する．

プログラムの判定結果には，時間と誤答による減点を反映した得点ルールを適用する(図 7)．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をパラメタとする．満点は，標準時間内に誤答なしで最終テストにパスした場合の得点である．ここから，誤答回数と経過時間で減点する．前者は，最終テストの誤答 1 回につき 1 回，後者は，標準時間からの超過を 5 分単位で減点する．ただし，コンテストの制限時間内に正答すれば，減点を打ち切って，最低でも基準点を与える．最終テストに正答できなかった場合は，予備テストの部分点のみが得られ，減点は行わない．

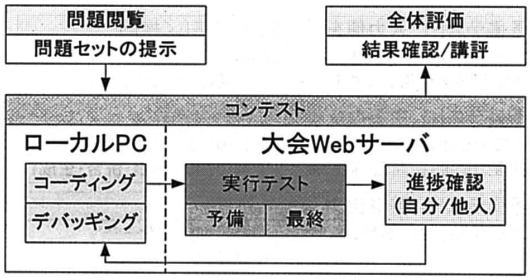


図 1 コンテスト形式の演習の進行

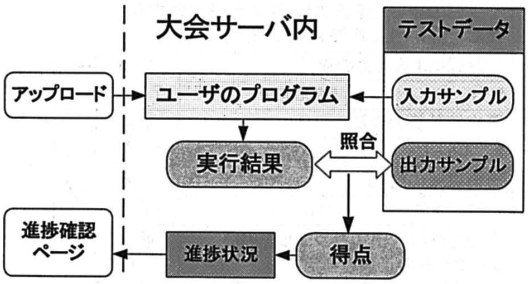


図 2 プログラム判定の手順

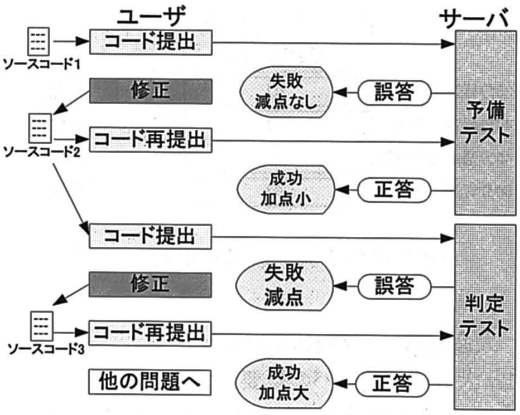


図 3 予備テストと最終テストによる判定

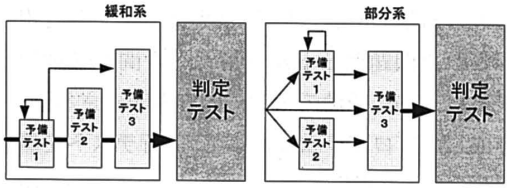


図 4 予備テスト系列

予備テスト1：入力値をデータとして正しく格納	
入力の順序とデータ型 不定個入力の番兵での打切り	不適な入力範囲の確認 入力値の計数と振分け
予備テスト2：主要な反復処理や配列操作	
反復の構造(パス、ネスト) 反復の範囲(初期値、継続条件)	配列のシフト(追加・削除・逆順) 配列の加工(分割・併合・生成)
予備テスト3：整形出力や例外処理	
整形出力(桁数、区切、表形式)	例外処理(レアケース) 境界値検査

図5 実行テスト系列における確認項目

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

図6 入出力サンプル例

	満点100		
	基準点 70	30	
35分経過 誤答0回	70	25	95点
40分経過 誤答1回	70	15	85点
45分経過 誤答3回	70		70点
50分経過 誤答5回	70		70点

図7 最終テスト誤答後の得点

3. 大会運営サーバ tProgrEss

3.1. 大会運営サーバの概要

以上のような演習を円滑に進めるため，大会運営サーバ tProgrEss を開発する．Web ベースの CGI として実装し，システムの全体構成は，図 8 の通りである．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．実行テストの結果は，正答，誤答，無限ループ，異常終了，静的エラー，不正ファイルの 6 段階で判定する(図 9)．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(図 10)．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，進捗状況公開ページで，各自の判定結果は教室全体にも公開される(図 11)．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．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不十分なコードや理解不足の点に注意を与える．

3.2. コンテストの実施形態

当初は，授業中のコンテストのみを想定していたが，演習中に解けなかった問題への再挑戦や，自宅学習へのスムーズな連携を考え，コンテストの実施形態として，教室型と宿題型の両者を取り入れる．

教室型は，授業時間内に 60 分前後(30〜90 分)の制限時間を設けて，教室内で一斉に行う．基本的な事項で，例題の類題として，15〜30 分程度で解ける問題を主に出題する．一部のコーディングが済んだテンプレートを用意する．予備テストは，通過点としての確認で，多くの学生が最終テストにパスすることを想定する．予備テストは，緩和系が有効で，簡単なテストケースからクリティカルなケースへと誘導する．部分系は，出力の一部のみを照合する場合に有効である．コードを書き直すような部分系は，時間的に不向きである．

宿題型は，1〜2 週間の期間内に，学外からのアクセスも許容して行う．応用的な事項の組合せで，60 分程度はかかる問題を主に出題する．テンプレートは，日本語による算法の説明を詳細に行うが，コードは余り含めない．全員が完答できるとは限らず，各自の実力に応じて，予備テストでの部分点を目指す．予備テストは，独立した個々のコードブロックの確認や，関数

の個別のテストなど，部分系が有効である．

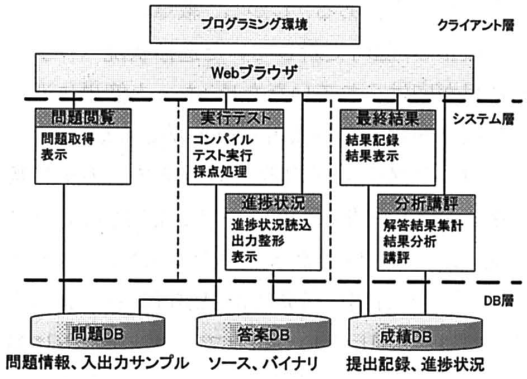


図 8　tProgrEss の全体構成

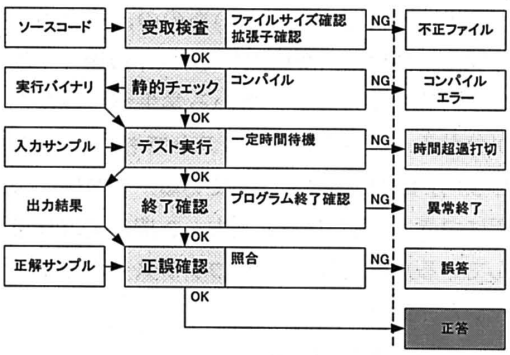


図 9　実行テストによる 6 通りの判定

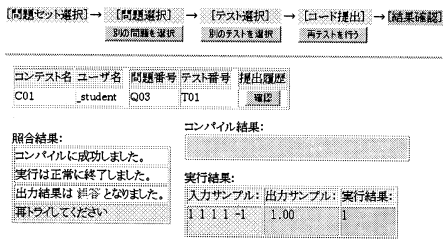
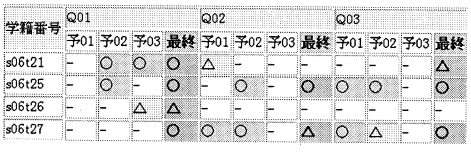


図 10 判定結果確認ページ



学籍番号	Q01				Q02				Q03			
	予01	予02	予03	最終	予01	予02	予03	最終	予01	予02	予03	最終
s06t21	-	○	○	○	△	-	-	-	-	-	-	△
s06t25	-	○	-	○	-	○	-	○	○	○	-	○
s06t26	-	-	△	△	-	-	-	-	-	-	-	-
s06t27	-	-	-	○	○	○	-	△	○	△	-	○

図 11 進捗状況公開ページ

4. 授業実践と運用評価

4.1. 授業実践の概要

本研究の提案である，コンテスト形式のプログラミング演習，およびコンテスト運営支援システムの有効性について，授業での実践を通じて検証を行う．まず，予備実験として，システムの自動採点の機能と GUI の簡単な評価を行った．運用は，2007 年度前期の「プログラミングⅡ(再履修)」と 2007 年度後期の 3 年生向けの応用プログラミング演習において実施した[5]．前者では，授業中の演習課題のプログラム正誤判定ツールとして自動採点機能を提供し，ソースコードの自動採点機能の実効性を検証した．後者では，60 分の制限時間を設け，教室型コンテストに近い形態で実施を行った．その結果に基づき，システムの改良や問題の精選を行い，本論で報告する運用実験を行った．

4.2. 教室型コンテストの実施

運用実験としては，2007 年度後期の 1 年生向けの「プログラミングⅡ」の授業において，教室型コンテストを実施した．本研究での本来の対象者向けの授業であり，時間や誤答による減点も行うコンテストとした．コンテストは，15 回の授業のうち，最後の 2 回の授業で，2 時限の授業の後，60 分のコンテスト時間を確保して実施した．該当の授業の受講者は，1 年生 81 名(再履修 1 年を含む)で，教師 1 名，TA4 名で演習を行っている．コンテストへの参加者は，1 回目は 70 名，2 回目は 72 名であった．なお，1 回目のコンテストの 1 週前の授業後に，システムの使い方に慣れてもらうため，簡単な 4 つの問題を用意してトライアルを実施した．

コンテストに出題した問題は，1 回のコンテストにつき 5 問で，簡単な問題を 3 問，それよりもやや応用的な問題を 2 問とした．各問題には，緩和系の予備テストを 3 個設定した．出題範囲は，授業の中で過去に出題された問題の類

題や，教科書の問題の発展形などである．利用する文法事項は，いずれも C 言語の基礎的な内容である，条件分岐，多重反復，多次元配列などである． 2 問以上の正答を達成目標とし，簡単な問題を 100 点満点，やや応用的な問題を 200 点満点とした．1 回目のコンテストには，1 回につき満点の 5%の誤答減点を科し，2 回目には，1 回目の同様の誤答減点と，問題の標準解答時間を 5 分超過するごとに 5%の時間減点を科した．

出題した問題は，1 回のコンテストにつき 5 問で，簡単な問題を 3 問，それよりもやや応用的な問題を 2 問とした．各問題には，緩和系の予備テストを 3 個設定した．出題範囲は，授業の中で過去に出題された問題の類題や，教科書の問題の発展形などである．利用する文法事項は，いずれも C 言語の基礎的な内容である，条件分岐，多重反復，多次元配列などである．2 問以上の正答を達成目標とし，簡単な問題を 100 点満点，やや応用的な問題を 200 点満点とした．1 回目のコンテストには，1 回につき満点の 5%の誤答減点を科し，2 回目には，1 回目の同様の誤答減点と，問題の標準解答時間を 5 分超過するごとに 5%の時間減点を科した．

4.3. 教室型コンテストの結果

2 回のコンテストとも，半数程度の学生が達成目標である 2 問正答に達していた．1 問 20 分程度を目安としていたが，コンテスト中に提出状況の変化を見ると，想定よりも時間が必要であり，実際には 1 問 30 分程度かかっていた．コンテストへの提出回数，教室進捗確認ページの閲覧回数などから，多くの学生がシステムを利用して問題に取り組んでいると分かった．平均提出回数は 12 回程度で，およそ 5 分に 1 度程度の頻度でシステムを利用していた．一方で，提出回数に比べて，進捗状況の確認回数は 1.5 回程度で余り多くない．これは，問題を解くことで精一杯で，他者を気にする余裕のない学生がいたようである．しかし，教卓のスクリーンに投影し，随時更新していた進捗状況ページには目が向けられていたため，もっと簡単に進捗状況を閲覧できる仕組みが必要である．

コンテストの提出数と最終テストの正答数を 5 分単位で集計したところ，図 12 のようになった．1 回目に比べ，2 回目のコンテストでは前半 30 分間に活発に提出が行われている．これは，2 回目ということで，コンテストに慣れてきたために積極的になっていると考えられる．また，後半の提出回数は，1 回目と 2 回目にあまり大きな差はないものの，最後の 5 分間には提出回数が増加しており，最後まで諦めずに問題に取り組んでいる学生が多いことが分かる．どちらのコンテストでも，30 分から 35 分頃に提出が減少している．これは，1 問につき 30 分程度で正答し，2 問目に取り組んでいるためと考えられ，コンテスト中のイメージと一致する．

4.4. 学生へのアンケート

コンテストに参加した学生に対してアンケートを実施した．各質問に対して，(4)「当てはまる」から，(1)「当てはまらない」まで 4 段階評価で，回答を行ってもらった．最後に，自由回答の項目も設けた．アンケートは，72 名に実施し，65 名から回答を得た．回答平均の結果が 2.5 ならば，どちらとも言えない結果であったと判断した．

段階的作成を支援する中間目標と部分仕様に基づいた入出力サンプルを用いた実行テストの効果に関しては，表 1 の質問を行い，結果を得た．多くの項目で平均を上回っており，積極的に実行テストを利用し，予備テストを中間目標として取り組んでいるようである．

サーバ側での自動判定による問題解答の達成感の効果，部分点と減点を考慮した得点ルー

ルと進捗状況の公開による競争の効果に関しては，表2の質問を行い，結果を得た．9つの質問のうち，7つの質問で肯定的な結果となった．特に時間のある限り新しい問題に取り組んだ，という質問で高い結果となっており，コンテストで最後まで諦めずに取り組む意識が高いという結果になった．なお，2-1-2 の質問のみ否定的な質問のため，数字が低いほど良い傾向を示している．

4.5. 宿題型コンテスト

補助的な運用実験として， 2007 年度後期の「記号処理論」の授業で，宿題型コンテストも，行った．2 年生を中心として，63 名が参加した．2 週間から 3 週間程度の期限を設け 2 度実施した．出題した問題は，1 回のコンテストにつき 8 問で，簡単な問題を 3 問，やや応用的な問題を 3 問，応用的な問題を 2 問とした．各問題には，緩和系の予備テストを 3 個設定した．授業の中で説明されたアルゴリズムをプログラムとして実現することが目的である．利用する文法事項は，いずれも C 言語の基礎的な内容である，条件分岐，多重反復，多次元配列などである．4 問以上の正答を達成目標とし，難易度に関わらず 100 点満点とした．最終テストの減点には，長期間で自由に解答できる宿題型のため，誤答減点のみを 1 回につき 5%を科した．

実施結果は，達成目標の 4 問以上に正答した学生が 63 名中，1 回目の宿題では 40 名，2 回目の宿題では 53 名であった．2 回の宿題での総提出回数はおよそ 3000 回で，1 問の正答までに，平均して 6 回程度の提出が行われていた．宿題型のコンテストの提出数を提出日ごとに集計したところ，締切の近くになると提出数が急激に増え，締切直前になってから一気に作成している学生が多かった．また，授業のある水曜日周辺の提出も増えている．宿題型のコンテストを活性化させるには，早めの解答を促すように先着順に高得点を与えるなど，得点ルールの工夫を行う必要がある．

学生からの宿題型コンテストへの感想として，何度もソースコードを提出するのが面倒だ，という意見がみられた．これは，一部の学生が，システムをコンテスト問題の最終確認にのみ用いているためと考えられる．1 つずつテストに挑戦するだけでなく，一度に全てのテストの結果が分かるなど，ユーザの利用形態に応じた機能が必要である．また，宿題型では，長期間の実施のためか，進捗状況の閲覧は，約 6 回であった．教室型コンテストに比べ，時間に余裕があれば，結果を確認することが分かる．

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．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，段階的なプログラム作成を誘導する．緩和系と部分系の予備テスト系列の構成方法と，判定における照合基準を論じた．また，演習支援サーバ tProgrEss を開発し，プログラム判定の機能と進捗状況の GUI について述べた．システムを利用し，教室型と宿題型の各形態として，実際の授業でコンテストを実施した．提出履歴の分析から，学生の利用状況や振舞いを考察し，システムの機能の改善を検討した．

今後の課題として，問題作成のガイドラインを検討し，学習事項を網羅する問題 DB を構築する．実行結果の正誤判定だけでなく，教師の目視によるコード確認も行い，各学生への個別指導を支援する機能も導入する．これらを踏まえ，運用実験を継続し，教育効果を検証する．

文　献

[1] 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).

[2] 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/

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

[4] 倉田英和, 他, “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).

[5] 倉田英和, 他, “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).

[6] 倉田英和, 他, "初級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", JSiSE 研究報告, Vol.22, No.5, pp.17-24, (2008).

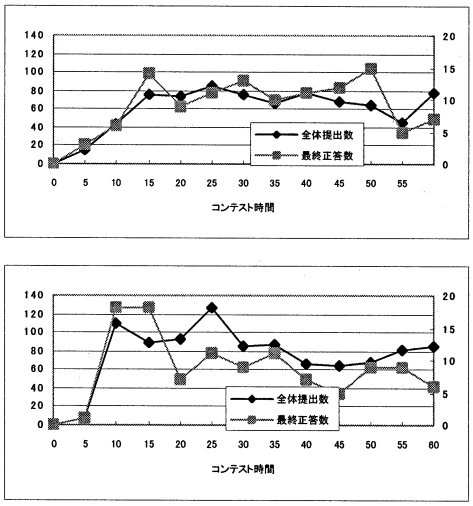


図 12 全体提出数と最終テスト提出数の推移

表 1 理解度に関するアンケート

質問番号	質問内容	平均
2-2-1	問題を選択する際，予備テストの情報も考慮した	2.85
2-2-2	解けそうな予備テストを中間目標として選び，解答の目標とした	2.66
2-3-1	予備テスト１．２をヒントとして，解き方の方針に利用した	2.68
2-3-2	予備テストの入力サンプルを使って各自の PC で実行した	2.82
2-4-1	予備テスト１，２を途中の確認に利用した	2.88
2-4-2	予備テスト３を最終テスト前の確認に利用した	3.11
2-4-3	入力の範囲を限定した予備テストは中間目標として役に立った	2.83
2-4-4	出力の一部を無視するような予備テストは中間目標として役に立った	2.49
2-5-2	各自で個別にテストが行えるので自分のペースで取り組めた	2.75

表 2 意欲の促進に関するアンケート

質問番号	質問内容	平均
2-1-1	時間のある限り新しい問題に取り組んだ	3.38
2-1-2	ノルマを達成した後は，他の問題へのやる気が少なくなった	2.10
2-4-5	最終テストの正答を目指さず，予備テストで部分点を狙った	2.40
2-4-6	最終テストの正答を目指したが，途中で部分点狙いに変えた	2.29
2-5-3	予備テストの正答を重ねることで確実に解いている実感が持てた	2.58
2-5-4	すぐに結果が分かるので，次の問題に取り組む意欲が増した	2.82
2-6-1	他人の結果を見て負けないよう自分も問題に取り組んだ	2.77
2-6-2	時間減点を気にして素早く集中して解いた	2.63
2-6-3	誤答減点を避け，最終テストの前に予備テストで慎重に提出した	3.03