4C－5

プログラムの構造と動作の理解支援*

本間 昭次　清水 洋子　深谷 哲司†
株式会社 東芝　研究開発センター　S&S 研究所‡

1 はじめに

既存のプログラムの改造や不具合の修正，再利用をする場合，プログラムの理解が必要になる．そのために，仕様書などの文書を参照する，ソースコードを読む，プログラムの動作を確かめる，といったことを行う．

詳細設計について文書を参照しようとすると，必ずしも最新情報に更新されているとは限らず，結局，ソースコードを見ることになる．

ソースコードを読む場合，特に膨大な量のソースコード全部に目を通すことは難しい．そこで，まず構造を理解しようとする．C 言語では，プログラムの構造を見るために関数のコールグラフが使われるが，コールグラフはグローバル変数が考慮されていないため，グローバル変数を介した関数間の関係を見逃す可能性がある．

プログラムの構造がある程度分かったら，次に手続き呼出しがどのような順序で行われるかといった動作を理解しようとする．動作を理解するためには，プログラムを実際に動かすか，または動きをシミュレートしながら，内部の様子を調べることが必要になる．内部の様子を見るためにブレークポイントを指定してプログラムを停止させ，内部を覗くという方法が主にデバッガに見られる(例えば，UNIX の gdb)．しかし，この方法で得られる情報は断片的な情報で，手続き呼出しの順序といった動作の流れを理解するのには向いていない．

本稿では，プログラムの理解を構造と動作の二点から支援し，上記の問題を解決する方法を提案する．まず，プログラムの構造についてグローバル変数を含めて表現する方法について述べる．次に，プログラムの動作をログに記録し，視覚化する方法について述べる．なお，対象言語はC言語とする．

2 構造理解の支援

プログラム構造の理解を支援する手法として，関数とグローバル変数の関係を表す方法を提案する．提案する方法では，どの関数がどの関数を呼出しているか(どの関数に呼び出されているか)，どの関数がどのグローバル変数の書込みや参照をしているか(どの関数に書込みや参照をされているか)，を木構造で表現する(以後，この表現方法を関数・グローバル変数グラフと呼ぶことにする)．これは関数のコールグラフまたはコールドグラフにグローバル変数の情報を付加したものとも言える．この表現方法により，これまで関数のコールグラフだけでは分からなかったグローバル変数を介した関数のつながりを知ることが可能となる．

関数・グローバル変数グラフでは，関数とグローバル変数について，「使用する」または「使用される」という観点からつながりをプログラムから抽出し，再帰的に表現する．「使用する」，「使用される」がどのような場合を表すかを表 1 に示す．

表 1 使用する，使用されるの意味

	意味	記号
A が B を使用する	関数 A が関数 B を呼び出す	A{　B
	関数 A がグローバル変数 B の値を変更する	A{→ B
	関数 A がグローバル変数 B を参照する	A{← B
	関数 A がグローバル変数 B の参照と変更をする	A{⇔ B
A が B に使用される	関数 A が関数 B の中で呼び出される	A　}B
	グローバル変数 A が関数 B の中で値を変更される	A ←}B
	グローバル変数 A が関数 B の中で参照される	A →}B
	グローバル変数 A が関数 B の中で参照と変更をされる	A ⇔}B

図 1 (1) に示したサンプルプログラムに対する関数・グローバル変数グラフによる表現方法を図 1 (2), (3) に示す．関数と変数を区別するために，関数には後ろに()を付けた．また，関数や変数の間の関係を矢印と中括弧を組み合わせた記号で表現した．記号の意味を表 1 に示す．

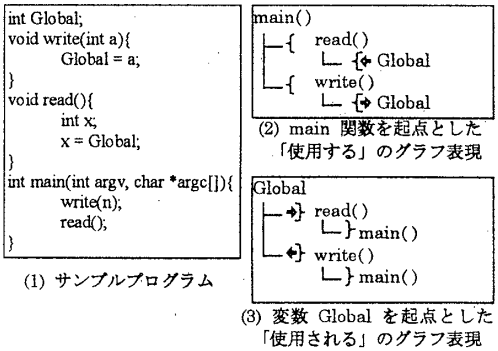

図 1　関数・グローバル変数グラフの例

* Understanding Program Structure and Behavior
†Akitsugu HOMMA, Yoko SHIMIZU, Tetsuji FUKAYA
‡Systems & Software Research Laboratories, Research Development Center, Toshiba Corporation

3 動作理解の支援

プログラムの関数とグローバル変数の静的なつながりを視覚化することによってプログラムの構造を理解したら，次にプログラムの動的な理解が必要になる．

プログラムの動的な理解には，プログラムを動作させたときの，ある時点での内部の状態とその変化の様子を知ることが有効である．

内部の様子を見る方法として，主にデバッガではブレークポイントを設定して，その時の内部の状態を覗く．しかし，それは断片的な情報の切り出しであり，プログラムの動作を流れとしては捉えにくい．

それに対し，プログラムの実行過程をログとして記録し，後からログを参照すれば，プログラムの状態が変化する様子を時系列に表示することが可能となる．さらに，ログを元にプログラムを逆方向にトレースすることも可能となる．また，並列プログラムに対する動作もトレースしやすい．欠点としては，すべての動作について細かくログを取ると，動作が遅くなり，ログの量も膨大になることなどが挙げられる．

本稿では，ログの量を抑えて記録し，プログラムの動作を視覚化して，動作理解を支援する方法について提案する．

3.1 動作理解支援支援

出力するログを絞り込み，かつ必要な情報を取り出すために，スライシング技術[1]を応用する．ユーザが着目する箇所に対する依存関係を調べ，依存関係がある箇所についてだけログ出力命令をソースコード中に挿入する．このソースコードをコンパイルして実行することにより，ログが出力される．出力されたログはログビューアで表示する(図 2)．

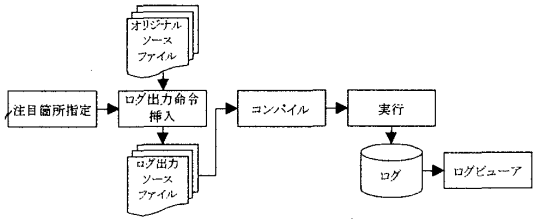

図 2　動作理解支援

3.2 ログ出力命令挿入

まず，ユーザは動作を調べたい箇所の変数や関数をソースコード上で指定する．次に，指定された箇所に対して依存関係がある箇所を探索する．この際，スライシング技術を利用する．最後に，探索された箇所についてログを出力する命令を挿入する．出力するログの内容を表 2 に示す．ログは，変数については参照と代入がなされたときに，関数については呼び出しと復帰のときに出力する．

表 2 出力するログの内容

変数	代入	位置(ファイル名，行) 変数名 ポインタ 代入された値とそのサイズ
	参照	位置 変数名
関数	呼出	位置 関数名 関数ポインタ 戻り値とそのサイズ
	復帰	位置 関数名 関数ポインタ

3.3 ログビューア

ログ出力命令を挿入したソースファイルをコンパイルして実行することによって得たログを見るために，ログビューアを用意する．ログビューアは，関数・グローバル変数の動的関係図，変数の値や関数の戻り値，ソースコードの三つのウィンドウからなる(図 3)．

関数・グローバル変数の動的関係図は，関数とグローバル変数について呼び出し，復帰，書込み，参照といった関係について，時系列に沿って表示する．関数・グローバル変数の動的関係図上のある時点に対応する変数の値や関数の戻り値，ソースコードを表示する．また，ソースコードに対して，ブレークポイントを設定して，そこまで実行したときの関数・グローバル変数の動的関係図や変数の値，関数の戻り値を表示する．

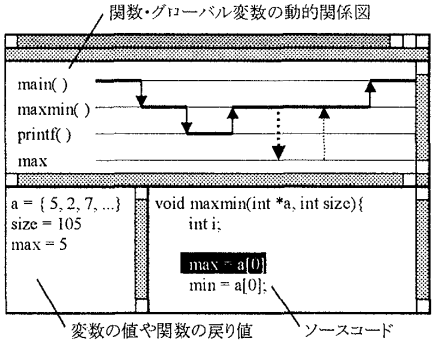

図 3　ログビューア

4 おわりに

本稿ではプログラムを理解するため，構造と動作という二つの側面からプログラムを視覚化する方法を提案した．構造については，関数とグローバル変数のつながりを表現し，理解を支援する．動作については，構造からだけでは分からない変化の様子を，関数とグローバル変数の関係を時系列に表示し，それに対応した変数の値やソースコードを表示することによって，理解を支援する．また，動作の記録を取る際，スライシング技術を応用し，出力するログの量を抑えて実行速度の低下やログの量の膨大化を抑え，かつ必要なログを取ることを提案した．

参考文献

[1] 下村 隆夫，プログラムスライシング技術と応用，共立出版，1995．