C プログラムの構文理解支援方法の提案

2014SE002 安達有希　2014SE057 牧野美里　2014SE080 小原友貴

指導教員：蜂巣吉成

1　はじめに

我々が C プログラムを学習してきた中で，難しいと感じたのがコンパイルエラーとなるプログラムの理解や修正であった．何がどう間違っているのかを理解することを難しく感じていたが，エラーの多くが構文の間違いであり，構文の理解をしていないことが原因であることが分かった．構文を意識せず理解もしていないまま，講義資料や教科書を真似てプログラムを書くだけでは，プログラムの間違いにも気がつきにくく応用も利かない．しかしそのような学習者は我々を含め周りにも多くいた．そのような学習者に構文を意識させ理解を促すことができれば，自身が書いたプログラムをより深く理解でき，構造が類似した構文どうしや他言語での構文を学ぶための抽象化能力を養うこともできる．

本研究の目的は，C プログラムの構文理解支援方法を提案することである．学習者が書いたプログラムの構文に対し，構文図式とそれを元にした図式を表示することで，視覚的に構文の理解を促す方法について考察する．プログラム中で確認したい構文を学習者自らが選択し，図式を表示させる方法についても考察する．

2　関連研究

プログラムの可視化や理解支援を行うツールの開発や研究はいくつか行われている．

2.1　HelpMeOut

HelpMeOut[1] は，エラーの解決方法をクラウドで共有し，エラーが起きた時に適切な修正方法を提案するツールである．ソースコード中のエラーのある行を選択し，HelpMeOut を参照すると，提案パネルが表示され，いくつかの解決策が表示される．解決策にはそれぞれ詳細な情報を表示したり正しい記述をそのままコピーしてエラーを直すボタンがある．HelpMeOut は，時間の経過とともに変更されるソースコードのログをデータベースへ収集している．そのログを基に，提案パネルはエラーに対する解決策を示している．データベースから修正の優先順位リストを解決策を表示している画面へ表示し，新しく追加された修正の説明をデータベースへ送信することで解説を集めている．修正方法を提示するが，ただその通りにプログラムを修正していると，必ずしも構文を理解することには繋がらない．

2.2　clang

clang[2] は C 言語向けのオープンソースのコンパイラである．gcc よりも詳細で理解しやすいエラーメッセージが表示されるので，誤りの修正が容易になっている．Listing1 のような構文に誤りがあるプログラムをコンパイルしたときに表示されるエラーメッセージである図 1，図 2 を比較すると，どちらも else の前に if や式がないことを指摘している．プログラムの中には if も式も書かれているので，構文を理解していない学習者にとっては，これらのエラーメッセージを頼りにプログラムの修正を行うことは難しい．

Listing 1　if-else に誤りのあるプログラム例

```
if (n%i == 0)
   printf("%d", i);
   n = n/i;
else
   i++;
```

```
14se057@2014-pc:~/sotuken/sample_compile$ cc fac.c
fac.c: 関数 ‘main’ 内:
fac.c:14:3: エラー: ‘else’ の前に ‘if’ がありません
14se057@2014-pc:~/sotuken/sample_compile$
```

図 1　Listing1 の gcc のコンパイル結果

```
14se057@2014-pc:~/sotuken/sample_compile$ clang-3.9 fac.c
fac.c:14:3: error: expected expression
                else
                ^
1 error generated.
14se057@2014-pc:~/sotuken/sample_compile$
```

図 2　Listing1 の clang のコンパイル結果

2.3　構文図式によるコンパイラ動作の可視化

岩崎らの研究 [3] はコンパイラの内部の処理状況を，機能単位ごとに分けて考え，構文図式や構文木などを用いて可視化を行っている．構文解析については，再帰下降パーサが処理している構文図式を次々と重ねて表示し，その動作を可視化している．コンパイラの動作理解を目的としており，本研究が対象とするプログラミング学習者の構文理解には向かない．

2.4　プログラム自動可視化ツール Avis

Avis[4] は，プログラムの読解支援を目的に，Java のソースコードを解析して，プログラムの流れを理解するためのフローチャートや，プログラム実行時の振舞いを理解するための逐次型実行経路図などを出力する．構文を理解させるための支援は行っていない．

3　理解支援方法の提案

3.1　構文理解の重要性

本研究を進めていくにあたり，指導者と学習者の認識の違いがあることが分かった．指導者がプログラミングを教える際，資料や参考書を利用する．それらを利用し学習する側が構文を意識して学習していくと考えていることが多いようだが，学習者はその意図を理解していない．構文に着目しないまま資料や参考書を利用する学習者も多い．

Listing2 と Listing3 はどちらも正常にコンパイルでき，意図した通りの結果が得られるソースコードの一部分である．構文を理解していなければ，なぜ書き方が異なっていても同様の結果になるのかは理解ができない．

Listing 2　if-else が複合文のプログラム例

```
if (num){
    puts("The number is not zero.");
}else{
    puts("The number is zero.");
}
```

Listing 3　if-else が単文のプログラム例

```
if (num)
    puts("The number is not zero.");
else
    puts("The number is zero.");
```

Listing4 も本来繰り返したい文の前が空文となっており，コンパイルができても意図した結果は得られない．構文自体は間違っていないので実行するまで間違いに気付かない．

Listing 4　for が空文のプログラム例

```
for (i=2; n>1;);
    if (n%i == 0){
       printf("%d", i);
       n = n/i;
    }else
       i++;
```

これらは資料や参考書を構文にも着目しながら読んでいれば，なぜそうなるのか理由を理解できる．また，構文を意識することで構文どうしで似ている部分やそうでない部分に注目でき理解も深まりやすくなる．構文を理解していれば C 言語だけでなく他言語にも応用が利き，構文を学ぶための抽象化能力を養うこともできる．

3.2　本研究の特徴

構文を可視化する方法として，構文図式，構文木を挙げる．構文を表す方法として構文図式，BNF を挙げ，学習者にとってどちらが理解しやすいかを比較した．

可視化の方法を比べた 2 つにおいて，プログラムの流れの把握は矢印で表現できる構文図式の方が容易であるが，再帰部分は分かりにくい．構文木は構文を解析した結果を木構造で表したものであるので，学習者がプログラムの構造がどのようになっているかや構文をあらかじめ学習者が理解している必要があり，本研究の目的には適していない．構文を表す方法において，BNF はテキスト形式での表示であるので，構文図式のように図で表す方が視認性が高いと考えた．以上を踏まえて構文図式を基に可視化を行う．

3.3　扱う構文について

本研究で扱う構文は次の 3 つである．

- if 文
- while 文
- for 文

これらは我々が演習をしていた際によく利用したものであり，学習者がこれらの構文を利用する機会が多いと判断した．構文理解を促すのに適していると考える．

3.4　提案する構文図式

本研究では，我々が教科書として使用していた [5] に記載されているものを参考に，図 3 のような構文図式をそれぞれ構文ごとに用意した．ノードの形には意味があり，終端記号を丸囲みで表し，中でも構文の種類を表す予約語は二重丸囲みで表している．式や文など非終端記号は四角囲みで表している．構文図式の各ノードには w_lp などの名前が付いており，丸囲みや四角囲みの下に表示している．

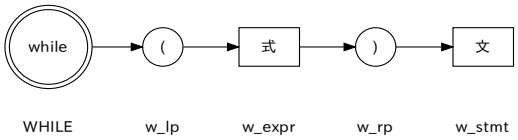

図 3　while の構文図式

学習者が書いたプログラムを解析し，その内容を構文図式の形式に反映させた図式の例を図 4 に示す．

非終端記号を表す四角囲みのノード内に学習者の書いたプログラムを表示する．学習者の書いたプログラムの構文部分に欠如している部分があれば，図 5 のようにその部分のノードにはプログラムではなくノードの名前が入るようにした．ただし，for 文の式の部分が空式になっている場合は欠如ではないので，ノード名は入らない (図 11)．

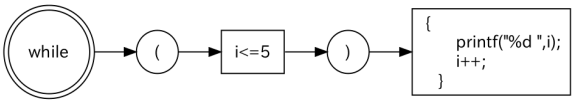

図 4　学習者のプログラムを反映させた図式 1

図 3 のような構文図式と学習者のプログラムの図式である図 4 や 5 を上下に並べて表示することで，学習者は自身が書いたプログラムの構文部分がどのような構造になっているのかを比較しながら確認することができる．

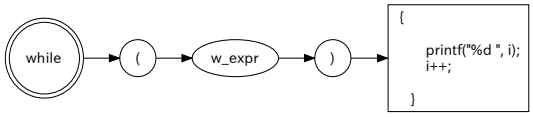

図 5　学習者のプログラムを反映させた図式 2

4　設計・実装

4.1　設計

本研究では，図式の生成ツールの作成と，生成した図式を表示するための方法を考案した．ツールは図式の生成ツールと HTML 生成ツールの 2 つがある．

図式の生成ツールは，学習者が自分の書いたプログラムに対して，TEBA[6] による解析結果を用いてプログラムに出てくる全ての if 文，while 文，for 文に対して，それぞれ図式を描画するための dot ファイルと，そのファイルから Graphviz[7] を用いて図式を jpg ファイルに変換したものを生成する．

HTML 生成ツールは学習者のソースプログラムと図式の生成ツールで出力された画像ファイルを入力とし，if 文，while 文，for 文の予約語部分に構文図式の画像を表示するためのリンクを含んだ HTML ファイルを出力する．

4.2　実現

ツールは Perl を用いて作成した．図の生成ツールは 372 行，HTML 生成ツールは 91 行である．

4.3　ツールの利用方法

学習者は自身が書いたソースプログラムを入力として，4.1 節の 2 つのツールを実行し，出力された HTML ファイルをブラウザで開く．ブラウザ上でリンクの貼られた学習者の書いたプログラム本文が表示され，学習者はそこから自分が見たいもののリンクを選択し図式の表示を行う．

例えば，Listing5 のプログラムであれば，図 6 のようにブラウザで表示する．学習者がリンクを選択すると図式が図 7 や図 8 のようにポップアップで表示される．

Listing 5　プログラム例

```
while (i<=5) {
    if (i%2==1){
        printf("%d␣is␣odd␣number.\n",i);
        i++;
    } else {
        printf("%d␣is␣even␣number.\n",i);
        i++;
    }
}
```

5　考察

文献 [5] に示されているサンプルプログラム 205 編のうち，if 文，while 文，for 文の 3 種類の構文が含まれている

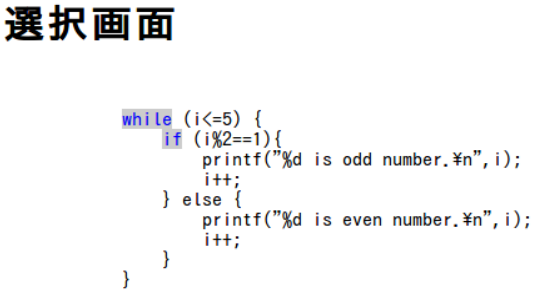

図 6　Listing5 プログラム例の表示

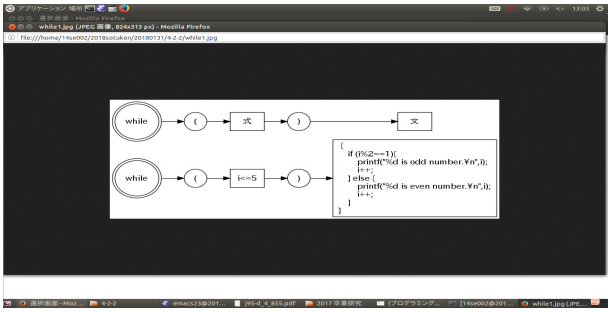

図 7　Lisiting5 の while の図式をポップアップで表示

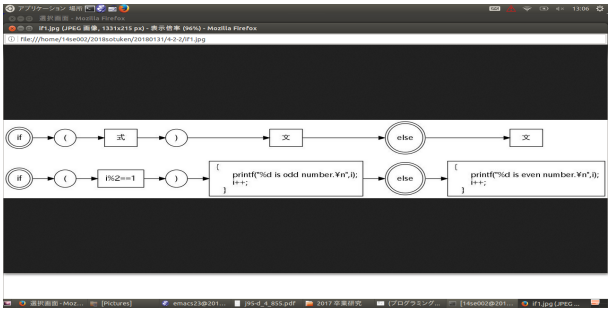

図 8　Lisiting5 の if の図式をポップアップで表示

134 編に対して図式の生成ツールを使用したところ，すべてのプログラムで適切な図式の生成を行うことができた．

3.1 節で述べた Listing2 と Listing3 のような書き方が異なっていても同じ結果を返すプログラムも，それぞれ図 9，図 10 を見ることによって文が複合文の場合でも単文の場合でも，どちらも同じ文であることが分かる．

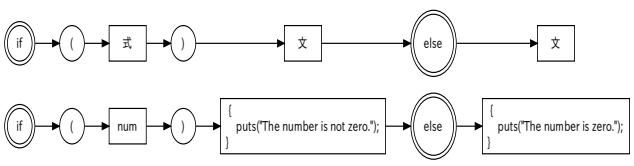

図 9　文が複合文の例の図式

Listing4 のような実行可能であるが，意図した結果にな

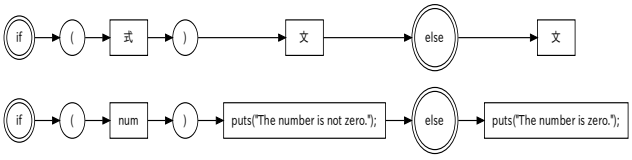

図 10　文が単文の例の図式

らないようなプログラムも図式にした図 11 を見れば本来繰り返したい文が空文となっていることがその原因であることが分かる．

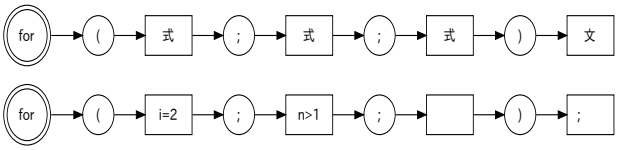

図 11　Listing4 の図式

構文図式の生成は構文ごとにそれぞれ行われ，学習者の書いたプログラムがほぼそのままの形で図式に反映されるので，複数の入れ子の複雑なプログラムや記述が乱れていて見づらいようなプログラムでも自分の書いたプログラムが構文的にどうなっているかが分かる．構文を理解していない学習者だけでなく，資料や参考書を真似ているだけの構文を意識していない学習者にも，構文を学ばせるのに有効である．

構文部分に誤りのあるプログラムからでも構文ごとにそれぞれ図式の生成が行われる．誤りのあるプログラムからも図式の生成が行えることによって，構文に起因するような意味的な誤りに気づくことができると考える．

2.2 節で述べた Listing1 のような誤りを含んだプログラムからは，図 12，図 13 の構文図式が生成される．図 12 を見れば 1 つの文しか if 文の文として認識されていないことや複数の文を複合文にし，1 つの文として記述する必要があることが分かる．また，エラーメッセージからは誤りの原因が分かりにくいと述べていたが，図 12 だけでなく図 13 も生成することによって，なぜこのようなエラーメッセージが表示されるのかも分かる．

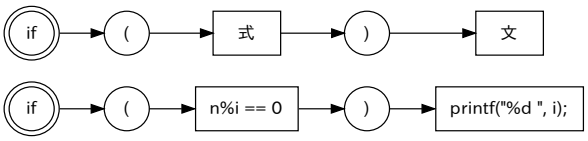

図 12　Listing1 の図式 1

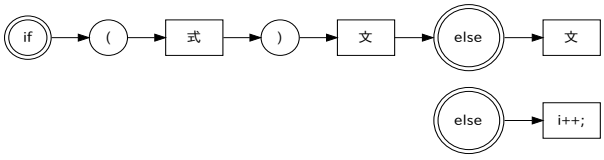

図 13　Listing1 の図式 2

6　おわりに

本研究では，C プログラムの構文理解支援方法の提案をした．学習者が書いたプログラムの解析結果を用いてそれぞれ構文ごとに構文図式の形式で表した図式を表示し，視覚的に構文の意識・理解を促せるようにした．また，図式の表示は選択式にし，書かれたプログラム内に図式のリンクを作ることでプログラム全体の構造を理解しやすくなった．本研究で使用した構文は 3 つだが，この他にも構文を増やしていくことと，学習者が実際にツールを使用したときの評価を行うことが今後の課題である．

参考文献

[1] Björn Hartmann, Daniel MacDougall, Joel Brandt, Scott R. Klemmer :“ What Would Other Programmers Do?Suggesting Solutions to Error Messages ”, CHI 2010:Understanding and Supporting Programming, pp.1019-1020 (April 10-15, 2010, Atlanta, GA, USA)

[2] 公式サイト：The LLVM Compiler Infrastructure, https://llvm.org/

[3] 岩崎克治，辻野嘉宏，都倉信樹 :“ 教育を目的としたコンパイラの動作の可視化手法とその効果について ”，情報処理学会研究報告コンピュータと教育 (CE)，1990-CE-011，pp.1-8(1990).

[4] 喜多義弘，片山徹郎，冨田重幸 :“ Java プログラム読解支援のためのプログラム自動可視化ツール Avis の実装と評価 ”，電子情報通信学会論文誌 D，Vol.J95-D No.4，pp.855-869(2012).

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

[6] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満 :“ 属性付き字句系列に基づくソースコード書き換え支援環境 ”，情報処理学会論文誌，Vol.53 No.7，pp.1832-1849(2012).

[7] 公式サイト：Graphviz - Graph Visualization Software，http://www.graphviz.org/