初心者が作成した反復構造を含むプログラムの誤り場所指摘アルゴリズム

久保田　聡[*1]・青木　征男[*2]・渡邉　高司[*3]

An Algorithm which Points Out Errors in a Program including Loop Structures Made by Beginners

Satoshi Kubota, Yukio Aoki and Takashi Watanabe

Synopsis

An algorithm which points out program's errors in loop structures is developed. It inspected a program made by beginners that has no compile error and can run, but has wrong output. The algorithm was applied for some sample programs and it pointed out 80% of errors. We will make a system based on this algorithm which points out locations where errors have occurred.

1．研究の背景と目的

初心者を対象としたプログラミング教育において，教員の負担は大きい。教員は学習者の作成したプログラムの誤りを調べ，その誤りを指摘する。初心者の作成したプログラムには誤りが含まれている場合が多く，初心者は自分の作成したプログラムが正しいかどうかの判断方法すら知らない。そのような初心者が誰の手助けも無く誤り場所を見つけるのは困難である。教員にとって誤りを探すのは難しい仕事ではない。しかし，多数の学習者に対して行うことは大きな負担となる。

そこで，初心者の学習を支援するために，いくつかのシステム[1]-[8]が多くの研究機関から報告されているが，それらのシステムが実用される機会は少ない。

今までに，本学でもプログラミング教育支援のためのシステムがいくつか作成されている。プログラムの一部を空欄にして回答を入力させるシステム（空欄補充型）[1][2]や，ブラックボックステストを行い正誤を判断するシステム（Black Box型）[3]が作成された。しかし，[1][2]は自由なアルゴリズムによるプログラムができない。また，初心者でも誤りのある場所を指摘されれば，その指摘から何故誤ったのか解る場合が多い。しかし，Black Box型では何処で誤ったのか判らない，といった問題がある。

自由なアルゴリズムでプログラムを作成でき，誤った場合に誤り場所を指摘できるシステム[6]を直井らが開発した。これは，問題ごとに複数の回答例を用意する事で精度よく誤り場所を指摘するシステムである。しかし，問題ごとに複数の回答例を用意しなくてはいけない為，教員は1問追加するために多くの労力を必要とする。また，いろいろな方法を使って，アルゴリズムが類似したプログラムを1種類の正解例プログラムで表現するシステム[7][8]が存在するが，まったく違うアルゴリズムで正解プログラムを作成できるなら，そのアルゴリズム用の正解例プログラムを用意する必要がある。たとえ最初の一回だけとはいえ，このような多くの労力を必要とするシステムでは教員は使用を見合わせるだろう。

そこで，我々は自由なアルゴリズムでプログラムを作成でき，誤った場合に誤り場所を指摘でき，かつ，事前に用意する情報を最小限に抑えたシステムABSS（Ayamari Basho Shiteki System）[4]を作成した。しかし，このシステムは逐次構造と分岐構造のみで構成され

＊1　大学院工学研究科　　＊2　電子情報工学科　　＊3　㈱ブレイクスルー
平成11年9月30日受理

たプログラムに対してしか使用できない。

そこで，本研究ではこのABSSを反復構造を含むプログラムの誤りも指摘できるように拡張するために，反復構造に誤りのあるプログラムの誤り場所を指摘するアルゴリズムを作成した。

2．誤り場所指摘システムの方針

誤りは大きく分けると，次の2つに分けることができる。

(1) コンパイルエラーが起こり，プログラムを実行できない誤り。

(2) コンパイルエラーが無く，実行できるが，期待した結果が得られない誤り。

学習者はコンパイラが指摘したエラーメッセージにより，上記(1)の誤りに気付く。しかし，(2)の誤りには気付かない場合が多い。そこで，我々は上記(2)のプログラムの誤り場所を指摘するアルゴリズムの検討を行った。

ABSSではシステム側で用意する情報を

(a) 問題文

(b) 正解プログラム

(c) 検査データ

(d) 正解プログラムに検査データを与えた場合の実行結果（正解データ）

の4つに限定する。これらの情報を元にして，プログラムの誤り場所を見つける。

学習者が問題文を読み，プログラムを作成する。システムは学習者の作成したプログラムを実行し，その実行結果と正解データを比較し，一致しなければ誤りがあると判断し，4章で説明するアルゴリズムを適応して誤り場所を指摘する。

3．反復構造に含まれる誤り

3．1　反復構造の構成

反復構造は一般的に以下の4つから構成されている。これを図1に示す。

A．初期化部分

B．反復の条件部分

C．反復構造の本体部分

D．再初期化部分

Aは反復の条件式に使用される変数を初期化する部分，Bは反復の終了条件部分，Cは繰り返される命令部分，DはBで使用される反復条件の再初期化部分である。

3．2　反復構造に含まれる誤りの種類

反復構造に含まれる誤りの種類を以下の10種類に分類した。

(1) 繰り返す回数が一回多い

(2) 繰り返す回数が数回多い

(3) 繰り返す回数が一回少ない

(4) 繰り返す回数が数回少ない

(5) 一回も繰り返さない

(6) 無限に繰り返す

(7) 初期化が誤っている

(8) 再初期化が誤っている

(9) 条件式がまったく違う

(10) 繰り返し本体に誤りがある

4．誤り場所の指摘方法

3．2の10種類の誤りについて，それぞれの誤り場所指摘アルゴリズムを考えた。以下の方法を順番に用いて，誤り場所を指摘する。

4．1　繰り返す回数が一回多い，一回少ない

もっとも単純な誤りである。ほとんどが条件式の"＝"

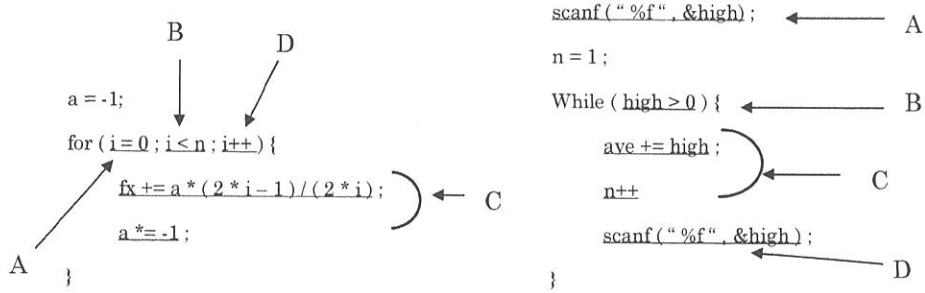

図1：反復の構造

の付け間違いによって起こる誤りである。そこで，条件式に"＝"がついている場合には"＝"を外し，ついていない場合には"＝"を付加した修正プログラムを実行し，その実行結果と正しい実行結果を比較することによって誤り場所を指摘する。

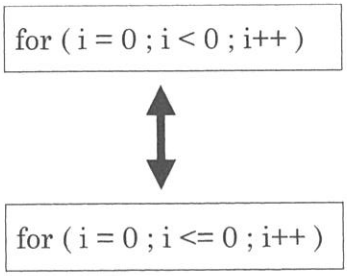

図2　"＝"を付加する

4．2　繰り返す回数が数回多い，数回少ない

反復の条件式が"$a<b$"なら，"$a<b+1$"のように条件式の右辺に +1 を行って実行し，その結果が正しい実行結果に近づけば「数回少ない」と指摘する。また，反復の条件式に −1 を行い，実行した結果が正しい実行結果に近づけば「数回多い」と指摘する。

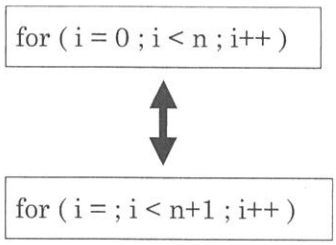

図3　(a)＋1を行う

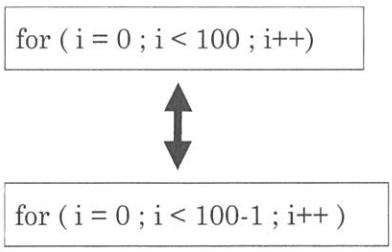

図3　(b)－1を行う

4．3　一回も繰り返さない

この誤りはほとんどの場合，反復の条件式が逆になっている。そこで，反復の条件式を否定する。すなわち「$i<n$」なら「$i \geqq n$」とし，「$i \geqq n$」なら「$i<n$」と条件式を変更して実行した結果を正しい実行結果と比較することによって誤り場所を指摘する。

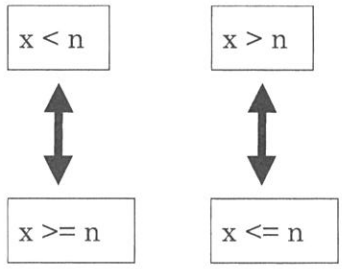

図4　条件式を否定する

4．4　無限に繰り返す

反復条件式が永遠に満たされ続けることによって起こる誤りである。多くの場合，反復条件式に使用される変数の値が変更されていない，すなわち，変数の再初期化が行われていないことが原因である。そこで，反復構造の中に変数の再初期化用の命令が存在しなければ誤りと指摘する。

4．5　初期化が誤っている，再初期化が誤っている，条件式がまったく違う，繰り返し本体に誤りがある

スライシング [5] を行って誤りを指摘する。この方法は先に開発したABSSのアルゴリズムをそのまま用いた。

5．机上検討

4章に示したアルゴリズムに対して以下のソースコードを用いて机上検討を行った。本学で行われた1995年度前期試験の下記の3問に対する1年生97名の答案の内，誤りのあるプログラムを分析した。

問題1：整数 n と実数 x を入力し，x の n 乗を求める。n が0や負の場合も計算すること。ただし，*pow* 関数をつかってはいけない。*for* 文を使うこと。配列を使ってはいけない。x に0は入力されないと仮定してよい。

問題2：複数の学生の身長 *high* を *cm* 単位で入力し，平均値 *ave* を求め，小数第一位まで出力する。負の身長が入力されたらデータの終わりとする。*while* 文を使うこと。配列を使ってはいけない。

表１：誤り場所指摘結果

No.	誤　り　の　種　類	問題番号			合計（人）	指摘
		1	2	3		
1	繰り返し回数が一回多い	1			1	○
2	繰り返し回数が数回多い					
3	繰り返し回数が一回少ない	2		3	5	○
4	繰り返し回数が数回少ない					
5	初期化が誤っている					
6	再初期化が誤っている					
7	無限に繰り返す		1		1	○
8	一回も繰り返さない	3	2		5	△
9	条件式がまったく違う	3	2		5	×
10	繰り返し本体に誤りがある	19	4	36	59	△
11	まったく違うプログラムになっている	1		2	3	×
12	変数の初期化が違う	2	4		6	×
	合計	31	13	41	85	

○：指摘できた
△：指摘できないものもあった
×：指摘できなかった

データ数を入力してはいけない。

問題３：項数 n を入力し，下記の級数の第 n 項までの和 sum を求め，小数第５位まで出力する。配列を使ってはいけない。

$$sum=\sum_{i=0}^{n}(-1)^{i+1}\frac{2i-1}{2i}$$

答案の誤りを12種類に分類し，各々の問題に対する誤った答案数を表１に示す。これらの誤りのあるプログラムに対して４章で説明したアルゴリズムを適用した。

問３に対する解答プログラム図５(a)に本アルゴリズムを適用した場合を説明する。まず学習者の作成したプログラムを実行し，実行結果と正解データを比較し，一致していないのでプログラムに誤りがあると判断した。次に誤り場所を指摘するために４章で述べたアルゴリズムを適応した。すなわち，４．１節の方法に従い図５(b)のように変更して実行し，実行結果を正解データと比較した。一致していないので，次に４．２節に従い図５(c)のように変更して実行した。実行結果と正解データと変更前の実行結果とを比較した。正解データに近づいていないので，次に４．３節に従って図５(d)のように変更して実行した。実行結果と正解データとを比較した。一致するので，誤りの種類は〝一回も繰り返さない〟誤りであり，誤り場所は *while* 文の条件式であるとわかった。

以上のような机上検討を全てのプログラムに対して行った結果を表１に示す。誤り場所を指摘できた項目については○を，指摘できなかった項目に対しては×とした。空欄は誤りプログラムが無かったことを示す。

６．考　察

６．１　無限ループ

No.８の〝一回も繰り返さない〟誤りは，５個のうち４個は指摘できたが，１個は指摘できなかったので△とした。この指摘できなかった誤りプログラムを図６に示す。図６の誤りの場合，*for* 文の条件式"$i<n$"を否定すると無限ループになる。このような無限ループは，変数の再初期化は行われているので，４．４節で示したアルゴリズムでは誤りを指摘できない。しかし，次のようにすれば指摘できることが判った。

反復条件が"$i<n$"の時，大きい方から小さい方を引いた値"$n-i$"が再初期化により大きくなれば無限ループになる。このような考えを指摘アルゴリズムに組み込むことによって指摘できる。

６．２　命令が不足している

表１ No.10の〝繰り返し本体に誤りがある〟プログラムが△になっているのは，問題(2)において〝必要な処理が抜けている〟誤りがあり，この誤りについては指摘できなかったためである。

７．結　論

初心者が作成したプログラムの誤り場所を指摘するアルゴリズムを考案した。４章に６．１節のアルゴリズムを加え，それを97名の試験の問題３問に対する解答プロ

```
#include<stdio.h>
main(){
        float high,ave,i=1;
        scanf("%f",&high);
        while(high<0){
                ave+=high;
                scanf("%f",&high);
                i++;
        }
        ave=ave/i;
        printf("%f %f %f",ave,high,i)
}
```

(a)学習者の作成した誤ったプログラム

```
#include<stdio.h>
main(){
        float high,ave,i=1;
        scanf("%f",&high);
        while(high<=0){
                ave+=high;
                scanf("%f",&high);
                i++;
        }
        ave=ave/i;
        printf("%f %f %f",ave,high,i)
}
```

(b)4.1 節に従って変更したプログラム

```
#include<stdio.h>
main(){
        float high,ave,i=1;
        scanf("%f",&high);
        while(high<0+1){
                ave+=high;
                scanf("%f",&high);
                i++;
        }
        ave=ave/i;
        printf("%f %f %f",ave,high,i)
}
```

(c)4.2 節に従って変更したプログラム

```
#include<stdio.h>
main(){
        float high,ave,i=1;
        scanf("%f",&high);
        while(high>0){
                ave+=high;
                scanf("%f",&high);
                i++;
        }
        ave=ave/i;
        printf("%f %f %f",ave,high,i)
}
```

(d)4.3 節に従って変更したプログラム

図5　誤り場所を指摘できるプログラム例

```
if ( n<0 ){
        for ( i=0 ; i<n ; i++ ){
                y*=x;
        }
}
```

(a)誤ったプログラム

```
if ( n<0 ){
        for ( i=0 ; i>n ; i-- ){
                y*=x;
        }
}
```

(b)正しいプログラム

図6　条件式を否定すると無限ループになるプログラム

グラムに適用したところ，85個の誤りのうち67個の誤り場所を指摘でき，本アルゴリズムが有効であることが机上検討で判った。

今後，本アルゴリズムをABSSに組み込み，誤り場所指摘システムを作成する予定である。

参考文献

[1] 下野聡久，諏訪省三，中本和彦，山本良範：Cプログラム間違い指摘システム，久留米工業大学卒業研究論文，1995年3月

[2] 河田直樹，坂田浩一：WWWを利用したC言語教育支援システム，久留米工業大学卒業研究論文，1998年3月

[3] 黒岩和行，川本泰幸：Cプログラミング支援システムの試作，久留米工業大学卒業研究論文，1996年3月

[4] 渡邉高司，青木征男：プログラミング初心者を対象とした誤り場所指摘システムの試作，電子情報通信

学会技術研究報告書，ET98-60，pp41-48，1998年
[5] 下村隆夫，「プログラムスライシング技術と応用」，共立出版株式会社，1995年7月7日
[6] 直井将行，福田妙明，上野晴樹：知的プログラミング教育環境INTELLITUTORにおけるプログラミング理解システムALPUSの拡張，情報処理学会第55回全国大会講演論文集(4)，pp464-465，平成9年
[7] 服部徳秀，石井直宏：ソースコードのバリエーション除去システム，電子情報通信学会論文誌，Vo.J80-D-I，No.1，pp.50-59
[8] 服部徳秀，石井直宏：プログラミング演習の評価サポートシステムの構築，教育システム情報学会誌，Vol.14，No.1，pp.21-28