4F−4

言語Cプログラム読解支援ツール

繁田英之，藤原泰行，玉木裕二，並木美太郎，高橋延匡
東京農工大学　工学部　電子情報工学科

1．はじめに

ソフトウェアのライフサイクルにおいて，保守は非常に大きな比重を占めている．当研究室でも，研究の引継やプログラムのバージョンアップのためのプログラムの保守が問題となっている．それらのプログラムは，主に言語 C で書かれており，総行数は，20 万行をこえている [1]．

プログラムを読むときには，一度紙に出力する場合が多い．紙でプログラムを読む場合には，(1) プログラムの読解に必要な書込みを行うことができる，(2) 何枚にも分けて出力することが可能なので，関数などを対応させながら読むことができる，という利点がある．しかし，(1) 出力されるプログラムの量が多くなればなるほど，見たい情報を検索する手間も増えてしまう，(2) プログラムを直接，訂正・編集することができない，という欠点もある．そこで，それらの欠点を補うために，コンピュータを用いることを考えてみた．本稿では，我々の研究室内での利用を想定した言語 C プログラム保守（読解）支援環境と，プロトタイピングしたツールについて述べる．

2．プログラムを読むときの問題点

言語 C のプログラムを読んでいるときに生じる問題点として次の 3 点を経験した．

(1) 各ファイル内に存在している識別子の定義とその参照位置が把握しづらい．

(2) プログラム中のマクロ名が，どんな値を持っているのかがすぐに分からない．

(3) 構造体などの複雑なデータ構造をプログラム中のメンバ名から理解しづらい．

これらの問題点は，特にプログラムの規模が大きくなったり，分割コンパイルのために定義場所を探すことが困難になるために起こると考えられる．

3．言語 C プログラム読解支援ツールの設計

3．1　ユーザに与える情報

言語 C プログラム読解支援に必要と考えられる情報は，

(1) 識別子（マクロ，変数，関数，typedef した型名，タグ名）の定義実体

(2) 関数の参照関係

である．

3．2　言語 C プログラム保守（読解）支援の全体構成

言語 C プログラム保守（読解）支援の全体構成を次の図 1 に示す．

各部分の説明を次に記す．

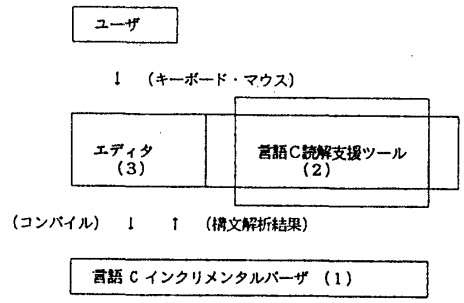

図 1 言語Cプログラム保守（支援）の全体構成

(1) 言語 C インクリメンタルパーザ

言語 C インクリメンタルパーザは，構文解析結果（解析木）をメモリに保持し，この解析木を再利用し，任意の部分を構文解析することができる．また，その解析木と，ソースプログラムとの対応情報を持っている [2]．今回設計したツールでは，この構文解析結果を利用する．

(2) 言語 C プログラム読解支援ツール

言語 C プログラム読解支援ツールは，次のような機能を持つ．

(a) 言語 C インクリメンタルパーザの作成した構文解析木を基に識別子の定義実体位置，関数の参照関係のための表を作成する．

(b) エディタのバッファから，文字列を切り出す．

(c) 切り出された文字列と表とマッチングさせる（表の検索）．

(d) ユーザから要求された情報を表示する．

(3) エディタ (ALTHEA) [3]

エディタは，基本的にはソースプログラムの編集のみに用いる．また，言語 C プログラム読解支援ツールが画面上から文字列を切り出すために，ツールとエディタのバッファを共有する．

4．言語 C プログラム読解支援ツールのプロトタイピング

4．1　プロトタイピングの方針

上記の情報が，有効であるかどうかを調べるために，まず，ツールをプロトタイピングする．プロトタイプで用意する機能を，(1) まず 3.1 の問題点 (1) で指摘した識別子の定義実体や関数の参照関係を表示させる，(2) 操作性を向上させるためにマウスを使用する，とした．したがって，プログラムの編集機能は設けない．あくまでも，識別子の定義実体，関数の参照関係を見せ，その有効性を確かめることを目的とする．

A Supporting Tool for Reading Language C Programs
Hideyuki SHIGETA, Yasuyuki FUJIWARA, Yuji TAMAKI, Mitarou NAMIKI and Nobumasa TAKAHASHI
Tokyo University of Agriculture and Technology

識別子名	ファイル名	型	所属関数名	初め	終わり
getline	test.c	3	NIL	15	22
・	・	・	・	・	・
・	・	・	・	・	・

型：0−マクロ
1−変数
3−関数
4−typedef 型名
5−構造体タグ名

図 2.1 定義実体格納表

関数名	ファイル名	範囲始め	範囲終わり
getline	test.c	15	30
・	・	・	・
・	・	・	・

図 2.2 関数の有効範囲表

4．2 プロトタイプの内部仕様

識別子の定義実体や，関数の参照関係の表示を実現するために次の 4 種類の表を用意する．

まず，定義実体を格納するためには，次の 2 種類の表を用意する．

(a) 定義実体格納表（図 2.1）

(b) 関数有効範囲表（図 2.2）

この 2 つの表で定義実体を管理する．定義実体表の“ファイル名”，“初め”，“終わり”は，識別子名のファイル内での定義実体場所を表している．関数有効範囲表は，変数名が重なったときに必要となる．マウスで指定した変数名の位置を読みとり，その位置からどの関数にいるのかを判断させ，適切な定義実体を表示できるようにする．

また，関数の参照関係を格納するために，次の 2 種類の表を用意する．

(c) 関数名格納表（図 3.1）

(d) 参照関係格納表（図 3.2）

表番号	関数名	型	ファイル名	自分を参照している関数へのポインタ	自分が参照している関数へのポインタ
0	main	0	test.c	NIL	・ →※
1	getline	1	test.c	NIL	NIL
2	write	1	test.c	NIL	NIL
3	outpout	1	test.c	NIL	NIL
4	NIL	NIL	NIL	NIL	NIL

型：0−ファイル内ローカル関数
1−グローバル関数

図 3.1 関数名格納表

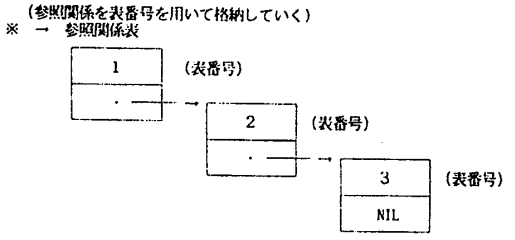

図 3.2 参照関係格納表

この 2 つの表で関数の参照関係を管理する．“ファイル名”は，関数が存在しているファイル名を表す．“型”は，関数がファイル内ローカル関数か，ファイル内グローバル関数かどうかを示す．関数名格納表から参照関係格納表へは，ポインタでリンクしており，参照関係格納表には，関数名が格納されている表の番号を格納する．

4．3 使用例

実際のプロトタイプの使用例を図 4 に示す．使用法は，識別子の名前にマウスのカーソルを合わせ，定義実体を見るときは左ボタンをクリックし，関数の参照関係を見たいときには，自分を参照している関数の一覧は左ボタンを，自分から参照している関数の一覧は右ボタンをそれぞれ，ダブルクリックする．図 4 は，自分を参照している関数の一覧表を見たところである．

5．おわりに

本稿では，言語 C プログラム保守（読解）支援のための全体構成とプロトタイプしたツールの設計について述べた．今後この設計をもとに，ツールを開発し，使用経験をフィードバックさせて，より使いやすい言語 C プログラム保守（読解）支援環境を目指す予定である．

参考文献

[1] 森田雅夫他 4：“言語 C の CAI システムの設計とそのプログラミング実行環境の開発”，シンポジウム報告書集 pp.177-186，情報処理学会，1991.1

[2] 玉木裕二：“OS/omicron における言語 C インクリメンタルパーザの基本設計”，情報処理学会第 40 回全国大会，1990，3

[3] 小林伸行，並木美太郎，高橋延匡：“OS/omicron 上の日本語エディタ”，情報処理学会第 38 回全国大会 1988，3

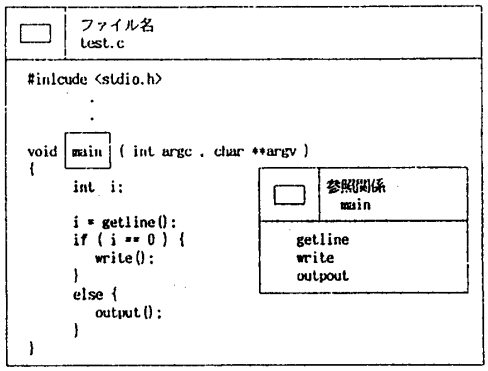

図 4 ツールの使用例