2008－CE－93 (5)
2008／2／16

プログラミング演習のための進捗モニタリングシステム

内藤広志[†] 齊藤 隆[††]

プログラミング演習で学生のプログラムを採点する負荷を軽減するためにコンピュータ支援評価(Computer-Aided Assessment, CAA)システムが利用されている．CAAシステムではソフトウェア工学で提案された様々な技法を用いて複数の観点からプログラムを評価し，その評価値の合計を課題の点数とする分析的な採点法をとっている．しかし，この方法では学生の誤りを発見し指導に役立てるのは困難である．本稿では，演習中の学生の進捗状況を把握するために開発したHerculesの機能とGUI，検査項目の記述法を述べる．Herculesは400人規模の演習で利用している．Herculesの機能について教員にアンケート調査をおこない演習の効果的な実施に有効であると評価を得た．

Monitoring the Progress of Students on Programming Courses

Naito Hiroshi[†] and Saito Takashi[††]

On programming courses in computer science education, many Computer- Aided Assessment (CAA) systems have been developed and used to reduce the assessment work-load of teachers. Using techniques studied at software engineering research, CAA systems give an analytical assessment of a student's program from different point of view. But teachers are difficult to find problems which students encounter during course hours using this assessment method. This paper describe Hercules system monitoring the state of student's progress, using assessing student's program by suitable method for detecting student's errors. Hercules have been used on programing course with 400 students. Results of a questionnaire for teachers about features and GUI of Hercules shows that Hercules is useful for teaching programming.

1. はじめに

大学の1，2年次のプログラミングの演習では対象となる学生が多いため，大規模な演習教室を使って実施しなければならず，学生スタッフ（TA）を含めた多くの人的リソースを必要とする．しかし，教員やTAが教室を巡回して学生の質問に答えるだけでは効果的な演習を実施するのは難しい．学生の進捗状況を把握し，多くの学生が犯している誤りを発見し，その原因を理解した上で，学生に注意をしたりヒントを与える必要がある．そのためにコンピュータの支援が必要である．

プログラミング課題を採点する教員の負荷を軽減することを目的に，コンピュータ支援評価（Computer-Aided Assessment, CAA）システムが開発されて，欧米の大学で利用されている[1),2)]．プログラミング課題を評価するために，CAAシステムではソフトウェア工学で提案された様々なテスト技法が利用されている[1)]．テスト技法は，プログラムを実行してプログラムの機能などを検査する動的な評価と，プログラムを実行せずにソースコードを解析する静的な評価に分類される．動的な評価には，プログラムを実行し，出力データが期待される値であるかを検査する技法と，プログラムに含まれる関数やメソッドを実行し，その戻り値が期待される値であるかを検査する技法（JUnit[3)]）がある．静的な評価には，コメント数やインデント付けなどのコーディングスタイルを検査する技法や，プログラムの行数や制御構造の入れ子数などのソフトウェアメトリクスによってプログラムの品質を計測する技法[4)]がある．また，正規表現で指定したパターンがソースコード中にあるかを検査する技法[5)]や，プログラムを構文解析して生成した構文木を検査する技法[6)]がある．CourseMarker[5)]ではこれらの技法で得た評価値に重みをかけた値を合計しプログラムの点数とする分析的な採点法を採用している[7)]．しかし，この方法では再帰的関数の課題で再帰を用いずにfor文を

† 大阪工業大学 情報科学部 情報メディア学科
Department of Media Science, Faculty of Information Science and Technology, Osaka Institute of Technology
†† 大阪工業大学 情報科学部 情報システム学科
Department of Information Systems, Faculty of Information Science and Technology, Osaka Institute of Technology

表 1　学習テーマとプログラミング課題

回	学習テーマ	課題数	平均行数	最大行数	最小行数
1	演習の準備と配列	7	26	48	14
2	スタックと応用 1	6	45	58	18
3	スタックと応用 2	9	57	76	18
4	待ち行列とその応用	6	70	91	26
5	リストとその応用	7	56	101	26
6	ポインタの基礎	8	17	26	13
7	構造体の基本	9	20	25	15
8	構造体と関数 1	11	28	55	11
9	構造体と関数 2	10	41	100	26
10	連結リスト 1	9	30	44	18
11	連結リスト 2	7	31	41	24
12	再帰的関数と 2 分木	8	28	49	17
13	2 分木と 2 分探索木	8	25	34	20
	合計	105	3722		

使うなどの問題の題意に沿わないプログラムをその点数だけから発見するのは難しく，多くの学生を一斉指導する場合，学生の評価結果データを詳細に調べて指導に必要な情報を得るのは効率的ではない．また，多くの CAA システムが用いている構文解析などのテスト技法はプログラムにコンパイルエラーがある場合は利用できない．しかし，学習内容やプログラミング言語の理解不足などによって学生は自力でコンパイルエラーを解決することが困難なことが多い．そのため，コンパイルエラーを含むプログラムに対して実行可能なテスト技法が必要である．そのため，これらの問題点を解決する進捗モニタリングシステム Hercules を開発し，C 言語，JAVA 言語，XML 言語を用いる複数の教科で 2003 年度より実際に使用して機能を改良・拡張してきた[8)~10)]．

以降，ポインタや構造体などの C 言語の高度な機能とデータ構造を教える「C 演習 II」を事例として，2 章では，本演習教科の学習目標とプログラミング課題の評価項目について述べ，3 章でプログラミング演習の実施上の問題点を述べ，4 章で進捗モニタリングシステムの要件を述べる．5 章でこの要件を満たすために開発した Hercules のシステム構造と検査内容の記述法について述べ，7 章で Hercules を利用した教員へのアンケート調査の結果を述べ，8 章でまとめを述べる．

2. プログラミング演習の学習目標と評価項目

2.1 学習目標と学習内容

C 演習 II では，繰り返しや配列などの C プログラミングの基本を履修した学生に対して，次を学習目標としてプログラミング演習を実施している．

(1)　構造体，ポインタなどの高度な C 言語の文法と使用法を理解する．

教科書[11)] の 161 ページの struct4.c を元に次の仕様のプログラム work77.c を作りなさい．
(1)　入力した点の座標が直前に入力した点と同じかを調べて実行例のように出力します．

```
% ./work77.exe
3 4
3 4
2 個目の (3, 4) は直前の点と同じです．
% ./work77.exe
7 5
5 7
2 個の点には直前と同じ点はありません。
```

(2)　main 関数では次の局所変数だけを使用します．

```
struct point p1;
struct point p2;
int n = 0;        /* 入力した点の個数 */
```

図 1　プログラミング課題の例

(2)　リスト，スタック，連結リストなどのデータ構造の実現法と利用法を理解する．

(3)　上記の学習内容を応用した 100 行程度のプログラムを作成できる．

(4)　データ構造の抽象化を考慮してプログラムを作成できる．

これらの学習目標を達成するために，学習内容を基本的な項目に絞り，**表 1** の 13 回の学習テーマを設定している．各回の学習テーマに対して学習目標（単元目標）を明確にし，それをスモールステップとなる下位目標（パフォーマンス目標）に分解し，下位目標ごとに筆記課題やプログラミング課題を設けている．そのため，表 1 に示すように，多くのプログラミング課題を出題している．表 1 の行数は正解プログラムの行数（コメント行を除く）の平均，最大値，最小値である．5 回および 10 回から 13 回は，データ構造の実装部分と利用部分を複数ファイルに分割したプログラムとなっているため，全体の行数は 100 行を越えるが，表 1 にはそのうちで主に学生が作成する部分の行数を示している．

これらのプログラミング課題の特徴として，課題の学習目標が具体的かつ明確であることだけでなく，限られた時間の中で効率的に学習目標を達成するため，ゼロからプログラムを作成するのではなく，課題目標の説明用の例題プログラムを修正してプログラムを作成する問題が多いことが上げられる．例えば，第 7 回の構造体の課題として**図 1** を出題している．この課題 work77 の学習目標は構造体の比較である．構造体のメンバ同士を比較するコードを求めるために，使用する変数を制限している．なお，スペースの関係で図 1 には必要な 5 個のテストケースのうちの 2 つだけを示し，またテストデータも簡略化している．

2.2 プログラミング課題の評価項目

プログラムを作成するには，プログラム言語，コンピュータの動作原理，ライブラリ関数，データ構造とアルゴリズム，問題解決法，テストとデバッグ，実行環境と開発ツールなど多様な知識と能力を必要とする[12]．そのため，学生の作成したプログラムをどのような観点で評価するかは，教科の学習目標や教員の考え方によって様々であり，統一的な基準は存在しない[1]．例えば，文献 13) では，正確性，プログラミングスタイル，プログラムドキュメンテーション，設計ドキュメンテーションの 4 カテゴリの規準を提案している．しかし，2.1 で述べたように，C 演習 II の学習目標は大きなプログラムを設計することではなく，基本的な項目の理解が中心のため，次の 3 つの項目でプログラムを評価している．

(1) 合目的性：学習目標を満たしたプログラムを作成しているか．
(2) 正確性：プログラムの機能が正しく，特定の入力に対して指定された出力を生成するか．
(3) プログラムの構造：合目的性や正確性を達成するために必要なプログラム構造となっているか．

3. プログラミング演習の問題点

C 演習 II の 2007 年度の履修学生は 300 名で，3 つの演習室にわけ，同じ内容の演習を同時に実施している．担当するスタッフは 1 クラス当り 2，3 名の教員と 2 名の TA である．各回のパフォーマンス目標の構造に従って，教科書[11] の該当ページと対応する課題を順に記述したウェブ資料を演習の教材として作成している．学生は，ウェブ資料を閲覧し，教科書の該当ページを読み必要な概念を理解した後，Linux マシン上で gcc コンパイラや emacs エディタなどの基本的なツールを使ってプログラムを作成する．ウェブ資料に沿って学生は自分のペースで演習を進めることもできるが，学習目標の理解を確実とするために，教員は次の指導をおこなっている．

(1) 学習内容の要点を説明する．
(2) 学生の進捗状況を見て，補足説明をしたり，ヒントを出す．
(3) 多くの学生がプログラムを作成し終えたところで，正解プログラムを示して解説する．

以前は教員は TA と共に演習教室内を巡回して学生の質問に答えることで，クラス全体の進捗状況や問題点を推測していたが，学生とのやり取りから得られる情報だけではいつフィードバックを与えたらよいかを判断することが難しい．また，他のクラスの進捗状況がわからないため，クラス毎に進捗が異なってしまうという欠点があった．

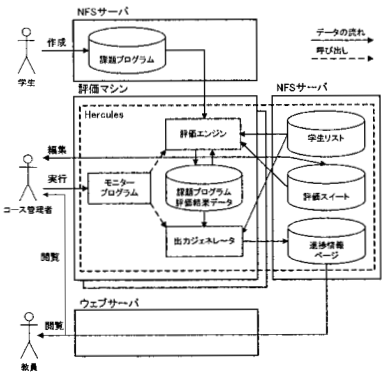

図 2 Hercules のシステム構成

4. 進捗モニタリングシステムの要件

前章で述べたような問題点があるため，プログラミング演習の進捗モニタリングシステムには次の要件が必要である．

(1) **クラスおよび学生の進捗の提示**
ヒントや正解プログラムを学生にいつ示すべきかを判断するため，作成中のプログラムを検査し，学生個人やクラス全体の進捗状況を提示する必要がある．

(2) **誤りの発見とランク付け**
補足的説明やヒントなどの指導を与えるため，学生の誤りを検出し，指導の観点から誤りのランク付けをおこなう必要がある．

(3) **不完全なプログラムの検査**
学生が作成中のプログラムを検査するために，コンパイルエラーがあるプログラムに対しても検査ができなければならない．

(4) **高いパフォーマンス**
多数の学生の進捗状況を把握するためには，プログラムを高速に検査するための高いパフォーマンスが必要である．

(5) **誤りの原因を分析するためのサポート**
進捗状況や誤りを提示するだけでなく，誤りの原因を分析し教員が指導するために必要な情報を提供することが必要である．

5. Hercules の概要

5.1 システムの構造と機能

Hercules は，**図 2** のように課題の評価内容を記述する評価スイート，各クラスの登録学生の名前などを記

述する学生リスト，評価スイートを実行して学生の課題プログラムを評価する評価エンジン，評価結果データから進捗表示ページを生成する出力ジェネレータ，評価エンジンと出力ジェネレータを呼び出して各クラスの学生のプログラムを評価し，進捗情報を生成するモニタープログラムから構成される．

Herculesでは，学生の進捗を把握するために，学生はメールやHTMLフォームを使って完成したプログラムを提出するのではなく，ホームディレクトリ中の指定されたファイルに課題プログラムを記述する．学生のホームディレクトリはNFSサーバーに作成されているので，Herculesは学生が作成中のプログラムを参照できる．

評価エンジンは，学生リストに書かれた学生のディレクトリを調べ，学生の課題プログラムをNFSサーバーより評価マシンのローカルディスクへコピーする．評価スイートを用いてコピーした課題プログラムを評価し，その評価結果データを評価マシンのローカルディスクに保存する．

出力ジェネレータは，評価結果データから全クラスの進捗状況を表示するHTMLページ（**図7**）と，各クラスの進捗状況とそのクラスの学生の進捗状況を表示するHTMLページ（**図8**）を生成する．

モニタープログラムは，評価エンジンと出力ジェネレータを繰り返し呼び出すことで，演習時間中の学生の課題プログラムを検査し，進捗情報を生成する．HTMLページには30秒毎に表示を更新する指定をMETAタグに書いているので，進捗表示ページが一定時間ごとに参照されて現在の進捗情報が表示される．

評価マシンにはLinuxマシンを使用し，主にシェル言語（tcshとbash）を用いて実装し，文字処理などの低レベルの処理にはC言語やPerl言語などを用いている．

要件（4）の高いパフォーマンスを得るため，学生の課題プログラムが以前に評価したときから変更がない場合は評価処理を実行せず，ローカルディスクに保存した評価結果データを使用して進捗表示ページを生成する．また，複数のLinuxマシンを使ってクラスや課題毎にモニタープログラムを並行に処理している．

教員とTAは，これらの進捗表示ページをブラウザで閲覧し，担当する演習教室の進捗状況を調べ，学生を指導する．本演習では，Herculesを演習中に実行するために1名の教員をコース管理者に当て，並列実行しているモニタープログラムの実行時間を監視しロードバランスの調整をしたり，進捗表示ページを監視して評価内容が誤っていた場合は評価スイートやウェブ

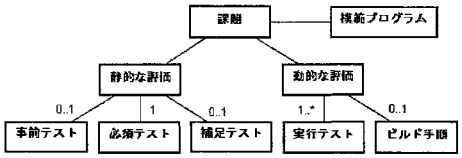

図3 評価スイートの構造

資料ページを修正するなどの作業をしている．なお，進捗表示ページはアクセス制限をかけており，教員，TA，コース管理者だけが参照できる．

5.2 評価スイートの構造とテストの実行順番

2.2で述べた評価項目を検査するため，検査によって検出される誤りの致命度によって静的な評価は事前テスト，必須テスト，補足テストにわけて記述し，動的な評価を実行テストに記述する．UML言語のクラス図で表現した評価スイートの構造を**図3**に示す．多重度が1のものは省略している．つまり正解プログラムと必須テスト以外は省略できる．

事前テストには，2.1に述べたように例題プログラムを修正してプログラムを作成する課題が多いため，例題プログラムをコピーしただけのファイルか調べる検査を記述する．また，必要なヘッダファイルをインクルードしているかなどコンパイルエラーの原因となるコードを調べる検査を記述する．

必須テストには，合目的性の検査および合目的性を満たすために必要なプログラムの構造の検査を記述する．例えば，構造体の定義が学習目標の場合は課題で指定されたメンバをもつ構造体があるかの検査を記述する．プログラムの構造の検査として，課題の仕様で

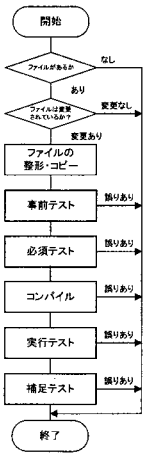

図4 テストの実行順番

指定された関数があるか，必要な入力関数があるか，if 文や for 文を使っているかなどを記述する．

ビルド手順は課題プログラムが複数ファイルから構成されるときに記述し，ファイル名のリストや実行ファイルを生成するためのコンパイル法を記述する．

実行テストには，プログラムの正確性を検査するためのテストケースを記述する．エラーテストケースを含めた任意個数のテストケースを記述でき，各テストケースに対して，入力データ（コマンド引数またはファイル名），出力データの検査，関数の呼び出し関係の検査を記述する．

補足テストには，冗長なコードなどのプログラムの構造の検査を記述する．例えば，必要以上の個数の if 文や for 文がないか，不要な関数の使用がないかを記述する．

これらのテストは，**図 4** に示す順番で実行する．まず，課題プログラムのファイルの有無を検査する．ファイルがあった場合はその修正日時を調べ，前回に評価処理を実行してからファイルが変更されている場合だけ評価処理を続ける．次のファイルの整形・コピー処理では，コーディングスタイルのゆれを少なくするため，空行，コメント，余分な空白をソースコードから削除してから整形処理をしたものを評価サーバにコピーする．以降の処理はこの整形したファイルに対しておこなう．

評価値は点数ではなく，○（正しい），△（冗長なコード，出力データが正確でない），▲（実行エラー），×（コンパイルエラー，学習目標を満たしていない），★（ファイルが存在しない）の 5 段階を付ける．

5.3 検査項目の記述法

図 3 の各テストの検査項目は**表 2** の検査コマンド☆とシェル言語を使って記述する．表 2 のように，検査コマンドには静的な評価用と動的な評価用のものがある．必須テストと補足テストは**図 5** のように 1 つのファイルにまとめて記述する．実行テストは**図 6** のようにテストケースごとに 1 つのファイルに記述する．検査コマンドの-N オプションは検査項目を識別するための番号を指定するもので，-e オプションは検査項目の評価値を指定するものである．例としてあげた図 5 と図 6 は図 1 の課題 work77 のテストを簡略したものである．課題 work77 には 5 つのテストケースがあるので図 6 と同様な実行テストを更に 4 つ記述している．

必須テストには「課題の前提条件」，「学習目標」，「プログラムの構造」の順に検査項目を記述する．fhaveStructDef.sh や fhaveFunction.sh など検査コマンドを使うことで，プログラミング言語の構成要素の検査を宣言的に記述できる．また，fpatternInFunction.sh を使うことで，関数の定義中に変数の定義があるか，if 文や for 文など制御文があるか，関数呼び出しがあるかなどを正規表現を使って検査することができる．検査コマンドでは**表 3** のパターン式を引数に指定できるので，複雑なパターンを指定して検査できる．図 5 で使用している変数 s はゼロ個以上の空白を表し，変数 sym は識別子を表す変数である．

実行テストでは図 6 のように fpatternInOutput.sh コマンドを使用して標準出力を検査する．必須テストと同様に表 3 のパターン式を使うことで出力のゆれや数値計算の誤差に対応した検査項目を記述できる．

正規表現を指定した検査コマンドとシェル言語の機能を使って静的な評価をすることで，要件（1）のコンパイルエラーのあるプログラムの検査が可能となる．

5.4 進捗情報の提示法

要件（1）のクラスおよび学生の進捗状況を提示するために，5.1 で述べたようにモニタープログラムは学生のプログラムを検査した結果から図 7 の全クラスの進捗表示ページと図 8 のクラス別の進捗表示ページを出力する．全クラス進捗表示ページは単元の課題名（workc1 など）を横方向とし，クラス名（p1 など）を縦方向として全クラスの進捗状況を表示する．各セルの上部には課題を評価した日時を表示し，前回の評価以降にソースファイルを更新した学生数を括弧の中に表示する．セルの下部には課題の評価値の度数をパーセントで表示する．このページによって他クラスとの進捗状況の比較ができる．

図 7 の全クラス進捗表示ページでクラス名をクリックすると，図 8 のクラスの進捗表示ページを表示する．このページは HTML フレーム機能を使って記述しており，左フレームに各課題の評価日時，ファイル更新学生数，評価値の度数を，右上フレームに課題の誤りのリストを，右下フレームに各学生の座席番号，学生番号，各課題の評価値とファイルの変更日時を表示する．

要件（2）の誤り情報の提示のために，左フレームの課題名をクリックすると，右上フレームにその課題で検出した誤りを人数の多い順に並べて表示する．各行には，誤りの度数（誤りをした学生の数），6 桁のエラー番号，評価値，エラーメッセージを表示する．また，要件（5）の誤りの原因を分析するために，右上フレームに表示している誤りエラー番号をクリック

☆ この他に JAVA 言語や XML 言語用の検査コマンドも提供している．

表 2 C 言語用の検査コマンド

種類	検査コマンド	機能
静的な評価	fcompareWithBasefile.sh	ファイルの類似度を検査する
	fhaveStructDef.sh	構造体の宣言があるかを検査する
	fhaveMember.sh	構造体の宣言にメンバが定義されているかを検査する
	fhaveNotMember.sh	構造体の宣言にメンバが定義されていないかを検査する
	fhaveFunction.sh	関数の定義があるかを検査する
	fhaveNotFunction.sh	関数の定義がないかを検査する
	fpatternInFunction.sh	関数定義中にパターンがあるかを検査する
	fpattern.sh	ファイル中にパターンがあるかを検査する
	fgetPattern.sh	ファイル中の何行目にパターンがあるかなどを調べる
動的な評価	fexecute.sh	プログラムを実行する
	fhaveOutput.sh	標準出力の有無，サイズの上限および下限を検査する
	fhaveNotOutput.sh	標準出力がないかを検査する
	fhaveErrorOutput.sh	標準エラーの有無，パターンがあるかを検査する
	fpatternInOutput.sh	標準出力中にパターンがあるかを検査する
	fgetPatternInOutput.sh	標準出力中の何行目にパターンがあるかなどを調べる
	fcallgraph.sh	関数が呼ばれた回数，どの関数から呼ばれたかを検査する

表 3 パターン式の種類

パターン式		検査内容
INCLUDE	*pattern* [MORE|EQUAL|LESS] *occurence*	正規表現 *pattern* の出現回数を *occurence* と比較する
LINE	*pattern from to*	正規表現 *pattern* が *from* 行から *to* 行までにあるか調べる
AND	*pattern1 pattern2*	正規表現 *pattern1* と *pattern2* を含む行があるか調べる
OR	*pattern1 pattern2 ...*	いずれかの正規表現を含む行があるか調べる
NOT	*pattern1 pattern2 ...*	いずれかの正規表現を含む行がないか調べる

```
if ( $argv[1] == "MUST" ) then
  # 課題の前提条件：int 型のメンバ x と y からなり，点を表す構造体型 point の宣言があるか？
  fhaveStructDef.sh     -N 01 -e "×" point || exit 1
  fhaveMember.sh        -N 02 -e "×" point int x || exit 1
  fhaveMember.sh        -N 03 -e "×" point int y || exit 1
  fpatternInFunction.sh -N 04 -e "×" int "main(void)" INCLUDE "^${s}struct point ${sym}" || exit 1
  # 学習目標：構造体の比較．構造体型 point のメンバ同士の比較があるか？
  fpatternInFunction.sh -N 05 -e "×" int "main(void)" INCLUDE "${sym}\.x${s}(==|\!=)${s}${sym}\.x" || exit 1
  fpatternInFunction.sh -N 06 -e "×" int "main(void)" INCLUDE "${sym}\.y${s}(==|\!=)${s}${sym}\.y" || exit 1
  # プログラムの構造（必須）：標準入力から点データを scanf 関数を使って入力するコードがあるか？
  fpatternInFunction.sh -N 07 -e "×" int "main(void)" OR      "while${s}\(" "for${s}\(" || exit 1
  fpatternInFunction.sh -N 08 -e "×" int "main(void)" INCLUDE "scanf${s}\("  || exit 1
  fpatternInFunction.sh -N 09 -e "×" int "main(void)" INCLUDE "EOF"          || exit 1
  fpatternInFunction.sh -N 0a -e "×" int "main(void)" INCLUDE "printf${s}\(" || exit 1
else if ( $argv[1] == "SHOULD" ) then
  # プログラムの構造（補助）
  fpatternInFunction.sh -N 0b -e "△" int "main(void)" INCLUDE "while${s}\(${s}scanf.*EOF${s}\)" || exit 1
  fpatternInFunction.sh -N 0c -e "△" int "main(void)" INCLUDE "${sym}${s}=${s}${sym}" || exit 1
endif
exit 0
```

図 5 課題 work77 の必須テストと補足テスト

すると，誤りを犯している学生のソースプログラム，実行結果などを含むエラーの詳細情報を表示する．

6. 評価スイートの作成法と問題点

文献 14) では，評価スイートを作成する際，プログラムの機能，入力，出力，エラー処理を記述した評価仕様を作成する必要があると述べている．C 演習 II のような基本的な概念を学ぶ演習では，これらの他に課題の学習目標を明確にする必要がある．2 章では C 演習 II の課題は学習目標が明確であるという述べたが，図 1 の例のように問題文に学習課題を明示的に書いていない．課題で何をすべきかを学生に考えさせることが学習効果を高める上で有効と考えているからである．そのため，評価スイートを作成する際，正解プログラ

```
# ファイル work77-1.txt を入力データとしてプログラムを実行する.
setenv HTITLE  'INFILE'
setenv HINFILE "$SAMPLEDIR/work77-1.txt"
fexecute.sh || exit 1

# 出力があるかを調べる.
fhaveOutput.sh -N 10 -e "×" || exit 1

# 標準出力データを調べる.
fpatternInOutput.sh -N 11 -e "▲" INCLUDE "2 個.*\(${s}3${s},${s}4${s}\).*同じ" || exit 1
fpatternInOutput.sh -N 12 -e "△" INCLUDE "2 個目の${s}\(${s}3${s},${s}4${s}\)${s}は直前の点と同じです" || exit 1

exit 0
```

図 6　課題 work77 の実行テスト

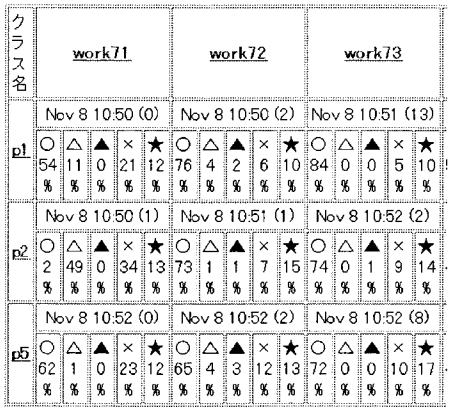

図 7　全クラスの進捗表示ページ

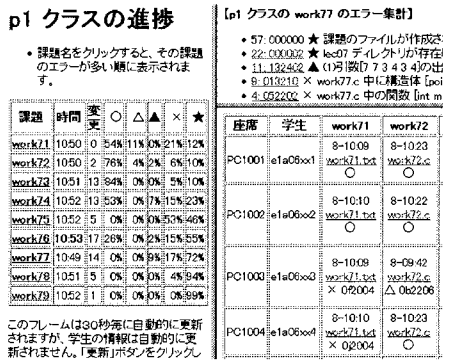

図 8　クラスの進捗表示ページ

ムと問題文を検討して学習目標を含めた評価仕様を明らかにし，5.2 章で述べた構造に従って検査項目を分類し，重要度の高い検査項目が最初に検査されるように記述する．次に正解プログラムを用いて Hercules を実行し，評価スイートが妥当であるかを確かめる．問題文は自動採点が可能なほど曖昧さがないか，十分

表 4　テスト種類ごとの検査項目数の平均

回	事前テスト	必須テスト	補足テスト	テストケース数	実行テスト	総計
1	2	6	1	3	17	26
2	3	10	1	4	43	57
3	3	9	4	4	30	46
4	4	13	2	4	44	63
5	8	11	1	4	35	55
6	2	5	1	2	9	17
7	4	10	1	4	12	27
8	7	14	1	3	17	38
9	3	13	3	4	23	43
10	6	8	3	2	16	33
11	5	9	1	3	21	35
12	5	10	4	4	23	41
13	7	11	1	4	19	37
平均	5	10	2	3	23	39

な個数のテストケースがあるかも検討するので，評価スイートが完成するまでに 1 時間ほどかかる．

プログラミング課題では多様な解答が可能であるため，事前に完全な評価スイートを作成するのは難しい．そのため，演習時間中に評価結果データを調べて検査項目を追加・削除したり，実行順を変更する必要がある．**表 4** に示すように評価スイートの検査項目の数は学習テーマによって異なり，複雑なものは 100 個となる課題もある．評価スイートの作成は試行錯誤的プロセスとなり多くの時間がかかるため，効率的に評価スイートを作成するためのツールが必要である．

7. アンケート調査

Hercules の機能と GUI を評価するために 10 名の教員を対象にアンケート調査を実施した．**表 5** がアンケート調査の集計である．教員は Hercules を使って演習を実施するだけでなく，プログラミング課題の締め切り後，評価結果データを調べて学生の誤りの原因を解説するページを作成している．表 5 の 1〜5 は演習を円滑に進める観点からのアンケート項目で，6〜9

表 5 アンケート評価
Table 5 Questionnaire about the system features

	質問内容	良い	普通	悪い
1	進捗状況ページをよく参照しましたか	9	1	0
2	進捗状況ページの操作方法はよいですか	5	5	0
3	誤りの原因を分析できましたか	10	0	0
4	クラス全体の進捗がわかりましたか	8	2	0
5	学生へのフィードバックに有用でしたか	7	2	0
6	課題解説を作成するのに効果的でしたか	7	1	0
7	正しい解答が誤りになっていましたか	3	3	2
8	誤った解答が正解になっていましたか	2	6	1
9	評価値は適切でしたか	2	7	0
10	課題の作成・改善に有効でしたか	6	3	0

は課題解説を作成する観点からのアンケート項目で，10 は教材を作成する観点からのアンケート項目である．なお，7 と 8 では「良い」が「少ない」を「悪い」が「多い」を表わす．

1〜5 の項目では「良い」と解答した教員が多く，Hercules を演習で利用することの有用性が評価された．ただ，2 の操作性の評価が低いので教員の要望を調査して操作性を改善したい．8〜9 では「普通」と解答した教員が多い．これは，新規のプログラミング課題では演習の前に作成した評価スイートの品質が悪く，演習中に評価結果データを見ながら評価スイートを修正するため，提示された進捗状況が不正確であるためである．10 では「良い」の解答が多い．実際，演習中にモニタリングしていると誤理解の多様さに驚く．学生が作成中のプログラムをモニターすることは課題の質を改善する上で非常に有益である．

8. おわりに

本学では，C 演習 II 以外のプログラミング演習（学生数は 600 人規模）でも Hercules を利用している．これらの大規模な演習でも，教員は進捗表示ページを監視し，正解率が 30%となった場合や正解率が変わらない場合に補足説明やヒントを出したり，正解率が 60%となったら正解プログラムを提示するといった教授法がおこなえ，進捗状況を把握することで適切な指導が可能となった．ただし，このように進捗状況だけを見て演習をおこなうと，学生の学習スピードによっては演習の進行が遅くなってしまうので，事前に課題当りの時間配分を示したタイムチャートを作成して演習をおこなっている．

しかし，評価スイートを作成するコストは高く，shell 言語で実装しているために評価パフォーマンスは十分とは言えない．Hercules の機能や実装法などを更に評価し，多くの演習で利用可能なモニタリングシステムに改良していきたい．

参 考 文 献

1) Ala-Mutka, K.: A Survey of Automated Assessment Approaches for Programming Assignments, *Computer Science Education*, Vol. 15, No.2, pp.83–102 (2005).
2) Douce, C., Livingstone, D. and Orwell, J.: Automatic test-based assessment of programming: A review, *Educational Resources in Computing*, Vol.5, No.3 (2005).
3) Beck, K.: *Test-Driven Development: By Example*, Addison-Wesley (2002).
4) Beizer, B.: *Software Testing Techniques*, Intl Thomson Computer (1990).
5) Higgins, C., Gray, G., Symeonidis, P. and Tsintsifas, A.: Automated assessment and experiences of teaching programming, *Educational Resources in Computing*, Vol. 5, No. 3 (2005).
6) Truong, N., Roe, P. and Bancroft, P.: Static Analysis of Students' Java Programs, *6th conference on Australasian computing education*, pp.317–325 (2004).
7) Olson, D.M.: The reliability of analytic and holistic methods in rating students' computer programs, *9th SIGCSE technical symposium on Computer science education*, pp. 293–298 (1988).
8) 内藤広志：プログラミング演習の総合支援システムの概要，第 3 回情報科学技術フォーラム (2004).
9) 内藤広志：プログラミング課題の採点システムの拡張可能なフレームワーク，第 68 回全国大会，情報処理学会 (2006).
10) 内藤広志：プログラミング演習の進捗状況のモニタリングシステム，第 68 回全国大会，情報処理学会 (2006).
11) 内藤広志，齊藤　隆：演習でマスターする C 言語とデータ構造，共立出版 (2006).
12) Robins, A., Rountree, J. and Rountree, N.: Learning and Teaching Programming: A Review and Discussion, *Computer Science Education*, Vol.13, No.2, pp.137–172 (2003).
13) Smith, L. and Cordova, J.: Weighted primary trait analysis for computer program evaluation, *Computer Science Education*, Vol.20, No.6, pp. 14–19 (2005).
14) Isong, J.: Developing an automated program checkers, *20th annual CCSC South Central conference*, pp.218–224 (2001).