April 1981

Report. No. STAN-CS-81-848

On the Problem of Inputting Chinese Characters

by

Chih-sung Tang

Research sponsored in part by

National Science Foundation

Department of Computer Science

Stanford University
Stanford, CA 94305

On the Problem of Inputting Chinese Characters

Chih- sung Tang
(Department of Computer Science, Stanford University)
(Institute of Computing Technology, Academia Sinica)

1. The Problem.

Many people are currently seeking a good way to do input and output of Chinese characters on computers. It seems to me that a lot of confusion about this problem is also prevalent, due perhaps to a misunderstanding of the problem. Basically, people are apt to think of input and output with Chinese characters as a simple generalization of the input/output problem for conventional characters, while in fact it is not. For example, inputting and outputting of Chinese characters are themselves two separate information processing problems. I think that a clarification of the problem may be a helpful step towards its solution.

People often say that the inputting and outputting of Chinese characters is an important and difficult problem, at least for Chinese-speaking people. But why is it important? Why difficult? I think the most convincing answer to the first question might be that without Chinese characters there are a great many important applications of non-numeric information processing in business, management, information retrieval, etc., that cannot be well developed in the Chinese-speaking society. So when we talk about the inputting and outputting of Chinese characters, what we mean is to do it in the context of an information processing program, not just to input or output the characters as an end in itself.

In this sense, Chinese character input/output is quite a different problem from what we may call a Chinese character writing system. I once saw a system of the latter kind in a famous research laboratory. The user can write a Chinese character stroke after stroke before the terminal, just like we write a Chinese character on paper. The author also claims that his work deals with the input problem for Chinese characters! This kind of work may have its significance as part of research studies on graphics or artificial intelligence, but it is far from becoming a practical input means for a real application since few programs can be allowed to receive their input at such a slow speed.

Furthermore, the problem of inputting and outputting Chinese characters in an information processing program is' a different problem from that of building a

The publication of this report was supported in part by NSF grant IST-7921977.

Chinese typewriter; the former need not presuppose the latter. People accustomed to considering the input and output of Chinese characters as a generalization of the input and output of conventional characters are quick to identify the input/output problem with questtions of typewriter design. But in fact, it is more difficult to build a typewriter, because (1) a typewriter must deal with an universal dictionary, (2) a typewriter cannot take advantage of the fact that input/output can be treated as a subproblem in a larger information processing environment, and (3) a typewriter that can type only Chinese is not useful in information processing, while one that can type both Chinese and conventional characters has too big a keyboard.

Let me explain these points in more detail. In an ordinary information processing problem, the vocabulary is always limited for a special application. In most cases we could even force the user to use only a certain restricted special-purpose dictionary. In other applications such as a library system, it appears that a universal dictionary may be required; yet it is still possible to make use of the principle of "divide and conquer" by partitioning a large dictionary into several special-purpose ones, for example by sorting the books of a library into different subcategories. In this way, many difficult problems can be ameliorated.

When we treat the input/output problem as a subtask in an information processing program, it is possible not only to specialize the vocabulary with respect to the problem at hand, but also to treat the dictionary as a private resource: A user can even establish a dictionary according to his or her own personal taste, and one user might have several dictionaries, each corresponding to one special inputting 'method. Thus the binding time for the choice of an input method can be delayed to the very time of input. This is a very important concept in the approach that we will discuss below.

The last but not the least important point we shall emphasize is that any Chinese input/output system must also be able to input and output conventional characters; otherwise it would be only a Chinese text-copying system and would be of little relevance to information processing. No information processing program can be written with only Chinese characters and no other symbols. Indeed, I once saw a computer equipped with such a system, and I doubt if it has ever been used in any information processing program.

Next, inputting and outputting of Chinese characters must be considered as separate information processing problems. The only connection required between them is to find a one-to-one or many-to-one correspondence between the input code of a word and the the internal representation of its output code. Some people try to impose an exact structural image of the output pattern onto its input code, and this always makes the input code unnecessarily complicated. I designed a system of this sort around 1973 and finally found it impractical. I have also seen one such system in the United States; this approach is often related to the construction of Chinese typewriters.

The output problem itself should be divided into two distinct parts: (1) to output recognizable Chinese characters quickly and efficiently, and (2) to deal with characters that are aesthetically beautiful. Both of these problems have by now been resolved satisfactorily, at least to some extent, but the input problem is still stubborn. So the remainder of this paper will concentrate on the input problem.

Thus, the problem we are going to discuss is *to find an easy, quick, and convenient way to input Chinese characters together with other conventional characters, in the contest of an information processing environment, with the result that the input code of each Chinese character can be used to identify the internal representation of its output code almost uniquely.*

Is this an easy problem? If we neglect the "easy, quick, and convenient" requirements above, the problem is not so hard as it seems. For example, I know that in some places there have been Chinese telegraph operators who can readily recite all the digital codes of Chinese characters. To them the inputting problem is obviously trivial. But to simplify the input problem at the expense of asking the user to be trained to remember too many conventions is not an acceptable approach, for a method that is accessible only to highly trained experts can never become popular.. Speed is another important consideration. It is well known that an English word contains about 4 to 5 letters, on the average, so it is reasonable to expect that each Chinese character be input with no more than 4 or 5 keystrokes. On the other hand, speed is only one of the criteria by which a method can be judged; it would be a mistake to lay too much emphasis on speed at the expense of other qualities.

"Convenience" is another important requirement: No method would be acceptable if it were hard to handle. Once I saw a system that can input both Chinese and conventional characters; since the keyboard is limited (too big a board being inconvenient to operate), each key must be used to represent two or more conventional characters together with 2 or 3 different Chinese radicals. You can imagine what a mess this is! I am afraid that it would take very long for most people to find one key, and I suspect there would be many printing errors on each page. Can this be considered a good design?

Finally, the uniqueness of representation by which each output code is identified from the input code is another crucial point. In fact, until now no significant method exists that is able to establish absolute uniqueness, and some systems suffer from too many ambiguities. For example, the idea of inputting Chinese characters according to their Pinyin spelling often has this defect. (Incidentally, the Pinyin method has still other defects; for example, Chinese people speak too many different kinds of dialects or non-standard mandarins.) A good method must satisfy two requirements: (i) The percentage of exceptional ambiguities should be at most a certain acceptable limit, e.g., 1.5%; (ii) the system should afford a good method to treat these exceptional cases.

Consequently, judged by these standards, Chinese character inputting is indeed a difficult problem.

2. A Dictionary-Independent System.

It seems to me that there is a very simple but useful concept that has been almost neglected by most systems I have ever encountered, namely to distinguish and separate the input system (the program to accept the input code) from the input method (the method of expressing each Chinese character by its input code or the method of striking the keys). For this reason alone, many systems are built according to one fixed input method. It is even worse to build a mechanical-electronic input device in this way. Such a design is bad not only because no input method can as yet really be considered as the best and in need of no further modification, but also (perhaps more importantly) this approach itself leaves the method without flexibility. Since the basic problem is to establish a one-to-one or many-to-one correspondence between the input code and an internal representation of the output code of each word, all such information is contained in a dictionary. As long as the user inputs Chinese and other characters using a method consistent with whatever dictionary is being used, the requirement is fulfilled, and the program system is implemented to search the dictionary. So it is natural to implement it as dictionary-independent software. If a system is constructed in this way, any change of input method requires only a change of dictionary. The system is always stable; more importantly, such a system is flexible in the following practical ways: (1) One system can adapt to many different input methods. (2) Different applications can have different dictionaries all residing in one system. (3) The dictionary can always be extensible. (4) Each word or character can have several different input codes. (5) Even when one method is considered as the best, different users or different situations may require some minor alterations, and such modifications are easily tolerated. Users who wish to input some particular words in a novel way, according to their special taste, can do so, even if they choose to use completely different principles for some characters than for others. For example, there may be a user whose dictionary is built according to the method I shall introduce in the next section; but when special ambiguous words like "  ", "  ", "  ", and "  " occur, it would be perfectly right for the user to input these words with some other method that seems natural (e.g., Pinyin without four tones).

In order to manage such a system, we suggest a command of the following form:

@HANZ ((file name) ; (extensible part))

Here the "file name" denotes the dictionary to be used, and the "extensible part" specifies input/output code pairs for new words not contained in the dictionary or new input codes for words already in it.

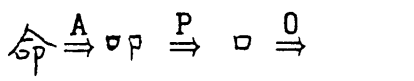
3. An Input Method.

In this section, a particular input method will be introduced that seems to me able to fulfill the demands listed in previous sections. This method consists of three basic points: (i) It illustrates a new way to define the constituents that form a Chinese character. (ii) It also provides a way to arrange the constituents of a word into linear order. (iii) It does not require an exhaustive listing of all parts of a Chinese character, when partial information is sufficient.

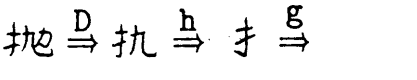
The method for defining the constituents of Chinese characters is based on an interesting idea that Chang Hung-fang [1] has introduced to search for Chinese words in a dictionary. It seems to me that is the method not very successful for his original purpose (to search in a dictionary), but it will indeed be very useful for inputting Chinese characters. His basic idea is to think of the constituents of Chinese characters as if they were similar in shape to Latin letters. For example, the constituent "阝" (side ear) looks very much like a letter "p", and "攵" looks a lot like the letter "k", etc. Obviously, these constituents do not correspond exactly with traditional Chinese radicals, but they have the following virtues: (i) To code Chinese characters in this way, we can use the same keyboard to input both conventional symbols and Chinese constituents. (ii) Since both types of symbols have similar shape, it is easy for keyboard operators to remember. (iii) Each of these constituents can cover a rather large part of a word, so four or fewer constituents will usually suffice to identify a word.

Adopting Chang's basic idea, we can modify it slightly and generalize it to 'other conventional characters that appear on standard keyboards, as in Table 1 at the end of this report. (Table 1 is still subject to modification; the user can make his own conventions according to the principles discussed above.) Chang calls the constituents the *shape constructs,* or simply "constructs", of the character.

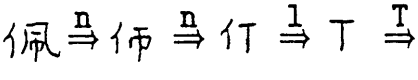
To arrange the constituents of a word into linear order, we have tried several methods (cf. [2]). The best of these seems to be to arrange them by successively cancelling the largest construct encountered at the four corners of a word, proceeding clockwise from the upper right corner. If the word has been exhausted before four steps are completed, it ends with fewer than four symbols, otherwise it is encoded by four nonblank symbols. Let us look at some examples:



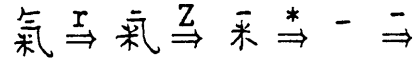
code = APO



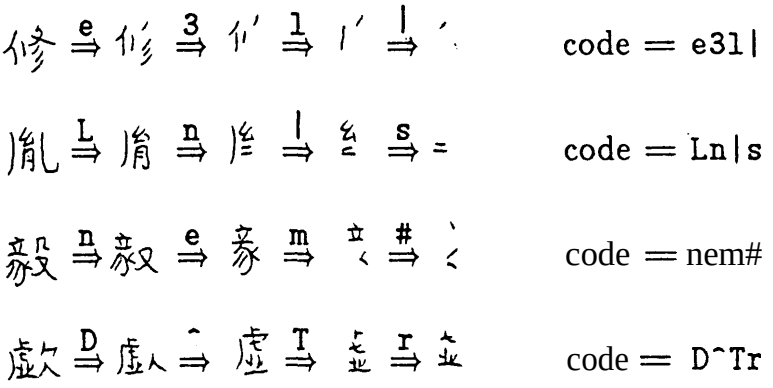
code = Dhg



code = nnlT



code = rZ*-



In each case the first character is removed at the upper right, the second at the lower right, the third at the lower left,, and the fourth at the upper left,.

The last example seems not very satisfactory, for it has too big a remainder. But in fact, its input code is already sufficient to identify the internal representation. Fewer ambiguities would arise if we were to choose five steps as the maximum for each word.

Note that about 90 conventional symbols are also considered to be special "Chinese characters," i.e., Chinese characters of one constituent, and there are a few real Chinese characters that happen to contain only one constituent; these might be confused with conventional characters. In order to avoid such confusion, we adopt the convention that Chinese characters having only one constituent are preceded by the symbol "1". For example, "1R" is "尺"; "1h" is "九". But there are some symbols corresponding to more than one character. In these cases, we can use "12", "13", "14" as prefixes, to distinguish them according to their order of occurrences in Table 1; e.g., "1m", "12m", "13m", and "14m" represent "艹","巾", "爪", and "勿", respectively.

4. Chinese Characters in Programming Languages.

Since the Chinese characters of. interest are those occurring in an information processing program, it is natural to consider the information processing of Chinese characters as a problem related to programming languages. We have considered this problem in two ways: (i) To design a new language with abundant facilities to do Chinese character information processing. (ii) To implement a Chinese version of existing famous programming languages, e.g., Chinese Cobol, Chinese Pascal, etc.

For problem (i), we have designed a language system with this goal in mind. The XYZ family of languages [3] contains the following features:

(1) In order to diminish the number of reserved Chinese words occurring in a program, we never use natural language words as grouping symbols like "**begin... end**"; instead, we always use mathematical brackets for this purpose. Furthermore, we always use one reserved word together with a mathematical symbol to

replace multi-word delimiters; e.g., we use the structure "BRANCH[B: S_1; : S_2]" instead of "if' B **then** S_1 **else.** S_2", etc.

(2) Every reserved word has both a Chinese counterpart and an English counterpart; sometimes they also have a math symbol as the third choice. The user can choose among them optionally. Thus, the reserved word for looping has three counterparts: "ITERATE" or "重复" or "*".

(3) The user can define both Chinese and Latin identifiers. But we expect the users not to let Chinese identifiers be the name of functions, for this is never done in traditional Chinese mathematics textbooks; the use of Chinese for function names would not. be readable.

(4) The language contains a new type CHINESE; a variable of this type can be assigned a Chinese string as its initial value.

By means of the input method shown in previous sections, we have been implementing a Chinese Cobol. We treat it in following simple way: Suppose we already have a compiler for English Cobol in the computer; we implement a preprocessor to accept the program written by Chinese Cobol. It translates the program into English Cobol, prints out both texts, and then transfers to the existing compiler to compile and execute. If the program contains user-defined identifiers for which the user wants to have a readable English correspondent, he or she must input a table indicating the correlation; otherwise the system will give each such identifier a standard name such as Xxi. So our input method can help us to solve such problems as creating a Chinese Cobol, and these problems are often considered difficult to deal with.

Acknowledgments. An input system has been implemented by Jianxin Wang in The Institute of Computing Technology, Academia Sinica, Beijing, along the lines suggested here; the research for this paper has been finished at Stanford in 1981. The author wishes to express his gratitude to Profs. McCarthy and Knuth for their support, especially to Prof. Knuth, without whose kind help this paper would not have its present form.

References

[1] Chang Hunk-fang, "A method to search for words in a Chinese dictionary." Draft, 1976. El Popola Cinio, P.O. Box 77, Beijing, China.

[2] Tang Chih-sung and Wang Jiangxin, "Chinese character inputting and higher level languages." Chinese Journal of Computers 1(1979), 55-60.

[3] Tang Chih-sung, He Tian-Mu, and Xu Fu-Ching, "The common base language of XYZ," Computer Science (Chinese) No. 2 (1980), l-21.

CORRELATION BETWEEN ASCII SYMBOLS AND CONSTRUCTS OF CHINESE CHARACTERS

SYM	CON	CON_VARIANTS	EXAMPLES	SYM	CON	CON_VARIANTS	EXAMPLES
A	亼	亽, 𠂉	命, 令, 错	a	石		硕
B	阝	巳, 乃	邓, 追, 孕	b	白		白 鬼
C	匚(w)	亡, 𠄎	匡, 亡, 民	c	卬(小)	卬	仰, 贸
D	刀	𠃌, ⺈, ⺈, 力	召, 方, 色, 欠, 加	d			
E	[illegible]	[illegible], 斤, [illegible], 彐, 王	印, 辰, 段, 归, 理	e	巳	已, 己, 巳, 巳, 又	导, 忌, 民, 记, 皮
F	下	下, 𠃑, 𠂆, ㄑ	卡, 正, 足, 后, 登	f	彳	亍	待, 稚
G	[illegible]		卯.	g	扌	予	拉, 序
H	廿	廾	苗, 弊	h	九	门	抛, 闪
I	工	工, 工, 工	巫, 贡, 功, 壬	i	讠		识
J	亅	丁, 丁, 亅	永, 行, 可, 寸.	j	礻		神
K	[illegible](w)	片, 爿	畏, 衣, 版, 戕	k	衤(小)	丬	神, 衣, 状
L	乚	乚	北, 心	l	亻		化 修
M	冂	内, 罒, 雨	咒, 丙, 而, 皿, 雷	m	罒	巾, ⺌, 爪, 勿	带, 帽, 爱, 貌, 瓜, 毅
N	川	儿	训, 流	n	冂	月, 几, 凡	同, 月, 风, 且, 凯
O	口(w)	日, 耳	国, 母, 耳	o	口(小)		呈
P	尸		展 眉	p	卩	卩	节, 假
Q	日		明	q	夕	夕, 𠂇	外, 互, 登
R	尺		尽 咫	r	卜	𠂉, 卜	虑, 生, 卦
S	弓		张	s	幺	纟	幼, 红
T	丅	厂, 丆	不, 居, 贝	t	七	匕, 九, 七, 弋	块, 比, 龙, 宅, 戈
U	凵	凵	凶 击, 臼	u	厶	厶, 山, 𠃊	参, 云, 虫, 以
V	丷(w)		兑 光 曾	v	丷(小)		金 曾
W	山	屮	岐, 朔	w	⺍	⺍	堂 学
X	乂	夂, 丈	交, 女, 史	x	儿	几	元, 见
Y	丿	刂, 刂, 丩	归 为, 刻, 临刻	y	犭		狗
Z	乙	乁, 之, 辶	乞, 飞 气, 之, 追	z	子	子	孔, 享

(cont.)

SYM	CON	CON-VARIANTS	EXAMPLES	SYM	CON	CON_VARIANTS	EXAMPLES
$	帮	韦	贯,伟	%	小	中	劣,重
&	足	是,艮,走	促,匙,很,赴	@	西		西 酉
#	井	廿,开,廾,止,豆	畊,革,研,共,步,立	"	ıı	川	业,齐
~	冖	⺌	冥 空,心	_	一	....	上,马
+	十	十,寸,ナ	卖,协,木,南	-	一		西
*	米	水,氺	粉,冰,恭	/	丿	ノ	戊,龙
^	人	入,八	木,余,分	\	㇏		廷
!	氵	讠	泳	?	了	了,子,丫	承,甬,疏
|	丨		丰 修	=	=	二	月,亏
<	:		豕	>	㇇	㇆,𠃌	买,也,为
:	:		冬	;	冫		冷
.	•(中)		为	,			
'				`	•(上) ノ		之,乏
(				)			
C				]			
{				}			
0				5	5	ㄣ,ㄋ	与,马,乌,亏
1				6			
2				7	7	丁,丁	马,乌,司
3	廴	彡	廷,参	8	言		喑
4	忄		忙	9			

Table 1