言語Cプログラムの読解を助ける出力ツール

3C−6

藤原泰行，繁田英之，玉木裕二，並木美太郎，高橋延匡
(東京農工大学)

1．はじめに

プログラムの保守が必要な現場では，効率よく正確に作業を行うためのプログラムに関する文書が重要な役割を担っている．そのために，当研究室でもプログラムの文書化を支援するための研究が行われてきた．これらは，クロスリファレンスなど，プログラムに関する情報を抽出，整理して紙へ出力するものであった．しかし，プログラムを記述しているコード群はただのテキストとして印字されていた．そこで，本研究では言語Cのソースプログラムを読みやすく加工し，プログラムの読解を助けることを目的とする．これはいわゆる Pretty Printer 的なことであるが，単純にプログラムを「整形」するだけではなく，積極的に「加工，変換」を試みようとするものである．

2．プログラムの「読みにくさ」の原因

プログラムの「読みにくさ」の原因は，大別すると次に示す2つの側面から考えることができる．

(1) プログラム自身に依存するもの
- ①データ構造やアルゴリズムが複雑，高度で，静的に解析すること自体難しいもの
- ②アルゴリズムの実現の仕方（プログラミング）に問題があるもの

など．

(2) プログラムの出力形式（字面）に依存するもの
- ①言語の構文，仕様
- ②コーディングスタイル
- ③紙への出力のしかた

など．

本研究は上記の (2) に関するものである．

2．1　言語Cの構文，仕様

言語Cの構文や仕様における読みにくさの原因を挙げる．

(1) 文字列の種類

Cのプログラムでは，データ構造やアルゴリズムを記述するコード，プリプロセッサ命令，コメント文が混在している．読みやすさの点からは，区別しやすい状態であってほしい．

(2) 括弧
- ①ブロック構造の表現がインデントと中括弧 {} しかないので，プログラムを読む段階でわかりやすいとは言えない．
- ②Cのプログラムでは丸括弧 () が多用されるので，括弧の対応をとりにくい．

(3) 演算記号

Cには豊富な演算子が用意されているが，違う演算にもかかわらず同じ記号を共用している場合があり，紛らわしい．

2．2　コーディングスタイル

コーディングスタイルの違いから感じる読みにくさを考える．

(1) 中括弧の位置

言語Cではブロックを中括弧 {} で表すが，括弧を書く位置がプログラマによって違う．中括弧はプログラム中に多く現れるので，括弧の対応を誤って理解する場合が考えられる．

2．3　紙への出力のしかた

プログラムをただのテキストとして出力した場合の読みにくさを考える．

(1) トークンについて

プログラム中のトークンは，識別子，予約語など数種類ある．しかし，一般的に，プログラムの出力の際はすべての文字は同じ扱いを受けている．読み手のために各種トークンは区別されやすい状態であってほしい．

(2) 文の折返し

紙の幅には限界がある．したがって，文が折り返され，次行の1カラム目から印字されることがある．頻繁にあるとプログラムの見通しを悪くする．

(3) 改ページ

文の途中で改ページされると，トレースを円滑に進めることが難しくなる．ループの途中の場合はページ間を行き来せねばならない．また，括弧の対応も取りづらくなる．したがって，動作を誤って理解する可能性も高い．

3．読みやすさ向上のために

ここでは，前章で述べた読みにくさの例を解消する改善策を考え，プログラムの読みやすさの向上をねらう．

3．1　言語Cの構文・仕様に関して

文字フォントを変える，補助線を使うなど，Cの構文や仕様を補うことで，読みやすさの向上をめざす．

(1) 文字列の種類に関して

プリプロセッサ命令の文字フォントを変え，他のコードとの区別をしやすくする．また，コメント文は "/*"，"*/" を取り除き，網掛け文字にする．コントラストによって，他の文との区別をつけやすい．

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

(2) 括弧に関して

①インデントと縦方向の補助線を使うことで，ブロックの範囲を明確にする．

②意識する必要性の低い，選択文，繰返し文の丸括弧を取り除き，中の文を斜体で印字する．また，「条件文」に下線を引いて目立たせる．

(3) 演算子に関して

演算子の文字フォントを変え，一部の演算子は記号も変換する（表１）．

表１．変換規則

変換前	変換後
&&	and
||	or
間接演算子 *	上付きの *
*	←
<=	≦
>=	≧
!=	≠

３．２　コーディングスタイルに関して

プログラムの管理，保守を複数人で行う場合，コーディングスタイルが統一されていると読解作業の効率が高くなる．したがって，入力されるスタイルを問わず，出力系が統一スタイルを提供する．

３．３　紙への出力のしかたに関して

文を整形するなど，スタイルを統一し，プログラムの体裁を整えることで読みやすさの向上をめざす．

(1) トークンについて

トークンによって，次のような出力をする．

・予約語　　：　斜体強調

・識別子

　マクロ名　：　フォントを変える

　typedef名　：　強調

　タグ名　　：　強調

　関数名　　：　定義位置で文字サイズを大きくする

(2) 文の折返し

文を折り返す必要がある場合は，演算順位の低い箇所で文を折り，インデントを行って印字する．

```
#include <stdio.h>

#define   IN   1   /*    単語の中    */
#define   OUT  0   /*    単語の外    */

/*   入力中の行，単語，文字のカウント   */
main()
{
     int  c, nl, nw, nc, state;

     state = OUT;
     nl = nw = nc = 0;
     while ((c = getchar()) != EOF) {
          ++nc;
          if (c == '\n')
               ++nl;
          if (c == ' ' || c == '\n' || c == '\t')
               state = OUT;
          else if (state == OUT) {
               state = IN;
               ++nw;
          }
     }
     printf("%d %d %d\n", nl, nw, nc);
}
```

ソースプログラム

(3) 改ページ

関数定義の区切りがよいところで改ページすることが考えられる．ひとつの関数が大きく，１ページに収まらない場合は，制御文の切れ目で改ページする．

４．出力の例

図１に変換したプログラムの例を示す．この例では関数定義から関数名，引数，戻り値を抽出し，関数のヘッダにしている．関数名が目立つので探しやすい．

また，関数定義の部分に「その関数が参照する関数」と，「その関数を参照する関数」の名前と定義位置の情報を付加する．ソースとは別にクロスリファレンスを用意する場合よりも参照しやすい．

５．今後の課題

今後も検討すべき点として，

(1) 変換する記号と変換に使用する記号

(2) 使用する文字フォントとフォントの組合わせ

(3) 全体のレイアウト

を考えている．筆者のまわり人々に意見を求め，参考にしながら検討を続けるつもりである．

６．おわりに

本稿では言語Ｃのプログラムを加工し，読みやすくすることによりプログラムの読解を助ける試みについて述べた．「読みやすさ」の客観的な尺度を導入するかわりに，実際にプログラムを読んで感じた不便な点や不満な点を改善するというアプローチをとった．したがって，個人の感覚，視点に捕らわれたところがあり，今後も考察を重ねる必要があるだろう．

```
#include <stdio.h>

#define   IN   1          単語の中
#define   OUT  0          単語の外

   入力中の行，単語，文字のカウント

ENTRY
関数名：main
引数　：なし
戻り値：int
この関数が参照する関数：printf
この関数を参照する関数：なし

     int  c, nl, nw, nc, state;

     state = OUT;
     nl = nw = nc = 0;
     while (c = getchar()) != EOF
          ++nc;
          if c == '\n'
               ++nl;
          if c == ' ' or c == '\n' or c == '\t'
               state = OUT;
          else if state == OUT
               state = IN;
               ++nw;

     printf("%d %d %d\n", nl, nw, nc);
```

変換結果

図１．変換例