談　話　室

日本語とCOBOL*

西　村　恕　彦*

はじめに

わが国において計算機の事務応用が普及し，COBOLが実用化するにつれて，日本の文字あるいは文法の利用への要求は強くなっていくものと予想される．現在のCOBOLシステムに日本の文字あるいは文法を導入したものを現在の計算機に適用すると，いくつかの問題がおこる．それについて，若干の注意を述べる．

まず現在のCOBOL文法のままで日本の文字が用いられる点をあげる．次にわずかに文法を修正するだけで，大きな骨組みをかえることなしに，日本の文字を広く利用する方法を考える．さいごに，全面的に日本語を採用するために解決あるいは合意せねばならない点をひろいあげる．

星印(*)で引用したのはDOD: COBOL, Edition 1965 第Ⅲ節のページである．

1. DOD COBOL

現在のDOD COBOLの文法規則によれば，入出力データ，定数，注記には仮名文字が入ってきてさしつかえない．

以下に三つの定理をかかげ，その根拠をしめす．

定理 1: 入出力データ

COBOLで書いたプログラムは，その処理する入出力データに仮名が含まれてよい．

*-6-28, 6-57: データ項目のクラス

計算機の文字の組にある可能な文字を内容とするデータ項目はALPHANUMERICである．

形式（ピクチャ）のXは計算機の文字の組にある任意の可能な文字を内容とする文字位置をあらわす．

*-2-3: 英数字

alphanumeric characterとはCOBOLの文字の組（51字）にある任意の文字である（この定義はうえの二つより狭く，おそらく，まちがいであろう）．

定理 2: 書いたとおりの文字定数

書いたとおりの文字定数（nonnumeric literal）を環境部門，データ部門，手続き部門に書くときに，仮名が含まれていてよい．

*-3-3: 書いたとおりの文字定数

書いたとおりの文字定数は任意の可能な文字の列である．

定理 3: 注　記

動詞NOTEのあとや見出し部門の注記パラグラフは仮名が含まれてよい．

*-4-2: 注記パラグラフ

注記パラグラフは可能な文字の組に含まれた文字の任意の組み合わせであって，文やパラグラフの形式にあわせて書く．

*-7-51: 文字列

動詞NOTEのあとには可能な文字の組にある文字の任意の組み合わせを列にして書く．

2. 仮名 COBOL

現在のDOD COBOLの大部分の文法規則と予約語の表とをそのままにしておいて，ただデータ名や手続き名などに仮名を用いてよいことにすると，これまでのCOBOLの文法や翻訳ルーチンにほんのわずかの変更を加えるだけで相当に親しみやすい，読みやすいプログラムを書けるようになる．国際的な共通性との矛盾も少なくてすむ．

これを仮名COBOLとよぶ．実例としてはFACOM-230-10, NEAC-2200などがある．

以下にDOD COBOLによる定理をかかげ，それにたいする変更提案を述べる．

仮名COBOLコーディング例:

* Some Comments on COBOL with Japanese Letters, by Hirohiko NISIMURA, Electrotechnical Laboratory of MITI.
** 通産省電気試験所

READ オヤーファイル AT END GO TO オワリ.
MOVE ミダシ TO ミダシ-W.
IF キンガク = 0 GO TO ケイサンオワリ.

定理 4: 語

一般の語は，仮名を含まない．手続き名，呼び名(mnemonic-name)？，記録様式名（mode)，作成者のきめた名前（implementor-name）などは語である．

＊-2-37: 語

語は以下の37種の文字のみからなる30字以内の文字列である．英字 A～Z，数字 0～9，ハイフン（ただし最初や最後には書けない)．書いたとおりの定数とピクチャの文字列とはこの定義の例外になる．

＊-2-23，2-28，2-33，2-19，2-24 などをみよ．

提案 4: 語

語を以下の文字のみからなる文字列とする．英字，数字，ハイフン（ただし最初や最後には書けない)．長音記号（ただし最初には書けない)，仮名，中黒(ただし最初や最後には書けない)．

定理 5: 名前

少なくとも1字の英字を含む語であって，データ名，条件名，登録名，新動詞（new-verb）などに用いる．

＊-2-11，2-8，2-22，7-27 などをみよ．

提案 5-1: 名前

少なくとも1字の英字または仮名を含む語とする．

提案 5-2: 語と名前

語を，英字語，英字名，仮名語にわける．英字語と英字名との規則は定理 4，5 のとおりとする．仮名語とは以下の文字からなる文字列とする．仮名，長音記号，中黒．仮名と英字との混用をみとめない．

提案 5-3: 仮名語

提案 5-2 においてデータ名，手続き名などを仮名語に限定する．つまり英字語，英字名は予約語にのみ用いる．仮名語はプログラマの作る名前とする．

3. 日本語 COBOL

事務用プログラミング言語として，日本語の文字と文法とを全面的に採用した方式を日本語 COBOL とかりによんでおく．この考え方をフランス語，ドイツ語，イタリア語に適用した例は，ECMA: COBOL Translations にみられる．その効果には次のものが考えられる．

（1） 英語の COBOL を教育，説明，議論する助けになる．

（2） 文書化（documentation）がよい．

（3） 計算機プログラミングをより浅い，広い層に普及できる．

（4） 国際的な共通性がまったくない．

日本語 COBOL には NEAC 2200 などがある．

3.1 動詞の活用形

英語の COBOL の命令や句は次の形にしたがうのが自然である．

命令: 動詞 〔名詞〔分離詞 名詞〕…〕
句: 名詞 連結詞 名詞〔分離詞 名詞〕…

日本語 COBOL では命令は次の形にするのが自然であろう．

命令:〔名詞〔分離詞 名詞〕…〕動詞

この最後の動詞については，終止形，命令形，連用形の三種類の活用形が考えられる．もうひとつの代案は漢語サ変動詞を使って，終止形を語幹で代用するのである．

例 1: 終止形，句読点つき

ファイルを読む；空ならば終りに行く．甲を乙に写す．甲を丙に加える丸める丁に写す．戊に行く．

例 2: 終止形，句読点なし

ファイルを読む空ならば終りに行く．甲を乙に写す甲を丙に加える丸める丁に写す戊に行く．

例 3: 命令形，句読点つき

ファイルを読め；空ならば終りに行け．甲を乙に写せ．甲を丙に加えよ丸めよ丁に写せ．戊に行け．

例 4: 命令形，句読点なし

ファイルを読め空ならば終りに行け．甲を乙に写せ甲を丙に加えよ丸めよ丁に写せ戊に行け．

例 5: 連用形，句読点つき

ファイルを読み；空ならば終りに行く．甲を乙に写す．甲を丙に加え丸め丁に写す．戊に行く．

例 6: 連用形，句読点なし

ファイルを読み空ならば終りに行く．甲を乙に写し甲を丙に加え丸め丁に写し戊に行く．

例 7: 漢語サ変語幹，句読点つき

甲を乙に転記．甲を丙に加算．戊を実行．

3.2 分かち書き

仮名で文章を書くには文節分かち書きがもっとも自然であろうが，翻訳ルーチンはむつかしい．単語分かち書きはその逆である．

例 8: 文節分かち書き

データ・レコードハ デンピョウ.
エート ビート シートヲ デーニ クワエル.

例 9: 単語分かち書き

データ　レコード　ハ　デンピョウ.

エー　ト　ビー　ト　シー　トヲ　デー　ニ　クワエル.

分かち書きの方式はあいまいなものがあるので，翻訳ルーチンは安全なように，また予約語の表は大きく作らねばならない．これは英語でも単数形と複数形との対立としておもてにあらわれている．

例 10：文節分かち書き

コオガ　オツヨリ{ダイデナイナラバ / ダイデ　ナイナラバ / ダイデ　ナイ　ナラバ / ダイデナイ　ナラバ}

語の表記法が日本語での慣用として十分に固定していないものがあるので，ここでも安全を考えねばならない．たとえば合成語をつなぐのに，いきなり仮名をつなぐか，ハイフンでつなぐか，中黒でつなぐか．音引きの表記は，ヒョウキ，ヒョオキ，ヒョーキがありうる．助詞は「ハ，ヘ，ヲ」か，「ワ，エ，オ」か．字母としては長音記号とハイフン，中黒と終止符，引用符と濁点などの区別．

3.3　数式と条件式

日本人にとっては数式や条件式は各種の演算記号を用いて数学的な記法によるのがもっとも自然でわかりやすいだろう．しかしそのためには AND, OR, NOT に相当する記号を COBOL の文字の組につけくわえないといけない．

演算記号を用いないで式を記述することができる．しかし普通の演算記号の位置は英語の語順にはあっているが，日本語の語順にはあっていない．

例 11：演算記号

B*B-4*A*C>0∧A≠0∨M

例 12：英語

B times B minus 4 times A times C is greater than zero and A is not equal to zero or M

例 13：直訳

B かける B 引く 4 かける A かける C が大なり零かつ A 等しからず零および M

例 14：日本語

B に B をかけ 4 に A をかけ C をかけて引いたものが零より大きくかつ A が零でないかまたは M

すなわち英語と日本語との語順は次のように対照される．

英語：(名詞) 動詞 (名詞)

日本語：(名詞　名詞) 動詞

4.　シフト記号と濁点

文字のビット構成が 7 ビットあるいは 8 ビットで，英数字と仮名とが単一の文字で識別できるときには，ある字母の列はあるビットの列へ一意に変換できる．しかし 1 字 6 ビットのような記憶媒体（入出力および内部記憶）においては，シフト記号を利用することが多い．また濁点のとりあつかいにも問題がある．

4.1　桁　数

シフト記号を用いたときには，ある字母の列の長さ (size) のきめかた，およびある字母の列をあるビットの列へ一意に変換する仕方が問題になる．まずその前者については文法書に以下の規則がある．

*-2-16　標準のデータ形式

COBOL のデータ部門でデータを記述するさいの概念で，無限の横幅と縦長さとをもった印刷ページ上のデータ形式のみかけにあわせて，データの性質や特性を表現する．計算機の内部や特定の外部媒体上にデータをたくわえる形式ではない．

*-2-5　標準の文字列

データ項目の内容の文字列で，その長さが標準のデータ形式どおりに測られるもの．

*-6-73　長さ (size) 句

データ項目中の文字の数で，標準のデータ形式によって測られた長さを指定する．

これらによると入出力データと書いたとおりの定数とについては，その長さの測り方は明白である（以後においては英字シフト記号を §，仮名シフト記号を 『 であらわす）．以下のいくつかの表現はすべて長さ 3 の文字列をあらわす．

“オバQ”

“『オバ§Q”

“『『§『オ『バ『§§Q§§”

“オハ　印字後退　〃§Q”

あとのほうの例は濁点の表現に関するもので，計算機によって次のようないろいろの表現がありうるが，いずれも長さ 3 と考えるべきである．

“ゴジラ”

“コ〃シ〃ラ”

“スペースカットの〃　コ　スペースカットの〃　シラ”

“〃　印字後退　コ〃　印字後退　シラ”

これらの例のうち“オバQ”と“ゴジラ”以外の表現を DOD のとおりに長さ 3 として，翻訳ルーチンを

作成することは，不当に手数がかかるものと考えられる．

シフト記号と濁点との長さ（size）のかぞえ方は，次の三つのどれかになろう．

（1） コーディング用紙，穿孔符号，内部記憶，印刷，入出力データが8ビット程度の符号からなり，英字，数字，仮名，濁音などがそれぞれ一意の符号であらわせるならば，何も問題はない．

（2） コーディング用紙，穿孔符号，内部記憶，印刷，入出力データのすべてについて符号が共通で，しかもシフト記号や濁点が常に1桁の桁位置を占有するならば，標準のデータ形式の定義を少し拡大解釈して，シフト記号や濁点をも固有の文字とみなす．ただしシフト記号が桁位置を占有するにもかかわらず，印刷字母としては空白になるときには正規の間隔との区別がつきにくく，デバッグや文書化に不利である．

（3） コーディング用紙，穿孔符号，内部記憶，印刷，入出力データにおいてシフト記号や濁点が，ある場合には1桁の桁位置を占有し，また他の場合には桁位置を占有しないような金物やシステムが存在する．たとえば外部8ビット，内部6ビットとか，外部6ビット，内部8ビットとか，あるいは外部，内部とも6ビットだが印刷のときだけスペースカットされるとかの例がそれぞれある．

4.2 語の同定識別

シフト符号や印字後退があると，字母列をビットの列にかえる操作が一意でなくなる．

“コ 印字後退 〃 シ 印字後退 〃ラ”

“コシラ”印字後退（4回）〃〃スペース（2回）

これらのデータの内容の大きさを比較して大きいとか等しいとかいうことができるであろうか．

より深刻な問題はソースプログラム上にある．まず単純な問題は正書法に関するものでそれによるとソースプログラムの行の右端はマージンR（たとえばコラム 72）で区切られることになっている．ソースプログラムの行にシフト記号や濁点が含まれていると，この右端のマージンRのきめかたに注意がいる．ただしこれは現在の文法でも強く規制されているわけではない（*-10-1）．また復改したときにシフトの効果がつづいているかどうかもきめる必要がある．

しかし次のような命令において，各データ名や予約語をどうやったらうまく識別できるだろうか．それだけの手間を翻訳ルーチンに負荷させてよいだろうか．それともプログラマかパンチャの責任にすべきだろうか．名前はすべてシフト記号で囲むことにすべきだろうか．提案5-3によるとそうなる．

MOVE ウルトラQ TO F6セブン

MOVE『ウルトラ §Q TO F6』セブン

§MOVE『ウルトラ §Q §TO §F6』セブン

§M§O§V§E『ウ』ル』ト』ラ §Q §T§O §F §6』セ』ブ』ン

翻訳ルーチンを作るときに，これらのシフト記号や濁点を含んだ文字列を一意のビット列にかえる手続きを組みこんでおくか，さもなければ，ある字母列を一意に文字列にかえる手順をプログラマかパンチャに教えておかないといけない．

このことはまた，COBOL で書いたプログラムの共通性を，コーディング用紙あるいは印刷用紙のレベルで（つまり字母のレベルで）きめるか，それとも機械媒体のレベルにまでおよぼすのかという議論に結びつく．現実には紙テープにしろ，紙カードにしろ，符号が各社まちまちだから，機械媒体におさめられたプログラムの共通性はすぐにはのぞめないだろう．

それでもおそらく，ある字母列が与えられたときにそれを穿孔符号の列にかえる一意の手続きを標準として定める必要があるのではなかろうか．それによって少なくとも翻訳ルーチンの作成にあたって考慮すべき条件を減らすことができよう．

参考文献

1) Department of Defense: COBOL, Edition 1965.
2) 富士通：FACOM 230-10 小形電子計算組織解説書
3) 富士通：Compiler-Oriented Computer FACOM 230-10 について，第7回プログラミング・シンポジウム報告集，C 84-114, 1966-1.
4) 日本電気：NEAC-シリーズ 2200，プログラミング システム，日本語コボル概説書，1966-12.
5) 電子協：NEAC-シリーズ 2200，カナコボルコンパイラ-D, 1965-11.
6) 関根智明監訳：COBOL 1965 年版，予定．

（昭和41年12月6日受付）