プログラミング言語 16－7
（１９８８．５．１３）

プログラム言語で使う文字コード

和田 英一
東京大学工学部計数工学科

プログラム言語で書くプログラムのうち計算機入力用には金物表現が重要であり、それには使う文字セット、文字コードを規定する必要がある。プログラムの中でも注釈や文字列定数は比較的自由に文字セットが拡張できる。この報告では、これまでの主要なプログラム言語で使った文字セットとコード(FortranのEBCDIC, APL文字コードなど)について概観する。さらに、現在主として使われている文字セットの ASCIIコードの構造を述べる。 ASCIIコードを基本としたいくつかの 8ビットコード系が整備されつつある。また日本語プログラミングの関係ではかつてはカナによるプログラム言語があった。データとしては漢字かなの2バイトコードが一般的に使えるようになったが、シフトJISはなるべく使わないようにしたい。

Character codes used in programming languages

Eiiti Wada
Department of Mathematical Engineering, University of Tokyo
Bunkyoku, Tokyo 113 JAPAN

For programs in hardware representation to be input into computers, graphic character sets (and their codes) must be defined. Even when the syntax is defined strictly, it is relatively easy and often permitted to expand the character sets for literal constants and comments to include kanji and hiragana characters.

Character sets used in historical languages are surveyed. Then structure and related issues on ASCII code is discussed. Several 8 bit standard character sets are now being prepared which are likely used in programs. Structure of Shift JIS code is finally discussed and it is recommended not to use the Shift JIS code in programming environment.

はじめに

プログラム言語で記述したプログラムには人に見せるためのものと計算機に入れるためのものがあるとAlgol 60ではいっていた。つまり規準言語 (reference language)、出版言語 (publication language)、金物言語 (hardware language) があることになっていた [1]。規準言語は言語を定義するためのものという感じである。Algol 60の委員会はもっぱら規準言語にばかりかかわっており、金物言語はインプリメンター任せだったため、いろいろな流儀の金物言語が作られ、それもあまり便利でないものだったので結局Algol 60のコミュニティは育たなかった。他の成功したプログラム言語は「はじめに金物言語ありき」という利用上のことを考えた発想で出発している。

金物言語というとそれを表現するための図形文字セット（とそのコード）が重要であり、現在は ASCIIコード全盛ともいう感じであるが、その他にもプログラム言語で使っている図形文字セットはいろいろあるので、この報告ではそれをサーベイしてみたい。なお、図形文字というのは復帰改行のような制御文字に対する言葉だがプログラム言語の場面では、まず図形文字が中心話題なので以下では単に文字セット、文字コードということにする。

プログラム言語における文字の使用区分

プログラムは文字の並んだものという構文規則もありうる筈だが実用上はもっと詳細に定義がしてある。それをみるとプログラム言語での文字の使われ方は文字の使われ方のうるささについて次の 4通りに分類できよう。

1. 演算子 (+ := など)、綴りの記号(if begin など)、区切り記号(; など)
2. 識別名 (i instream foo など)
3. 文字列定数 ("argument type error")
4. 注釈 (comment standard name table initialization)

このうち下のものほどインプリメンターが文字セットを自由拡張できる。プログラム言語の文法書にそう記述してあるものもある。

テレタイプコード

最初の電子計算機は大学や研究所みたいなところで作られたから入出力にはフレクソライタ（電動タイプライタ）やテレタイプを使うことが多かった。国際のテレタイプは 5単位で英字の大文字、数字、特殊記号だけの文字セットである (ITA 2 International Telegraph Alphabet No.2)。これは欧文モールスコードと同じ文字セットである。ケンブリッジ大学の Edsacはテレタイプを使っていたが、10進 2進のコード変換を簡単にするため、コードは国際テレタイプのそれとは同じではない。テレタイプのコードは紙テープにパンチしたとき、くずをなるべく出さないためか出現頻度の高い文字には 1の少ないコードをわりあてていた。モールスコードが短かいのと同様である。従ってQWER...に1234...が対応していると数字は 2進法のパターンとはきれいには対応せず、 Edsacではそのためコードを変えてしまった。東大の TACはEdsacをまねして作ったからコードも同じだったかと思う。

高橋研のPC-1は国際テレタイプと基本的には同じで 6単位のコードのものを使っていた。

EBCDIC Fortran 48文字セット

TACが完成して少し経った頃Fortranが登場した。これは IBMで開発したので IBMカードの文字を使う。これは図 1に示すような48文字セットであるが(,), =, -は事務用のセットでは□, %, #, &になっていた。このうちFortranが使っていたのは47文字で 8-4パンチの -は何のためにあったのかよくわからない。

Fortranとほぼ同様に古い歴史のLisp1.5はIBM 704用に作られたので文字セットはFortranと同じである。なにしろ記号アトムは英数字だしその他に( ) . しかいらないのだからこんなにシンタックスシュガーを採り入れない言語も珍しい。しかしLisp1.5のマニュアルには次の記号アトムには対応する文字が値として結合していると書いてある。

```
dollar $   slash /          lpar (    rpar )    comma ,    period .
pluss  +   dash -(11punch)  star *    blank     eqsign =
```

-に 11パンチと注がついているのは 8-4パンチにもうひとつ使わない -があったことを示している。
Cobolもほぼ同じ文字セットであった。

日本語プログラムとカナ文字

我が国でも Fortran, Algol, Cobolの処理系が次第に作られるようになってくるとカナ文字の使えるものがでてきた。つまりカナ文字 Fortranとカナコボルである。面白いことにこのふたつはカナ文字の使える場所が違っていた。カナ文字 Fortranはは綴りの記号のところをカナ文字化した。例えばReadをヨメのように書く。一方カナコボルは識別名にカナを許した。それにはコンパイラの字句読込みルーチンを多少直し、名前表にカナ文字列も入るようにするだけでよい。 Fortranの方はもともと行の途中の空白は無視するという構文規則だから、構文規則は特殊文字だけに頼らざるを得なかったのだが、文の識別にカナ文字という強力な助っ人がきたのでコンパイラ作りは容易になったものと察せられる[2][3]。

カナ文字による日本語プログラミングはその後しばらく見なかったけれども水谷氏の小朱唇も綴りの記号をカナ文字にするプログラム言語である。識別名はローマ字しか使えない[4]。

APL

APLが現れたときはその異質な文字セットと右から左という演算順序は人々を驚かせたものだ。これは特殊な端末

からしか使えない、つまりAlgol 60の金物表現の二の舞かと思われたが、 IBMは APL用のタイプエレメントまで用意したので APLは愛好者の間では使われるようになった。この文字セットのコード表は図 2の通りで図形文字セットとして登録してある。

数学者は次々と記号、記法を発明し思考の節約をはかるものだが、既存の文字セットを後生大事にして begin endifなどやらず新しい文字セットと端末を作る態度にはある意味で感心させられる。こういうプログラム言語は恐らく APLだけであろう。(BackusのFPもイリノイでインプリメントしたがどういう文字セットだったか忘れた。)

ASCIIコード

この業界で標準のような顔をしているのがASCIIコードである。ASCIIはもともとAmerican National Standard Code for Information Interchangeだから、 ASCIIコードというのはコードがだぶって正しくないという人もいるが ASCIIコードといわないとすわりが悪いのでそう呼ぶことにする(図 3の左)。

これは ISO 646という 7ビットの標準で決めた基本コード表というのがあるのだがその仲間である。このコード表には10ヶ所ほど各国で自由に使っておい場所(各国場所)がある。また2/3, 2/4はそれぞれナンバーサインかポンド、カレンシーサインかドルを入れることとなっている。もし各国で特に何も入れないときはそこに適当な記号を入れたIRV(International Reference Version)というコード表があり、この記号を使うことになっている。ANSIはIRVの2/4の場所を $にしたものを自国の標準としたこれがASCIIである。(よく見ると7/14も少し違う。)

IBMはパソコン以外はEBCDICで悠然としているが、世の中には ASCIIの端末が多く出回り、そのためASCIIに準拠したプログラム言語が増えてきている。Pascal, C, Ada, CommonLispなどがそうである。しかしこれらの言語も ASCIIがなければプログラムが書けないというのではない。例えばPascalは Fortranの48文字程度でプログラムが書けるよう、ある程度の文字の代用を決めている。[,], ↑は(.,.),@を使ってよい。 Adaもそうで | は! #は :のように決めている。Adaのこの手当ては各国場所を避けたかったからである。

ISOのコードについてはよく知らないのだが少し前の ASCIIは5/14, 5/15が↑, ←になっていた。(今は ^ と_ である) Smalltalkは値を返すのに↑を、代入に←を使っているからその仕様は←, ↑の時代に決まったものであろう。キィボードや fileoutフォーマットのファイルから ^ や_ を読み込むと画面には←や↑がでるから一安心である。

プログラム言語で使うことを考えると ^ や_ より↑や←の方が適しているような気がする。10年ほど前に当時院生だった近山君が8080用に「小さいLisp」を作った。これは< が car, > が cdrのように特殊文字 1字が基本関数名になっていた。↑や←はそれぞれrplaca, rplacdに使うと丁度よいという話もあったがそれはそうしなかったと思う。

ASCIIコードは今や RIVを駆逐せんばかりの勢いである。 2/4にカレンシーサインの入った IRVの端末など実在しないし誰も使おうとしないからである。

ISO 8859 8ビットコード

↑が ^ になったのは ^ を発音区別符号に使うためと思われる。 IRVのうち " はダイアレシスに、, はセリダに、はアキュートアクセントに、 ` はグレーブアクセントにも使えるようにできている。そして ^ はサーカムフレックス兼用とした。これらの発音区別符号を使うためにはバックスペースが必要である。また欧州にはまだ変な字もありこれでは欧州各国で通信するには不便だというので、テキスト通信用のコードの標準をISOとCCITTで決めた。ISO 6937というのがそれである。ISO 6937は 7ビットの使い方もあるが一応は 8ビットコードで、左半分に IRVを置き、右にはノンスペーシングの発音区別符号やその他の文字を入れ、 320くらいの字が使えるようになっている。しかしこれは使ってみるとテキスト通信にはなるほど便利だが文字によって 1バイトだったり 2バイトだったりで処理には不便だということになった。欧州の一部ならもっと少ない字数でも役に立つというので西欧州用の 8ビット 1バイトコード(191字 (左半分に95字、右半分に96字) というコード表を作った。これがISO 8859のパート 1である。この頃から2/0のスペースも図形文字の字数に入れたりするようになった。この左半分は ISOのコード表なのにASCIIになっている。同様にして8859の別のパートも作られているがパート 1が 8ビットシングルバイトの標準となるであろう。これがすぐプログラム言語用になるかどうかは予測がむずかしいが13/7と15/7に×と÷が入っているのでこれは使えると思う(図 3)。

漢字コード

JIS X 0208 (昔の呼び方ではJIS C 6226) が制定されたときは入出力はどうすればよいのだろうかと思ったが、その後ドットプリンタやビットマップディスプレイの普及のおかげで漢字処理は急に身近かなものになってきた。

junetの通信も漢字かなのものが圧倒的に多くなってきた。当然プログラム言語の中でも漢字を使いたいという希望がでてき、もちろん当面は文字列定数と注釈が主だがいろいろな工夫が行なわれてきた。

その第一はシフト JISであった。これは 8ビットの環境で 1. JIS X 0201のローマ字またはASCII 2. JIS X 0201の片仮名、及び 3. JIS X 0208の漢字の 3種類をシフトコードなしに使おうとするものである。 8ビットのコード表をみると列0, 1にはC0という制御文字が入っている。列 2から 7にはローマ字が入っている。列8, 9はC1の制御文字だがJIS X 0201を見るとここは未定義である。片仮名は列10, 11, 12, 13に収まり(10/0は左へ入れると空白のところだから、これも使わず未定義)列14, 15も未定義である。そこで第 1バイトが未定義なら漢字コードとし、漢字コードなら第 2バイトはC0制御文字のところを除いて使うというふうにすればJIS X 0208の文字は収まるというのが骨子である。

これはMS-DOSで採用されたので、しばらくは一世を風靡したがいろいろ問題があって見直されている。特にunixで

は \をエスケープにしているのに、ここに第 2バイトが現れて変なことを起こす。またC1を使うのに不便だし、大体JISの表といちいち変換（つまりシフト）しなければならない。というわけでJIS X 0201の片仮名（半角カナということにあり）を犠牲にし、右半分にゆったりと漢字を入れる EUCの考えが登場した。半角カナは濁点、半濁点の字は電報のように 2字分使うので全角カナの方がずっと自然であるが、半角カナを使いたいという希望も後をたたず、EUCではこれをG2に指示し SS2で使うことにしている。漢字にはまだ、78年版と83年版の問題、補助漢字符号の問題などあるが、筆者が情報規格調査会の日本語機能委員会で提案しているのは左半分はISO 8859のように ASCIIにする；右は83年版のJIS漢字にする（それも通常は78年版にある文字だけを使う）というものである。ASCIIにすると ¥がなくなり \が復活する。¥は漢字の方を使うのである。

漢字コードの 3区には全角の英字や数字がある。これでプログラムの本体を書いてもよいかとか、全角の fooと半角(ASCII)の fooは同じ識別名なのかとか話題は尽きないが、これはASCIIでも FOOと fooをどう見るかと同様な問題と思われる。

IBM PC 256文字セット

筆者はその辺の事情に疎いがECMAでは最近 128文字セットの検討をはじめた。それはIBM PCとSystem/2がマルチリンガルな 256文字セットを使っているからという理由だそうだ(図 4)。これまでの規格の図形文字集合は96文字が単位だったから、ここでハタと困っているわけだが、一説には32文字コードを用意し、これで例えば列8，9に呼び出すという案もあるらしい。まだこれはどういう方向に発展するかわからない。

128文字セットといえば、Metafontの本にあるStanfordや TEXのコードを思い出す。

プログラム言語で使う文字セットは落ち着くにはまだまだ時間がかかるであろう。

おわり

参考文献

[1] 米田信夫、野下浩平：Algol 60 講義　共立出版 1979

[2] 近藤頌子、原田賢一、土居範久：カナ文字 Fortran
情報処理 Vol.5, No.4 (July 1964) pp.222-223

[3] 西村恕彦：日本語と Cobol
情報処理 Vol.8, No.3 (May 1967) pp.157-160

[4] 水谷静夫：小朱唇言語仕様
東京女子大学日本文学科 1986

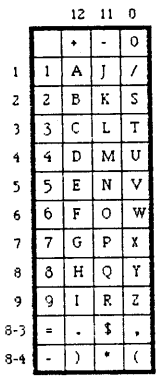

図 1

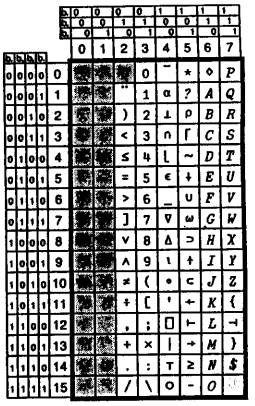

図 2

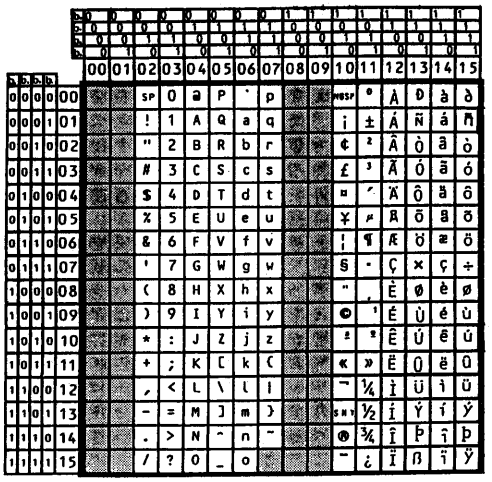

図 3

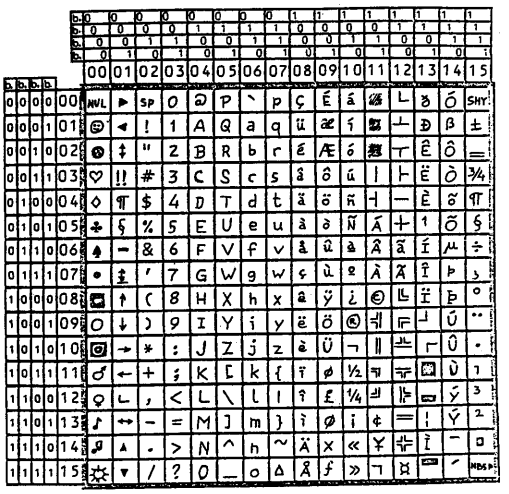

図 4