# 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

2017SE089 田中裕人　2017SE091 戸田順也
指導教員：蜂巣吉成

## 1　はじめに

C 言語ではポインタを用いて動的なメモリの確保，解放や自己参照構造体の定義を行う．自己参照構造体を利用し，リストや木構造といった基本的なデータ構造を実現する．ポインタやデータ構造などを講義や参考書で学習する際，その構造を図示することが多い．学習者は講義や参考書で図を見て操作を理解したつもりでも，プログラムがどのように動作するのかわからず，コードを記述できない学習者は多い．特にポインタの操作はメモリ空間を意識する必要があり，他のデータ型と比べて難解である．

プログラムを解析する際にプログラムを実行しながら解析して得られる情報を本研究では実行履歴と呼ぶ．学習者のプログラムの実行履歴を取得して可視化する様々な動作理解支援ツールは理解支援の目的に応じて作成される．例えば，自己参照構造体を指すポインタの操作によるメモリ領域の動的な確保，解放を学習者に理解させるなら，メモリ領域の割当の情報や関数や変数の情報の可視化が必要になる．再帰関数や繰り返し文などの反復制御によるポインタやノードの動作の初学者向けの理解支援であれば，メモリ領域の割当の情報は取得せず，関数や変数の情報の可視化を行えば，十分な理解支援が可能である．このようにツールの理解支援の目的や学習者の習熟度に応じて必要な情報や適した可視化形式は異なる．特に代入文の作成を明示的にサポートしているツールは少ない．

本研究では C 言語プログラムの動作理解支援ツール作成の効率を向上させるために，実行履歴を取得する処理の共通化と取得した実行履歴に対する API をプラットフォームとして提案する．またプラットフォームの評価のために，取得した実行履歴を表で出力するツールを作成する．

ポインタの操作は他のデータ型の操作と比べて特に難解なので，先行研究 [1] のような学習者が図のポインタの指す先を操作することで，操作に対応する命令文の候補と更新された図を出力する機能はプログラムの理解支援に有効である．本研究ではポインタのデータの更新と命令文の候補の出力を行う API を提案する．ツールは各メモリ領域にどのような変数が定義されているか，どのようなコードを記述すれば意図通りの処理が実現できるか確認できる．

## 2　関連研究

文献 [2] は 2013 年までの 30 年間の理解支援ツールを調査している．多くの可視化ツールに対する利用者の関わりはアニメーションの速度の変更や表示する画像の変更などであり，本研究で扱おうとしている実行履歴に対する操作からソースコードを生成するツールはなかった．可視化された情報に対して利用者が変更できるようなツールの必要性も述べている．

SeeC[3][4], PythonTutor[5][6], Willow[8] は学習者が実行したプログラムの実行結果からデータ構造などを可視化している．SeeC は学習者の操作によってプログラムを逐次実行して変数や関数の関係を可視化する．ただし変数宣言のみ，コードの記述量が少ないなど，可視化のみでは支援が足りない場合がある．PythonTutor は，ブラウザ上で学習者が記述したコード逐次実行しオブジェクトを可視化する．Python だけでなく Java，C，C++，JavaScript，Ruby に対応している．Willow はデータ構造に応じて可視化方法を変更できる．

文献 [7] で提案されている PVC は C 言語の変数，ポインタ，配列，動的に割り当てられたメモリとそれらの関係を可視化する．PVC は動的なメモリ管理やファイル入出力なども扱える可視化ツールである．インストールの手間なく使えるように Web 上で動作する．Web ブラウザでソースコードを入力して可視化ボタンを押すと可視化される．ブラウザで可視化された情報を見て，ソースコードを変更し，その結果を即座に可視化できる．しかし，これらのツールでは可視化された情報を利用者が操作してソースコードを生成する機能はない．

可視化された情報の操作からソースコードを生成するツールとして文献 [1] のツール，VIE[9] がある．文献 [1] のツールは，C 言語を対象にリストに対する操作を実現するプログラムの動作を可視化するツールを提案している．学習者の操作に対応する代入文を出力することで実現したい処理に対応するコードの記述支援ができる．VIE では，学習者の操作に応じてコードの候補を提示するシステムを構築し，対話的に学習支援ができる．可視化領域に作成されたオブジェクトに対して自由にマウスで操作することができ，それに応じたコードを提示する．文献 [1] のツールはリストのみ，VIE は main 関数内のポインタのみ扱える．

## 3　プラットフォームの提案

### 3.1　プラットフォームの概要

学習者のプログラムの実行履歴を可視化するツールに対しプラットフォームを提案する．本研究では，命令文の実行後のメモリ上の変数や関数の情報を実行履歴として扱う．プラットフォームを用いて作成される動作理解支援ツールは，実行履歴の取得，ポインタの指す先の更新，命令文の出力をプラットフォームの API を用いて処理し，API から得た情報を図示する処理を JavaScript のプログラムで

記述し，ブラウザ上で可視化するツールを想定している．

提案するプラットフォームは C プログラムの取得，解析，データファイルの出力を Java で実現し，解析されたデータをツールで取得するための API，ポインタの更新，命令文の候補の表示を行う API を JavaScript で実現する．データファイルは JSON 形式で出力する．

### 3.2 プラットフォームの設計

プラットフォームの設計指針を以下に示す．

1. 様々な動作理解支援ツールに対して十分な情報を取得するための API を提供する．
2. ポインタに関して実行履歴の情報を変更できる．

プラットフォームがツールに提供する実行履歴の内容は関連研究の動作理解支援ツール群の可視化内容から定める．変数や関数の関係，データ構造を可視化するツールである SeeC, Willow，リストのツール [1] では，画面に関数のコールスタック，変数名，変数の値，変数の型，アドレス，ポインタの指す先，構造体のメンバの情報を表示している．メモリ空間を可視化するツールである PVC では，画面にメモリアドレス，スタック，ヒープ，グローバル領域の変数の情報を表示している．本研究ではこれらの情報をツールに提供する API を実現することで，変数や関数の関係，データ構造，メモリ空間を可視化するツールの制作に対する支援を行う．

指針 2 は具体的にはポインタの更新によってメモリに対するコードの参照ができることである．コードから図を生成するだけではなく，実行履歴の情報を変更できることで図からコードを生成することが可能であり，学習者の理解支援に有効であると考えた．

### 3.3 クラス定義

指針 1 に従い，実行履歴の情報を表すクラスを定義する．情報の取得はクラスのメソッドにより行う．図 1 にクラス図を示す．

それぞれのクラスの説明を次に示す．

History **クラス**　文を実行した後のメモリ情報が複数ある列を表す．

**メソッド**　任意の文の実行した後の情報を取得する getSTMTs, getFirstSTMT, getSTMTByContent, getSTMTByLine，getSTMTByContentAtCount，getSTMTByLineAtCount がある．

ExecutionSTMT **クラス**　任意の命令文を実行した後の情報全体を表す．実行されたプログラムの行数 line，命令文のコード content を持つ．加えて，同じ文を反復して実行する場合を区別するために ID が割り振られている．

**メソッド**　命令文の実行後の各メモリ領域の情報を取得する getHeap，getGlobal，getFunctionStackTop，getFunctionStack がある．また前後の文の実行後の情報を取得する getNextSTMT，getPrevSTMT がある．

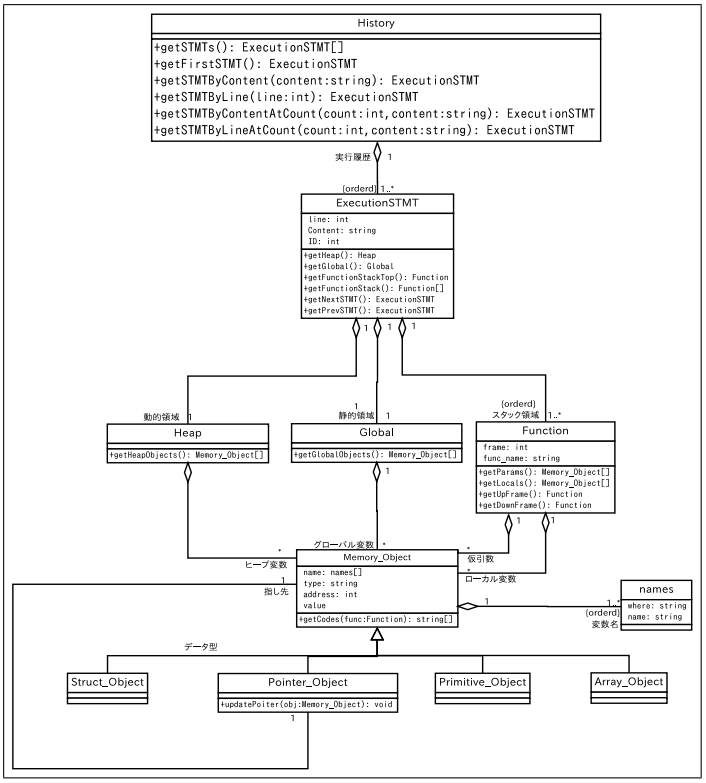


図 1　実行履歴を表すクラス図

Heap **クラス**　ヒープ領域の情報を表す．

**メソッド**　ヒープ領域上の変数の情報を取得する getHeapObjects がある．

Global **クラス**　グローバル領域の情報を表す．

**メソッド**　グローバル変数の情報を取得する getGlobalObjects がある．

Function **クラス**　実行中の関数の情報を表す．関数のフレーム frame，関数名 func_name を持つ．

**メソッド**　関数の仮引数やローカル変数の情報を取得する getParams, getLocals がある．また上下のフレームの関数の情報を取得する getUpFrame，getDownFrame がある．

Memory_Object **クラス**　文を実行した後のメモリ情報を表す．型 (type)，番地 (address)，値 (value) を持つ．変数名は names オブジェクトの配列で格納されている．変数の型ごとにサブクラスを持つ．

**メソッド**　ポインタの更新によって変更された変数の参照方法を確認する getCodes がある．

names **クラス**　変数の参照方法を表す．フレーム (where) とそのフレームからの参照方法 (name) を持つ．

Struct_Object **クラス**　構造体型の変数の情報を表す．

Pointer_Object **クラス**　ポインタ型の変数の情報を表す．

**メソッド**　ポインタの更新をする updatePointer がある．

Primitive_Object **クラス**　基本の型の変数の情報を表す．

Array_Object **クラス**　配列型の変数の情報を表す．

変数の参照方法は，その変数を異なる関数からスコープする場合やポインタから参照する場合など，複数の参照方法が存在することがあるので，変数の情報を持つ Memory_Object においては，変数名は names クラスのオブジェクトの配列で取得し，変数名とその変数を呼び出す関数を対応づける．

### 3.4　ポインタの更新

学習者の誤りの提示や代入文の作成を明示的にサポートするツールの作成のために，本研究ではポインタを題材として学習者の図の操作に対応してデータの更新と命令文の候補の出力を行う機能の実現を補助する API を提案する．先行研究 [1] のような，学習者自身が図のポインタの指す先を操作することで，操作に対応する命令文の候補を出力する機能とデータの更新をして図を更新する機能を実現するために，ツール側からポインタのデータの更新を行うためのメソッドとして Pointer_Object クラスに updatePointer メソッド，図の操作に対応する代入文の出力を行うためのメソッドとして Memory_Object クラスに getCodes メソッドを定義する．

updatePointer は変更したい Pointer オブジェクトの指す先の Memory_Object オブジェクトを引数に取り，指す先のアドレス value を指す先の Memory_Object のアドレスに変更する．updatePointer の実行後はポインタに指されていた変数と新たにポインタに指される変数の参照方法が更新される．getCodes は関数のフレーム frame を引数に取り，Memory_Object を参照するためのコードを文字列で返す．複数のコードで参照できる場合は，すべてのコードを返す．updatePointer メソッドと getCodes メソッドはツールが可視化する図を更新する機能に用いられることを想定している．

ツール実行中に学習者から更新したいポインタ変数とその指す先の変数を情報を受け取り，updatePointer メソッドを利用することで，ツール実行中に可視化している変数オブジェクトのデータを更新することができる．これにより，ポインタの値の変化やポインタが構築するデータ構造の変化を可視化することができる．また，getCodes メソッドで図の更新によるデータ構造の変化だけでなく，ポインタに指されていた変数や，ポインタに新たに指される変数の参照方法の変化についても理解支援ができる．

### 3.5　実行履歴表示ツールの実現

#### 3.5.1　ツールの概要

API の動作を確認するために，実行履歴を取得して表で出力するツールを作成した．自己参照構造体は配列，構造体，ポインタなどの主要なデータ構造やメモリ領域の動的な確保，解放などの C 言語学習において重要な要素を広く網羅しているので，自己参照構造体から成る二分探索木の操作をする C プログラムを対象として実験した．図 2 はツールの全体像である．ツールは画面左上にプログラムと図の操作，中央に実行履歴を可視化した表，右上に代入文の情報をそれぞれ示す．プラットフォームを利用して，二分探索木のノードの操作をするプログラムを可視化するツールを表として可視化する機能を実現する．表はグローバル領域の変数の情報を表示する global，ヒープ領域の変数の情報を表示する heap，スタック領域の関数と変数の情報を表示する func の 3 つから構成される．

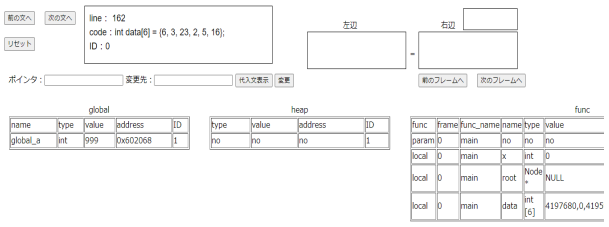


図 2　実行履歴表示ツールの実行画面

ツールでは次の操作が可能である．

- 可視化する実行履歴を任意の命令文の実行後のものにできる．
- ポインタと指す先のノードを指定し表を更新できる．

ツールは以下の手順を用いて情報の取得を行う．

1. getStatementByLine や getStatementByContent メソッドで可視化するプログラムの行番号，Statement オブジェクトを取得する．
2. Statement オブジェクトから各メモリ領域のオブジェクトを取得し，それらのオブジェクトが所持する Memory_Object オブジェクトを取得することで，表の作成に必要な関数や変数の情報を得られる．
3. プログラムの状態の進退は，Statement オブジェクトの getNextStatement，getPrevStatement で新しい Statement オブジェクトを取得することで実現される．

#### 3.5.2　ポインタの変更

ポインタの更新の例として図 3 の heap の表の ID2 のポインタの指す先を ID7 の変数に変更する操作を挙げる．更新したいポインタ変数の ID と新たな指す先の変数の ID を入力することによって表の操作ができる．ID を入力し，代入文表示のボタンを押すと，図 5 のように左辺に指す先を変更するポインタのコードを出力し，右辺に変更先のポインタの指す先の代入文の候補が出力される．また，代入文の候補は関数によって異なるので前のフレーム，次のフレームのボタンで左辺，右辺の代入文の候補を変更できる．変更ボタンを押すと，図 4 のように heap の表の value が変更される．変更された value は赤で強調される．

heap

| type | value | address | ID |
|---|---|---|---|
| int | 6 | 6303760 | 1 |
| struct node * | 6303792 | 6303768 | 2 |
| struct node * | 6303824 | 6303776 | 3 |
| int | 3 | 6303792 | 4 |
| struct node * | 6303856 | 6303800 | 5 |
| struct node * | 6303888 | 6303808 | 6 |
| int | 23 | 6303824 | 7 |
| struct node * | NULL | 6303832 | 8 |
| struct node * | NULL | 6303840 | 9 |
| int | 2 | 6303856 | 10 |
| struct node * | NULL | 6303864 | 11 |
| struct node * | NULL | 6303872 | 12 |
| int | 5 | 6303888 | 13 |
| struct node * | NULL | 6303896 | 14 |
| struct node * | NULL | 6303904 | 15 |

図 3　変更前の heap

heap

| type | value | address | ID |
|---|---|---|---|
| int | 6 | 6303760 | 1 |
| struct node * | 6303824 | 6303768 | 2 |
| struct node * | 6303824 | 6303776 | 3 |
| int | 3 | 6303792 | 4 |
| struct node * | 6303856 | 6303800 | 5 |
| struct node * | 6303888 | 6303808 | 6 |
| int | 23 | 6303824 | 7 |
| struct node * | NULL | 6303832 | 8 |
| struct node * | NULL | 6303840 | 9 |
| int | 2 | 6303856 | 10 |
| struct node * | NULL | 6303864 | 11 |
| struct node * | NULL | 6303872 | 12 |
| int | 5 | 6303888 | 13 |
| struct node * | NULL | 6303896 | 14 |
| struct node * | NULL | 6303904 | 15 |

図 4　変更後の heap

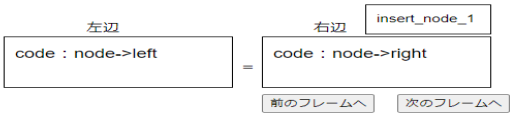


図 5　ポインタ 2 の指す先を 7 に変更する代入文の候補

## 4　考察

### 4.1　自己参照構造体で構築された二分探索木の図示

3.5 で扱った，自己参照構造体で構築された二分探索木の操作を行うプログラムに対して，表形式以外の可視化方法を検討する．図 6 と図 7 は，図 3 と図 4 の実行履歴を Graphviz[10] を用いて可視化している．ノードを丸，ポインタを矢印で表し，学習者の構築した二分探索木の構造を可視化することで学習者は意図通りの構造が構築されているか，任意のノードを参照する方法を確認できる．ここに，ポインタ更新の API である updatePointer や getCodes を用いて，図の操作と操作に対応する命令文の候補の出力を行う機能を実現することで，学習者はプログラムのポインタの指す先の誤りの修正が可能であると考えられる．

変数の実行履歴は，呼び出されている関数ごとに取得可能なので，任意の再帰関数からスコープできる変数のみを図示することで，木全体の表示だけでなく，再帰処理中の部分木のみを表示することも可能である．これにより，木構造が木全体から部分木へと帰納的に分割される様子を可視化することができ，再帰における部分問題化の考え方の理解につながると考えられる．

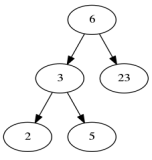


図 6　図 3 の図示

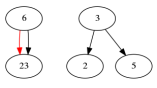


図 7　図 4 の図示

## 5　おわりに

本研究では C 言語を対象に学習者のプログラムの動作理解支援ツール作成における実行履歴を取得する処理の共通化と取得した実行履歴に対する API をプラットフォームとして提案した．API として，オブジェクトの関連を辿ることで実行履歴を取得する API や，図の操作によりポインタの指す先を更新する機能を補助する API，操作に対応する代入文の候補を出力する機能を補助する API を定義した．また，API の動作を確認するために，実行履歴を取得して表で出力するツールを作成した．

今後の課題として，ポインタ型以外の変数の更新のための API の実現，実行履歴データファイルの容量縮小が挙げられる．

## 参考文献


[1] 坂公博, 奥村航：自己参照構造体を用いるプログラムの理解支援ツールの提案，南山大学理工学部 2019 年度卒業論文.

[2] Juha, S., Ville, K., Lauri, M.：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.

[3] SeeC - program visualization and debugging for novice C programmers, available from `https://seec-team.github.io/seec/index.html`(accessed 2020-07-24).

[4] Matthew, H.E. and Chris, M.:Runtime error checking for novice C programmers. CSEIT ’13, pages 19, Singapore, 2013. Global Science & Technology Forum.

[5] Python Tutor -Visualize Code Execution, available from `http://pythontutor.com/`(accessed 2021-01-09).

[6] Philip, J.G.:Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education, SIGCSE ’13, March 2013, Pages 579584.

[7] Ishizue, R., Sakamoto, K., Washizaki, H., Fukazawa, Y.：PVC: Visualizing C Programs on Web Browsers for Novices, SIGCSE ’18, February 21-24, 2018, Baltimore, MD, USA.

[8] Pedro, M., Leopoldo, T.：Willow: A Tool for Interactive Programming Visualization to Help in the Data Structures and Algorithms Teaching-Learning Process, SBES 2019, September 2327, 2019, Salvador, Brazil.

[9] 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，pp143-150.

[10] Graphviz - Graph visualization software, available from `https://graphviz.org/`(accessed 2021-01-09).