プログラミング演習における進捗状況把握方法の提案

2010SE012 浅井祐太　2010SE065 石川航　2010SE117 松井暁生
指導教員：蜂巣吉成

1　はじめに

現在，情報系の大学などでは，プログラミング教育が行われており，学習者は演習形式で，実際にコーディングを学ぶという方法が広くとられている．学習効果をあげるために，教員が学習者の進捗状況を把握し，適切な指導を行うことは重要である．本研究では，進捗状況とはソースコードの記述内容の変化過程とする．記述内容が模範解答に近づけば，出題者の意図に沿って進捗が進んでいると言える．記述内容が変化しても模範解答に近づかないときは，学習が誤った記述をしているか，別解を記述している場合である．

教員が教室を巡回することで，学習者の進捗状況をある程度把握することが可能であるが，講義時間には限りがあり，各学習者に割ける指導時間は限られたものとなる．従来の方法では，学習者の状況を逐次確認するとともに，それを多人数を相手に行う必要があるといった点から，進捗状況を十分に把握できていない場合が存在する．教員が学習者の進捗状況を容易に把握することができれば，学習者毎に問題箇所への指摘ができる．さらに，全体として問題箇所の傾向を把握し，学習者全体に適切なフィードバックを返すことで，学習効果を上げることができる．

学習効果を上げることを目的として，教育支援システムがいくつか提案されている．[1]〜[4] ではソースコード編集履歴やコンパイル・実行結果，エラー履歴などを基に分析を行っている．しかし，これらの方法では，学習者がどのようにソースコードを記述しているのかというコーディング過程に関する情報の取得・分析を十分に行っていないので，コーディング内容まで把握するのは困難である．

我々の研究室では，これらの問題を解決するために Java 言語を対象として学習者の記述途中のソースコードを取得して，字句，文，文同士の構造で模範解答と比較・分析することで，進捗状況を把握する方法について考察している [5]．特定の演習課題についての分析結果を述べているが，その分析方法の一般性や，分析結果の教員への提示方法については提案されていない．

そこで本研究では，プログラミング演習において学習者の進捗状況を把握する方法を提案する．進捗状況を把握するためのプラットフォームとして Web ベースの統合開発環境（Web Integrated Development Environment：以下 WebIDE）を利用することで，学習者の情報を自動的に取得・分析・通知を行う．

本研究における課題を挙げる．

- ソースコードの分析方法
- 別解の判断方法
- 教員への通知方法

進捗状況を把握するための方法として，学習者のソースコードと模範解答との比較および学習者同士での比較を提案する．ソースコードを字句数や演算子数，制御文数，制御式数，制御文の構造といった観点から比較や分析を行うことで，学習者間での記述方法の違いや，意図しない記述などにも対応することができる．昨年度の我々の研究室の研究では，JavaServlet を用いて WebIDE を実現し，コンパイルには Java CompilerAPI を利用していた．この方法では Java 言語以外への対応が困難なので，本研究の WebIDE ではコンパイル・実行機能については，[4] でも利用されている Web サービスの Ideone[6] を用いる．

2　関連研究

プログラミング教育において，学習者の学習状況を把握するシステムはこれまでも研究されている．

[2] では，ソースコードを逐次取得し，熟練者がコンパイルエラー情報と照らし合わせることで正しい記述をしているか分析している．また，ソースコードのトークンに基づく編集距離にも着目し，その値の傾向の分析も行っている．

[4] では，プログラミング講義を対象にコーディング遅延や停滞を可視化するシステム（C3PV）を開発し，学習状況の把握を行っている．相対的に状況の悪い学習者を提示することで，限られた数の教員や TA でも効率よく指導を行うことを可能にしている．

これらの研究は，いずれも学習者の状況を把握をすることができるが，ソースコードの記述内容にまで踏み込んだ分析を行っていない．そのために，どのような記述をしているのかという具体的な内容は把握することができない．

3　Web 開発環境による情報取得

プログラミング演習において，教員は学習者に課題を提示した後，進捗状況を把握するために巡回を行うことが多い．試行錯誤していたり，質問する学習者がいた場合に，教員はその度にソースコードを確認する必要がある．個人の指導に時間を多く割いてしまう場合があり，限られた演習時間で全体の進捗状況を把握することは難しい．

3.1　Web の利用

学習者全体の状況を分析・把握しようとした場合，ソースコードをはじめとした学習者の情報を一箇所に集める必要がある．そこで本研究では Web を利用することでソースコード等の学習者の情報を一箇所に集約する．学習者の PC と同ネットワーク上のサーバに情報を集約することで，全体や個人の進捗状況をリアルタイムに把握することができる．ソースプログラムの保存や履歴管理にはソフトウェ

アリポジトリを用いる方法もあるが，一般にリポジトリで管理されるのは，一定の経験を積んだプログラマが作成したテスト済みのプログラムであり，プログラミング学習者の記述途中のプログラムの管理には向いていない．

Eclipse 等の IDE で記述途中のソースコードを保存する方法がある．この方法は，学習者の編集途中のプログラム取得に適していると言えるが，学習者の PC に Eclipse やプラグインのインストールやバージョンアップなどが必要となり，環境構築に手間や時間がかかる．本研究では，Web の開発環境を実現することでソースコードを取得し，環境構築よる手間や時間の軽減を図る．

3.2 Web ベースの統合開発環境

本研究では，学習者の進捗状況を把握するためのプラットフォームとして，WebIDE を提案する．プログラミングを学ぶ上で必要な機能（以下基本機能）に加え，ソースコードの取得や分析を行うことで，進捗状況を把握する．また，WebAPI を利用したコンパイル・実行を行うことで，多言語へも対応することができる．WebIDE を実現するにあたり，プログラミングに必要最低限な機能（以下基本機能）と，進捗状況を把握するために必要な機能（以下進捗状況把握機能）を次に示す．

基本機能

- アカウント管理機能
  学習者が WebIDE 上でログインする際，アカウント管理用データベースと通信しログイン認証を行う．
- ファイル管理機能
  各学習者が作成したソースファイルをサーバに学習者ごとに保存する．ファイルに対して行うことができる操作としては，選択・保存があり，いずれも WebIDE 上で行うことができる．
- コンパイル・実行機能
  学習者が記述したソースファイル・標準入力をコンパイル・実行し，WebIDE 上に実行結果の表示を行う．本研究では IdeoneAPI を利用してコンパイル・実行の処理を行う．

進捗状況把握機能

- 学習者のソースコードなどを逐次取得・分析し，進捗状況を教員へ通知する．

非機能要求としては，多くのプログラミング言語に対応していることや，操作性や処理速度などにおいて，ローカル環境と比べても学習に支障がないシステムであることが挙げられる．今回実現する WebIDE は，コンパイル・実行機能について IdeoneAPI を利用するが，インターフェースを対応させることで他の WebAPI も利用することができる．また，進捗状況把握機能以外にも機能拡張が容易となるように，各データを管理しやすいフォーマットで扱うような設計とした．

3.3 IdeoneAPI の利用

ソースコードのコンパイル・実行には，Ideone.com[6] の WebAPI を利用する．Ideone は 40 種以上のプログラミング言語に対応したオンラインコンパイラであり，ソースコードを送信することで，コンパイルおよび実行結果を受け取ることができるサービスである．数十行程度のプログラムであれば平均 4〜5 秒でコンパイル・実行とその結果を WebIDE に表示することができ，学習をするにあたって支障のない速度で処理が行える．また，無限ループするプログラムに対しては実行を強制終了させる等の処理も考慮されている．ただし，本研究の WebIDE では，ファイルの入出力やネットワークで通信を行うプログラム，system 関数やシステムコールを利用するプログラム，新規にウィンドウを作成しグラフィックスを表示させるプログラムの実行には対応していない．

4 進捗状況把握方法

2 節で述べたように既存の研究では，ソースコードの字句や式，文，文の構造までを分析対象とはしていない．また，コード行数用のメトリクスを利用しても，模範解答との字句や構造としての比較などを行うことは難しい．進捗状況を把握するためには，ソースコードの字句や構造などに踏み込んだ分析を行い，時間の変化と共に，学習者が記述している具体的な内容を把握することが必要である．

本研究では，プログラミング演習において学習者がソースコードをどのように記述しているかを把握し，その結果を用いて学習者全体でどのような記述があるのかを分析することで，進捗状況を把握する．しかし，記述内容の分析においては，同じ処理のプログラムであっても学習者毎に記述方法は異なる場合が多く存在する．そこで本研究では，学習者による記述方法の違いを吸収するために，4 つの分析方法を用いる．

字句による分析では，模範解答に出現する字句と，模範解答には含まれない字句をそれぞれ抽出することができる．模範解答に沿った記述をしている，または模範解答とは異なる字句を用いて学習を進めており，別解を作成しているなどの学習者の状況を把握できる．

式による分析では，式の出現回数と記述人数を調べ，学習者が字句をどのような組み合わせで記述しているかを把握することができる．

文による分析では，文の出現回数と記述人数を調べることにより，字句と式の組み合わせがどのようになっているかを分析する．これにより，制御文や条件式をどのように記述しているのかを把握することができる．

文構造による分析では，文同士の連接関係と親子関係を分析することで，学習者が模範解答と同じ構造で解答を記述しているか，別の構造で記述しているかを把握することができる．

5 実験と検証

5.1 実験方法

本研究で提案した WebIDE を学生の学習者 25 名に利用してもらい，1 分ごとに学習者のソースコードを取得した．学習者には次の関数を作成する 4 問を出題した．

- 問題 1：文字列をコピーする関数
- 問題 2：階乗を計算する関数
- 問題 3：文字列の長さを返す再帰関数
- 問題 4：文字列から指定された文字を削除する関数

字句，式，文，文の構造の各分析において，単純に登場回数や使用人数を分析することは可能であるが，字句の登場回数や使用人数の一覧を見るだけで進捗状況をすぐに把握することは困難である．それぞれの分析方法において，解答を作成する上で重要である箇所の登場回数や使用人数のみを教員に通知することで，進捗状況の把握を容易にする．

実験で得られた問題 1〜3 の学習者のソースコードより，以下の箇所を分析することで学習者の進捗状況を把握できると仮定した．

- 演算子
- 制御文の条件式
- 制御文と条件式の関係
- 制御文の構造

これらを問題 4 に当てはめて分析し，解答例と比較して進捗状況を把握することで，重要な箇所の妥当性を確認する．問題 4 の解答例をソースコード 1 に示す．

ソースコード 1 問題 4 の解答例

```
1  void strdelete(char *str, char c)
2  {
3      char *t;
4      for (t = str; *str != '\0'; str++) {
5          if (*str != c) {
6              *t = *str;
7              t++;
8          }
9      }
10     *t = '\0';
11 }
```

5.2 字句による分析

字句による分析では，演算子を分析する．

表 1 演算子の集計結果

10 分時点			20 分時点			30 分時点		
字句	人	回数	字句	人数	回数	字句	人数	回数
=	19	72	=	19	90	=	18	81
++	19	37	!=	18	28	!=	18	30
!=	18	23	++	18	47	++	18	50
==	15	16	==	17	21	==	15	16
+	1	3	−	3	4	−	2	6
&&	1	1	+	2	3	<	1	2

表 1 より，この問題では，解答例で登場している演算子を学習者が多く記述している．=と ++ 演算子は人数と回数の両方が多いが，!=演算子は人数に対しての回数が少ない．解答例にない==演算子を多くの学習者が使用していることから，!=ではなく==を比較に用いてることが把握できる．また，10 分時点と 20 分時点で記述のある &&や + 等の解答例に記述のない演算子が，30 分時点では減少していることから，学習者の記述が解答例に近づいていることが把握できる．

5.3 式による分析

式による分析では，制御文の条件式を分析する．

表 2 条件式の集計結果

10 分時点		20 分時点		30 分時点	
*str!='\0'	15	*str!='\0'	13	*str!='\0'	16
*str==c	11	*str==c	12	*str==c	8
!='\0'	3	!='\0'	6	*str!=c	7
*str!=c	2	*str!=c	6	!='\0'	6
==c	2	==c	3	==c	4

表 2 より，解答例にある条件式*str!='\0' は，学習者でも多くの人数が記述しており，解答例に近い記述をしているといえる．*str!=c に関しては使用している人数は時間ごとに徐々に増加しているが，記述している学習者は多くはない．解答例には使われていない条件式*str==c が 10 分時点では多く記述されているが，時間ごとに徐々に人数が減少している．学習者は*str!=c の代わりに*str==c を記述しているが，時間経過毎に徐々に*str!=c に書き換えていることが把握できる．

5.4 文による分析

文による分析では，制御文と条件式の関係を分析する．

表 3 制御文と条件式の組み合わせの集計結果

10 分時点			20 分時点			30 分時点		
制御文	条件式	人数	制御文	条件式	人数	制御文	条件式	人数
if	*str==c	11	if	*str==c	11	for	*str!='\0'	8
for	*str!='\0'	9	while	*str!='\0'	7	if	*str==c	8
while	*str!='\0'	6	for	*str!='\0'	6	if	*str!=c	7
for	!='\0'	3	for	!='\0'	6	while	*str!='\0'	7
if	*str!=c	2	if	*str!=c	6	if	==c	3

表 3 より，解答例にある制御文 for と*str!='\0' については，10 分時点から 20 分時点で減少し，30 分時点でまた増加している．ここで学習者の記述で for と!='\0' を確認すると，10 分時点から 20 分時点で増加し，30 分時点で大きく減少している．学習者は for 文の条件において，*str の代わりに自身が宣言した変数を使用して試行錯誤し，解答を記述していることが把握できる．解答例の if と*str!=c の組み合わせについては，時間経過毎に徐々に増加しており，解答例に近づいていることが把握できる．また学習者の記述で*str==c の記述が経過 10 分と 20 分時点で多く 30 分時点で減少しており，条件式を書き換えて解答例に近づいていると把握できる．

5.5 文の構造による分析

文の構造による分析では，制御文同士の入れ子構造の組み合わせについて分析する．

表 4 制御文の構造の集計結果

10 分時点		
組み合わせ		人数
for(*str!=’\0’)	if(*str==c)	6
while(*str!=’\0’)	if(*str==c)	3
for(!=’\0’)	if(==c)	2
while(*str!=NULL)	if(str==c)	1
for(*str!=’\0’)	if(*str!=c)	1
20 分時点		
組み合わせ		人数
for(*str!=’\0’)	if(*str==c)	6
while(*str!=’\0’)	if(*str==c)	4
for(*str!=’\0’)	if(*str!=c)	3
for(!=’\0’)	if(==c)	2
while(*str!=’\0’)	if(*str!=c)	1
30 分時点		
組み合わせ		人数
for(*str!=’\0’)	if(*str==c)	6
while(*str!=’\0’)	if(*str==c)	2
while(*str!=’\0’)	if(*str!=c)	2
for(*str!=’\0’)	if(*str!=c)	2
while(*str!=’\0’)	if(str==’c’)	1

表 4 より，解答例の for(*str!=’\0’) と if(+str!=c) の組み合わせは，一部の学習者のみが記述している．全体を通して for(*str!=’\0’) と if(*str==c) の組み合わせが多く，多くの学習者はこの組み合わせを記述していることが把握できる．また，時間経過毎に制御文の組み合わせの種類が増加している．これは学習者が解答に悩み、独自の様々な意図の変数を作成したことで，組み合わせが学習者毎に異なったからと考えられ，悩んでいる学習者の状態を把握できる．

5.6 教員への提示方法

重要な字句や式等のみについて分析することで学習者の進捗状況を把握することが出来た．しかし字句や式を記述している人数を示すだけでは，一目で進捗状況を把握することは難しい．よって記述人数や回数をグラフで表す．本分析の文の構造による分析の 10 分，20 分時点のグラフを図 1 に示す．色の濃いグラフが解答例にある構造であり，10 分から 20 分にかけて解答例にある構造が増加していることにより，解答例に近づいていることが把握できる．

6 おわりに

本研究では，学習者の進捗状況の把握を実現するために，取得したソースコードを字句や式などの単位から分析を行う進捗状況把握方法を提案した．WebIDE を用いて学習者からソースコードをリアルタイムに取得し，提案した方法の妥当性を検証した．検証の結果，誤った記述や別解を検出することができ，指導すべき内容や指導自体が必要な状況かどうかを判断することが可能となった．今後の課題として，提案した方法や通知方法を WebIDE 上に実現する必要がある．また，コンパイルエラーや実行結果からも分析を行うことで，ソースコードのみでは把握できない状況に対応させることも今後の課題である．

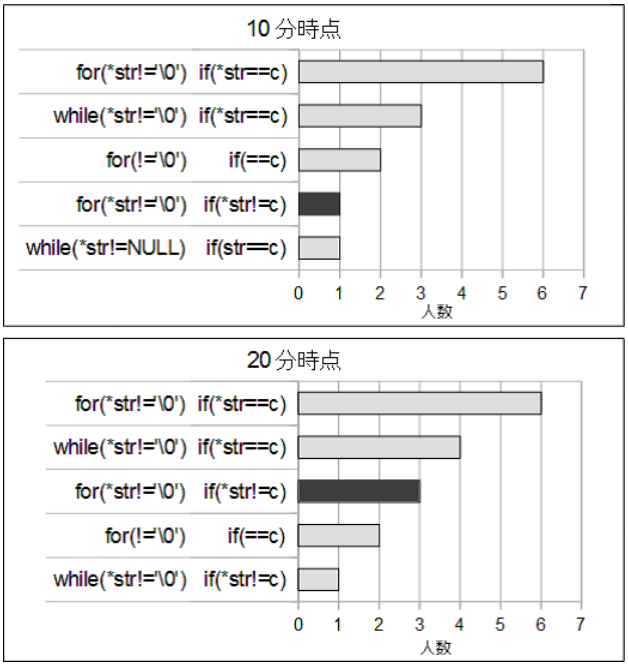

図 1 10 分 20 分時点 文の構造のグラフ

参考文献

[1] 長谷川伸，松田承一，高野辰之，宮川治，“プログラミング入門教育を対象としたリアルタイム授業支援システム”，情報処理学会論文誌，Vol.52，No.12，pp.3135-3149，2011.

[2] 伏田享平，玉田春昭，井垣宏，藤原健二，吉田則裕，“プログラミング演習における初学者を対象としたコーディング傾向の分析”，電子情報通信学会技術研究報告，SS，ソフトウェアサイエンス，Vol.111，No.481，pp.67-72，2012.

[3] 倉澤邦美，鈴木恵介，飯島正也，横山節雄，宮寺庸造，“プログラミング演習における一斉指導のための学習状況把握支援システムの開発”，電子情報通信学会技術研究報告，ET，教育工学，Vol.104，No.703，pp.19-24,，2005.

[4] 井垣宏，齋藤俊，井上亮文，中村亮太，楠本真二，“プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案”，情報処理学会論文誌，Vol.54，No.1，pp.330-339，2013.

[5] 鈴木那由多，高山直，寺尾勇紀，“プログラムの抽象度を用いたコーディング過程把握手法の提案”，南山大学情報理工学部・数理情報学部 2012 年度卒業論文，2013.

[6] Sphere Research Labs，Ideone.com，available from <http://ideone.com/>.