様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　５　月　２９　日現在

機関番号：　３３９１７
研究種目：　基盤研究(C)（一般）
研究期間：　2014～2016
課題番号：　２６３５０３４４

研究課題名（和文）Web統合開発環境を利用したプログラミング教育支援方法

研究課題名（英文）Support for Programming Education using Web-based Integrated Development Environment

研究代表者

蜂巣　吉成（Yoshinari, Hachisu）

南山大学・理工学部・教授

研究者番号：３０３１９２９８

交付決定額（研究期間全体）：（直接経費）　　2,000,000円

研究成果の概要（和文）：情報系の大学におけるプログラミング演習を想定し、(1)学習者の進捗状況の把握、(2)学習者のソフトウェアテストに対する意識向上に関して研究を行った。(1)について、ソースコードの編集過程から制御構造や制御式を分析したり、学習者が試した実行の入出力から進捗状況を把握する方法について提案した。(2)について、学習者がテストケースを自分で適切に設計できるようになることを目指し、学習者の作成したテストケースを評価してアドバイスを行うシステムを提案した。

研究成果の概要（英文）：In programming exercises at universities, we propose a way for understanding students' coding processes by extracting control structures and expressions from their source codes and input and output data on executing their programs. We also propose a test case evaluation system which evaluates test cases written by students and gives advices to them in order to make effective test cases by themselves.

研究分野：　ソフトウェア工学

キーワード：　プログラミング教育　WebIDE　ソフトウェアテスト

１．研究開始当初の背景

情報系の大学においてプログラミング教育は重要であり、講義に加え、演習形式で学生が PC でプログラムを作成して、プログラミングを学ぶことが多い。われわれは勤務校で 10 年以上プログラミング教育に携わり、その経験の中で次の 2 点がプログラミング演習において重要であるという考えに至った。

(1) 学習者の進捗状況の把握

演習形式では教員は教室を巡回し、学習者からの質問などに対応することで、学習効果をあげることが期待できる。しかし、学習者自身がどこで躓いているか分からず質問できなかったり、質問することに抵抗を感じたりする場合があり、必ずしも十分な指導を行えているとは限らない。また、学習者自身ではプログラミングが進んでいると感じていても、実際には誤った考え方で進めている場合や、学習意図とは異なる解答をしている場合がある。これらを解決して学習効果を上げるためには、学習者がどのようなソースプログラムを記述しているかを教員が適切に把握する必要がある。

(2) 学習者のソフトウェアテストに対する意識向上

テストは、ソフトウェア開発において、ソフトウェアが仕様を満たしているかを確認するために重要である。プログラミング演習でも同様であり、課題で作成するプログラムを説明した文章から、学習者がプログラムの仕様を考えてテストケースを設計し、実際にテストを実施しプログラムの動作確認を行わなければならない。しかし、課題にはいくつかのデータに対する実行例が掲載されている場合が一般的で、学習者はその実行例の通りに動作すれば十分であると考えてしまう場合がある。学習者にテストの重要性を意識させ、作成すべきプログラムの仕様について十分に考えてプログラミングを行い、テストケースを設計させることが必要である。

２．研究の目的

本研究では、上記の問題を解決するために Web 統合開発環境(WebIDE)を利用したプログラミング教育支援方法を提案する。学習者は WebIDE 上で、プログラムを編集・コンパイル・実行する。

(1)の進捗状況把握のために、WebIDE が自動的に作成途中のプログラムや実行履歴を取得・保存して定性的に分析し、学習者の進捗状況を教員に提示する。これにより、教員は教室を巡回することなく、学習者のプログラミング状況を把握することが可能になる。

(2)のソフトウェアテストの意識向上のために、テストケース評価システムを提案する。教員は演習開始前にテストケースが満たすべき基準(同値クラスや境界値条件など)を考え、評価基準として記述する。学習者が WebIDE でテストケース表を作成すると、評価基準に従って評価される。評価基準を満たさない場合は、学習者にテストケースの評価結果が提示され、再度テストケースについて考えさせる。これにより、学習者に作成すべきプログラムの仕様やテストの重要性について理解させる。

３．研究の方法

(1)の進捗状況把握について、①ソースコードの編集過程から分析する方法と②実行履歴から分析する方法の 2 つのアプローチについて考察した。①では、自動保存された作成途中のソースコードを構文解析し、制御構造や条件式などを抽出して、進捗状況を把握する。②では演習課題では同値クラスなどに基づいた実行例が提示されることが多いことに着目し、学習者の実行における入出力から進捗状況を把握する。

(2)のテストケース評価システムについては、演習課題毎にテストすべき値や入力データ数などが異なり、学習者によってテストケースに異なるデータを用いることもあるので、それらの差異を吸収して評価基準が記述できなければならない。評価基準そのものにフォールトが含まれることを避けるためには、極力、簡潔に記述できることが望ましい。本研究では、参考書の演習問題を分析し、入力データのデータ構造を定義して評価基準を記述する方法を提案する。

４．研究成果

本研究で作成した WebIDE の実行画面の例を図１に示す。上部がソースコード編集部で、プログラムを編集すると、1 分毎に自動で保存される。下部がテストケース評価表である。左の欄にプログラムに対する入力、中央の欄に期待される出力をテストケースとして記述する。テストケースは「入力の追加」ボタンにより追加、「削除」ボタンにより削除できる。「採点」ボタンにより、テストケースが評価される。「実行」ボタンを押すと、WebIDE がプログラムのコンパイル、テストドライバの生成とテストの実行を行い、その出力が右の欄に表示される。

図１ WebIDE の画面例

(1) 学習者の進捗状況の把握

①ソースコードの編集過程に基づいた分析

学習者の編集中のソースコードを制御構造などを元に同値類分割することで、学習者全体の進捗状況を把握する方法を提案した。ソースコードの細かい差異を吸収するために、抽象化した制御構造・条件式を用いて同値関係を定義した。各同値類の制御構造からプログラムの正誤が把握でき、同値類に分類された学習者の人数から全体の進捗状況が把握できる。

制御構造を用いた分析

学習者全体の進捗状況を把握するには、処理の大まかな流れを表すソースコードの制御構造の傾向を知ることが有効である。本研究では、ソースコードから制御文や再帰関数に関する部分を抽出したものを略式制御構造と呼ぶ。実際に抽出するのは、制御文の予約語である if、else、for、while とその制御範囲を表す{}、および再帰関数名である。略式制御構造を用いて、学習者のソースコードを同値類分割する。

条件式を用いた分析

制御構造を用いた分析で大まかな進捗状況を把握し、さらに条件式を用いて細かい分析を行う。本研究では制御文の条件式の演算子、定数、角括弧(配列)、再帰関数呼出しの実引数などを抽出したものを略式条件式と呼ぶ。抽出する基準として変数名も考えられるが、変数名は学習者によって異なる可能性があるので、抽出しない。

図 2 は a の n 乗を求める課題の編集途中のソースコードとそれに対する略式制御構造と略式条件式を示したものである。ソースコードは波閉じ括弧などが不足しているが、それらを補完したのちに構文解析する。略式条件式の丸括弧に囲まれた 3 つの式は、それぞれ if, for, for の丸括弧の条件式などに相当する。略式制御構造と略式条件式を見ると、0 を基準に場合分けして、それぞれの繰返しがおおよそ記述されていることがわかる。

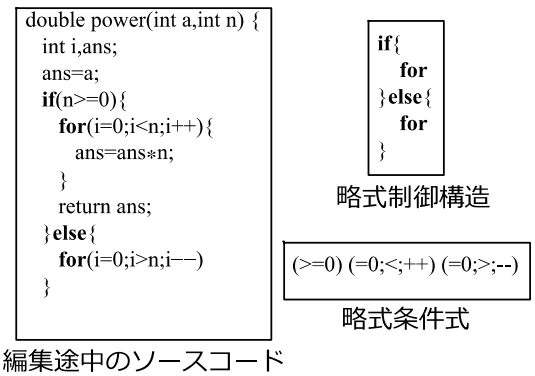

図 2 略式制御構造と略式条件式の例

実験と検証

学部 3・4 年生の学習者 27 人に対して行ったプログラミング演習において、一定時間ごとに取得したソースコードから同値類分割を行い、教員が略式制御構造と略式条件式から元のソースコードを推測して、正誤判断を行うことができるかを検証した。図 3 は同値類分割の例である。左に略式制御構造を、右に略式条件式を記述している。数字はその記述をした人数であり、グレーは正解と推測されたものである。

if{
for
}
else{
for
}
	(>0) (;>0;--) (;<0;--)	1

	(>0) (=0;<;++) (=0;;)	1
	(>0) (=0;<;++) (=0;<;++)	1
	(>0) (=1;<=;++) (=(−)1;>=;--)	1
	(>=0) (=0;<;++) (=0;>;--)	4
9	(>=0) (=;>0;--) (=*(−)1;>0;--)	1

図 3 演習における同値類分割の例

実際のソースコードからも正誤判定を行い、正解と不正解の適合率と再現率について検証した。正解の適合率は正解と推測したものの中で実際に正解だったものの割合、正解の再現率は実際に正解だったものの中で正解と推測できたものの割合である。略式制御構造などから正しく正誤判断を行うことができれば、学習者への指導へと繋げることができる。演習開始から一定時間ごとの正誤判断の適合率と再現率を下表に示す。

	正解の適合率	不正解の適合率	正解の再現率	不正解の再現率
5 分後	-	1.00	-	0.96
10 分後	0.58	1.00	1.00	0.75
15 分後	0.90	0.93	0.90	0.93

10 分後の正解の適合率と不正解の再現率が低いが、それ以外は 0.9 以上で比較的高い。10 分後のソースコードを確認したところ、編集途中または同値関係以外の部分で不正解ではあるが、制御構造と条件式は正しく記述できている学生が多かった。

学生のプログラムが間違っているかどうかを略式制御構造と略式条件式から判断することが可能であり、躓いている原因などのソースコードの内容を把握できると言える。しかし、条件式に対しての正誤判断を行うのには時間を要した。模範解答を用意して、学習者のソースコードと比較して、正誤を自動判定するといった対処が必要になると考えられる。

②実行履歴に基づいた分析

プログラミング演習において、問題に掲示された実行例を学生が試すというプロセスに着目し、学生の進捗状況を把握する方法を提案した。過去の進捗状況把握方法は①のようにソースコードを利用するものやコンパイル状況などを利用するものがほとんどで、実行履歴に着目した研究はなかった。

学習者に提示される実行例は同値クラスや境界値に基づいていることが多い。様々な

ソースコードがある中で、学習者の実行例の結果が正しければ同値クラスの処理が正しく、間違っていれば処理が間違っているということが分かる。つまり、教員は学生のソースコードを見なくても、おおよその内容が把握できると考えられる。また、実行例が 2 つ提示されている場合、学習者は図 4 の状態遷移図にしたがってプログラミングを進めていくと推測され、その状態から進捗状況が判断できる。

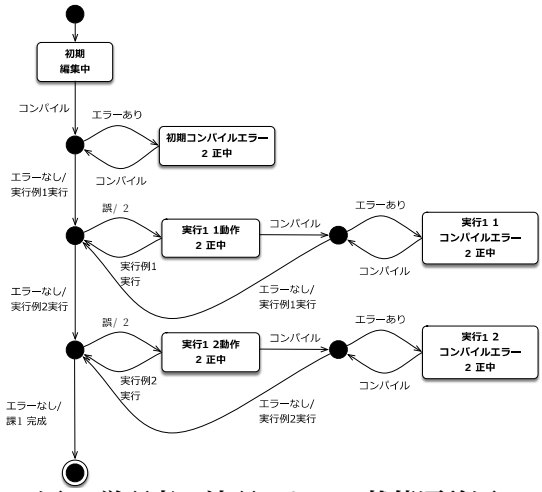

図 4 学習者の演習における状態遷移図

実験と検証

WebIDE を用いて学部 3 年生の学習者 11 人に対してプログラミング演習を行い、コンパイル状況や実行時の入出力を取得し、進捗状況が把握出るか検証を行った。

実験では問題で提示された実行例以外の数値を試す学習者も見られたが、それらはすべて実行例と同じ同値クラスの数値であった。すべての学習者が実行例の順番通りに試し、動作が不適なら修正、再び実行といった形で進めていたので、概ね図 4 の状態遷移図の通りに演習を進めていたと言える。

実数 x の n 乗を計算する問題では、実行例 1(2^3)、実行例 2(2^0)、実行例 3(2^3)を提示したが、実行例 3 が不適の学習者が 4 人いた。この場合、負の整数乗の計算ができていない可能性が高いので、教員側からは負の整数乗の計算の部分の指導を全体に行うことが望ましいと考えられる。実際にソースコードを確認すると、3 人は想定通りの間違いであった。1 人は計算部分ではなく変数の型に関する間違いであった。この間違いは実行例からの判断は難しく、①で示したようなソースコードの分析方法と併用するとよいと考えられる。

今後の展望

①と②の方法を相互補完的に組み合わせることで、進捗状況の概略を容易に把握でき、必要に応じて詳細な進捗状況を分析できるようになると考えている。教員向けに進捗状況を可視化するシステムを構築し、実際の演習でその評価を行う予定である。模範解答との自動比較についても検討していく。

(2)テストケース評価システム

WebIDE を用いて、学習者のテストケースを取得・評価し、学習者へのフィードバックを行うシステムを提案した。テストケースを一括で実行することも可能なので、学習者がテストを行う度に同じ入力を与えるなどの手間を減らすことができる。

テストケース評価機能

本研究では、学習者に十分なテストケースを作成させる方法として、教員が課題毎にあらかじめテストケースの評価基準を記述し、学習者が設計したテストケースが評価基準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する。評価基準をパスしない場合には、学習者にアドバイスを表示して、不足しているテストケースを把握させることでテストについて実地で学ばせる。評価基準やアドバイスは機械的に決めるのではなく、学習者の習熟度レベルや課題の趣旨に応じて教員が記述することとした。これにより、エラーの入力を考えるかどうかなど、演習を柔軟に行える。

本研究で試作した WebIDE は PHP を用いて実現している。PHP では正規表現などのテキスト処理のライブラリもあるので、テストケースの評価も PHP で行うこととし、評価基準を PHP の条件式を用いて、次のようにタブ区切り形式でテキストとして簡潔に記述できるようにした(以降､タブを TAB と記述する)。

評価基準番号 TAB 判定条件 TAB アドバイス

「判定条件」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「アドバイス」が表示される。同値クラスを評価するための基準が複数ある場合などは、同じ「評価基準番号」を指定することで、評価基準をグループ化できるようにした。

例えば、年齢(整数値)を入力して、20 歳以上は成年、20 歳未満は未成年と表示をする課題の評価基準は、入力データを変数 age として、成年(評価基準番号 1)、未成年(評価基準番号 2)、エラー(評価基準番号 3)の場合について、次のように記述できる。テストケースが 19 と 20 の場合は、評価基準番号 1 を満たすが、評価基準番号 3 と 2 の 0 歳の場合を満たさないので、アドバイスが表示される。

```
1 TAB age==20 TAB 成年の最低年齢
1 TAB age>=20 TAB 成年の場合
2 TAB age==19 TAB 未成年の最高年齢
2 TAB age==0 TAB 0 歳
2 TAB 0<=age && age<20 TAB 未成年の場合
3 TAB age<0 TAB エラー(年齢がマイナス)
```

判定条件の記述において、課題毎に入力データの型や個数が異なることが問題となる。本研究では、課題毎にテストケースの入力デ

ータの構造を定義し、そこで定義された変数を用いて判定条件を記述する。
　データ構造は型と変数と出現回数の組で定義する。本研究では、C 言語の参考書の例題などを参考に、型を整数、実数、文字列の 3 つに分類し、整数と実数については、さらに 0 と正負の範囲で 5 種類に分けた。数値の入力については 0 を境界として範囲が指定できる課題が多く存在したので、このように設計した。
　プログラムの入力データの構造について、次のように分類した。

・単独のデータ。1 つの変数に対応する。
・複数のデータから構成されるデータ。
　・単一の型から構成される複数データ
　　配列の課題などにおける入力データ
　・異なる型から構成される複数データ
　　構造体の課題などにおける入力データ

　複数のデータを入力する場合、入力データの個数が実行前に決まる一定個数の課題と、実行時に決まる不定個数の課題がある。不定個数の課題を次の 2 種類に分類した。

・最初に個数を入力
　最初に個数を入力し、その個数分だけデータを入力する。
・ある条件まで入力
　EOF、あるいは、負の数が入力されるまでのようにある条件になるまでデータを入力する。

　以上のことを踏まえ、入力データ構造の記述方法を設計した。データの出現回数を定義する方法は、正規表現に類する形式とし、+ は 1 回以上、 *は 0 回以上とする。出現回数が記述されていなければ、1 回だけの入力とみなす。それ以前に入力された整数を出現回数に用いることもできる。
　例えば、データ数を入力した後、その個数分、名前(文字列)、年齢(0 以上の整数)、身長(0 以上の実数)を繰り返し入力する課題のデータ構造は、次のように記述できる。ここで定義した変数に対する判定条件をテストケースの評価基準として記述する。

(uint n)
(string name, uint age, udouble height){n}

　複数の入力データに対する処理を記述したり、頻繁に用いられる処理を記述したりするために関数を定義する方法を設計した。本研究では、関数を「リストを引数として受け取り、値を返す処理」として汎用的に定義し、関数呼出し時の実引数で対象となるリストを区別する方法を採用した。実引数において、変数を [] で囲んで記述した場合は 1 レコードによる変数のリストとし、変数名だけを記述した場合は全レコードによる変数のリストとする。関数の本体は PHP で記述する。

評価
　提案システムにより、学生がテストケースの作成ができるようになるかを確認するために、プログラミングを学習済みの学部 3 年生 7 人に、ボウリングの点数を計算する課題のテストケースを作成してもらった。教員は最初に問題について説明をした後は学生には指導は行わずに、自習形式で行った。
　1 回のテストケース作成でカバレッジ 100%が 2 名、50%が 2 名、25%が 3 名であった。提案システムのアドバイスを元にテストケースを作成し直して最終的に 6 名の学生がカバレッジ 100%を達成した。最初 100%ではなかった学生が 100%になるまでのテストケース作成回数は、2 回、4 回、10 回、59 回であった。提案システムによるテストケース作成の効果はある程度認められるが、作成回数の多い学生には必要な値などを明示したより直接的なアドバイスをフィードバックしたり、教員が直接指導したりといった方法が必要になると考えられる。

今後の展望
　多くの演習課題を通して提案システムの評価や改良を行うこと、効果的なアドバイス記述などのテストケース作成演習におけるノウハウの獲得などを行なっていく。

(3)その他のプログラミング学習支援
①プログラミング学習用プルーフリーダ
　プログラミング学習において「きれいに書く修行」を支援するために、学習者が作成したプログラムが教育者の意図に合致しているかをチェックするプルーフリーダを試作した。プログラミング学習では、学習者は課題の実行例などを参考にプログラムの動作確認をし、実行例と同じ出力が得られたら課題を終了していることが多い。しかし、学習者が作成したプログラムは簡潔性、明瞭性、一般性に欠ける場合があり、必ずしも教育者が意図した「きれいに書かれたプログラム」とは限らない。「きれい」の判断基準は課題毎に異なるが、これを学習項目として整理し、教育者が作成した模範解答プログラムと学習者のプログラムを比較して、教育意図に合っていない「汚く書かれたプログラム」を発見するプルーフリーダを試作した。
②誤り修正課題作成システム
　プログラムの誤り修正課題とその正誤判定を行うプログラムを自動生成するツールを提案した。誤り修正課題は意図的に誤りを混入させたプログラムを学習者に提示し、その誤りを正しく修正させる課題であり、デバッギングやコードリーディングの能力向上に有効である。実際の演習を行い、誤り修正課題演習の評価を行った。作成した誤り修正課題は Web で公開している
(http://ecq.tebasaki.jp)。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計９件）
[1] 蜂巣 吉成，小林 悟，吉田 敦，阿草 清滋.「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」コンピュータソフトウェア，6 ページ，採録決定，査読あり

[2] 蜂巣 吉成，吉田 敦，阿草 清滋.「プログラムの誤り修正課題および正誤判定プログラムの自動生成」情報処理学会論文誌: 教育とコンピュータ，Vol. 3 No. 1，pp. 64-78，2017 年 2 月、査読あり

[3] 小林 悟，蜂巣 吉成，吉田 敦，阿草 清滋.「データ構造と関数定義に着目したプログラミング学習者用テストケース評価記述方法の提案」ソフトウェア工学の基礎 XXIII―日本ソフトウェア科学会 FOSE 2016，pp. 33-42，2016 年 12 月，査読あり

[4] 吉田 敦，蜂巣 吉成，阿草 清滋.「ウェブアプリケーション開発のための PHP オンライン部分評価器の試作」ソフトウェア工学の基礎 XXIII―日本ソフトウェア科学会 FOSE 2016，pp.217-222，2016 年 12 月，査読あり

[5] 蜂巣 吉成，吉田 敦，阿草 清滋.「命令型プログラミング言語における初学者向け動作理解支援ツールの提案」ソフトウェア工学の基礎 XXII―日本ソフトウェア科学会 FOSE 2015，pp. 97-102，2015 年 11 月，査読あり

[6] 吉田 敦，蜂巣 吉成.「前処理指令に対する制約のない前処理前コードの構文解析手法」情報処理学会論文誌，Vol. 56 No. 2，pp. 592-610，2015 年 2 月，査読あり

[7] 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 XXI―日本ソフトウェア科学会 FOSE 2014，pp. 241-250，2014 年 12 月，査読あり

[8] A. Yoshida，Y. Hachisu.“A Pattern Search Method for Unpreprocessed C Programs Based on Tokenized Syntax Trees,”Proc. of 14th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM2014)，pp. 295-304，2014 年 9 月，査読あり

[9] Y. Hachisu，A. Yoshida.“A Support System for Error Correction Questions in Programming Education,”Proc. of International Conference on e-Learning 2014 (EL2014)，International Association for Development of the Information Society (IADIS)，pp. 249-258，2014 年 7 月，査読あり

〔学会発表〕（計４件）
[1] 加藤 宗一郎，吉田 敦，蜂巣 吉成，桑原 寛明，阿草 清滋.「ファイルへの入出力に基づく実行履歴の構造化手法の提案」情報処理学会 ソフトウェア工学研究会 2017-SE-195，8 ページ，2017 年 3 月，早稲田大学（東京都新宿区）

[2] 蜂巣 吉成，吉田 敦，阿草 清滋.「学習項目を利用したプログラミング学習用プルーフリーダの試作」日本ソフトウェア科学会第 33 回大会，6 ページ，2016 年 9 月，東北大学（宮城県仙台市）

[3] 加藤 大典，蜂巣 吉成，吉田 敦，阿草 清滋.「変数に着目した部分プログラム抽出によるコードレビュー手法の提案」電子情報通信学会信学ソフトウェアサイエンス研究会 SS2014-73，Vol. 114，No. 510，6 ページ，2015 年 3 月，沖縄県青年会館(沖縄県那覇市)

[4] 蜂巣 吉成，吉田 敦，阿草 清滋.「プログラミング演習におけるコーディング状況把握方法の考察」情報処理学会研究報告コンピュータと教育研究会 2014-CE-125，8 ページ，2014 年 6 月，公立はこだて未来大学（北海道函館市）

〔図書〕（計１件）
[1] Y. Hachisu，A. Yoshida.“A Web based System for Error Correction Questions in Programming Exercise (chapter 6),”Artificial Intelligence Technologies and the Evolution of Web 3.0，IGI Global，pp. 124-143，2015 年 2 月

〔その他〕
ホームページ等
Error Correction Questions
http://ecq.tebasaki.jp
C 言語の誤り修正課題を公開している

６．研究組織
(1)研究代表者
蜂巣 吉成（HACHISU YOSHINARI）
南山大学・理工学部・教授
研究者番号：30319298

(2)研究分担者
吉田 敦（YOSHIDA ATSUSHI）
南山大学・国際教養学部・教授
研究者番号：50283495