WebIDE を用いた プログラミング初学者向けのテスト評価支援方法の提案

2012SE010 浅野 勝　2012SE114 小林 悟
指導教員：蜂巣 吉成

1　はじめに

大学におけるプログラミング演習では，コーディングだけでなく，プログラムの誤りや漏れを発見するためのテストも必要不可欠な工程として重要である．現在のプログラミング演習は，学習者が作成したソースプログラムや実行結果を最後に提出して演習を終了し，それを評価する方法が一般的である．学習者においても実際に提出するのはソースコードであり，コーディングに注力するので，テストに時間をかけないことも多い．演習課題によっては，プログラムの入力と出力が記述された実行例が提示されている．これは単に実行例を示すだけでなく，学習者にテストケースを考えさせるために，意図的に必要なテストケースを網羅していないことがある．学習者の中にはこの実行例を動作させることをもって，プログラムが正しく書かれているかテストをしたと判断する者もいる．この学習方法では，教員が学習者の実際に行ったテストを把握しづらく，テストについて学習者へのフィードバックを行うことが難しい．こうした背景から，プログラミング演習において，学習者にテストの重要性を理解させること，学習者が適切なテストケースの作成ができるようになることは困難である．

2013 年度の卒業研究 “プログラミング演習におけるテスト支援方法の提案”[1] では，Web ベースの統合開発環境 (Web based Integrated Development Environment: WebIDE) を構築し，学習者が行ったテストが十分であるか評価し，不十分だった場合は十分なテストを設計できるように支援する環境を提案している．この研究で提案された方法には 2 つの問題点がある．学習者のテストケースを評価するために，演習課題毎に教員が評価基準とコメントを記述する方式を採用しているが，入力数が不定でテストデータに構造をもつような演習課題に対応することができない．このような課題は一定数存在し，構造体の学習をする上でも必要な課題である．また，評価基準は複数のデータに対する処理を記述することが多いので，関数を用意しているが，あらかじめ定義された関数しか使うことができず，記述できない評価基準が存在する．

本研究では学習者が課題の仕様を満たしているか十分に確認できるテストを設計できるようになることを目的とする．学習者は境界値分析，同値分割などを知識としてだけでなく，経験的に学ぶことによってテストを設計する力を実につけることができる．そのためのアプローチとして，2013 年度の研究で提案された WebIDE を用いて，2 つの問題点の解決を図る．評価基準の記述方法を入力データの構造を定義するようにする．入力データの型と名前，入力回数を宣言することで，評価基準内において宣言した変数が使用できる．これにより構造をもったデータが扱えるようにした．関数については専用の関数ライブラリを用意して，教員が自ら任意の関数を追加することで，評価基準の記述方法に拡張性をもたせた．これらの問題点の解決によって，学習者にテスト技法を経験的に学ばせることのできる演習課題を多く扱えるようになり，学習者のテスト設計を支援することができる．

2　関連研究

Edwards[2] は，指導者の作成したプログラムとテストケースを用いて，学習者が作成したプログラムとテストケースを評価するシステムを開発している．テスト駆動開発に沿ってプログラムを作成し，学生の「テストの有効性」,「テストの完全性」,「コードの正確さ」の 3 つの基準で採点を行い，その結果を学習者に即時フィードバックすることで，テストに対する動機付けと支援を行っている．

吉田ら [3] は，指導者でなく，学生自身がコードを評価することができる手法 Triple Checked Testing を提案している．これらのチェックを全て行うことでテストケースと実装が要求仕様を満たしているか確認することができ，満たしていない場合は漏れを指摘する．さらに，テスト駆動開発に基づいて演習を進めることで，コードを書くことだけでなく，ソフトウェアテストやコードリーディングの学習を支援する．

文献 [2][3] は，指導者が課題のプログラムとテストケースを用意すれば，どのような問題にも対応することができるが，誤りや漏れが検出された際に，具体的にどこが誤りであるのか，どのような漏れがあるのかを学習者に伝えることができないので，本研究が対象とする初学者には向いていない．文献 [1][5] では，教員が課題のテストケース毎にコメントを設定することで，具体的な誤りや漏れを学習者に通知し，テストを支援している．

上河内ら [4] は，Java 言語を対象に，学習者がテストをどこからどのようにどの程度行うかを理解するために，コードをテストできる能力，テストするメソッドの決定，テストデータの作成，テストプログラム作成と実行の 3 ステップで定義した単体テストプロセスに対するテスト方法学習支援を提案している．オブジェクト指向言語の単体テストを計画してテストプログラム (テストドライバ) を記述できるまでが目標なので，本研究よりもプログラミングの理解がある学習者向けである．

3　従来の学習者のテストケース評価支援

3.1　WebIDE の機能

文献 [1][5] では，WebIDE を用いてテストケースの作成，実行，テストケース評価機能を実現している．学習者がプログラムとテストケースを入力すると，WebIDE がテストドライバの自動生成とテストの実行を行う．

ローカル PC の端末上で実行した場合は，プログラム実行時の入出力を教員が把握するのは難しい．しかし，WebIDE では Web サーバ上で実行を行うので，その度に学習者毎にログが生成されて入出力の結果を取得することが可能である．

3.2　テストケース評価機能

テストケースを評価する方法には，学習者が行ったすべてのテストケースを取得し，教員がそれを直接確認する方法や教員の作成したテストケースを取得し，教員の作成したテストケースと学習者の作成したテストケースを比較するなどの方法がある．文献 [1][5] では，学習者に十分なテストケースを作成させる方法として，学習者が設計したテストを教員が課題毎にテストケースの評価基準を記述して用意し，それをどの程度パスするかによって，テストケースを評価する方法を提案している．評価基準をパスしない場合には，コメントを表示して学習者に即時フィードバックを行う．評価基準やコメントは，学習者の習熟度や課題の趣旨に応じて教員が自ら記述することによって，演習を柔軟に行うことができる．

3.3　従来の評価基準記述方法

課題毎にテストケースの評価基準を記述するには，評価基準を記述するための方法が必要になる． 既存のプログラミング言語で記述することも可能だが，記述量が多くなり，教員への負担になる．文献 [1][5] では， 以下の専用の評価基準記述方法を用意することで，簡潔に評価基準が記述できるようになっている．

評価基準番号 [tab] **判定基準** [tab] **コメント**

判定基準はテストケースにおける入力データが満たすべき条件であり，その条件を満たすテストケースがなかったときに，学習者にコメントが表示される．1 つの評価基準に対して，2 つ以上の判定基準がある場合は，評価基準の種類を評価基準番号とし，判定基準と区別することで，同じ評価基準内であっても， 判定基準毎にコメントを設定することができる．テストカバレッジは，評価基準番号を基準として，どれだけの評価基準番号の判定基準が満たされているかの割合で示される．

判定基準において，入力データは先頭から順に$1, $2, 最後の入力データは$$，最後から 1 つ前の入力データは$$_1 と記述する．total=$1+$2+$3+$4+$5 のように変数を定義することもできる．例えば，$1 から$5 までの合計が 380 以上であるという評価基準は次のように記述する．

```
total = $1+$2+$3+$4+$5
1 [tab] total>=380 [tab] 合計が 380 以上になるテストを行ってください
```

3.4　従来の評価基準記述能力

文献 [1][5] の評価基準記述方法は， 入力数が一定である演習課題の評価基準を記述することができる．入力データの型については，数値型，文字型，文字列型，これらの複合型についていずれも記述できる．また，入力数が一定でない演習課題であっても，入力されるデータに構造を持たないような課題の評価基準を記述することができる．ある数が入力されるまで整数を読み込むような課題や EOF まで文字を読み込むような課題が例として挙げられる．しかし入力数が一定でなく， 入力されるデータに構造を持つような課題の評価基準を記述することはできない． 最初に読み込んだ人数分の体長データ (身長, 体重) を読み込むような課題が例として挙げられる． この場合，i 人目の身長， 体重データは，それぞれ$(2*i)，$(2*i+1) に格納されるが，文献 [1][5] の記述方法では，このような記述はできない．

3.5　従来の評価基準記述方法における問題点

南山大学で行われているプログラミング演習課題の中で，2012 年度プログラミング基礎演習，および 2013 年度プログラミング応用実習を対象に，演習課題の分類・分析を行ったところ，入力数が一定でなく，入力されるデータに構造を持つような演習課題は，構造体の学習部分で見られた．構造体を用いた演習課題に対して評価基準を記述できないことは問題である．

また，文献 [1][5] の評価基準記述方法では，判定基準内で使われる関数は，合計や最大値，最小値などあらかじめ用意されたものだけで，教員はその関数しか利用することができない．しかし，演習課題によっては特殊な処理を行う関数が必要になる場合も考えられる．本研究では教員自身が関数を用意できる評価基準記述方法を提案する．

4　データ構造を考慮した評価基準記述方法の提案

本研究では，3.5 節で提示した問題点を解決するため，次の機能を提案する．

- 入力データ構造の定義
- 判定基準内で使われる関数を教員が用意できる仕組み

入力数が一定でなく，入力されるデータに構造を持つような演習課題に対して，より有効的な記述方法として入力データ構造の定義を提案する．また，複数のデータに対する処理方法を明確にし，教員が関数を定義する方法を提案する．代替案として入力数が一定でなく，入力されるデータに構造を持つような課題に対応するために，3.4 節にあるように入力データを計算式を用いて記述する方法も考えられる．しかし，この方法は計算式に間違いが含まれた場

合，可読性が低く誤りが分かりにくい．また，学習者の入力が課題の仕様に対して，正しいことが前提なので誤った入力に対する指摘はできない．

教員の模範解答から変数・構造体の宣言部分を抜きだして入力データ構造の定義とし，残りの部分を手動で記述する方法も考えられる．しかし残りの部分をプログラミング言語で直接記述するのは入力データ構造の定義を抜きだしたとしても，本研究の方法より手間がかかると考えた．

4.1 入力データ構造の定義

本研究では，入力データの扱いを自動変数を割り当てる方法から，入力データ構造を定義する方法に変更した．

本研究での入力データ構造の定義の記述方法は

(入力データの型 データ名){入力回数}

のようにする．

このように入力データ構造を定義することで，今まで評価基準を記述できなかった演習課題に対応しただけでなく，学習者の記述した入力が不適切な場合に，メッセージを表示することが可能である．

入力データ構造に対して，次のデータ型を用意した．

- int, double(任意の整数，実数)
- pint, pdoule(正の整数，実数)
- nint, ndouble(負の整数，実数)
- uint, udouble(0 または正の整数，実数)
- znint, zndouble(0 または負の整数，実数)
- str(文字，文字列)

以上の 11 種類を用意すれば整数と実数の区別ができ，‘負の数が読み込まれるまで’ といった条件でのデータマッピングも可能になると考えた．このようなある条件でのデータマッピングについては，上記の型による判別のみ可能となる．‘999 が読み込まれるまで’ といった任意の数を条件に指定できないが，ほとんどの課題が 0 を境界とする数を条件としていたので十分と考えた．

入力回数は，そのデータと同じ構造のデータが何回入力されるかを記述する．この項目が記述されていない場合は，1 回だけの入力とする．{} の中に定数や変数を記述することで入力回数を決定でき，入力回数が決まってない場合は*(0 回以上出現) と記述する．また，n 行 m 列行列の入力などにも対応するため回数を +，*，-，/ を用いて四則演算でも表せるようにした．

本研究での記述方法は，このように入力データ構造を定義した上で，評価基準を記述していく．

例えば，最初に入力した人数分の身長，体重を読み込むような課題で，身長が 120cm の場合のテストを行う場合 (遊具等の身長制限を判定するプログラムが考えられる)，評価基準は次のようになる．

[入力データの宣言]

```
(int num)
(double height, double weight){num}
```

[評価基準]

1[tab]height == 120 [tab] 身長が 120cm の場合のテストを行なってください．

ここで，num には，人数が読み込まれ，height，weight には，それぞれ身長，体重のデータが読み込まれる．入力回数の部分に num と記述することで，(height, weight) を num 回読み込むという記述が可能となる．

同じ構造のデータを複数読み込む場合，全てのデータに対して，同じテストを行う必要性は低い．今回のように n 人のデータを読み込む場合，n 人全員の身長が 120cm である場合のテストを行う必要性はない．本研究では同じ構造のデータを複数回読み込む場合，複数のデータの中で 1 つでも，条件を満たすテストケースがあれば良いこととする．この例の場合，1 人でも身長が 120cm であるテストを行えば良いことになる．しかし複数の座標を読み込みその座標間の点間距離を求める演習課題で，すべての座標が等しい場合のテストを行う場合などは上記のような記述方法では記述できない．このような演習課題に対しては 4.3 節の関数による判定基準の記述で対応することが可能である．

4.2 関数の定義

文献 [1][5] では，判定基準の記述を簡単にするために，判定基準内でいくつかの関数を使えるようにしている．関数は判定基準を記述する上で必要な場面が多く，本研究の記述方法にも導入したい．しかし，入力データ構造を定義するように評価基準の記述方法を変更したので，関数の仕様もそれに合わせる必要がある． 4.1 節の例を図 1 に示

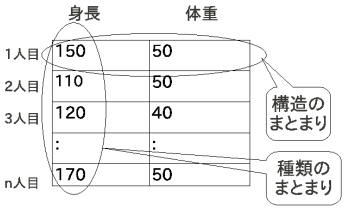

図 1 入力データ構造のグループ分け

す．この図での，縦のグループを種類のまとまり，横のグループを構造のまとまりと定義する．関数はこれら 2 種類のまとまりに対して利用できる必要がある．

構造のまとまりと種類のまとまりで関数を利用する方法として，それぞれに別の関数を用意する方法が考えられる．構造のまとまりに対する funcRecord 関数，種類のまとまりに対する funcHeight,funcWeight 関数というようにそれぞれに関数を用意する方法である．しかしこの方法では，それぞれの関数をいくつも用意しなければならず手間がかかる．

本研究では関数は配列に対して処理を行うものとした．関数の実引数として入力データ名を記述した場合，種類のまとまりの配列とし，実引数を [ ] で囲って記述した場合，[ ] 内のデータを要素とする構造のまとまりの配列とする．図 1 を例にすると，種類のまとまりはそれぞれ，func(height)，func(weight) と記述でき，構造のまとまり

は func([height,weight]) と記述できる．また専用の関数ライブラリを用意し，そこに任意の関数を追加することで事前に用意されていない関数も使用できるようにした．これにより，教員は追加の関数を準備するとき，配列を受け取り，その配列に対して処理を行う部分のみ作成すればよく，教員への負担の増加を抑えることが可能である．この関数は，PHP で記述する．これはこの関数が動作する WebIDE が PHP で実装されているので，それに合わせたためである．

関数の記述方法の代替案として，マクロ用の言語を提供し，教員が関数を記述することなども考えられるが，新たな言語を教員に覚えてもらうことの負担や，PHP の記述力の高さを考慮し，この方法を選択した．

4.3 関数による判定基準

次のような入力データ構造で表される n 個の座標 (x,y) を読み込み，それらの点の距離を求めるような課題を例とする．

```
(int n)
(int x, int y){n}
```

このような演習課題ですべての座標が等しいテストを行うという判定基準を用意する場合を考える．判定基準を x == y とした場合，ある点の x 座標と y 座標の値が等しいという条件になってしまい，記述したい条件とは異なる．

読み込んだ配列の要素が全て等しい値なら true を返す equal_all 関数を用意すれば，equal_all(x) && equal_all(y) と記述することで，すべての座標が等しいテストを行うという判定条件にできる．このように関数を活用すれば，すべての入力が等しい，最初の入力や最後の入力が 最大値，最小値になるという判定基準も記述できる．

5 考察

5.1 記述能力

2012 年度プログラミング基礎演習と 2013 年度プログラミング応用実習，文献 [6][7] にある演習課題について，評価基準変換ツールを用いて変換させたところ，対応している課題は，WebIDE 上で正しく動作するコードを生成することができた．

本研究での記述方法では評価基準を記述できない課題として，最初の入力がデータ構造を示すような課題 (最初に 1 を入力した場合は学生番号，名前を入力する．最初に 2 を入力した場合は教員名，担当教科を入力する等) は，データ構造を定義することができず，評価基準を記述できなかった．このような課題は共用体の課題である．本研究の記述方法では，共用体の記述ができないが，共用体は初学者向けの内容ではないため，本研究では扱わない．また，行列などを扱う課題で対角行列を二次元配列として処理を行うような判定基準の記述はできない．これは，本研究の入力データ構造の定義がレコード型の記述方法になっているからである．入力データ構造の定義を二次元配列としても記述できればこのような判定基準の記述も可能となる．ベクトルや行列などの数学的な処理が多い場合はこのような定義が有効である．

5.2 テストの冗長性について

学習者が新たなテストを追加してもカバレッジが上がらなかった場合，必要ないテストか重複したテストかがわからないことがある．また，学習者がランダムテストを行なって評価基準を満たすことを防ぎたい．そのために本研究では，他のテストで既に評価基準を満たしているテストケース，どの評価基準も満たさないテストケースについてもコメントを表示する必要があると考えた．この機能は 3.3 節でのカバレッジ計算の機能を応用して実現した．

6 おわりに

本研究では学習者が十分なテストを設計できるようになるために，WebIDE のテストケース評価機能において，データ構造を定義し，ユーザ定義の関数が記述できるように拡張した．

今後の課題としては，教員の負担をさらに軽減するために，入力データ構造や問題分類からの判定基準の雛形の用意や，コメントの候補を挙げるといった評価基準の半自動生成が挙げられる．また，学習者のテスト設計に実際に有効であるのか評価する必要がある．

参考文献

[1] 伊倉慶太，加藤綾真，加藤優磨，“プログラミング演習におけるテスト支援方法の提案”，南山大学情報理工学部 2013 年度卒業論文，2013.

[2] Edwards,S.H. “Using test-driven development in the classroom:Providing students with automatic, concrete feedback on performance”, EISTA’ 03,2003,pp.421-426.

[3] 吉田 英輔，角川 裕次，“テスト駆動開発に基づくプログラミング学習支援システム 初心者開発者のためのセルフトレーニングアーキテクチャ”，信学技報．ソフトウェアサイエンス，Vol. 105，No. 331，2005，pp. 27-32.

[4] 上河内頌之，松浦佐江子，“Java プログラミング初学者に対するテスト方法学習支援ツール”，信学技報．教育工学，Vol. 106，No. 364，2006，pp. 37-42.

[5] 蜂巣吉成，吉田敦，阿草清滋，“WebIDE を用いたプログラミング演習におけるテストケース評価システムの提案”，ソフトウェア工学の基礎 21 日本ソフトウェア科学会 FOSE 2014，2014，pp.241-250.

[6] 柴田望洋，赤尾浩，肘井信一，高木宏典，解きながら学ぶ C 言語，SoftBank Creative，2004.

[7] 柴田望洋, 新版明解 C 言語入門編, SoftBank Creative, 2004.