1L-1

C 言語における記憶域期間と有効範囲を考慮したメモリの可視化

青木 史勇　　篠埜 功

芝浦工業大学 工学部 情報工学科

1. 研究背景と目的

プログラミング教育によく用いられる言語の一つに C 言語がある。しかしプログラミング初心者にとって C 言語の習得は敷居が高く、特にポインタの学習で理解に時間を要したり、具体的な使い方が分からない初学者は多い。

本研究では、C 言語で記述されたプログラムの実行中のメモリ空間を可視化することで初学者のポインタの学習を支援するツールを作成する。このツールで記憶域期間や有効範囲を考慮した可視化を行うことにより、有効範囲外だが生存している変数の参照ができたり、あるポインタの参照先の変数の生存期間が過ぎた場合、そのポインタは参照してはいけないといった事項の学習を支援することを目的とする。

2. ツールの概要

本研究で作成するツールの目的およびツールの構成について述べる。

2.1 ツールの目的

デバッガによる情報を用いて、C 言語プログラム実行中のメモリを可視化するツールを作成する。このツールは、プログラム実行中の静的領域、スタック領域およびヒープ領域上にある変数を、有効範囲を区別できる形で可視化する。

2.2 ツールの構成

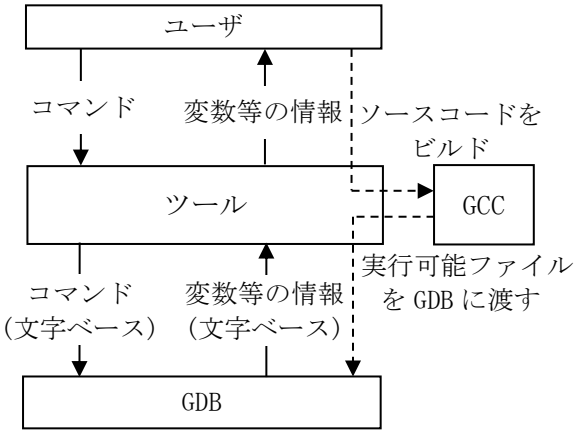

図 1. ツールの構成

ツールの構成を図 1 に示す。ユーザはソースコードをツール経由で GNU Compiler Collection (GCC)によってビルドする。ツールは作成された実行可能ファイルを GNU Debugger(GDB)によってデバッグし、GDB から出力される情報を基にメモリ内部の様子をグラフィカルに表示する。ユーザはボタンなどの GUI によって GDB にコマンドを送信したり、メモリ上に格納された変数の詳細な情報を閲覧することができる。

3. 実装

実装したツールの画面の構成及び可視化手法について述べる。

3.1 ツールの画面の構成

ツールは起動時に、デバッグするプログラムの C 言語のソースファイル名を要求する。このツールは分割コンパイルを部分的にサポートしており、最大で 3 つのソースファイルからなるプログラムを実行することができる。入力されたソースファイルのコンパイルが成功すると、ツールのメイン画面が起動する(図 2)。

メイン画面左上部分にはデバッグ中のプログラムのソースコードが表示され、次に実行される行が赤く表示される。

左下部分には GDB へコマンドを送信するボタンがあり、プログラムの実行開始やステップ実行を行うことができる。

画面右下部分はメモリ空間をグラフィカルに表示する部分である。

右上部分には、画面右下に表示されている変数が選択された際、その変数に関する詳細情報が表示される。

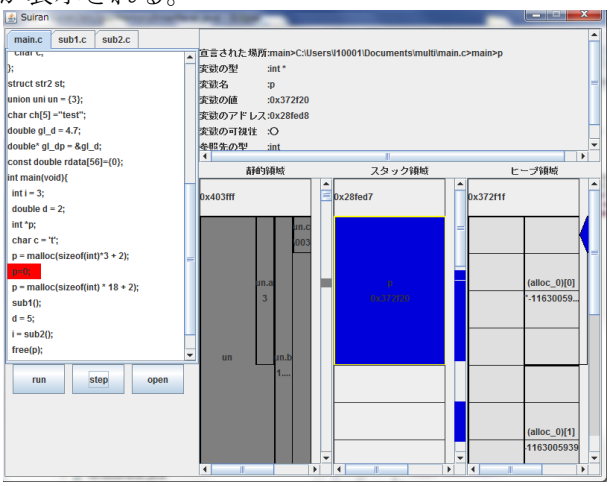

図 2. ツールの画面の構成

3.2 記憶域期間の表現

メモリ空間は静的領域、スタック領域、ヒープ領域の 3 つに分けて表示される。それぞれの

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.

メモリ領域は、変数のために確保されている領域の周辺のみが表示されるため、変数のための領域が確保される瞬間を捉えやすいようになっている。

3.3　変数の有効範囲の表現

変数は、パネルとして表示される。パネルは単色で塗りつぶされており、同一のブロック内で宣言された変数のパネルは同じ色で塗りつぶされる(図 3)。これにより、表示されている変数同士が同一の有効範囲を持つか否かが視覚的に分かるようになっている。

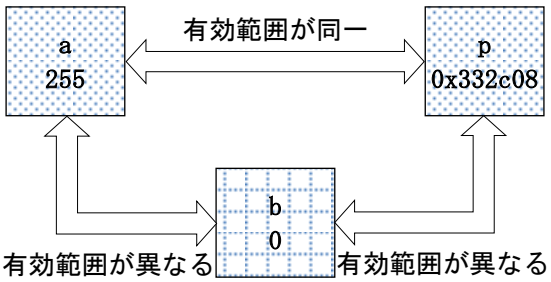

図 3. 色分けによる有効範囲の表現

3.4　変数の可視の表現

変数を表すパネルは、変数が可視であるとき明瞭に表示され、可視でないとき半透明で表示される(図 4)。これにより、再帰呼び出しによって同じ変数のための領域が複数割り当てられた場合も、変数の有効範囲が分かるようになっている。

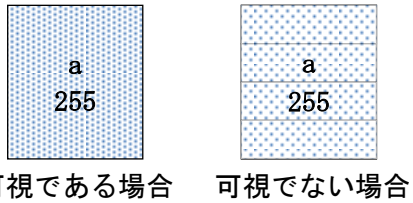

図 4. 透明度による可視の表現

3.5　ポインタの表現

プログラム実行中に生存しているポインタ変数が存在する場合、その参照先となる領域の右隣に、ポインタを表すパネルが表示される(図 5)。

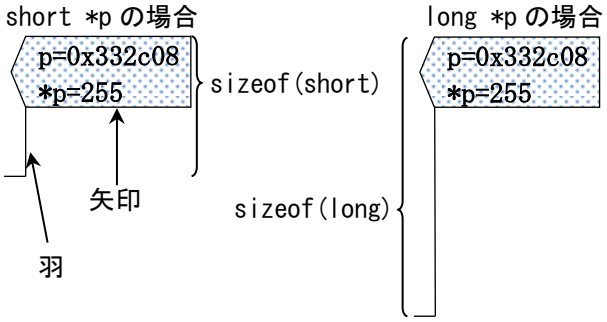

図 5. ポインタを表すパネル

パネルは矢印部分と羽部分に分かれている。矢印の先端はポインタの参照先を表し、矢印の胴体部分にはポインタ変数自身の値と参照先の値が表示される。羽の長さと矢印の高さを合計した長さはポインタ変数の参照先の型の大きさを表しており、ポインタの型と参照先の変数の型の一致、不一致を判断する目安になる。

4.　関連研究

メモリの可視化を行う研究として、メモリ空間を忠実に可視化した SuZMe[1]や、未定義の変数の区別や計算過程を可視化した佐賀らによる研究[2]などがある。これらと本研究の可視化手法を、○:サポートしている、△:一部サポートしている、-:サポートしていないの 3 段階で評価したものを表 1 に示す。本研究は変数の有効範囲や記憶域を明確に可視化する機能がある点で優れていると考えられる。

表 1. 関連研究との比較

	メモリ可視化	未定義可視化	有効範囲可視化	記憶域可視化
SuZMe	○	-	-	-
佐賀らによる研究	△	○	-	-
本研究	△	-	○	○

5.　まとめと今後の課題

本研究では、記憶域期間や有効範囲を考慮した C 言語プログラム実行中のメモリ状態の可視化ツールを作成した。

今後の課題として、実際にプログラミング初学者にツールを利用してもらい、ツールの機能や可視化の表現方法の変更、改善を行うことが考えられる。また、本ツールがデバッグ対象とするプログラムには多くの制約があり、以下のような点については改善が必要である。

(1) 有効範囲の区別はプログラム中に出現するすべての局所変数が関数の先頭でのみ宣言されていることを前提としており、入れ子になったブロック内で宣言された局所変数があると正常に動作しない。

(2) malloc 関数,calloc 関数,free 関数を除く C 言語の標準ライブラリに対応していない。

参考文献

[1] 小池伸弥, 郷健太郎: SuZMe：空間的なアドレス表現を用いたプログラム理解支援ツール, 電子情報通信学会論文誌 D, Vol. J95-D, No. 3, pp. 518-526.

[2] 佐賀誠司, 平嶋宗, プログラミングにおけるメモリ処理に関する誤りの可視化機能, JSiSE2010 学生研究発表会, pp. 223-226.

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.