プログラム状況認識行動支援システム

河田　進*　　宮武　明義*　　矢野　米雄**

The Support System for the Action to Recognize Program Conditions

Susumu KAWATA*　Akiyoshi MIYATAKE*　Yoneo YANO**

Synopsis

It is important that the learners detect and correct the errors by themselves to learn a programming technology. However, it is difficult for them to detect and correct the errors of a program. Therefore, we developed the support system for them. The system shows them the correct executing result and some error executing results with an explanation of error condition. Moreover, the system shows them the candidacies of error cause. By using the system, the learners can recognize the error condition of the executing result, and they can examine the error cause of the program.

１．はじめに

初級プログラマ（以下，学習者と呼ぶ）がプログラミング能力を習得するためには，各種データ構造やアルゴリズムの理論的な学習とともに，実際にプログラミングを行う実践経験が重要である。その際，学習者は例題プログラムを参考に，課題問題に対して以下のような活動を行う。

(1) 例題プログラムの目的，データ構造，アルゴリズムを理解する。
(2) 課題の目的を理解し，例題との違いを検討する。
(3) 課題で処理するデータの構造とアルゴリズムを決定する。
(4) 課題プログラムを記述する。
(5) プログラムをコンパイルする。
(6) 文法エラーがある場合，エラーメッセージを手がかりに例題プログラムと課題プログラムを比較しながら誤りを発見・修正する。
(7) プログラムを実行し，予想結果と実行結果を比較して正しい動作かどうかを検討する。
(8) 正しい動作でない場合，結果の違いから，学習者が決定したデータ構造やアルゴリズムに従って記述しているかどうか検討する。または，データ構造やアルゴリズムそのものを再検討する。

上記の(8)で示した，誤りを修正する作業（デバッグ）こそが，学習者がプログラミング能力を効率的に習得するために必須な活動である。しかしこの活動を行うためには，経験不足による２つの問題がある。

(1) 実行結果の違いが何を意味するのかが判断できない。
(2) プログラムの何が原因でその違いが生じたのかを類推できない。

この問題のために，学習者による誤りを修正するまでの時間は大きく，学習効率は高くない。この問題は，学習が進むにつれて徐々に解消していくが，学習効率を向上させるためには短時間で，実行結果の違いの意味とその原因を把握できるような支援が必要である。

そこで，以下の支援を行うシステム（以下，本システムという）を開発する。

(1) 課題ごとに，出現するかもしれない様々な実行結果を提示し，予想結果との違いの意味（状況と呼ぶ）を解説する。

* 情報工学科
** 徳島大学　工学部

(2) 課題ごとに，実行結果が生じる原因となりそうな候補を提示し，プログラムを検証させる。

「出現するかもしれない様々な実行結果とその状況」や「実行結果が生じる原因」は予め用意するのではなく，多くの学習者が同一の課題群のプログラミングを行う課程で出現・発見したものをシステムが認識行動支援データベース（以下支援ＤＢと略す）に蓄積し，他の学習者の学習時に利用する。また，教師が学習者を適切に指導するために，学習履歴も支援ＤＢに記録する。

本稿では，２章で本システムの概要および機能・他の研究との比較，３章においてシステムの構造，４章において実装方法やユーザインターフェースを示す。

２．システムの概要

システムが，学習者の作成したプログラムの実行結果に対してどの様な指導をするかを示す。

２．１　誤りの例と原因の候補

図１は３進数の足し算を行う正しいプログラムと実行結果である。それに対し，図２は初級プログラムが作成したプログラムの誤った結果の例と状況を説明したものである。

```
#include <stdio.h>
int main(void)
{
    unsigned short a,b,s;

    scanf("%hu%hu",&a,&b);
    if(a==2 && b==2) s=11;
    else if((a==1 && b==2) ||
            (a==2 && b==1)  s=10;
    else s=a+b;
    printf("a=%hu b=%hu a*b=%hu\n",a,b,s);
    return(0);
}
```

3進数の足し算表

a＼b	0	1	2
0	0	1	2
1	1	2	10
2	2	10	11

```
0 0
a=0 b=0 s=0
0 1
a=0 b=1 s=0
0 2
a=0 b=2 s=0
1 0
a=1 b=0 s=0
1 1
a=1 b=1 s=1
1 2
a=1 b=2 s=2
2 0
a=2 b=0 s=0
2 1
a=2 b=1 s=2
2 2
a=2 b=2 s=11
```

図１　正しいプログラムと実行結果

［例１］
a,bどちらかが2のとき11になっている
```
0 0
a=0 b=0 s=0
0 1
a=0 b=1 s=1
0 2
a=0 b=2 s=11
1 0
a=1 b=0 s=0
1 1
a=1 b=1 s=2
1 2
a=1 b=2 s=11
2 0
a=2 b=0 s=11
2 1
a=2 b=1 s=11
2 2
a=2 b=2 s=11
```

［例2］
(1,2)(2,1)の結果が3進数には無い3になっている
```
0 0
a=0 b=0 s=0
0 1
a=0 b=1 s=1
0 2
a=0 b=2 s=2
1 0
a=1 b=0 s=1
1 1
a=1 b=1 s=2
1 2
a=1 b=2 s=3
2 0
a=2 b=0 s=2
2 1
a=2 b=1 s=3
2 2
a=2 b=2 s=11
```

［例3］
aの値が全て0になっている
```
0 0
a=0 b=0 s=0
0 1
a=0 b=1 s=1
0 2
a=0 b=2 s=2
1 0
a=0 b=0 s=0
1 1
a=0 b=1 s=1
1 2
a=0 b=2 s=2
2 0
a=0 b=0 s=0
2 1
a=0 b=1 s=1
2 2
a=0 b=2 s=2
```

［例4］
実行できていない
```
0 0
セグメンテーション違反
0 1
セグメンテーション違反
0 2
セグメンテーション違反
1 0
セグメンテーション違反
1 1
セグメンテーション違反
1 2
セグメンテーション違反
2 0
セグメンテーション違反
2 1
セグメンテーション違反
2 2
セグメンテーション違反
```

図２　誤りの例と状況

表１から表４は誤りの例１から例４の原因となりうるものの候補である。

表１　誤りの例１の原因の候補

候補番号	誤り原因
1	結果が11になる判定が(2,2)だけになっていない
2	3進数足し算の表の組み合わせで判定されていないものがある

表２　誤りの例２の原因の候補

候補番号	誤り原因
1	(1,2) (2,1) を判定していない
2	(1,2) (2,1) を判定した際に結果を10としていない
3	(1,2) (2,1) を判定するif文を実行する前に別の判定で処理されている
4	(1,2) (2,1) を判定して結果を求めた後，別のif文で再判定されている
5	3進数の値(0,1,2,10,11)以外が結果として求まるよう記述してある

表３　誤りの例３の原因の候補

候補番号	誤り原因
1	aの入力がなされていない
2	aの入力後，判定前にaが代入などにより変更されている
3	aのデータ型と入力書式が適切な組み合わせになっていない

表４　誤りの例４の原因の候補

候補番号	誤り原因
1	scanfで変数の前に&が付けられていない

２．２　学習の流れ

学習者がシステムを利用した学習の流れを以下に示す。

(1) 学習者は，システムが提示する正しい実行結果と学習者プログラムの実行結果から誤りの例１から例４など状況のどれに相当するかを判断し，システムに入力する。

(2) 学習者は，システムが入力された状況によって選らんだ誤り原因の候補一覧（表１から表４などのどれか）の一つ一つについてプログラムを検査し，原因を特定する。

(3) 特定できた原因よりプログラムを修正後、再度実行結果を得て，(1) (2) を繰り返す。

２．３　教師の役割

システムを効果的に運用して学習効果を向上させるためには，教師の存在が重要である。以下では教師の役割を示す。

1)　課題実行環境の構築

プログラムを実行するためにはデータが必要である。しかし，初級プログラマがプログラムの正しさを検証できるようなデータを揃えるのは容易ではない。そこで，教師がデータファイル群とデータファイル群を自動で読み込み実行するコマンドファイルを用意する。図３はコマンドファイルの例である。

```
cc -o $1 $1.c
if test $? -eq 0
then
more $1.c >> $1.tmp
number=1
while test $number -ne 10
do
more $1.d"$number" >> $1.tmp
./$1 < $1.d"$number" >> $1.tmp
number=`expr $number + 1`
done
more $1.tmp
more $1.tmp >> $1.res
rm $1.tmp
fi
```

図３　実行コマンドファイル

2)　支援ＤＢへの情報

蓄積支援ＤＢに「出現するかもしれない様々な実行結果とその状況」や「実行結果が生じる原因」がまだ蓄積されていない状態では，学習者はシステムを利用できない。そのため，教師は学習者が作成したプログラムの結果を観察し，学習者に状況と原因の候補を説明するとともにこの 2 種類の情報を支援ＤＢへ蓄積する。また，支援ＤＢに記録されていない状況が発生した場合，学習者はシステムから提示された状況群から特定できないため教師に相談する。教師は，それが新しい状況であると判断した場合も同様の行為を行う。

3)　学習者の状況判断に対する指導

図３のコマンドファイルには，１つの課題に対する学習者プログラムのソースリストや実行結果が時間

に沿ってプログラミング記録として残るような仕掛けを施してある。また，支援ＤＢには，学習者が判断した状況や特定した誤り原因を日時とともに学習履歴として記録する。教師は，学習履歴とプログラミング履歴を観察し，学習者の判断ミスを指導することで，学習者の理解を支援する。

4) 学習履歴を利用した指導

プログラミング活動をしなければ学習記録やプログラミング記録は作成されない。教師は，この２つの記録を観察することで，学習状況が良好でない学習者を特定し指導する。

２．４　能力獲得へ期待される効果

本システムを活用することにより，学習者が以下のような能力を獲得することが期待できる。

(1) デバッグを行う習慣を身に付けることができる。
(2) システムを利用しなくてもプログラムの状況を認識したり，表現したりすることができる。
(3) システムから誤り原因の一覧を得なくても，原因を特定できるようになる。
(4) アルゴリズムやデータ構造の理解や応用できる能力の習得を効率的に行える。

２．５　他の研究との比較

ここでは，他の研究の支援方法と比較し，我々の支援方法の有効性について検証する。まず，誤り場所や誤り原因を特定することで，学習者のプログラミングを進行させるような支援を行っている方法 1)～4)では，学習者は，指摘された場所や原因に限って思考を進めるので，時間的に早く誤りを修正できる可能性がある。しかし，誤った記述と正しい記述の組合せに関する違いを理解できたとしても，その組合せ以外の場合については検討しないため，同種ではあるが異なる誤り方をした場合に誤り原因を特定できない。対して，我々は，あくまでプログラムの状況や誤り原因の候補を提示するだけであるので，学習者は自作プログラムの状況や誤り原因について複数の考察をしなければならず，より応用性の高い能力を獲得できる。例えば if 文において，条件式が逆になっているとき，「条件が逆である」という指摘を受けた学習者は，記述している条件の逆をどの様に書けば良いかは考えるため，問題の意味を表現する正しい書き方を学習できるであろう。しかしその際，境界に関する事柄を検討するとは限らない。我々の方法では，誤り原因の候補の中に条件の逆も境界に関する検討も入っているので，他の課題においても両方の可能性を検討できるようになる。

次に，振る舞いの可視化を行う支援方法[5)~7)]であるが，学習者は，観察された振る舞いがどの様な規則性のある現象を生じているかを理解する必要がある。それを理解できなければ誤り原因を特定できないので，初級プログラマにはそれを理解させるための訓練が必要であり，初期においてはガイドが必要であり有効である。よって，我々がプログラムの実行結果から状況を指摘することは有益である。

３．システム構造

図４は本システムの構造図であるが，ここではシステムを構成する支援ＤＢの構造や各サブシステムの処理内容について述べる。

３．１　支援ＤＢの構造

支援ＤＢは４種類のテーブルから構成されるが，図５に各テーブルの構造を示す。

(1) 正解テーブル

課題コードをキーとして，正しい実行結果を記録する。

(2) 状況テーブル

課題コードをキーとして，誤りのある実行結果とその状況を記録する。状況には課題別に状況コードを振る。

(3) 誤り原因テーブル

課題コード，状況コードをキーとして誤り原因を記録する。誤り原因には，課題，状況別に原因コードを振る。しかし，誤り原因の内容については同種とみなせる表現になる場合がある。例えば，図１の課題を行っているときは他にも３進数の掛け算や４進数の足し算の課題も行う。表１の候補２を「表の組み合

わせで判定されていないものがある」と表現すれば，同じ学習項目の他の課題で利用できる。また，表４は殆どの課題で利用できる。

そこで，誤り原因コードを大分類コード，中分類コード，小分類コードに分け，中分類コードが指定されていない場合は，同じ大分類コードに該当する全ての課題で利用する。さらに大分類コードが指定されていない場合は全ての課題で利用する。尚，これを実現するために，課題コードは大分類コードと中分類コードを連結したものにする。

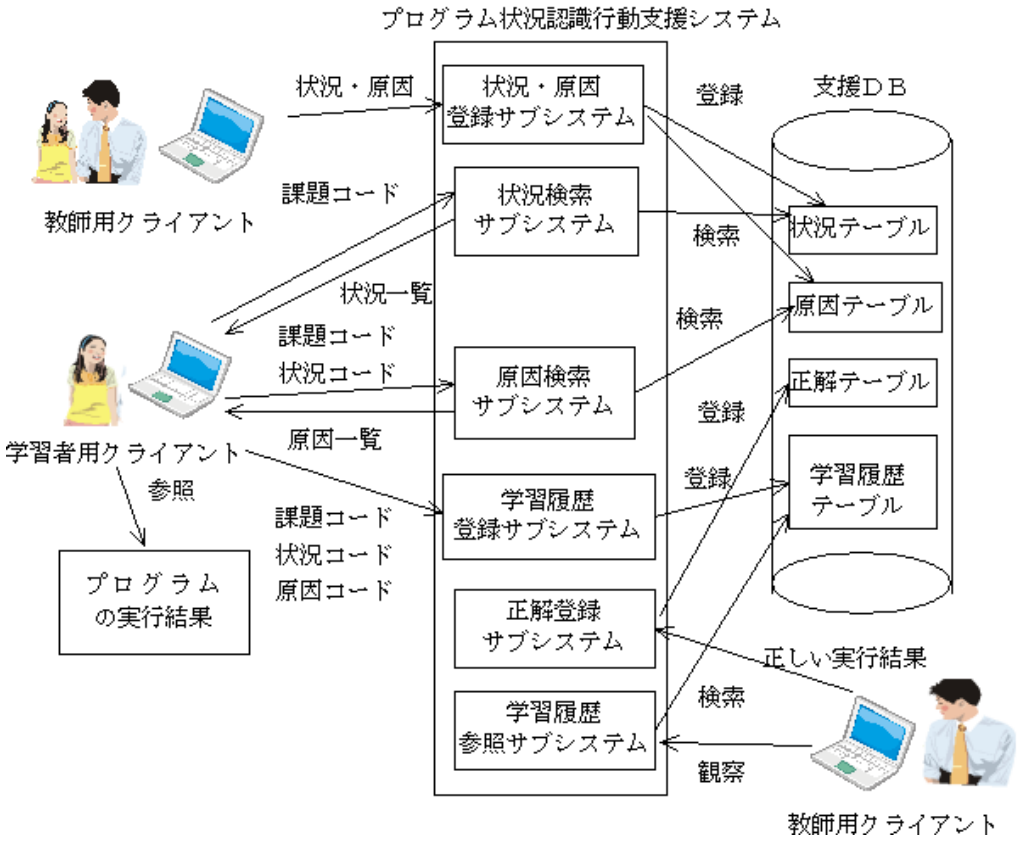

図４　プログラム状況認識行動支援システムの構造図

(4) 学習履歴テーブル

学習者コード，課題コード，日時をキーとして学習者が判断した誤りプログラムの状況コードと誤り原因コードを記録する。学習者プログラムが正しい結果を出した場合は，状況コードに完成を意味する値が入る。

正解テーブル

課題コード	正しい実行結果

状況テーブル

課題コード	誤りのある実行結果	状況コード	状況

誤り原因テーブル

課題コード	状況コード	誤り原因コード			誤り原因
		大分類コード	中分類コード	小分類コード	

学習履歴テーブル

学習者コード	日時	課題コード	状況コード	誤り原因コード		
				大分類コード	中分類コード	小分類コード

図５　支援ＤＢの構造

３．２　状況検索サブシステム

学習者は，状況検索サブシステムに課題コードを送信すると，サブシステムはまず正解テーブルから正しい実行結果を表示する。学習者が，学習者プログラムの実行結果と比較して同じでないと判断した場合，サブシステムは状況テーブルから課題コードに一致する全ての誤りのある実行結果と状況を検索し，一覧表を学習者に提示する。

３．３　誤り原因検索サブシステム

学習者は，状況検索サブシステムによって提示された誤りのある実行結果と状況の一覧を参照し，学習者プログラムの実行結果と同じ状況にある実行結果を探す。発見した場合，学習者は誤り原因サブシステムに課題コードと状況コードを送信する。誤り原因検索サブシステムは，状況コードに一致する誤り原因を誤り原因テーブルから検索する。さらに，検索結果から課題コードに一致するものを絞り込んで学習者に提示するが，その際，検索結果の誤り原因コードの中分類コードが指定されていない場合は大分類コードが課題コードの上位部分に一致するものを残す。さらに，大分類コードが指定されていないものは全て残す。

学習者プログラムの実行結果と一致する状況を発見できない場合は教師に相談する。

３．４　学習履歴登録サブシステム

学習者は，誤り原因検索サブシステムによって提示された誤り原因一覧を一つ一つ参照し，学習者プログラムの誤り原因を検証する。原因が特定できた場合はプログラムを修正・再実行する前に，学習履歴登録サブシステムに課題コード，状況コード，誤り原因コードを送信し，学習履歴登録サブシステムは学習履歴テーブルに登録する。誤りが無くなった場合，状況コードとして完成を入れる。

誤り原因を特定できない場合，学習者は教師に相談する。

３．５　状況・原因登録サブシステム

教師は，学習者が状況を特定できない，または状況は特定できたが誤り原因が特定できないと相談された場合，学習者が参照した状況一覧や誤り原因一覧，学習者プログラムの実行結果を参照し，学習者を指導する。

その際，新しい状況や誤り原因が出現したと判断した場合，状況・原因登録サブシステムを使って状況テーブルや誤り原因テーブルに追加する。

３．６　正解登録サブシステム

教師は，演習を行う前に課題の正しいプログラムと検証するためのデータを作成し，正解登録サブシステムを使って正しい実行結果を正解テーブルに登録する。

３．７　学習履歴参照サブシステム

教師は，学習者の演習活動状況を観察し，活動が活発でない者を指導しなければならない。そのため，学習履歴参照サブシステムを使って学習履歴テーブルを参照し，記録の存在や誤りがあるまま放置している課題の存在を確認する。

また，学習者のプログラミング履歴と学習履歴を比較し，学習者の状況判断の誤りを見つけて指導する。状況判断は正しいが，誤り原因の判断が間違っている場合は指導しない。試行錯誤によって誤り原因を自分で特定した方が能力を獲得できるからである。

４．実装

本システムはデータベースとネットワークを使ったＷｅｂアプリケーションとして実装する。学習者や教師はＷｅｂブラウザを使って本システムにアクセスできる。図６はユーザインターフェースの例である。

図６　本システムのユーザインターフェース

５．おわりに

現在筆者は，４０数名の学生にＣ言語によるプログラミングを指導している。彼らは週あたり４，５本のプログラムを作成・修正しているので，筆者は毎週２００本近いプログラムを添削しており，その負担は小さなものではない。本システムを活用することでこの作業を行う必要がなくなる。

しかし代わりに，学生のプログラム状況認識の誤りを指導しなければならないが，これがどの程度負担になるかは不明である。また，プログラム状況や誤り原因がデータベースに数多く登録されるまではシステムが有効に働かないので，登録作業も大きな負担になることが予想される。従って，システムを利用する初年度は教師の負担は大きいと予想されるが，２年目以降は負担が減り，学習効果も向上するものと期待している。

システムはまだ完成していないので，早急に完成させ実際の演習に利用し，有効性を検証したい。

参考文献

1)　高本明美, 藤井美知子, 田中稔, ”プログラミング学習支援を目的としたコンパイラメッセージに基づく誤り要因リストの自動生成”， 教育システム情報学会誌, Vol. 15, No. 1, pp. 3-12, (1998)

2)　鷹岡亮, 岡本敏雄, ” エージェントによるＣシェルプログラミングの診断機構について”， 信学技法ET96-92, pp33-38, (1996-12)

3)　海尻賢二, ” ゴール／プランに基づく初心者プログラムの認識システム”, 電子情報通信学会論文誌, Vol. J

78-D-Ⅱ, No. V2, pp. 321-332(1995)
4) 渡辺博芳, 荒井正之, 武井恵雄, ”事例に基づく初等アセンブラプログラミング評価支援システム”, 情報処理学会論文誌, Vol. 42, No. 1, pp. 68-78, (2001)
5) 松田憲幸, 柏原昭博, 平嶋宗, 豊田順一, “プログラムの振る舞いに基づく再帰プログラミングの教育支援”, 電子情報通信学会論文誌, Vol. J80-F-Ⅱ No. 1, pp. 326-335(1997)
6) 矢野将之, 藤崎邦博, 平嶋宗, 竹内章, “再帰構造の図式を導入したPrologプログラミング学習支援システム”, 教育システム情報学会誌, Vol. 18 No. 3-4(秋・冬合併号), pp. 319-327(2001)
7) 平嶋宗, 今田洋史, 池松秀則, ミヤツカラヤ, 竹内章, “探索アルゴリズムを対象とした対話的学習環境”, 教育システム情報学会誌, Vol. 18 No. 3-4(秋・冬合併号), pp. 264-273(2001)