関数の引数にポインタを用いたプログラムの可視化による動作理解支援ツールの提案

2015SE033 加藤ちひろ　2015SE048 松尾翔馬　2015SE054 森本朱音
指導教員：蜂巣吉成

1　はじめに

C 言語におけるポインタの学習は最大の難関といわれている．[4] ポインタの学習において，学習者は初め，ポインタ変数の宣言，間接演算子，アドレス演算子の使い方などを学習する．学習者はポインタの学習までに習う，記述したそのままの変数名を使った直接的な参照方法と，ポインタ変数を用いた間接的な参照方法との違いにつまづきやすい．C 言語の参考書では，main 関数内で普通の変数とポインタ変数を使った例示はよくされるが，関数の引数にポインタを用いた例示はあまりされていない．実際にポインタを使用するプログラムでは，main 関数内のみではなく関数の引数にポインタを用いた場合のほうが多い．main 関数内のみでポインタ変数の指す先のアドレスや値を操作するプログラムを理解していても，関数の引数にポインタを用いたプログラムの作成，理解ができない学習者が存在してくる．

理解ができない理由として，引数にポインタが用いられているときにポインタ変数が具体的にどの変数を指して，操作しているのかがわかりにくい，ということが挙げられる．仮引数では実引数の値のコピーを受け取る，という概念が理解できず，アドレス演算子 (&) を付ける理由がわからない学習者も存在する．C 言語の参考書では，例題の解説にメモリ空間を意識した図を用いることがあるが，学習者自身が作成したプログラムには図がないので，プログラムの動作のイメージがつかめない．コンパイル時にポインタに関する警告が出ることがあるが，警告文は英語で書かれており，翻訳したとしてもポインタを初めて学ぶ学習者には理解しづらい専門的な言葉が含まれている．ポインタの仕組みを理解していないと，コンパイル時のエラーメッセージを理解できず，プログラムの修正ができない．

本研究では C 言語を対象とし，関数の引数にポインタを用いたプログラムのポインタの動作を理解させ間違いのあるプログラムを修正できるように支援をすることを目的とする．変数を表した箱を関数ごとにまとめて表示し，ポインタ変数の指す先，変数の値の変化を図に表す．学習者自身が書いたプログラムから図を表示するので，使用者それぞれのプログラムについてのイメージができる．警告が出るプログラムについても可視化を行う．正しいプログラムの図と比較することで，何が誤っているのかを理解させる．また，文章で間違っている部分の指摘と修正案の候補を提示する．本研究は，ポインタを初めて学ぶ学習者を対象とし，配列や構造体，動的なメモリ確保については扱わない．コンパイルエラーが出るプログラムも対象外とする．

2　関連研究

プログラミング学習における動作理解支援を目的とした研究がいくつかされている．

VIE システム [1] では，プログラムの変数や関数を可視化し，対話的な理解支援を行う．VIE システムは，main 関数内のみのプログラムの状態の可視化を行う．学習者が可視化されたオブジェクトに対して操作をすると，操作後の状態に適したソースコードの例を表示する．ポインタを用いたプログラムの理解支援にも対応しているが，main 関数内のみの可視化なので，本研究の目的とする，関数の引数にポインタを用いたプログラムの理解支援は行えない．

文献 [2] は，C 言語を対象としており，ソースコードと変数の値を記述した表と，指定した表の列，行にあるポインタ変数と関連している変数の関係図を作成する．この関係図では，指定したひとつのポインタ変数とその指している変数の図のみしか生成されない．関数内に複数のポインタ変数が存在する場合，ポインタ変数ごとに指定をしなければならない．警告が出るプログラムに関しては，可視化はするが，間違いを指摘する文は表示されない．学習者は図だけでは間違いを見つけ，修正することが困難である．

SeeC[3] は，C 言語を対象とした初学者向けのツールである．SeeC では，main 関数から実行が始まり，学習者がボタンを押して実行を進めるのに伴って，その時々の変数や関数の関係図を表示する．警告が出るプログラムに関しても可視化を行うが，その際に出る間違い指摘文は“！”のみである．学習者はこのメッセージを見ただけではプログラムの修正方法を考えつくことが困難である．

本研究で作成した動作理解支援ツールは，Web 上で実現するので，先に挙げた関連研究すべてで必要であったツールのダウンロードを必要としない．ツール内で表示する図では，ポインタ変数と普通の変数の差に加え，int 型と double 型の違いも，色の違いや形の違いによって視覚的に明確にする．また，警告が出るプログラムに関しては，間違っている場所を探索し，ソースコードのどこで警告が出るのかということと，具体的な修正案を提示する．これにより，学習者はツールを使用し，ポインタ変数のプログラム上での役割を視覚的に理解しつつ，警告が出る場合は，間違い指摘文と修正案を見て，プログラムの修正を行うことができる．

3 動作理解支援ツール

3.1 学習者が起こしやすい間違い

我々の学習における経験などから学習者が関数の仮引数にポインタを渡す際の間違いを，次の場合に分類した．

A　ポインタ変数と指す先の変数の型の間違い

A-1　ポインタ変数がポインタ変数を指す間違い

A-2　不適切にポインタ型で宣言する間違い

A-3　基本型の間違い

B　ポインタ変数にアドレス値が代入されていない間違い

実引数と仮引数の関係把握に混乱しやすい，ソースコード 1 を例にあげる．仮引数がポインタである関数内で，さらに仮引数がポインタである関数を呼び出しているので参照関係を混乱しやすい．

main 関数内の sort3 の呼び出しにつられ，ソースコード 2 のような間違いを起こしやすい．2，3，4 行目は A-1 の間違いである．仮引数がポインタである関数を呼び出す際の実引数に闇雲にアドレス演算子をつけてしまうことが原因である．

ソースコード 3 の 2 行目は A-2 の間違いである．関数の引数にポインタを使用しないプログラムと同じように，関数の仮引数と同じ型で実引数の型を宣言している．

ソースコード 4 の 2 行目は A-3 の間違いである．ポインタまたは参照先の変数の基本型を異なる型で宣言している．

ソースコード 5 は B の間違いである．4 行目の sort3 関数の呼び出しの際に，実引数にアドレス演算子を用いていないので，変数のアドレスではなく値が渡され，ポインタが未定義の場所を指している．

以上の間違いの原因は，ポインタを十分に理解しておらずポインタを用いたプログラムの状態をイメージできていないこと，引数にポインタを用いた関数の仮引数と実引数の関係を正しく把握できないことであると考えた．これらの誤ったプログラムをコンパイルすると警告文が表示される．たとえば Listing6 のように「int *」へのキャストをせずに整数からポインタを生成していると警告されるが，ポインタを理解していないと意味がわからないので修正は困難である．

ソースコード 1　3 つの値を昇順に並べ替えるプログラム

```
1  void swap(int *a,int *b){
2      int temp=*a;
3      *a=*b;
4      *b=temp;
5  }
6  void sort3(int *x,int *y,int *z){
7      if(*x>*y) swap(x,y);
8      if(*y>*z) swap(y,z);
9      if(*x>*y) swap(x,y);
10 }
11 int main(void){
12     int p,q,r;
13     p=4; q=2; r=3;
14     sort3(&p,&q,&r);
15     return 0;
16 }
```

ソースコード 2　swap 関数の呼び出しを間違えているプログラム

```
1  void sort3(int *x,int *y,int *z){
2      if(*x>*y) swap(&x,&y);
3      if(*y>*z) swap(&y,&z);
4      if(*x>*y) swap(&x,&y);
5  }
```

ソースコード 3　p，q，r の型を間違えているプログラム

```
1  int main(void){
2      int *p,*q,*r;
3      p=4;q=2;r=3;
4      sort3(&p,&q,&r);
5      return 0;
6  }
```

ソースコード 4　sort3 の仮引数の型が間違っているか p，q，r の型が違うプログラム

```
1  int main(void){
2      double p,q,r;
3      p=4;q=2;r=3;
4      sort3(&p,&q,&r);
5      return 0;
6  }
```

ソースコード 5　main 関数で sort3 の呼び出しを間違えているプログラム

```
1  int main(void){
2      int p,q,r;
3      p=4;q=2;r=3;
4      sort3(p,q,r);
5      return 0;
6  }
```

Listing 6　ソースコード 5 のコンパイル結果の一部

```
source3.c: In function ‘main’:
source3.c:14:5: warning: passing
 argument 1 of ‘sort3’ makes
 pointer from integer without a cast
  [enabled by default]
     sort3(p,q,r);
     ^
sourse3.c:6:6: note: expected ‘int
 *’ but argument is of type ‘int’
 void sort3(int *x,int *y,int *z){
      ^
```

3.2 動作理解支援方法

本研究では関数の引数にポインタを用いたプログラムの状態を学習者が把握しやすいよう図としてプログラムの状態を可視化する．ポインタの指している先を的確に把握できるような表示をする．3.1 節で示した間違っているプログラムについても可視化を行い，間違い箇所に気づくことができるようにする．また同時に文章で間違いを指摘することでどのように間違えているかを具体的にわかるようにし，それを修正する案も提示する．

3.2.1 可視化方法

本ツールはプログラムを逐次実行して変数などの情報を取得し，各地点で図を表示する．ポインタに関する図の描画は次のように行う．

- 1 つの変数は 1 つの箱で表し，型は色で区別する
- 変数名は箱の上に，変数の値は箱の中に表示する
- ポインタは二重の箱で表し，ポインタのポインタは三重の箱で表す
- ポインタが指す先は矢印で表す．null ポインタの場合は箱に斜線を入れる
- 指す先が不明の場合，箱の下に“？”を表示する
- 変数を関数ごとにまとめて表示する

図 1 に，ソースコード 1 を可視化した図を示す．

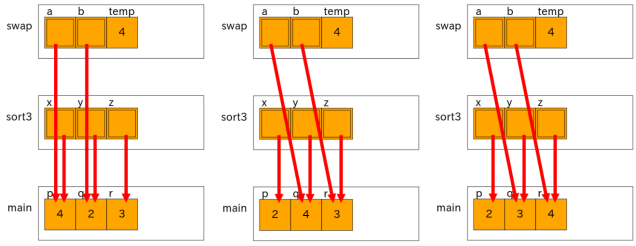


図 1　ソースコード 1 を可視化した図 (左：2 行目，中央：8 行目，右：4 行目)

型の違いを箱の大きさで区別する方法もあるが，図が見づらくなったりメモリ空間を勉強していない学習者にはわかりにくいので色で区別する方法を採った．int 型は黄色，double 型は青色で表現する．ポインタに関しては，宣言時の*p につられポインタを*p とするような間違った表現をしないよう，ポインタ変数は二重の箱，ポインタのポインタは三重の箱で描画し，視覚的にわかりやすくする．

3.2.2　間違い指摘文と修正案の表示方法

本研究で扱う警告の出るプログラムは 3.1 節で挙げた A，B とする．間違いの種類を判別し指摘文と修正案を表示する．ポインタに関する警告文が出た場合，原因を判定するのは困難であるが，対象者はポインタ初学者であるため初学者の学習内容に合わせた間違いに限定し支援する．ソースコード 2 ではポインタ変数がポインタ変数を指していることを指摘する．修正案は swap の呼び出しの際に&をつけてしまっている可能性があること，sort3 の仮引数の型を int に直すこと，swap の仮引数の型を int **に直すことの 3 つが挙げられるが，初学者の学習内容ではポインタのポインタの例示は少ないので，前者 2 つを表示する．

3.3　設計・実現

3.3.1　設計

3.2 節で提案した図を表示し，警告がでるようなプログラムに関しては，間違い指摘文と修正案を表示する．これらはプログラム実行時のある状態を可視化しているので，実行する行を進めてプログラムの状態を追うことが必要である．main 関数をスタートとして，学習者がボタンでプログラムの実行する行を進めるまたは戻すことでプログラムの状態を追えるようにする．

3.3.2　実現

ツールを実現するにあたり，ソースコードの変数名，変数の値，型，アドレス，関数名を取得する必要がある．これらの情報は GDB でプログラムを逐次実行させ情報を取得している．これらの部分は Java を用いて自動化し，取得した情報を JSON 形式でファイルに出力する．そのファイルのデータを元に 3.2 節で提案した図と間違い指摘表示を提示する．図は HTML の Canvas 要素を用いて描画し，間違い指摘表示は JavaScript を用いて実現した．ツールは Web 上で動作し，学習者はブラウザで C ファイルと標準入力の入力データを送信するとその情報を元に CGI でツールを動作させ，図 2 のツール画面を返す．左側に可視化しているプログラム，左下に間違い指摘文と修正案，中央に関数ごとに変数の情報を表示し，右側に可視化した図を表示している．右上のボタンで実行する行を進めたり戻すことができる．間違い指摘文，修正案は左下のボタンで表示または非表示にできる．

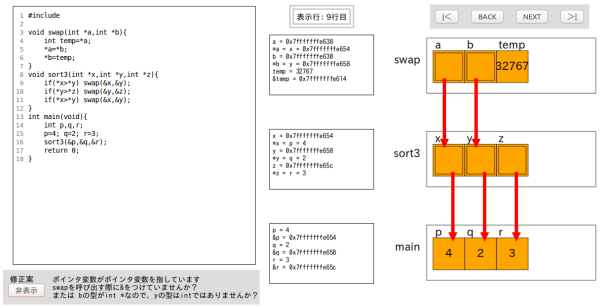
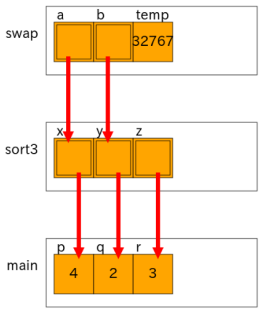


図 2　動作理解支援ツールの表示画面

図を見やすく表示するために，ツールには関数呼び出しを 3 段階まで，1 つの関数内の変数の数は 6 つまで，基本型は int 型，double 型に，ポインタ変数は int*，int**，double*，double**に限る制約を設けた．いくつかの参考書 [5][6] を調べたところ，この制約内で例題が扱える．

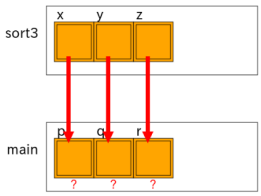
3.4　ツールの実行例

3.1 節で挙げた，間違いの例をツール実行した結果を図 3，4，5，6 に示す．正しいソースコード 1 は図 1 である．



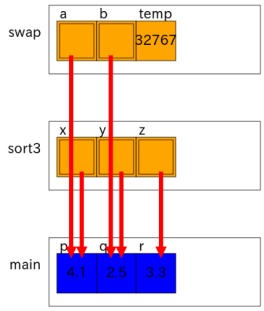
指摘文：
ポインタ変数がポインタ変数を指しています
修正案：
swap を呼び出す際に&をつけていませんか？
または b の型が int *なので、y の型は int ではありませんか？

図 3　ソースコード 2 の 2 行目の可視化と指摘文と修正案



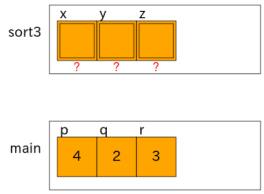
指摘文：
ポインタ変数がポインタ変数を指しています
修正案：
sort3 を呼び出す際に&をつけていませんか？
または z の型が int *なので、y の型は int ではありませんか？

図 4　ソースコード 3 の 4 行目の可視化と指摘文と修正案



指摘文：
ポインタ変数の型と指す先の型が一致しません
修正案：
z の型は double *ではありませんか？または r の型は int ではありませんか？

図 5　ソースコード 4 の 4 行目の可視化と指摘文と修正案



指摘文：
z の指す先が不適切です
修正案：
z にアドレス値を代入してください

図 6　ソースコード 5 の 4 行目の可視化と指摘文と修正案

4　評価

関数の引数にポインタを用いたプログラムの，正しくコンパイルできるものと警告文が出るものにツールを使用し，期待通りのツール画面を得られるか検証した．正しいプログラムとして C 言語の教科書 [5] のうち関数の引数にポインタを用いる例題 3 題と演習問題 2 題に提案したツールで可視化できるか確認したところ，表示することができた．警告の出るプログラムは 3.1 節で挙げた間違いの種類すべてに使用したところ，期待通りの図と間違い指摘，また正しいプログラムに導くような修正案を得られた．

5　考察

5.1　学習におけるツールの利用方法

学習者が間違ったプログラムを作成し，コンパイルの際に警告文がでた場合，警告文を読んだだけではプログラムがなぜ間違っているのかを理解し，それを修正することは難しい．本研究で作成したツールでは，次のように段階的に利用することで理解を促す．

1. 学習者のプログラムから図を表示する
   ツールを利用して，警告の出るプログラムから図を表示し，この図で間違いを理解して修正させる．
2. 模範解答プログラムから正しい図を表示する
   1 で理解できない学習者には，教員等があらかじめ作成した正しい図と比較させて，なにが間違っているのかを理解させる．
3. 修正案の表示
   2 でも修正できない学習者には，ツール画面の左下にある表示ボタンを押すことで, 間違いに合った指摘文と修正案を表示する．

5.2　ツールの拡張

本研究で作成したツールには，3.3.2 項で述べた制約があるが，これらは拡張可能である．現段階で，制約を超えている場合でも C のプログラムから情報を取得し，JSON ファイルに格納することができる．この情報をもとに Canvas で図を描画できるようにすることは可能であるが，矢印が重ならないよう見やすく表示したりするのが困難である．1 つの関数内の変数の数が制約を越える場合は，画面上にその図を描くために初めから余白ができてしまうので，見栄えが悪くなる．

6　おわりに

本研究では，C 言語における関数の引数にポインタを用いたプログラムの可視化による動作理解支援ツールを提案した．C 言語を対象とし，引数にポインタが用いられている関数間のポインタの関係性を可視化により明確にする。警告がでるプログラムの場合も可視化と修正案の提示を行い，間違いのある箇所と理由を明確にすることで，学習者自身でプログラムの修正ができる．今後の課題として，配列や構造体を含むプログラムの可視化を検討している．

参考文献

[1] 川崎雄登，平井佑樹，金子敬一：プログラミング学習のための可視化対話環境，情報教育シンポジウム 2014 論文集，Vol.2014，No.2，pp.143-150(2014).

[2] 河本健吾，山田恭裕：初学者向けのポインタを用いたプログラムの動作理解支援方法の提案，南山大学情報理工学部 2015 年度卒業論文 (2015).

[3] SeeC - program visualization and debugging for novice C programmers, available from <https://seec-team.github.io/seec/index.html>(accessed 2019-01-11).

[4] 前橋和弥：新・標準プログラマーズライブラリ C 言語ポインタ完全制覇，技術評論社 (2017).

[5] 柴田望洋：新・明解 C 言語入門編，SB クリエイティブ (2014).

[6] 大川内隆朗，大原竜男：かんたん C 言語，株式会社技術評論社 (2010).