清書プログラム†

杉原　厚吉††

1. はじめに

清書プログラムとは，入力データをあらかじめ指定した形式に変換して出力するプログラムである．たとえば，LISP の算譜を入力するとカッコの“深さ”に従って左カッコの位置をタテにそろえた見やすい形式に書き直したり，記事の原稿を入力すると新聞の枠に合わせて自動植字機を動かしたりするためのプログラムがその例である．しかし，これらの例は用途が特殊であったり入出力装置が大がかりなものであったりするため，あまり汎用性はない．これに対して，英文を清書するためのプログラムは，タイプライタが安価な出力装置として利用できるため広く使われるようになってきている．

英文の清書プログラムの一つに，RUNOFF と名付けられているプログラムがある．以下はこのプログラムの紹介とそれを使ってみた感想である．これは，著者が限られた場面——電総研の情報処理用計算機 TOSBAC 5600 の TSS システム——で限られた目的——投稿論文や手紙の清書——のために使用したつたない経験にもとづいたものであり，世の中にある清書プログラムのすべてを解説するにはほど遠いものであるが，清書プログラムの便利さと問題点の大筋がわかっていただければ幸いである．

2. システム構成

電総研では，図-1に示す構成で清書プログラム RUNOFF（以後，単に清書プログラムと呼ぶ）を使用している．

英文原稿の入力と修正を行うためのキャラクタディスプレイ端末，ディスプレイ画面を固定するためのハードコピー装置，原稿を記憶するためのファイル，清書原稿を出力するためのタイプライタ，それらを制御するための計算機から成り，これらを次の順序で使用する．ユーザは，まずキャラクタディスプレイ端末からエディタを介して英文原稿を入力し，ファイルに記憶する（これは，印刷作業の工程の中では，原稿用紙に文章を書くことに相当する）．このとき，清書形式を指定するための制御情報も英文原稿の中にはさみながら入力する．次に，この原稿を清書プログラムに入力し，その出力をディスプレイ画面に出してハードコピーをとる（ゲラ刷りを作ることに相当する）．もし必要があれば，ファイル中の英文原稿を修正する（ゲラの校正作業に相当する）．これを，修正の必要がなくなるまでくり返し，最後に，清書プログラムの出力をタイプライタに打出す（印刷に相当する）．

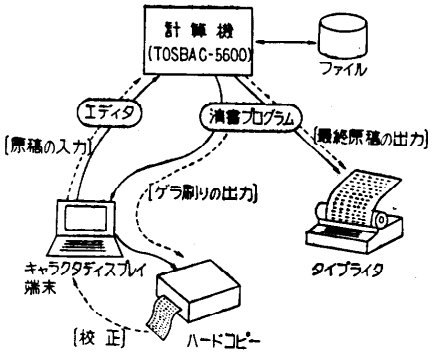

図-1　システム構成

3. 使用例

清書プログラムの基本的機能を例を使って説明しよう．

図-2（a）の原稿を清書プログラムに入力すると，図-2（b）に示す出力が得られる．入力（a）のうち，ピリオド“.”で始まる行は清書プログラムへ指示を与えるための制御情報，その他の行は英文原稿である．この例において最初の5行は全体の清書形式を指定するための大域的な制御情報である．まず，1ページが32行から成り（1行目），1行は50文字から成

† Program for Pretty Printing by Kokichi SUGIHARA (Information Sciences Division, Electrotechnical Laboratory).

†† 電子技術総合研究所

```
.PAPERLENGTH 32                          ┐
.LINELENGTH 50                           │
.REPLACE @                               ├ 大域的制御情報
.ALTERNATE %                             │
.JUSTIFY                                 ┘
.CENTER 1 ..... 局所的制御情報
Summary ..... 英文原稿
.SPACE 1 ..... 局所的制御情報
@@@@@We propose a preorder tree which    ┐
satisfies an order relation: Keys        │
placed at nodes appear in lexicographic  ├ 英文原稿
order, when the tree is traversed in     │
preorder.                                ┘
.BREAK ..... 局所的制御情報
@@@@@Algorithms for search, insertion    ┐
and deletion are given.                  │
%The number of key comparisons in the    ├ 英文原稿
search is investigated theoretically     │
and by simulation studies and compared   │
with the case of binary search trees.    ┘
.BREAK ..... 局所的制御情報
@@@@@We also give an algorithm for      ┐
constructing a preorder tree which       │
minimizes the number of key comparisons. ├ 英文原稿
%Simulation experiments confirm the      │
effectiveness of optimization.           ┘
.BREAK ..... 局所的制御情報
```

図-2 清書例1 (a) 入力

We propose a preorder tree which satisfies an order relation: Keys placed at nodes appear in lexicographic order, when the tree is traversed in preorder.

Algorithms for search, insertion and deletion are given. The number of key comparisons in the search is investigated theoretically and by simulation studies and compared with the case of binary search trees.

We also give an algorithm for constructing a preorder tree which minimizes the number of key comparisons. Simulation experiments confirm the effectiveness of optimization.

清書例1 (b)力出

る(2行目)ということが指定される．3行目(.REPLACE @)は，原稿中の“@”をすべて1文字分の空白に置替えることを指定する．4行目(.ALTERNATE %)は，原稿中の“%”が清書において左端に来たときは削除し，それ以外のときは1文字分の空白で置替えることを指定する(この指定の目的はすぐあとで明らかになる)．5行目(.JUSTIFY)は，単語間の空白の長さを調整することによって清書の右端をそろえることを指定する．

6行目以降の制御情報は，局所的な清書形式を指定するためのものである．6行目(.CENTER 1)は，“次の1行を左右の中央に打て”という制御情報であり，8行目(.SPACE 1)は，“ここに空白行を1行入れよ”という制御情報である．

9行目からは英文原稿が続くが，9行目の“@@@@@”は，3行目の制御情報と呼応して，パラグラフ

```
.BEGINPAGE
.CENTER 1
References
.SPACE 1
.INDENT 5
.UNDENT 4
[1]@M. Hoshi and T. Yuba: Searching in
preorder trees and optimum tree
construction,
.SCOREUNDER 1
Trans. IECE of Japan,
Vol. 58-D, No.4, p. 217 (April 1975).
.BREAK
.UNDENT 4
[2]@D. E. Knuth:
.SCOREUNDER 1
The art of computer programming, Vol. 1,
2nd ed., Addison-Wesley (1973).
.BREAK
.UNDENT 4
[3]@D. E. Knuth:
.SCOREUNDER 1
The art of computer programming, Vol. 3,
Addison-Wesley (1973).
.BREAK
.UNDENT 4
[4]@M. Miyakawa, T. Yuba, Y. Sugito and
M. Hoshi: Optimum sequence trees.
.SCOREUNDER 1
SIAM J. Comput.,
Vol. 6, No. 2, p. 201 (1977).
.BREAK
```

図-3 清書例2 (a) 入力

[1] M. Hoshi and T. Yuba: Searching in preorder trees and optimum tree construction, Trans. IECE of Janan, Vol. 58-D, No.4, p. 217 (April 1975).

[2] D. E. Knuth: The art of computer programming, Vol. 1, 2nd ed., Addison-Wesley (1973).

[3] D. E. Knuth: The art of computer programming, Vol. 3, Addison-Wesley (1973).

[4] M. Miyakawa, T. Yuba, Y. Sugito and M. Hoshi: Optimum sequence trees. SIAM J. Comput., Vol. 6, No. 2, p. 201 (1977).

清書例2 (b) 出力

の先頭に5文字分の空白を入れる効果をもたらす．また，パラグラフの先頭以外の各文の頭の“%”は，4行目の制御情報と呼応して，一般には文と文の間に2文字分の空白を入れる（ただし，右端をそろえるために空白がさらにふえることもある）が，ちょうど文の切れ目で改行するときは空白が入らないようにする，という効果をもたらす．また，14行目（.BREAK)は，ここで改行することを指定するもので，その結果，その直前の行では右端をそろえることはしない．

これらの制御情報の効果は，図-2（b）の清書結果と照らし合わせることによって理解いただけるであろう．英文原稿の中に挿入する制御情報は，初めての人には煩雑に感じられるかもしれないが，少し慣れると苦にはならなくなる．また，ここで注目すべき点は，清書したい紙面の大きさ等に変更があるときは，たいていは大域的制御情報——すなわち入力原稿（a）の最初の5行——を変更するだけで新しい形式に合わせることができるという点である．

入力原稿のもう一つの例と清書プログラムによるその出力を，それぞれ**図-3**（a）と（b）に示す．ただし，図-3（a）は図-2（a）に続く原稿の別の部分であり，したがって図-2（a）の最初の5行の大域的制御情報がそのままここでも効力を持っているものとする．

図-3（a）の1行目（.BEGINPAGE）は，ここで改ページすることを指定する．5行目（.INDENT 5)は，以下の原稿は左に5文字分の空白を設けて打出すことを指定する．6行目（.UNDENT 4）は，次の行の始まりは左へ4文字分はみ出して打つことを指定する．また，10行目（.SCOREUNDER 1）は，次の1行に下線をほどこすことを指定する．以下，同様である．これらの制御情報によって図-3（b）の清書結果が得られる．

以上の例は，清書プログラムの基本的機能を理解していただくために示したものである．このほかにも，紙面にページをふったり脚注を入れたり等の機能があるが，ここでは省略する(1)を参照)．なお，清書プログラムに指定を与えるための制御語は，最初の4文字だけで代用できる——たとえば，“.PAPERLENGTH 32”は“.PAPE 32”で代用できる．

4. 便 利 な 点

清書プログラムを使用してみて便利だと感じた点を以下に列挙する．

1) 修正が収束する．これは，清書プログラムを利用するためより，むしろ計算機のファイルに原稿を記憶しておき，その内容をエディタで自由に変更できることの効用である．
2) 大域的な形式変更が容易である．1ページの行数や1行の文字数の変更は，大域的制御情報をそれぞれ1カ所変えるだけで実現できる．たとえば国際会議等で研究発表をする場合には，査読用にA4判の原稿を作って投稿し，査読を通過した場合には予稿集の写真製版用のB4判の用紙に原稿を打ち直さなければならないことが多いが，このような形式の変更が簡単にできる．
3) 行の右端がそろうから，見た目にきれいである．
4) 清書原稿のレイアウトが対話的にできる．清書原稿の途中に図を入れるための余白を配置したり，原稿の長さを調整したりするために，清書プログラムの出力をディスプレイ端末に出し——すなわち，ゲラ刷りを作り——，ユーザがそれを見ながら制御情報を変更できる．

5. 不 満 な 点

何でも使っていると欲が出てくるものだが，清書プログラムの場合にも次のようなぜいたくな不満を抱くようになる．

1) タイプボールの取替えがめんどうである．現在のシステムで特殊文字を使いたいときには，“そこでタイプライタをいったん止める”という制御情報を入力原稿の中に入れておき，ユーザが手作業でタイプボールを取替えてからタイプライタを起動し，特殊文字が終わったところでもう一度タイプライタを止め，タイプボールをもとに戻さなければならない．特殊文字が何カ所も現れる場合にはこの作業が大変である．したがって，まず特殊文字のところは飛ばして1ページ分打ち，次にボールを取替えてそのページの特殊文字をまとめて打つという機能が追加されれば便利かもしれない．
2) 添字は手作業で打たなければならない．現在使用中のタイプライタでは半行分の行送りを計算機で制御できないため，添字は，1）の場合と同じようにタイプライタをいったん止め，手作業で打たなければならない．しかし，これは

我々の使っているタイプライタの特殊事情によるものであり，半行分の行送りを計算機で制御することは技術的にはそれほど困難ではない．

3) 音節で単語を切れない．手でタイプを打つときには，行の右にはみだしそうな長い単語は音節で切って，後半を次の行へまわすことができる．しかし，本システムでは，はみ出しそうになった単語はそっくりそのまま次の行へ送るため，単語と単語の間に大きな空白ができることがある．これを完全に解決するためには，英単語データベースを清書プログラムと組合せなければならないが，基本的な小数の音節規則を清書プログラムに組込むだけでもかなりの改善になるのではないだろうか．

4) 英文原稿と制御情報が混在することが気に入らない．大域的な制御情報は原稿の初めに置けばよいが，局所的な制御情報はそれぞれ原稿の特定の場所に挿入しなければならない．その結果，原稿が制御情報で“汚され”，清書プログラムと対にしなければ原稿ファイルが意味をもたなくなってしまう．普通の英文原稿の先頭に大域的制御情報をつけ加えておけば，あとはある程度“勝手”に形式を整えてくれる清書プログラムがあってもよいであろう．

6. おわりに

英文原稿を清書するためのプログラムの一例について，簡単な紹介と感想を述べてみた．いろいろ不満も述べたが，それにもかかわらず，「一度使い出したらやめられない」というのが率直な感想である．

また，使っているうちに自分の原稿を清書プログラムの特性に合わせるようになる．たとえば，数式で説明しても文章で説明してもかまわないところは文章にしてしまう——一般に，数式の方が制御情報が複雑になる——し，添字つきの変数“X_1, X_2”等は使わないで，かわりに“$X1$, $X2$”等を使うようになる．

なお，高分解能のドット・プリンタやインクジェット・プリンタが実現すれば，5章の1)，2)の問題はハードウェア的には解消するであろう．しかし，その場合には，積分記号等の特殊記号や図までも原稿ファイルの中に入れておきたいという要求が生じてくるのが自然だから，その結果，入力原稿の中に挿入する制御情報が次第に複雑になり，ソフトウェア上の問題が生じてくると思われる．

ここで紹介したプログラム RUNOFF は，清書プログラムのほんの一例に過ぎない．似たようなプログラムはすでに各方面で使われており（たとえば，東京大学大型計算機センターの ROFF システム），また最近では，ミニコンピュータとフロッピーディスクとタイプライタを組合せた清書専用のシステムも売出されているとのことである．今後も，清書プログラムはどんどん普及していくものと思われる．

参考文献

1) TOSBAC 5600「TEXT EDITOR, RUNOFF」概説書，東京芝浦電気 (1971).

（昭和54年7月17日受付）