プログラミング学習における理解支援のための実行履歴 3D 可視化方法の提案

M2023SE011 月原花菜
指導教員：蜂巣吉成

1　はじめに

プログラミング学習において，学習者は繰返し，特に 2 重ループがある場合，外側と内側のループがどのように実行されているか理解することが難しい場合がある．プログラムに誤りがあった場合，誤り箇所や修正方法が分からないことがある．文字列探索など複数のアルゴリズムが存在する場合，違いが分からないことがある．

文献 [1][2] では，繰返しのプログラムの実行履歴を表形式で 2 次元的に提示することで，プログラムの流れや変数の変化，繰返しの回数などを俯瞰的に見て確認できるようにし，デバッグの支援を行った．しかし，2 重ループを含むプログラムの実行履歴は繰返しの回数に応じて横に広がり，必要な情報が表のどこに書かれているか分かりにくく，プログラムの流れも理解しづらい．

本研究は C 言語学習者を対象に，直感的にプログラムの流れを理解できるように，繰返しプログラムの実行履歴を 3D で可視化する方法を提案する．2 次元的な可視化よりも，2 重ループの外側と内側がどのように実行されているかを直感的に理解しやすくなり，文献 [1][2] の実行履歴を可視化した表が横に広がる問題を解決する．模範解答と学習者プログラムの実行履歴を可視化，比較することで，学習者自身がプログラムの誤り箇所や改善箇所を発見し，修正できるようになると考える．

本研究の技術的課題として，直感的に理解しやすいプログラムの実行履歴可視化方法が挙げられる．重要な情報を視認しやすく，理解しやすいように表現する必要がある．プログラムを分岐のない連続した文のコード片であるセグメントで分割し，セグメント単位で実行履歴を扱う．セグメントをブロックで表現し，ループ回数分ブロックを並べたり積み重ねたりすることで，繰返した回数や繰返しの仕方が視覚的に理解しやすくなると考える．条件分岐を含む場合，実行されなかったセグメントは出力されず空間があるようにすることで，条件分岐が正しく行われたかどうかの判断がつきやすくなる．ブロック上に繰返しの条件式に出現する変数のみを表示することで，2 重ループの外側と内側のループの関係性を分かりやすくする．

2　関連研究

文献 [1][2] は，命令型プログラミング言語のプログラムの動作理解を支援するツールを提案している．プログラムの実行時における各命令文を実行した後のすべての変数の値と分岐文や繰返し文の条件式の真偽を自動的に記録し，それらを俯瞰できるように表形式で表示している．しかし，2 重ループ以上の繰返しを含むプログラムになった場合，表が横に広がり，必要な情報を見つけづらいという問題がある．本研究では，文字ではなくブロックでプログラムの実行履歴を可視化する．3D 上では 2 次元的に表現するよりも，狭い範囲で繰返した回数や繰返しの仕方が俯瞰的に見え，理解しやすいと考える．

文献 [3] は，実行時ログの理解容易性の向上を目的に，条件分岐と繰返しプログラムを対象にソースコードと時間軸を追加したログデータを 3 次元空間で表現し，変数の変遷を表現する方法を提案した．x-y 平面上にソースコードを表示し，実行されない部分を灰色にすることで実行される部分とされない部分を区別する．z 軸方向に繰り返される数だけソースコードを表示し，ループ回数が分かるようになっている．しかし，VR 上で文字は読みづらく，直感的に認識しづらいと考え，本研究では実行履歴をブロックを用いて表現する．

文献 [4] はプログラム可視化のサーベイ論文であるが，本研究のように実行履歴を 3D を用いて表現することでプログラムの理解支援を行う研究はなかった．

3　プログラム実行履歴の 3D 可視化方法

3.1　問題分析

2 重ループの外側と内側のループの関係性に次の 4 つが挙げられる．

1. 外側のループと内側のループは互いに独立する
2. 内側のループは外側のループに依存する
3. 外側のループは内側のループに依存する
4. 外側のループと内側のループは互いに依存する

本研究では，ループ A で計算・代入される変数の値がループ B の継続条件に関係するとき，ループ B はループ A に依存すると呼ぶ．

1 は互いに独立しているので，それぞれのループを分けて考えられる．2 と 3 は変数の値の変化とそれによる繰返しへの影響を把握できないと，プログラムの流れが理解できない．4 はより複雑に影響し合うので理解が困難である．

3.2　提案方法の概略

実行履歴を 3D で表現し，繰返しを可視化することで，プログラムの流れの理解支援を行う．3D で表現する理由は次の通りである．文献 [1][2] から，プログラムの逐次実行を上から下に向かって 1 次元的に表現し，繰返しを含む場合は右に向かって 2 次元的に表現するのは有効だと考える．しかし，2 重ループも 2 次元的に表現すると，見づ

らくなるなどの問題が生じるので，本研究では 3 次元的に表現するのが有効だと考えた．この考え方に従い 3 重ループ以上の繰返しを含むプログラムを可視化すると，4 次元以上の次元となり現実的ではない．これに対する対応方法は，4 章で考察する．異なるアルゴリズムによる複数のプログラムや，誤りを含む学習者プログラムと模範解答プログラムの実行履歴を 3D で可視化し，比較することで，それらの差異を理解しやすくなると考える．

実行履歴を用いず，フローチャートやプログラム依存解析などの静的解析を用いる可視化方法もある．しかし，3 と 4 の関係は実際に実行しないと内側のループで計算された変数の値が外側のループにどのような影響を与えるか分からない．

変数の値の変化を表で表す方法もあるが，アルゴリズムによっては変数が多くなり，必要な情報が見づらくなる．本研究では繰返しの条件式に出現する変数のみを表示することで，繰返しへの影響も読み取れるようにする．

3.3 前提条件

本研究は，C 言語プログラムの関数を対象とし，実行履歴取得ツール [5] などにより可視化に必要な情報は取得できるとする．本研究における実行履歴とは，実行されたセグメントの履歴のことで，繰返しのセグメントでは条件に関係する値を含む．

3.4 セグメント

本研究ではプログラムをコード片 (セグメント) の集合と捉え，3D を用いてセグメントをブロックとして表現し，実行履歴の情報を元にプログラムの流れを可視化する．これにより，表示する情報が必要以上に多くなるのを防ぎ，学習者が直感的にプログラムの流れを理解しやすくなると考える．セグメントの定義は文献 [6] にしたがう．

1. 制御文本体 (文) はセグメントである．ただし，複合文の波括弧は除く．
2. 制御文の予約語と制御式を囲む丸括弧の部分はセグメントである．
3. これら以外の連続した宣言と文はセグメントである．
4. 制御文の入れ子の場合は 1 から 3 のセグメントも入れ子になるが，このうち最内のものをセグメントとする．

セグメントに分割する例を Listing1 を元に説明する．Listing1 はデジタルルートを 2 重ループで求めるプログラムであり，3.1 節の 3 の 2 重ループである．

Listing 1 デジタルルート (2 重ループ)

```
1  int main() {
2      int num; //A
3      int n, sum;
4      num = 1999;
5      n = num;
6      while (n >= 10) {//B
7          sum = 0;//C
8          while (n > 0) {//D
9              sum += n % 10; //E
10             n /= 10;
11         }
12         n = sum;//F
13     }
14     printf("%d のデジタルルート
           : %d\n", num, sum); //G
15     return 0;
16 }
```

Listing1 は 2 から 5 行目までの「int num; int n, sum; num = 1999; n = num」，6 行目の「while(n >= 10)」，7 行目の「sum = 0;」，8 行目の「while(n > 0)」，9 から 10 行目の「$sum+ = n\%10$; $n/ = 10$;」，12 行目の「n = sum;」，それ以降の 7 つに分割できる．

3.5 可視化方法

プログラムから文献 [5] の実行履歴取得ツールなどを使用して，セグメント単位の実行履歴を取得する．セグメント，各セグメントごとのループした回数，変数の値に関する情報を取得しテキストファイルに記録する．そのテキストファイルから unity を利用し可視化する．セグメントをブロックで表現し，繰り返した回数分ブロックを並べていくことで，繰返し回数や仕方，条件分岐が正しく行われているかなどが直感的に理解できると考える．また，ブロックの表面上に繰返しの条件式に出現する変数の値を表示する．

3.6 ブロックの配置方法

実行履歴の可視化は次の手順で行う．

1. セグメント単位で実行履歴を取得する
2. 縦軸 (y 軸) の上から下にセグメントのブロックを並べる位置を決定する
3. 実行されたセグメントのブロックを決定した位置に配置する．このとき実行されなかったセグメントの位置は空になる
4. 繰り返しの場合，1 重ループ, 2 重ループの外側のループは横軸 (x 軸) に並べる位置をずらし，ブロックを配置する．2 重ループの内側のループは手前側 (z 軸) に並べる位置をずらし，ブロックを配置する

ブロックは，while を緑，for を青，if を赤，else を黄，else if を橙，その他を白で表現する．

以上をもとに可視化した例を図 1 に示す．例として実行履歴 A,B,C,D,E,D,E,D,E,D,E,D,F,B,C,D,E,D,E,D,F,B,C,D,E,D,E,D,F,B,G の一部を用いる．

図 1 上の数字 2,3,4 は手順の 2,3,4 番目を表している．Listing1 の実行履歴を可視化した図を図 2 に示す．

外側のループ回数を横側に，内側のループ回数を手前側に並べて表現することで，表で可視化するより横幅を抑えられる．プログラムの流れを俯瞰でき，繰返しの仕方や回数，条件分岐の有無も分かりやすい．

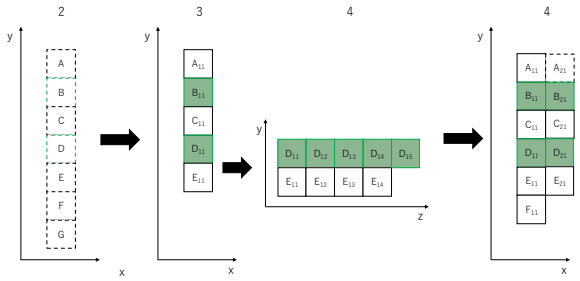

図 1　実行履歴可視化例図

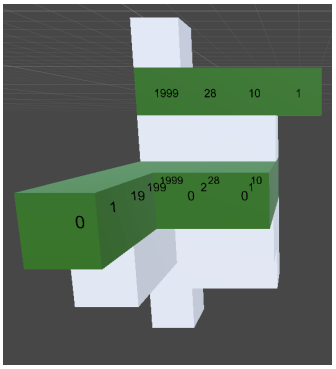

図 2　デジタルルート (2 重ループ) の実行履歴可視化図

4　評価・考察

本研究で作成したツールを実行しプログラムの実行履歴を 3D を用いて可視化することで，理解支援と間違い箇所の絞り込みにつながるか評価を行った．

4.1　理解支援の評価

デジタルルートを 1 重ループで実現したプログラムを Listing2 に，実行履歴を可視化した図を図 3 に示す．図 2 と図 3 を比較すると，ブロックの個数から繰返し回数に大きな違いがないことが分かる．図 3 の 2 段目の変数の値を見ると，n と sum の値を表示しているので，どのように計算が行われているか分かりやすい．しかし，図 2 の 2 段目と 4 段目の変数の値を見ても関係性が分かりづらい．繰返しの条件式に出現する変数の値だけでなく，繰返しで計算される変数 (今回の sum) の値も表示することで分かりやすくなると考える．

Listing 2　デジタルルート (1 重ループ)

```
1  int main() {
2      int num; //A
3      int n, sum;
4      num = 1999;
5      n = num;
6      sum = 0;
7      while (n > 0 || sum > 9) {//B
8          if (n == 0) {//C
9              n = sum; //D
10             sum = 0;
11         }
12         sum += n % 10; //E
13         n /= 10;
14     }
15     printf("%d のデジタルルート
           : %d\n", num, sum); //F
16     return 0;
17 }
```

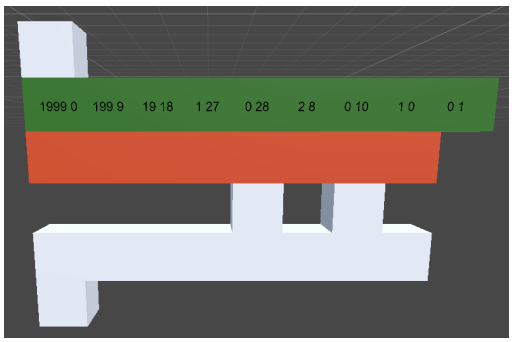

図 3　デジタルルート (1 重ループ) の実行履歴可視化図

KMP 法と BM 法で文字列 ABCXDEZCABACABAC から ABAC を探すプログラムを用意した [7]．実行履歴を可視化した図を図 4，図 5 に示す．図 5 のスキップテーブルを作成する箇所は，ブロックを表示すると横に広がり，主とする部分が見づらくなるので一部省略した．BM 法を用いた方が繰返し回数が少ないことが分かる．

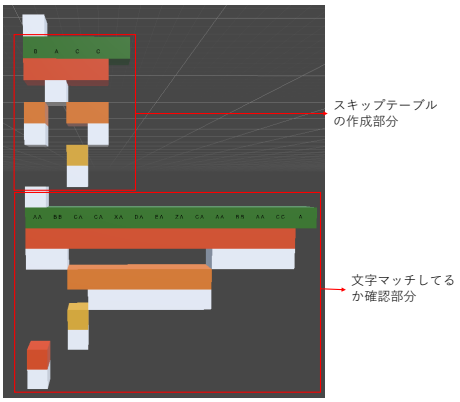

図 4　KMP 法の実行履歴可視化図

ブロックで表示することで，一目で繰返し回数が多い箇所が分かるので，プログラム実行において時間がかかる箇所の発見もできると考える．

4.2　間違い箇所絞り込みの評価

間違い箇所絞り込みの評価は，文字列 Nanzan から文字 a を 2 重ループで削除するプログラムを用いた．正しいプログラムと誤りありプログラムを可視化した図を図 6 と図 7 に示す．図 6 と図 7 の 2 段目を比較すると，Nanzan と Naza で繰返し回数と処理対象の文字が異なることが分かる．図 6 と図 7 の一番下の段は同じ処理の箇所だが，図 7

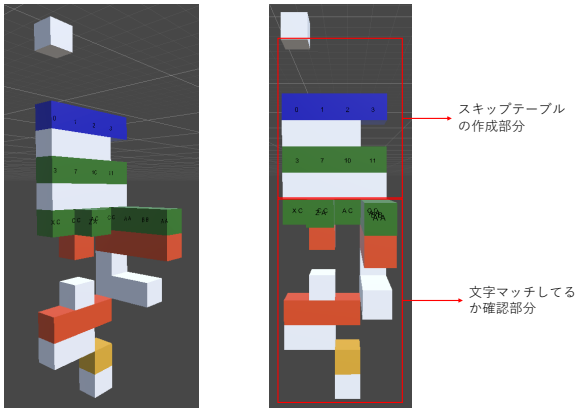

図 5　BM 法の実行履歴可視化図

は常にブロックが表示されている．ここから，条件分岐が適切でなく，本来実行されないセグメントも実行され，2 段目の繰返しに影響していることがわかる．可視化した模範解答プログラムの実行履歴と比較することで，全体の形や繰返しの回数などを比較でき，誤り箇所推定につながると考える．

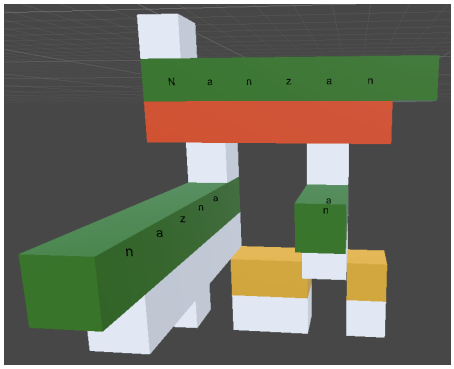

図 6　文字列の文字削除 (正しい) の実行履歴可視化図

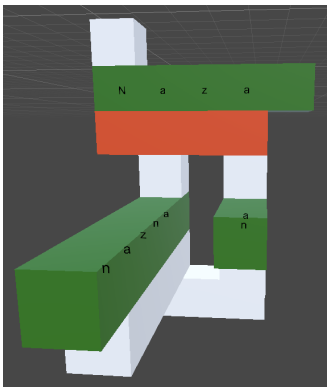

図 7　文字列の文字削除 (誤り) の実行履歴可視化図

4.3　3 重ループ以上の繰返しを含むプログラムの 3D 可視化について

3 重ループ以上の繰返しを含むプログラムの実行履歴を 3D 上で可視化する方法について考察する．逐次実行は 1 次元的に，1 重ループは 2 次元的に，2 重ループは 3 次元的に表現することで，複雑化して見づらくなる問題を解決し，俯瞰的に見やすくなると考え，3D での可視化を行った．しかしこの考え方だと，3 重ループは 4 次元，4 重ループは 5 次元と次元を増やすことになり非現実的である．そこで VR を用い，3 次元以上になる場合，外側の 2 重ループは 3D で表現し，内側の 1 重ループはブロック内に埋め込み，新たに 3D で表現することで対応可能と考える．また，内側の 1 重ループが重なっている部分のみ色を変化させることで，VR を用いる必要なく見ることが可能だと考える．

5　おわりに

本研究では，C 言語学習者を対象に，直感的にプログラムの流れを理解しやすくすることを目的に，繰返しプログラムの実行履歴を 3D で可視化する方法を提案した．今後の課題は，実行履歴からループ回数を取得するツールの作成と 3 重ループ以上の繰返しを含む場合の実行履歴を VR を用いて可視化するツールの作成が挙げられる．実際の教育現場で用いられた場合の実用性の調査などが挙げられる．

参考文献

[1] 蜂巣 吉成, 吉田 敦, 阿草 清滋: 命令型プログラミング言語における初学者向け動作理解支援ツールの提案, ソフトウェア工学の基礎 XXII (FOSE 2015), pp. 97-102 (2015-11)

[2] 長谷川 洸也, 川地 周作: 命令型プログラミングにおける動作理解支援に関する研究, 南山大学情報理工学部 2014 年度卒業論文 (2015)

[3] 鮎川 拓実, 深澤 良彰: オブジェクト指向プログラムにおける実行時ログの 3 次元表現とその操作, 情報処理学会第 85 回全国大会, pp. 219-220(2023-2)

[4] Juha Sorva, Ville Karavirta, and Lauri Malmi, ”A Review of Generic Program Visualization Systems for Introductory Programming Education”, ACM Transactions on Computing Education, Vol. 13, No. 4, Article 15, Publication date: November 2013.

[5] 田中 裕人, 戸田順也: C 言語学習者向けの実行履歴取得およびポインタ更新可能なプラットフォームの提案, 南山大学理工学部 2020 年度卒業論文 (2021)

[6] 澤田 侑希, 梅田 裕一郎, 蜂巣 吉成, 吉田 敦, 桑原 寛明: プログラミング演習におけるセグメントを用いたソースコードの誤り箇所特定方法の提案, ソフトウェア工学の基礎 XXIX (FOSE 2022), pp. 55-60 (2022-11)

[7] 柴田 望洋: 新・明解 C 言語で学ぶアルゴリズムとデータ構造第 2 版, pp. 292-303, SB クリエイティブ (2021)