1986

A method of storage and display of Chinese characters as graphic symbols

Lo-yi Chung

Rochester Institute of Technology
School of Computer Science and Technology

A Method of Storage and Display of Chinese Characters As Graphic Symbols

by
Lo-yi Chung

A thesis, submitted to the Faculty
of the School of Computer Science and Technology
in partial fulfillment of the requirements
for the degree of
Master of Science in Computer Sciences.

approved by: _____
Professor Peter Anderson

_____
Professor Guy Johnson

_____
Professor Evelyn Rozanski

date May 28, 1986

Title of Thesis: A Method of Storage and Display of Chinese Characters as Graphic Symbols

I, Lo-yi Chung hereby grant permission to the Wallace Memorial Library, or RIT, to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: July 16 '86 _____

Acknowledgement

I want to thank my family for their understanding and support. Special thanks to Dr. Robert Frisina and Mr. John Whitely for their support of the request for leave of absence during 1984/85 year. Encouragement, guidance and careful editing of the thesis by Dr. Peter Anderson, Mr. Guy Johnson, and Ms. Evelyn Rozanski are greatly appreciated. Without them it would not have been possible to have completed the degree requirements.

Table of Contents

INTRODUCTION

The majority of Chinese characters are picture-like symbols. Some of them were derived from their actual resemblance to a natural scene or to the shape of objects; others were abstract symbols of concepts, of similar pronunciation, and of composites of other symbols. The lack of an alphabet-like structure in Chinese characters makes internal representation, input/output and storage difficult under the current computer environment.

This thesis is concerned with the subject of storage and display of Chinese characters as graphic symbols. The hypothesis is that it is possible to save in storage by keeping similar parts of Chinese characters only once. The display method involves retrieving the parts and transformation needed, and showing each part to put the character together.

In addition to the review and library search, the author had the opportunity to spend 10 months in Taiwan, the Republic of China, to investigate the current Chinese computer systems and experiments. Description of current literature, development, and actual systems are included in the Literature Review chapter.

The UCSD PASCAL system is used to implement the thesis project on an IBM Personal Computer with 128K of memory and graphic capability. Computer graphics is the vehicle for display, and a link list method is used to store and retrieve Chinese characters. The program allows the user first to define basic components, or parts; then to create Chinese Characters via the defined components; finally, to retrieve Chinese characters by component. Discussion of the system specification is included in the Project Description chapter.

Conclusion and further recommendations are presented in the last chapter. Terms used throughout the report are defined in the Glossary section of the Introduction chapter.

1.1 Glossary

Component — a Chinese character or a root; it is a unique symbol, usually not easily generated with other symbols. The criteria used in this proposal is that a basic component should be recognized as a common form with no more than 8 lines.

Character — a Chinese character is a symbolic representation of Chinese language, usually conformed to a square box.

Root — a part of a Chinese character. There are hundreds of roots, some of them are characters themselves. Most of the roots are only components of other characters. Roots are also known as "radicals", see Appendix I for examples.

Chinese Phonetic System

There are 21 consonants, 16 vowels, and 5 tones. A Chinese character can be 'pronounced' by : one vowel, two vowels, one consonant and one vowel, or one consonant and two vowels. One of the five tones is added to further differentiate the sound of a character.

LITERATURE REVIEW

The purpose of the literature review is to describe Chinese characters in computer applications. The present chapter opens with a general description of Chinese characters. Current implementations of computer systems in Chinese are reviewed, and their methods are analyzed. Finally, display and storage of Chinese characters are further reviewed.

2.1 CHINESE CHARACTERS AND COMPUTERS

The data processing industry and computer development have been centered around the English language. The keyboard, printer, and internal exchange code have all been standardized for English for many years, and adapted for other Roman language systems. However, there are some fundamental differences between English, a phonetic, alphabet-based language, and Chinese, a picture-like language. There are three inherent traits of Chinese characters that make it difficult for computer processing:

First of all, the shape of Chinese characters is different from English. Most of the Chinese characters are pictographs. It is difficult to analyze and classify them into a limited number of parts, such as the alphabet in English. In addition, there are many homographs (same shape but different meaning), same meaning but different shapes (abbreviated vs. formal whole writing). There are also at least six popular fonts, such as Old Sung, Thin Black, etc. (see Appendix II). The sizes of a character may also affect the arrangement of its strokes in the square space. In other words, shapes of Chinese characters not only differ according to types of fonts and customs of writing, but also vary when sizes change.

In 1982, the Education Ministry published the Often-used Chinese Character Standard Shape Table and the Next to the Often-used Chinese Character Standard Shape Table. They are the first standardization efforts of the shape of Chinese characters. However, it takes time for the standard to be generally implemented. Even now, some publishers continue to correct or make new fonts.

Secondly, the number of Chinese characters is enormous. Currently, most applications on microcomputers contain about 16,000 Chinese characters with room for user-defined additions. The Often-used Chinese Character Standard Shape Table collects 4,808 characters. The subsequent Next to Often-used Chinese Character Standard Shape Table contains 10,740 characters. The Chinese Standard Exchange Code published by the Information Science Promotion Task Force has 13,053 characters. Both efforts are considered not comprehensive, but sufficient for daily use. An extensive Chinese Dictionary published by the Chinese Culture University contains 49,905 characters. Other popular dictionaries contain between 6,500 and 12,000 characters.

Third, the order of Chinese characters is not standardized. There are several indexing methods. The traditional root method is used in the dictionary; all characters are first grouped by roots and second by the number of strokes other than the root. The Chinese phonetic symbols method is well taught in grade school, has been very popular since the 1940's,

and is now being used as an appendix in the dictionary. The telephone book adapts the number-of-strokes method which counts the total number of strokes in each character and arranges them in ascending order.

There are also many other methods that observe the "corner" characteristics of a character and code it accordingly. Using all of the above one usually can not find the character in one step; a second or third selection is required. The writing of Chinese characters (the order of strokes) has a few basic rules, such as from left to right and top to bottom, but there is no standard order of strokes for each character.

2.2 IMPLEMENTATION OF COMPUTER SYSTEMS IN CHINESE

The non-standardized feature in today's Chinese computer provides an open field for competition. It also creates a desirable atmosphere to design systems to fit the needs of the users. The number of Chinese characters collected and stored in the computer, the input method, the storage and display of Chinese characters, and the output selection are specific as to the requirements of an application.

2.2.1 INPUT METHOD

Each Chinese character occupies a square space, and has a shape and meaning. The regular ASCII keyboard and representation has limited usage for inputting Chinese characters. In practice, ingenious ways are designed to accommodate input methods for Chinese characters. In general, the regular keyboard is preferred over the larger keyboard on small or

personal systems. The regular keyboard is especially attractive in combination with microcomputers for novice computer users. The larger systems, when a trained typist is available, is better suited to a large keyboard designed for Chinese characters. It was estimated that an experienced operator on a large keyboard can type three times as fast as an ordinary user on an ASCII keyboard.

Given the importance of the keyboard and the lack of a convenient way of inputting Chinese characters, input is considered the "bottleneck" of the development of Chinese computer systems. In a survey conducted by the Institute for Information Industry, it was found that there were 31 input methods as of February 1983. The following classification is an attempt to briefly describe them.

2.2.1.1. Direct keyboard entry.

A special Chinese keyboard is designed as part of the computer system. This keyboard contains the often used Chinese characters,

usually in the neighborhood of 6,000. This keyboard is large in size (about 23" in length, 15" in width, and 5" in height; weight about 26 pounds) and requires special training to operate. The characters are arranged in groups by frequency of use, roots or by the Chinese Phonetic Symbols, as specified by the user. Groups of Chinese characters are then arranged on several printed and laminated pages. The keyboard is touch sensitive; the operator selects a page first and flips the page on top of the keyboard. The operator touches the group first, and then specifies the position of the character wanted in the group. A beginning operator can type at 55 characters per minute if provided with 14 hours training. An experienced operator can type over 100 characters per minute.

2.2.1.2. ASCII keyboard entry

There are three common input methods that use ASCII keyboard.

a) Coding entry. A collection of Chinese characters are each assigned a unique code of a combination of A-Z, and 0-9.

b) Phonetic symbol entry. The regular ASCII keyboard is masked with the 37 phonetic symbols. Since the phonetic symbols are used in all grade schools, it is an easy way for most of the people to locate a character they know how to pronounce. However, there are many homonyms and a second selection is usually required to find the character wanted. A casual user can type roughly 20 characters per minute.

c) Root and part selection and combination. Defined roots and parts are assigned to the ASCII keyboard. When correctly selected in the right order, the character appears. Some training and familiarity with the language is required.

2.2.1.3. Optical Scan, Character recognition,light pen, and voice input.

No commercial product is available yet. Many researchers believe this is the ultimate solution to the problems of inputting Chinese characters into computers. The main reason for developing optical scan and voice input is to avoid the tedious steps required in a keyboard entry method, and to take advantage of the fact that each Chinese character is a single syllable unit. The Institute for Information Industry exhibits an experimental voice input system which takes a human voice, analyzes the pattern, compares it with prerecorded voice pattern, and shows the characters on the screen.

2.2.2 INTERNAL REPRESENTATION

Another concern is the internal representation for Chinese characters. Although there are many commercial products, there is still no standard exchange code for Chinese characters. In a study done in 1982 (Chen, 1982), there are 22 different internal codes among 31 Chinese computer products. The main reason is that there is not yet a collection of "all" Chinese characters. While it is possible to create a new English word through the use of a new combination of alphabets, a new Chinese character means a new assignment and an additional code to be added. The most frequently encountered new characters are those created in a person's name. Under consideration now is the limitation of the choice of names to available Chinese characters; it is still far from a legally and morally accepted rule.

Since most of the internal codes are arbitrarily assigned, a sort by internal code is meaningless. Sorting is usually done via a table built separately

by the manufacturers according to user needs. Many users also employ additional fields in their data base to indicate sorting order. In many cases an additional sorting field is cheaper than programming support from the manufacturers.

A task force has been formed by the government to discuss and recommend a standard exchange code. Two sets of exchange codes have been recommended: one for general use; the other for library and publishing use.

2.2.3 CHARACTER GENERATION

Special attention is also given to the appearance of the Chinese characters. Since Chinese characters as they are used in daily life (such as in newspapers) have an average of 8-9 strokes, a 16 x 16 matrix is the minimum requirement. Other improved images use 24 x 24 (IBM 3250 system), or even 40 x 40 (such as the Sha-ken, a Japanese Company, composition machine for printers).

There are two methods to create the original character sets. One is to prepare and digitize each character separately. The other one is to prepare common "parts" and then compose a character on screen. The first one is considered to have a better visual quality; however, it is expensive to prepare. Companies often patent their characters to protect such an investment. The second method is more economical both in preparation and in storage, and is considered "good enough" for users at home or in an office environment.

For specialized fonts, such as the ones used in headlines of newspaper, a set of boundary parameter is saved and stored. When the character is recalled, the boundary will be drawn first and then filled with outline color. The IPX (Ideographix, Inc.) system and ACCFONT, an experimental character generation system, both use this method in character display.

2.3 FURTHER REVIEW ON THE STORAGE AND DISPLAY OF CHINESE CHARACTERS.

Since the storage and display of Chinese characters are the main themes of the proposed thesis, three storage and display algorithms are further described. Because of the graphical nature of Chinese characters, all current display methods employ graphic mode and some forms of drawing.

2.3.1 VECTOR DRAWING

The Super Chinese DOS (SCDOS) and Super Chinese DOS.Extended (SCDOS.E) is a Chinese character generator for APPLE II and compatible computers. It is a typical example of low-cost (about U. S. $50) Chinese systems on microcomputers.

It uses roots and parts as the primary index method, and the Chinese Phonetic Symbols as the secondary index method. The ASCII keyboard is masked with both roots and parts, and the phonetic symbols. Each

character is represented by a code of 1-5 keyboard strokes. The order of the specified roots or parts is important in locating the character. A wild card is allowed in selecting the roots or parts. Both indexing methods will provide a group of characters when the given keystrokes are not specific enough.

SCDOS and SCDOS.E are one of the many systems that use the Chinese character set originally designed by P. F. Chou which uses a 16 x 16 matrix to show a character. The parts of a character are stored with starting and ending points, and lines are drawn in vertical, horizontal, or diagonal directions. It also allows a rectangle to be drawn when two diagonal points are given. Curved lines are not implemented.

The SCDOS and SCDOS.E provide 24,000 Chinese characters. There are also functions to create and store 665 additional new characters; new characters are generated by turning each pixel on or off in a 16 x 16 matrix. An interface with the EPSON printer is available.

2.3.2 ACCFONT

The ACCFONT (Automated Chinese Character Fonts) is a character generation system first published by K. Y. Cheng of the Institute of Information Science of the Academia Sinica in Taipei, Taiwan. The designers of ACCFONT recognized the fact that Chinese characters are primarily pictures and line drawings, and seek a mathematical solution to represent lines. If there are enough generalities, the mathematical formulae become a powerful representation to character strokes. The basic look of a Chinese character can be saved, and other typography can be generated through different sets of mathematical formulae. An underlying mission is to search for primitive patterns so that pattern recognition will be possible.

The ACCFONT system runs on a VAX machine. The theoretical base is that each Chinese character can be regarded as a graphic pattern. The pattern primitives are the strokes. The strokes can be combined into various radicals and radicals can be combined into

various Chinese characters. Therefore, every Chinese character can be generated syntactically by a pattern grammar. It is similar to D. E. Knuth's METAFONT system for the English alphabet. T. Y. Mei, J. Hobby and G. Gu also proposed and completed similar character generation systems for Chinese characters.

ACCFONT uses the cubic b-spline curve formulation to draw various curved lines in a Chinese character. It uses a tree structure to store various parts of a character. The operator needs to be familiar with the "commands" of the system. After the commands are accepted by the computer, the syntax parser then categorizes the position, length, slope, etc. of each stroke. This information is fed into another program to produce the desired shape. The result is put together in a window and output to a graphic CRT screen. The designer can correct previous commands and repeat the process until a satisfactory result is obtained and stored. Once a character pattern is stored, additional commands allow the output of such a character pattern in various fonts on screen or from a plotter.

2.3.3 BIT PATTERN CHARACTERS

Since the 16 bit microcomputer became available, more and more Chinese systems are designed and implemented with bit pattern Chinese characters.

The display advantage seems obvious; better visual quality quickly attracts general office users. The storage tends to be large. A 24 x 24 dot pattern character occupies 72 bytes in computer memory (24 x 24 /8 = 72). However, with large memory chips becoming cheaper, the concern for memory size diminishes rapidly.

The Institute for Information Industry assisted IBM in developing their Chinese computer characters, and later modified and edited the character set according the the Often-used Chinese Standard Shape Table, and the Next to the Often-used Chinese Standard Shape Table. The result is published as a technical report -- Chinese Character Dot Pattern and Internal Code Table with 3 disks of Chinese Character dot pattern in

IBM PC format. The report is for sale at a surprisingly low price (U. S. $45). Many Chinese computer and system manufacturers are developing firmware and software based on this Chinese character set. They are producing a group of middle price range products for serious office users. This set contains 13,494 characters, including not only the characters in the Often-used Chinese Standard Shape Table and the Next to the Often-used Chinese Standard Shape Table, but also characters that appear in the grade school text book. A set of Chinese characters in 36 x 36 dot pattern is under development by the Institute for Information Industry and will be available in 1986.

PROJECT DESCRIPTION

A PASCAL program was written to demonstrate that Chinese characters can be adequately constructed using standard computer graphic techniques. The program is primarily menu-driven. Figure 1 shows the main menu.

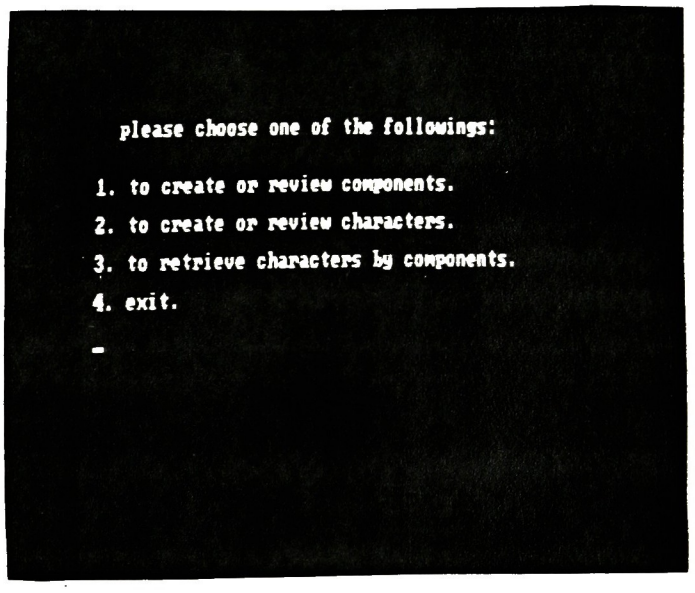


Fig. 1 The main menu screen

3.1 Functional Specifications

The main program logic centered around three categories of activities:

part I - create and storage of basic components.

part II - create and storage of characters using basic components.

part III - retrieval of characters by components.

Within each part, the user can direct the computer to do various tasks.

3.1.1. Part I

Part I consists of the following tasks. Figures 2 and 3 demonstrate the display of components and how to define a component.

add lines

A line can be defined by specifying the starting and ending points. Lines segments are the basis for constructing a component.

erase lines

While constructing a component, a line can be added or erased.

see completed components

Once a component is completed, there is an option to show the component on screen.

store components

Before leaving a session of creating components, an option is given to store newly added components on a disk or to leave without storing the results of the session.

load components from a disk file

All components stored on a disk can be retrieved and displayed on screen.

delete a component

Once a components is completed, lines can no longer be erased; an option to delete the component is available.

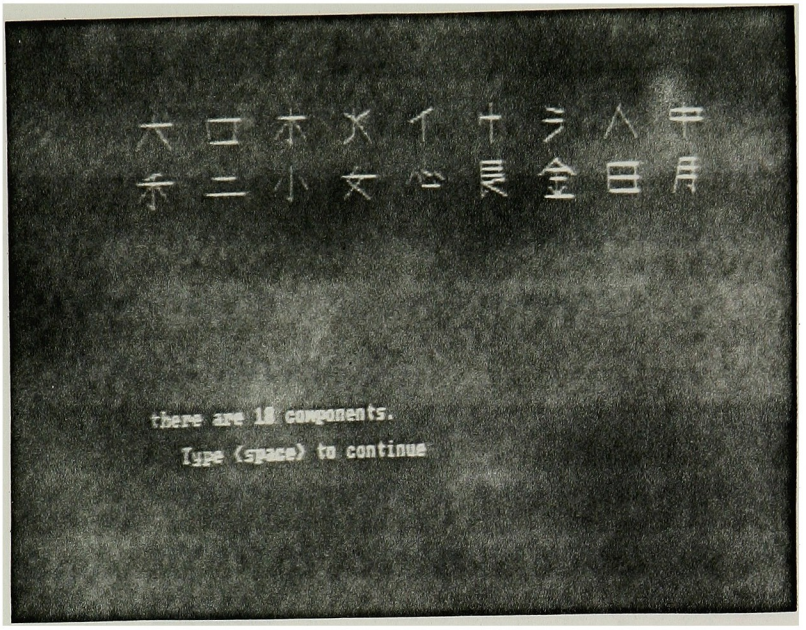


Fig. 2 Display of components

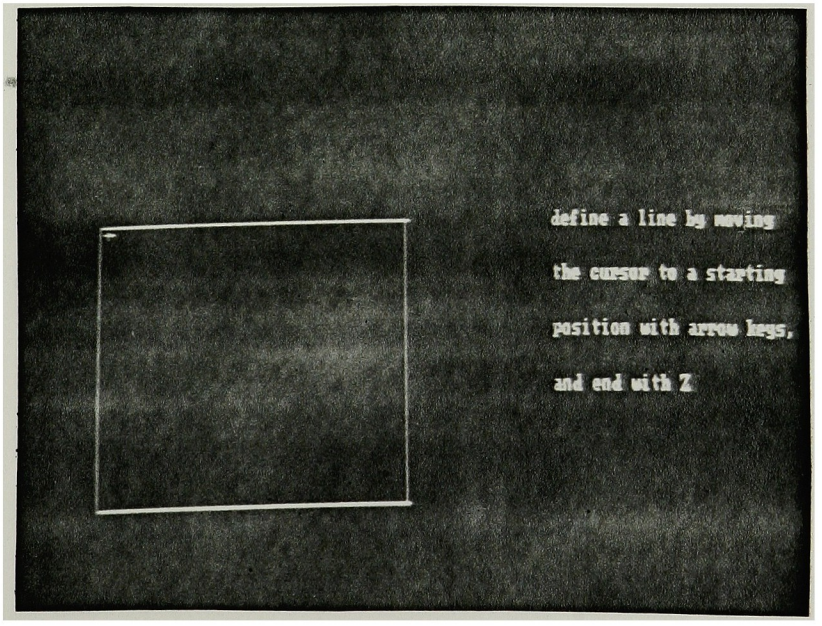


Fig. 3 Defining a line of a component

3.1.2. Part II

Part II, Create and Storage of Characters, consists of the following functions. Figures 4, 5 and 6 show the process of selecting components, and the composition of a character from selected components.

create a new character

A character is defined by putting components together. First, a component is selected; second, the position of the component is defined.

add parts

Up to 8 components can be added to compose a character.

modify parts

There is an option to change selected components in a character to another component.

store completed characters

When a character is completed, it can be saved on a disk or discarded.

load characters from a disk file

Once a character is saved on a disk, it can be retrieved, displayed and modified as needed.

delete a character

After a character is created, it is possible to delete it from the collection.

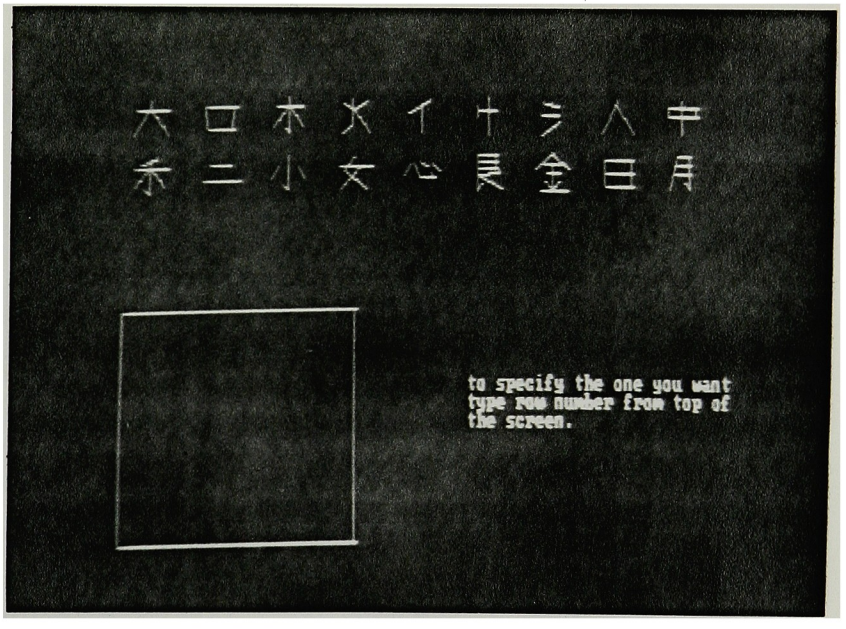


Fig. 4 Selecting a component to create a character

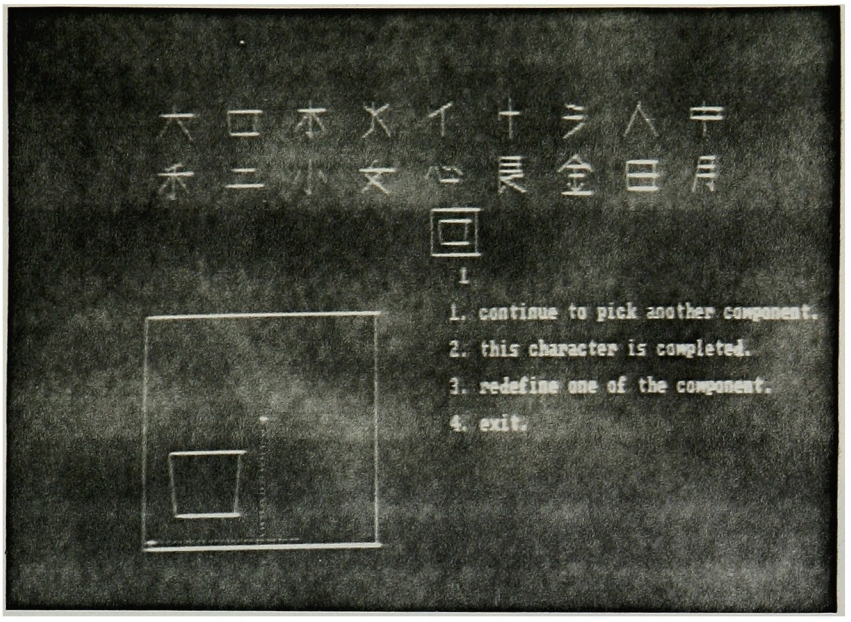


Fig. 5a Positioning the component in creating a character.

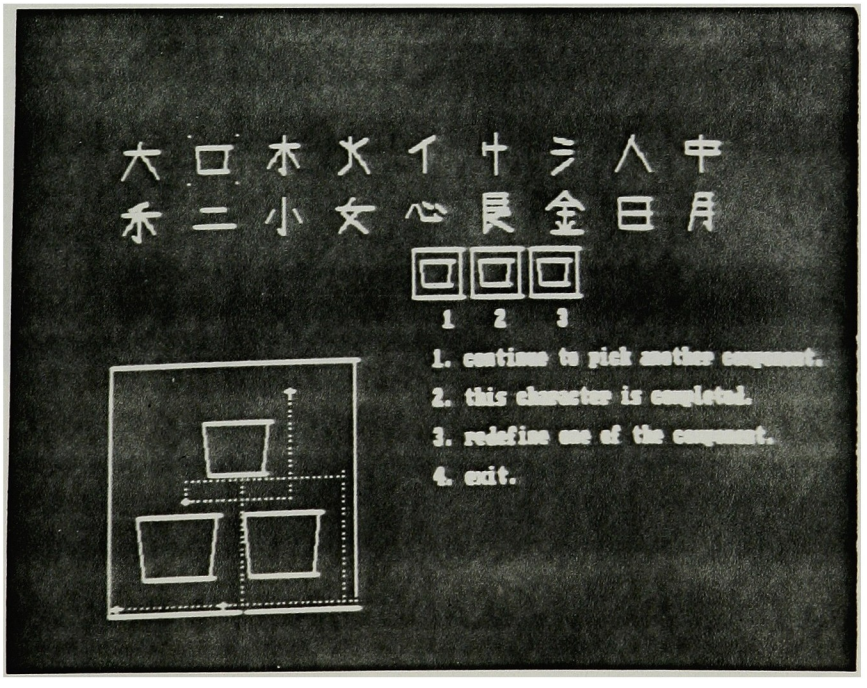


Fig 5b Creating a character with 3 components.

Fig. 6 Display of all completed characters.

3.3.3. Part III

Part III, Retrieval of a Character by Components, has the following functions. Figures 7 and 8 display a successful retrieval.

select a component

In retrieving a character, a component is first chosen.

retrieve a character or characters by components

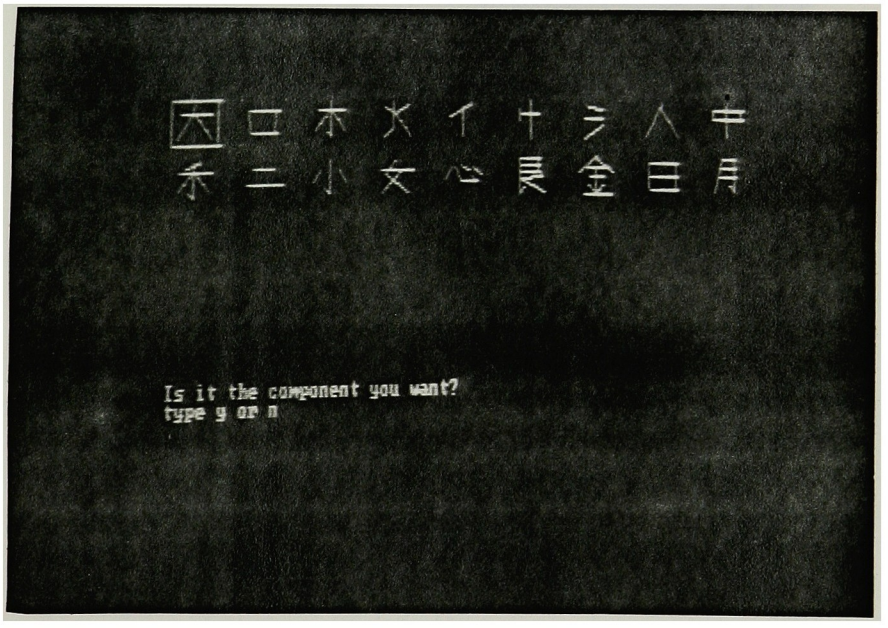


Fig. 7 Select a component for retrieval of characters containing the component.

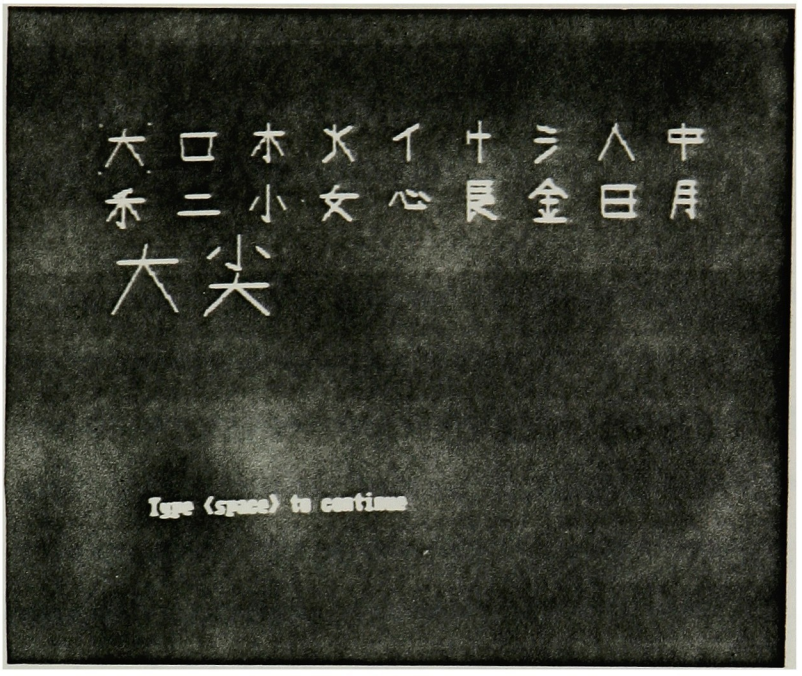


Fig. 8 Display of retrieved characters.

3.2 User Inputs and Outputs

The user is required to select from the menu screen each of the possible actions. In most cases, one keystroke will be sufficient.

It is necessary for the user to be familiar with the basic concept of using a 'disk file' as a storage device, and recognize main memory as temporary. The first part, creating components, outputs a component file on disk. The second part, creating characters, outputs a character file on disk. Both files are then used as input to the third part, retrieving a character.

The program was designed to 'degrade gracefully' when errors such as an invalid file name are encountered. The user will be given opportunities to recover from a mistake.

3.3 Limitations and Restrictions

The user should have some knowledge of the Chinese language to create components and characters effectively.

In this project, part I and part II were designed to demonstrate the graphic techniques and to generate a component file and a character file as the input files for part III. The number of components and characters displayed was limited by the size of the screen and the memory used by the UCSD PASCAL system.

In the absence of a 'mouse' device or a lightpen, the cursor movement and user responses were restricted to keyboard activities.

There is no printer facility that prints the graph screen; therefore, the system's output is only available on the screen.

3.4 System Specifications

The UCSD PASCAL system on IBM Personal Computer comes with many features and graphics capability. The ones that are used in the thesis project are discussed in this section.

3.4.1 Graphic functions

In addition to the UCSD PASCAL graphic functions, several general graphic functions are specified based on Newman and Sproull (1979). They are: start-at, line-to, set-window, set-viewport, world-to-screen, aspect-ratio, apply, transfer and scale. Since the UCSD PASCAL system on IBM PC does not provide a visible cursor when in high resolution black and white graph mode, a cursor was displayed through the use of a procedure. These fuctions are stored in the system library and used throughout the program.

3.4.2 Graph mode and text mode

The graph mode was used to display both text and graphics. Several 'text windows' were defined and used for display of menu selection. Once defined,

menu selection. Once defined, text windows were cleared before the next display with reset of viewport.

The text mode was called to display the four introduction text screens. The advantage of the text mode is that text display is easier with the use of the WRITELN statement, and it has a blinking cursor.

3.4.3 Units and Segments

Units and Segments were used to manage program files within the memory confines of the UCSD PASCAL system on an IBM Personal Computer. Units stay in the main memory all the time. However, they can be compiled separately. They are helpful in editing and compiling large programs. Segment routines may be swapped independently of the main program during execution, and therefore, can be used to help manage run time main memory.

The graphic functions were compiled separately as units, and included in the standard PASCAL system library, because they are used throughout the program.

The procedures that are unique to each of the three parts (creating components, creating characters, and sorting characters) were grouped under three "segments": segment procedures CREATE_COMPONENT, CREATE_CHARACTER, and SORT_CHARACTER. Procedures used by two or all of the three parts were included in the main program. The reason for using segment is primarily for the economy of main memory during execution of the program.

3.4.4 Graphics applications

The user defines line segments of a component on the screen with the use of the arrow keys, the program calculates the parameters of the line segments and records in the component record. Similarly, the user's selection of a component in creating a character is done on screen, the component's position in the character is calculated into scaling and translating factors and stored in a character record.

Components were treated as 'instances' during the drawing of a character. There are two transformation matrices: one for the component transformation in the character, the other for the transformation information of the character itself.

3.4.5 Records and pointers

Throughout part I and part II, there are two major kinds of records: component and character.

The component record has the following fields:

id

It is zero when the component is being deleted;

number of lines

It is the number of line segments contained in the component.

lines

Lines are array of starting and ending points of the line segments.

The character record has the following fields:

id

It is zero when it is being deleted.

number of parts

Number-of-parts is the number of components included in this character.

pointers

Pointers are arrays of component id.

transformation information

It is the scaling and transformation factors for each of the components included in the character.

See Appendix II for examples of actual contents of the component record and the character record.

In part III, retrieving characters by a specified component, additional pointers are added to the component record and to the character record. The additional pointer fields are used to make a link list for each of the component. Briefly, each component points to the component in a character that uses it. Each component part in a character then

points to the next character that uses it, until the pointer ends with a value of zero.

CONCLUSION

The thesis project demonstrates that it is possible to design a system to create, store and display Chinese characters on a microcomputer with graphic capability. The IBM Personal Computer seems adequate for the project. The UCSD PASCAL system is quite limited in terms of the use of available memory (up to 128K of memory no matter how much more is available), and its reliability in handling large programs (system crashed often when program exceeded 1500 lines).

The interactive feature proves to be necessary because it provides immediate feedback on the screen. The use of a menu makes sure that the user's response is valid and is a good way to direct the user to different levels of activities. The main logic of the whole project eventually relies on the design of the menu, which deserves more attention at the time of the initial design.

Other aspects of the project as a system for creating Chinese characters are:

1. Viewing quality of characters

   In general, the system is capable of generating characters that are less 'stiff' than the Super Chinese DOS System. However, it is difficult to compete with the thickened lines and rounded corner of the dot pattern characters. The quality of displayed Chinese characters via computer technology can be improved at the expense of memory, no matter which method is used.

2. Memory

   A considerable amount of memory is saved when a large number of Chinese characters is input to the system, because there are more duplicate parts. Memory consumption for 18 components and 20 characters takes 1120 bytes of main memory (18 x 10 + 47 x 20). If characters are stored as dot matrix characters: twenty of the 24 x 24 dot matrix characters take 1440 bytes (24 x 24 / 8 x 20), and twenty 40 x 40 dot matrix pattern characters uses 4000 bytes of memory

(40 x 40 / 8 x 20). As the number of components and characters increase, the memory requirement does not accelerate as those of the dot matrix characters. However, this method requires that all components reside in the main memory while a character is being composed or displayed. Dot matrix characters do not have such a restraint.

The utimate saving lies with the possible extension of the concept of meta-font. Assuming a basic form of a Chinese character is formed, the parameter list could be extended to include enlargement, reduction, and slanting factors. Various typefaces and different sizes of the same word can be generated based on the one description of this meta-font. Dot matrix characters do not have such a possible extension.

3. Flexibility

The preparation for Chinese characters as graphic symbols is less mechanical than the dot matrix characters. Certain aesthetic judgement becomes part of the process. The initial analysis of parts and the

relative position of parts in each character will be of vital importance.

Dot matrix characters can not be modified easily; they are also more expensive to prepare. Those prepared as part of this thesis can be modified without major efforts.

Hofstadter(1985) in "Metafont, Mathematics and Metaphysics: Comments on Donald Knuth's Article 'The Concept of a Meta-Font'" raised some interesting arguments. He questioned the existence of such a complete parameter list through which all possible variations of a meta-font, such as "A", can be generated. Chinese characters are much more complicated symbols than alphabets. As the author of the ACCFONT system, K. Y. Cheng indicated, if there are as many rules as there are exceptions, then rules will not achieve the economy as expected. Nevertheless, Meta-Font, ACCFONT, and many other research efforts have demonstrated that for a finite set of typefaces, English or Chinese, such a methodology will work. In applying computer

technology to printing, and to word processing, this method is adequate for a major portion of the Chinese vocabulary.

RECOMMENDATIONS FOR FURTHER STUDIES

- It may be possible to use the 'paging' concept to display large numbers of characters or components on the screen. Each page is separated by a keystroke from the user.

- Improve user interaction with a 'mouse' device, or a light pen, or a masked keyboard for selection of components.

- The current system may be applied to signmaking and other graphic symbol communication systems.

- When memory is not a problem, curved lines should be given as an option.

- Additional parameters can be added to modify component size and the margin between components when the character size changes.

Bibliography

Chen, Cheng-kuan "Applications in Chinese Computer" Information and Computer Vol. 3, no. 5, Nov. 24, 1982, p. 25-27, in Chinese.

Cheng, Kao-young, "Computer Calligraphy, ACCFONT", Information and Computer Vol. 3, no. 5, Nov. 24, 1982, p. 79-85, in Chinese.

Cheng, Kao-young, "A Brief Introduction to the Computer Graphic Research Results of the Institute of Information Science of the Academia Sinica", Computer Quarterly, Vol. 17, no. 2, Summer 1983, p. 2-15+, in Chinese.

Cheng, Kao-young and K. J. Chen, "A System Model for Chinese Character Fonts Generation", Proceedings of ICTP 1983, Tokyo, October 17-18, 1983, p. 53-61.

Education Ministry, Often-used Chinese Standard Shape Table, Taipei, Taiwan, Cheng-chen Book Company, June 1982.

Education Ministry, Next to the Often-used Chinese Standard Shape Table, Taipei, the Education Ministry, Oct. 1982.

Fan, Fung-ming "Computers Will Listen As You Talk", Information and Computer, Vol. 3, no. 5, Nov. 24, 1982, p. 69-74.

Hobby, John D. and Guoan Gu " A Chinese Meta-Font", Proceedings of ICTP 1983, Tokyo, October 17-18, 1983, p. 62-67.

Hofstadter, Douglas, Metamagical Themas: Questing for the Essence of Mind and Pattern, New York, Basic Books, Inc., c1985.

Hwong, Teh-wong "Often-used Characters in Data Processing", Computer Quarterly, Vol. 18, no. 2, Summer 1984, in Chinese.

BILBIOGRAPHY

Institute for Information Industry, Chinese Character Dot Pattern and Internal Code Table (with 3 disks in IBM PC format), Taipei, Taiwan, Institute for Information Industry, May 1984, Technical Report no. C-26, in Chinese.

Institute for Information Industry, Evaluation Report of the Survey of Input Methods and Input Hardware of Chinese Computer, Taipei, Taiwan, Institute for Information Industry, Jan. 1984. Technical report no. C-21, in Chinese.

Institute for Information Industry, Guide to Chinese Computers, Taipei, Taiwan, Institute for Information Industry, Sept. 1983, in Chinese.

IPX 4680 Automatic Chinese Phototypesetting system, Sunnyvale, California, Ideographix, Inc., no date, in Chinese.

Liu, Choa-shung and Pei-chi Ho, Apple II Super Chinese DOS - Super Chinese DOS.E, Rev. ed., Taipei, Jou-lin Books Co., Oct. 1983, in Chinese.

Newman, William M. and Sproull, Robert, Principles of Interative Computer Geraphics, 2nd edition, New York, McGraw-Hill, 1979.

Sha-ken Co. Ltd., Sha-ken Computer Composition Reference 130983E, 3130284BE, no date, in Chinese.

Wei, Dwan and Chang Ching-teh, "Stroke Distribution, Order of Stroke, and Input Method", Computer Quarterly, Vol. 16, no. 4, Winter 1982, p. 24-32, in Chinese.

Wu, Chia-lin, "Why There Is No Standard Chinese Character Exchange Code", Information and Computer, Vol. 3, no. 5, Nov. 24, 1982, p. 22-23, in Chinese.

Yuan, Wei-hsin, "Light, Computer, Chinese", Central Daily News, Jan. 17, 1985, p. 12 and Jan. 18, 1985, p. 12, in Chinese.

PRIVATE CONVERSATIONS AND VISITS

Chen, K. J., Researcher, Academica Sinica, Nov. 1984.

Chen, Ptolemy, Asst. Sales Manager, Homtai Computer Corp. (Dealer for Apple computer), April 1985.

Cheng, Kuo-young, Researcher, Academica Sinica, Nov. 1984.

Huang, T. J., President, Systex Corp., Feb. 1985.

Huang, W. D., Giantek Technology Corp., Nov. 1984.

Ma, Geng-shin, System Engineer, Institute for Information Industry, May, 1985.

Wong, Sales Representative, Sha-ken Co. Ltd. Oct. 1984.

Yeh, Chan-kwang, Vice President, Ideographix, Inc., Taiwan Operations, April 1985.

APPENDIX I

STANDARD ROOTS OF CHINESE CHARACTERS.

RADICAL INDEX

10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210,

0, ① 一 丨 丶 丿 乙 亅 二 亠 人 儿 入 八 冂 冖 冫 几 凵 刀 力 勹 匕 匚 匸 十 卜 卩 厂 厶 又 ③ 口 口 土 士 夂 夊 夕 大 女 子 宀 寸 小 尢 尸 屮 山 川 工 己 巾 干 幺 广 廴 廾 弋 弓 ⑤ 彐 彡 彳 ④ 心 忄 戈 戶 手 扌 支 攵 文 斗 斤 方 无 旡 日 曰 月 木 欠 止 歹 殳 毋 比 毛 氏 气 水 氵 火 灬 爪 爫 父 爻 爿 片 牙 牛 犬 犭 玄 玉 王 瓜 瓦 甘 生 用 田 疋 疒 ⑥ 癶 白 皮 皿 目 矛 矢 石 示 礻 禸 禾 穴 立 竹 米 糸 糹 缶 网 罒 羊 羽 老 而 耒 耳 聿 肉 月 臣 自 至 臼 舌 舛 舟 艮 色 艸 艹 虍 虫 血 行 衣 衤 襾 西 ⑦ 覀 見 角 言 谷 豆 豕 豸 貝 赤 走 足 身 車 辛 辰 辵 辶 邑 阝 酉 釆 里 ⑧ 金 長 镸 門 阜 隶 隹 雨 青 非 ⑨ 面 革 韋 韭 音 頁 風 飛 食 首 香 ⑩ 馬 骨 高 髟 鬥 鬯 鬲 鬼 ⑪ 魚 鳥 鹵 鹿 麥 麻 ⑫ 黃 黍 黑 ⑬ 黹 ⑭ 黽 ⑮ 鼎 鼓 鼠 ⑯ 鼻 齊 ⑰ 齒 龍 龜

9, 19, 29, 39, 49, 59, 69, 79, 89, 99. 109, 119, 129, 139, 149, 159, 169, 179 189, 199, 209,

APPENDIX II

POPULAR CHINESE FONTS

(copied from IPX4680 Automatic Chinese Phototypesetting System pamphet p. 20)

中文字體樣本

楷書體

東海企業 IPX ideographix 光電排版

宋體

東海企業 IPX ideographix 光電排版

黑體

東海企業 IPX ideographix 光電排版

字號

東東東東東東東東

東東東東東東東東東東東東東東東東東

東東東東東東東東東東東東東東東東東東東東東東東東

APPENDIX III

EXAMPLES OF THE COMPONENT RECORD AND THE CHARACTER RECORD.

component1 大
number of line is 4
  line number 1
  startx = 8.00000
  starty = 2.80000E1
   end x = 3.20000E1
   end y = 2.80000E1
  line number 2
  startx = 2.00000E1
  starty = 3.60000E1
   end x = 1.90000E1
   end y = 2.80000E1
  line number 3
  startx = 1.90000E1
  starty = 2.80000E1
   end x = 6.00000
   end y = 5.00000
  line number 4
  startx = 1.90000E1
  starty = 2.80000E1
   end x = 3.40000E1
   end y = 5.00000
component2 口
number of line is 4
  line number 1
  startx = 7.00000
  starty = 2.90000E1
   end x = 9.00000
   end y = 9.00000
  line number 2
  startx = 9.00000
  starty = 9.00000
   end x = 3.00000E1
   end y = 9.00000
  line number 3
  startx = 3.00000E1
  starty = 9.00000
   end x = 3.20000E1
   end y = 2.90000E1
  line number 4
  startx = 3.20000E1
  starty = 2.90000E1
   end x = 7.00000
   end y = 2.90000E1

component3 木
number of line is 5
  line number 1
  startx = 9.00000
  starty = 3.00000E1
   end x = 3.00000E1
   end y = 3.00000E1
  line number 2
  startx = 2.00000E1
  starty = 3.70000E1
   end x = 2.00000E1
   end y = 6.00000
  line number 3
  startx = 2.00000E1
  starty = 6.00000
   end x = 1.70000E1
   end y = 9.00000
  line number 4
  startx = 2.00000E1
  starty = 3.00000E1
   end x = 7.00000
   end y = 1.10000E1
  line number 5
  startx = 3.20000E1
  starty = 1.10000E1
   end x = 2.00000E1
   end y = 3.00000E1

```
put your disk in the disk drive 1, and type name of the
CHARACTER file, and press return key.

cdata
character1
number of part is 2
  part number 1
  is component number 1
  its transformation:
       scale x is  9.25000E-1
       scale y is  5.50000E-1
   translate x is  1.00000
   translate y is  1.00000
  part number 2
  is component number 12
  its transformation:
       scale x is  9.00000E-1
       scale y is  4.50000E-1
   translate x is  2.00000
   translate y is  2.10000E1
character2
number of part is 1
  part number 1
  is component number 1
  its transformation:
       scale x is  9.25000E-1
       scale y is  9.25000E-1
   translate x is  1.00000
   translate y is  1.00000
character3
number of part is 1
  part number 1
  is component number 11
  its transformation:
       scale x is  9.50000E-1
       scale y is  9.25000E-1
   translate x is  1.00000
   translate y is  1.00000
character4
number of part is 2
  part number 1
  is component number 17
  its transformation:
       scale x is  4.25000E-1
       scale y is  9.50000E-1
   translate x is  1.00000
   translate y is  1.00000
  part number 2
  is component number 18
  its transformation:
       scale x is  6.25000E-1
       scale y is  9.75000E-1
   translate x is  1.50000E1
   translate y is  1.00000
```


```
character5
number of part is 3
  part number 1
  is component number 2
  its transformation:
       scale x is  5.00000E-1
       scale y is  5.00000E-1
   translate x is  1.00000
   translate y is  1.00000
  part number 2
  is component number 2
  its transformation:
       scale x is  5.00000E-1
       scale y is  5.00000E-1
   translate x is  1.80000E1
   translate y is  1.00000
  part number 3
  is component number 2
  its transformation:
       scale x is  4.25000E-1
       scale y is  4.25000E-1
   translate x is  1.20000E1
   translate y is  1.80000E1
```

APPENDIX IV

PROGRAM LISTING

```
  1   0    0:d    1  (*$L PRINTER:*)
  2   2    1:d    1  program thesis;
  3   2    1:d    1
  4   2    1:d    1  {This is a thesis project for a M. S. degree in computer science --      }
  5   2    1:d    1  {A METHOD OF STORAGE AND DISPLAY OF CHINESE CHARACTERS AS GRAPHIC SYMBOLS.}
  6   2    1:d    1  {It is written on an IBM PC using UCSD PASCAL.                          }
  7   2    1:d    1  {                                Lo-yi Chung                            }
  8   2    1:d    1  {                                Sept. 6, 1985, rev. Apr. 28, 1986      }
  9   2    1:d    1
 10   2    1:d    1
 11   2    1:d    1
 12   2    1:d    1
 13   2    1:d    1  type
 14   2    1:d    1    pitch_range = 0 .. 12;
 15   2    1:d    1    octave_range = 0 .. 7;
 16   2    1:d    1    onebit = 0 .. 1;
 17   2    1:d    1    twomax = 0 .. 2;
 18   2    1:d    1    twobits = 0 .. 3;
 19   2    1:d    1    threebits   0 .. 7;
 20   2    1:d    1    fourbits = 0 .. 15;
 21   2    1:d    1    font_pattern = packed array [0 .. 63] of boolean;
 22   2    1:d    1    font_table   array [128 .. 255] of font_pattern;
 23   2    1:d    1    font_ptr = ^font_table;
 24   2    1:d    1    str_ptr = ^string;
 25   2    1:d    1    key_table = array [1 .. 16] of str_ptr;
 26   2    1:d    1    key_ptr   ^key_table;
 27   2    1:d    1
 28   2    1:d    1
 29   2    1:d    1  function button (select : twobits) : boolean;
 30   2    1:d    1
 31   2    1:d    1  procedure paddle (select : twobits; var result : integer);
 32   2    1:d    1
 33   2    1:d    1  procedure note (pitch : pitch_range; octave : octave_range; duration : integer);
 34   2    1:d    1
 35   2    1:d    1  function lightpen (var charxpos, charypos, pixelxpos, pixelypos : integer) :
 36   2    1:d    2    boolean;
 37   2    1:d    1
 38   2    1:d    1  procedure setkeys (tableptr : key_ptr);
 39   2    1:d    1
 40   2    1:d    1  procedure videomode (mode : threebits);
 41   2    1:d    1
 42   2    1:d    1  procedure setfont (table : font_ptr);
 43   2    1:d    1
 44   2    1:d    1  procedure bkgnd_color (color : fourbits);
 45   2    1:d    1
 46   2    1:d    1  procedure palette (color : onebit);
 47   2    1:d    1
 48   2    1:d    1  procedure settime (hour, minute : integer);
 49   2    1:d    1
 50   2    1:d    1  procedure gettime (var hour, minute : integer);
 51   2    1:d    1
 52   2    1:d    1  procedure select_printer (unitnum : twomax);
 53   2    1:d    1
 54   2    1:d    1  procedure select_remote (unitnum : onebit);
 55   2    1:d    1
 56   2    1:d    1
 57   2    1:d    1
```

```
 58  2   1:d   1
 59  2   1:d   1     procedure display_scale( min_x, min_y, max_x, max_y : real );
 60  2   1:d   1
 61  2   1:d   1     function aspect_ratio : real;
 62  2   1:d   1
 63  2   1:d   1     function create_figure( x_size, y_size : real ) : integer;
 64  2   1:d   1
 65  2   1:d   1     procedure delete_figure( screen : integer );
 66  2   1:d   1
 67  2   1:d   1     procedure viewport( min_x, min_y, max_x, max_y : real );
 68  2   1:d   1
 69  2   1:d   1     procedure align_cursor( x,y : real );
 70  2   1:d   1
 71  2   1:d   1     procedure fillscreen( screen : integer ; shade : integer);
 72  2   1:d   1
 73  2   1:d   1     procedure background( screen : integer; shade : integer );
 74  2   1:d   1
 75  2   1:d   1     function read_pixel( screen : integer; x, y : real ) :integer;
 76  2   1:d   1
 77  2   1:d   1     procedure set_pixel( screen:integer;
 78  2   1:d                          x,y:real;
 79  2   1:d   0                      shade:integer);
 80  2   1:d   1
 81  2   1:d   1     procedure getfigure( source_screen : integer;
 82  2   1:d                          corner_x, corner_y : real;
 83  2   1:d   0                      copymode : integer );
 84  2   1:d   1
 85  2   1:d   1     procedure putfigure( destination_screen : integer;
 86  2   1:d                          corner_x, corner_y : real;
 87  2   1:d   0                      copymode : integer );
 88  2   1:d   1
 89  2   1:d   1     function read_figure_file( title : string ) : integer;
 90  2   1:d   1
 91  2   1:d   1     function write_figure_file( title : string ) : integer;
 92  2   1:d   1
 93  2   1:d   1     function load_figure( index : integer ) : integer;
 94  2   1:d   1
 95  2   1:d   1     function store_figure( screen : integer ) : integer;
 96  2   1:d   1
 97  2   1:d   1     procedure activate_turtle( screen : integer );
 98  2   1:d   1
 99  2   1:d   1     function turtle_x : real;
100  2   1:d   1
101  2   1:d   1     function turtle_y : real;
102  2   1:d   1
103  2   1:d   1     function turtle_angle : real;
104  2   1:d   1
105  2   1:d   1     procedure move( distance : real );
106  2   1:d   1
107  2   1:d   1     procedure moveto( x,y : real );
108  2   1:d   1
109  2   1:d   1     procedure turn( rotation : real );
110  2   1:d   1
111  2   1:d   1     procedure turnto(  direction : real );
112  2   1:d   1
113  2   1:d   1     procedure pen_mode( state : integer );
114  2   1:d   1
```

```
115  2   1:d   1     procedure pen_color( shade : integer );
116  2   1:d   1
117  2   1:d   1
118  2   1:d   1
119  2   1:d   1  const
120  2   1:d   1      sc_fill_len - 11;
121  2   1:d   1      sc_eol   13;
122  2   1:d   1
123  2   1:d   1  type
124  2   1:d   1      sc_chset        = set of char;
125  2   1:d   1      sc_misc_rec     - packed record
126  2   1:d   1                        height, width : 0..255;
127  2   1:d   1                        can_break, slow, xy_crt, lc_crt,
128  2   1:d   1                        can_upscroll, can_downscroll : boolean;
129  2   1:d   1                      end;
130  2   1:d   1      sc_date_rec     = packed record
131  2   1:d   1                        month : 0..12;
132  2   1:d   1                        day :   0..31;
133  2   1:d   1                        year :  0..99;
134  2   1:d   1                      end;
135  2   1:d   1      sc_info_type      packed record
136  2   1:d   1                        sc_version : string;
137  2   1:d   1                        sc_date : sc_date_rec;
138  2   1:d   1                        spec_char : sc_chset; {Characters not to echo}
139  2   1:d   1                        misc_info : sc_misc_rec;
140  2   1:d   1                      end;
141  2   1:d   1      sc_long_string  = string[255];
142  2   1:d   1      sc_scrn_command = (sc_whome, sc_eras_s, sc_erase_eol, sc_clear_lne,
143  2   1:d   1                         sc_clear_scn, sc_up_cursor, sc_down_cursor,
144  2   1:d   1                         sc_left_cursor, sc_right_cursor);
145  2   1:d   1      sc_key_command    (sc_backspace_key, sc_dc1_key, sc_eof_key, sc_etx_key,
146  2   1:d   1                         sc_escape_key, sc_del_key, sc_up_key, sc_down_key,
147  2   1:d   1                         sc_left_key, sc_right_key, sc_not_legal);
148  2   1:d   1      sc_choice       = (sc_get, sc_give);
149  2   1:d   1      sc_window       - packed array [0..0] of char;
150  2   1:d   1      sc_tx_port      - record
151  2   1:d   1                        row, col,             { screen relative}
152  2   1:d   1                        height, width,        { size of txport (zero based)}
153  2   1:d   1                        cur_x, cur_y  : integer;
154  2   1:d   1                                  {cursor positions relative to the txport }
155  2   1:d   1                      end;
156  2   1:d   1
157  2   1:d   1    procedure sc_use_info(do_what:sc_choice; var t_info:sc_info_type);
158  2   1:d   1    procedure sc_use_port(do_what:sc_choice; var t_port:sc_tx_port);
159  2   1:d   1    procedure sc_erase_to_eol(x,line:integer);
160  2   1:d   1    procedure sc_left;
161  2   1:d   1    procedure sc_right;
162  2   1:d   1    procedure sc_up;
163  2   1:d   1    procedure sc_down;
164  2   1:d   1    procedure sc_getc_ch(var ch:char; return_on_match:sc_chset);
165  2   1:d   1    procedure sc_clr_screen;
166  2   1:d   1    procedure sc_clr_line (y:integer);
167  2   1:d   1    procedure sc_home;
168  2   1:d   1    procedure sc_eras_eos (x,line:integer);
169  2   1:d   1    procedure sc_goto_xy(x, line:integer);
170  2   1:d   1    procedure sc_clr_cur_line;
171  2   1:d   1    function  sc_find_x:integer;
```

```
172  2   1:d   1   function  sc_find_y:integer;
173  2   1:d   1   function  sc_scrn_has(what:sc_scrn_command):boolean;
174  2   1:d   1   function  sc_has_key(what:sc_key_command):boolean;
175  2   1:d   1   function  sc_map_crt_command(var k_ch:char):sc_key_command;
176  2   1:d   1   function  sc_prompt(line :sc_long_string; x_cursor,y_cursor,x_pos,
177  2   1:d   1                       where:integer; return_on_match:sc_chset;
178  2   1:d  21                       no_char_back:boolean; break_char:char):char;
179  2   1:d   1   function sc_check_char(var buf:sc_window; var buf_index,bytes_left:integer)
180  2   1:d                            :boolean;
181  2   1:d   1   function space_wait(flush:boolean):boolean;
182  2   1:d   1   procedure sc_init;
183  2   1:d   1
184  2   1:d   1
185  2   1:d   1
186  2   1:d   1         uses turtlegraphics, ibmspecial, screenops;
187  2   1:d   1
188  2   1:d   1         type
189  2   1:d   1             instance - array [1..3,1..3] of real;   { n[row,column]}
190  2   1:d   1
191  2   1:d   1         var
192  2   1:d   1             wxl,wxr,wyb,wyt:real;             {world coordinate limit  }
193  2   1:d   9             vxl,vxr,vyb,vyt:integer;          {viewport cordinate limit }
194  2   1:d  13             wvxx,wvyx,wvxy,wvyy,wvxa,wvya:real;    {viewing transformation}
195  2   1:d  25
196  2   1:d  25         procedure setwindow (x1,y1,x2, y2:real);
197  2   1:d   1         procedure setviewport(x1,y1,x2,y2:integer);
198  2   1:d   1         procedure worldtoscreen(x,y:real; var sx,sy:real);
199  2   1:d   1         procedure startat(x,y:real);
200  2   1:d   1         procedure lineto(x,y:real);
201  2   1:d   1         procedure apply(n:instance);
202  2   1:d   1         procedure translate(tx,ty:real; var Q:instance);
203  2   1:d   1         procedure scale(scalex,scaley:real; var Q:instance);
204  2   1:d   1         procedure clear_matrix(var M:instance);
205  2   1:d   1         procedure clearline(line:integer);
206  2   1:d   1         procedure draw_cursor(cursorx,cursory:real);
207  2   1:d   1         procedure erase_cursor(cursorx,cursory:real);
208  2   1:d   1         procedure drawbox;
209  2   1:d   1         procedure blank_area(x1,y1,x2,y2:integer);
210  2   1:d   1
211  2   1:d   1   {************************************************************************}
212  2   1:d   1
213  2   1:d   1     uses ibmspecial, turtlegraphics, screenops, graphics;
214  2   1:d   1
215  2   1:d   1     {ibmspecial, tutlegraphics, and screenops are standard IBM UCSD lib.  }
216  2   1:d   1     {graphics is a general graphic package written and added to the       }
217  2   1:d   1     {standard system library.}
218  2   1:d   1
219  2   1:d   1     const
220  2   1:d   1         max_line=8;             {max line for a component is 8      }
221  2   1:d   1         max_part=5;             {max no. of component in a character}
222  2   1:d   1         max_comp=18;            {max no. of component from file     }
223  2   1:d   1         max_char=20;            {max no. of character               }
224  2   1:d   1         comp_xs=0.2;  comp_ys=0.2;      {component scale factor     }
225  2   1:d   1         comp_xt=0.0;  comp_yt=70.0;     {component translate factor }
226  2   1:d   1         char_xs=0.4;  char_ys=0.4;      {character scale factor     }
227  2   1:d   1         char_xt=0.0;  char_yt=64.0;     {character translate factor }
228  2   1:d   1         leftmostx=0;  centerx=34;       {starting x position of mssg}
```

```
229  2   1:d   1
230  2   1:d   1     type
231  2   1:d   1        line_no=0..max_line;
232  2   1:d   1        part_no=0..max_part;
233  2   1:d   1
234  2   1:d   1        line= record
235  2   1:d   1            xypoint:array[1..4] of real;
236  2   1:d   1            end;
237  2   1:d   1
238  2   1:d   1        component - record
239  2   1:d   1            id:integer;
240  2   1:d   1            no_of_line:line_no;
241  2   1:d   1            lines:array[1..max_line] of line;
242  2   1:d   1            end;
243  2   1:d   1
244  2   1:d   1        transform= record       {transformation factor, sx,sy are scale     }
245  2   1:d   1                                {factors, tx,ty are translate factors.      }
246  2   1:d   1            sx,sy,tx,ty:real;
247  2   1:d   1            end;
248  2   1:d   1
249  2   1:d   1        character=record
250  2   1:d   1            id:integer;
251  2   1:d   1            no_of_part: part_no;
252  2   1:d   1            pointers: array[1..max_part] of 0..max_comp;  {ith record in comp}
253  2   1:d   1            info: array[1..max_part] of transform;
254  2   1:d   1            end;
255  2   1:d   1
256  2   1:d   1        comp_sort=record        {comp_sort is table_component plus pointers  }
257  2   1:d   1                                {to link to characters containing this comp. }
258  2   1:d   1            id:integer;
259  2   1:d   1            no_of_line:line_no;
260  2   1:d   1            lines:array[1..max_line] of line;
261  2   1:d   1            next_char:0..max_char;      {pointer to the next character.      }
262  2   1:d   1            next_part:0..max_part;      {pointer to the comp in the char.    }
263  2   1:d   1            end;
264  2   1:d   1
265  2   1:d   1        char_sort=record        {char_sort is table_character plus pointers  }
266  2   1:d   1                                {to link to characters sharing same component}
267  2   1:d   1            id:integer;
268  2   1:d   1            no_of_part:integer;
269  2   1:d   1            pointers: array[1..max_part] of 0..max_comp;
270  2   1:d   1            info: array[1..max_part] of transform;
271  2   1:d   1            next_char:array[1..max_part] of 0..max_char;
272  2   1:d   1            next_part:array[1..max_part] of 0..max_part;
273  2   1:d   1            end;
274  2   1:d   1
275  2   1:d   1     var
276  2   1:d   1        table_component:array[1..max_comp] of component;
277  2   1:d 1189        idno, nol:integer;          {id no, and no of line of component  }
278  2   1:d 1191
279  2   1:d 1191        Q:array[1..3,1..3] of real;     {Q is the cumulative transformation
280  2   1:d 1209                                            matrix for instance trans.  }
281  2   1:d 1209        T:array[1..3,1..3] of real;     {T is the total transformation matrix}
282  2   1:d 1227                                        {  for the character position       }
283  2   1:d 1227
284  2   1:d 1227        comp_file:file of component;
285  2   1:d 1593        char_file:file of character;
```

```
286  2   1:d 1940        s:string;                   {input character or component file  }
287  2   1:d 1981        c_file:string;              {output character or component file }
288  2   1:d 2022
289  2   1:d 2022        table_character:array[1..max_char] of character;
290  2   1:d 2962        ch_id_no:integer;           {current character no.              }
291  2   1:d 2963        ch_part_no:integer;         {current part no in character       }
292  2   1:d 2964
293  2   1:d 2964        picture:integer;            {=0 if there is no character xform  }
294  2   1:d 2965                                    {=1 if there is character xform     }
295  2   1:d 2965
296  2   1:d 2965        comp_list:array[1..max_comp] of comp_sort;
297  2   1:d 4189        char_list:array[1..max_char] of char_sort;
298  2   1:d 5329
299  2   1:d 5329        disp_flag:integer;          {=0 if component not yet display    }
300  2   1:d 5330                                    {=1 if component already display    }
301  2   1:d 5330  {*************************************************************************}
302  2   1:d 5330  {procedures designated as forward are those in the main section of the
303  2   1:d 5330   thesis program and being used in segments.                              }
304  2   1:d 5330
305  2   1:d 5330  procedure store_file( kind:string);
306  2   1:d    1      {depending on kind='component' or 'character', store info on a disk file }
307  2   1:d    1      forward;
308  2   1:d    1
309  2   1:d    1  procedure cursor_move(c:char; var  x,y:real);
310  2   1:d    1      {depending on the arrow key pressed on keyboard, the simulated cursor
311  2   1:d    1       moves according                                                    }
312  2   1:d    1      forward;
313  2   1:d    1
314  2   1:d    1  procedure read_file(kind:string);
315  2   1:d    1      {depending on kind='component' or 'character', read info from a disk file}
316  2   1:d    1      forward;
317  2   1:d    1
318  2   1:d    1  procedure show_a_component(var C:component);
319  2   1:d    1      {display a component on screen}
320  2   1:d    1      forward;
321  2   1:d    1
322  2   1:d    1  procedure disp_comp;
323  2   1:d    1      {display all components on screen}
324  2   1:d    1      forward;
325  2   1:d    1
326  2   1:d    1  procedure find_no( left_x:integer; var row,column,N:integer);
327  2   1:d    1      {calculate an index number N from row and colum specified by user   }
328  2   1:d    1      {left_x is the x position for message to display                     }
329  2   1:d    1      forward;
330  2   1:d    1
331  2   1:d    1  procedure pick_component(left_x:integer; var pick_no:integer);
332  2   1:d    1      {prompt user for row and column number for a selected component     }
333  2   1:d    1      {left_x is the x position for message to display                     }
334  2   1:d    1      forward;
335  2   1:d    1
336  2   1:d    1  procedure a_box (xscale,yscale,xtr,ytr:real);
337  2   1:d    1      {draws a box around a selected component or character               }
338  2   1:d    1      forward;
339  2   1:d    1
340  2   1:d    1  procedure erase_a_box (xscale,yscale,xtr,ytr:real);
341  2   1:d    1      {erase a box around a selected component or character               }
342  2   1:d    1      forward;
```

```
343  2   1:d   1
344  2   1:d   1  procedure draw_part(var M:integer; N:part_no);
345  2   1:d   1      {draw a component of a character, M is the character id no, N is the
346  2   1:d   1       component no                                                        }
347  2   1:d   1      forward;
348  2   1:d   1
349  2   1:d   1  procedure draw_character(var M:integer);
350  2   1:d   1      {draw a character, M indicates character id no                      }
351  2   1:d   1      forward;
352  2   1:d   1
353  2   1:d   1  procedure disp_character;
354  2   1:d   1      {display all characters in table_character                          }
355  2   1:d   1      forward;
356  2   1:d   1
357  2   1:d   1  segment procedure Create_Component;
358  2   1:d   1
359  2   1:d   1  {this is part 1 of a thesis --                                         }
360  2   1:d   1  {A METHOD OF STORAGE AND DISPLAY OF CHINESE CHARACTERS AS GRAPHIC SYMBOLS.  }
361  2   1:d   1  {Part 1 allows the user to create 'components' which are recognizable parts }
362  2   1:d   1  {of Chinese characters.                                                }
363  2   1:d   1  {                      Lo-yi Chung                                    }
364  2   1:d   1  {                      Sept. 10, 1985                                 }
365  2   1:d   1
366  7   1:d   1      var
367  7   1:d   1          startx,starty,endx,endy,x,y:real;
368  7   1:d  13
369  7   1:d  13
370  7   1:d  13  {*************************************************************************}
371  7   1:d  13  {the following procedures are unique to this segment only.              }
372  7   1:d  13
373  7   1:d  13  procedure erase;
374  7   1:d   1  {erase the most recently drawn line of a component while creating a component}
375  7   2:d   1      var
376  7   2:d   1          x,y:real;
377  7   2:0   0      begin
378  7   2:1   0          nol:=nol-1;
379  7   2:1   7          startat(table_component[idno].lines[nol].xypoint[1],
380  7   2:1  37                  table_component[idno].lines[nol].xypoint[2]);
381  7   2:1  69          pen_color(0);
382  7   2:1  72          pen_mode(4);
383  7   2:1  75          worldtoscreen(table_component[idno].lines[nol].xypoint[3],
384  7   2:1 105                        table_component[idno].lines[nol].xypoint[4],x,y);
385  7   2:1 139          moveto(x,y);
386  7   2:1 145          table_component[idno].no_of_line:=table_component[idno].no_of_line-1;
387  7   1:0   0      end;
388  7   1:0   0
389  7   1:d   1  procedure zero_component;
390  7   1:d   1  {initialize component record}
391  7   3:0   0  begin
392  7   3:1   0      idno:=1;
393  7   3:1   4      nol:=1;
394  7   3:1   8      table_component[idno].no_of_line:=nol;
395  7   1:0   0  end;
396  7   1:0   0
397  7   1:d   1  procedure save_point;
398  7   1:d   1  {save a line as two sets of x y value in the component record}
399  7   4:0   0  begin
```

```
400  7   4:1   0     with table_component[idno].lines[no1] do begin
401  7   4:3   23        xypoint[1]:=startx;
402  7   4:3   36        xypoint[2]:=starty;
403  7   4:3   49        xypoint[3]:=endx;
404  7   4:3   62        xypoint[4]:=endy;
405  7   4:2   75        end;
406  7   4:1   75    table_component[idno].no_of_line:=no1;
407  7   4:1   95    no1:=no1+1;
408  7   1:0    0  end;
409  7   1:0    0
410  7   1:d    1  procedure save_table;
411  7   1:d    1  {when one component is finished, advance id no., and set line no to 1}
412  7   5:0    0  begin
413  7   5:1    0     table_component[idno].id:=idno;
414  7   5:1   15     idno:=idno+1;
415  7   5:1   22     no1:=1;
416  7   1:0    0  end;
417  7   1:0    0
418  7   1:d    1  procedure drawline;
419  7   1:d    1  {let user draws lines in the square box, save as component when  finish,}
420  7   1:d    1  {or exit to finish the program}
421  7   1:d    1
422  7   6:d    1  var
423  7   6:d    1     x,y:real;
424  7   6:d    5     flag:boolean;       {flag=false when cursor is first drawn}
425  7   6:d    6                         {flag=true when cursor is not the 1st and needs to}
426  7   6:d    6                         {be erased}
427  7   6:d    6     j:integer;
428  7   6:d    7     c,c_cursor:char;       {user's response}
429  7   6:d    9
430  7   6:d    9     procedure menu_continue;
431  7   7:d    1        var
432  7   7:d    1           j:integer;  {local index}
433  7   7:d    2           answer:char;        {response of user}
434  7   7:d    3           boolean_c:boolean;  {for space_wait function}
435  7   7:d    4
436  7   7:0    0        begin
437  7   7:1    0           viewport(45,0,100,100);
438  7   7:1   13           fillscreen(0,0);            {blank menu area}
439  7   7:1   17           setviewport(0,5,60,60);
440  7   7:1   25           align_cursor(45,43);
441  7   7:1   33           writeln('1. create another component');
442  7   7:1   53           align_cursor(45,36);
443  7   7:1   61           writeln('2. see completed components');
444  7   7:1   81           align_cursor(45,28);
445  7   7:1   88           writeln('3. exit.');
446  7   7:1  108           sc_getc_ch(answer,['1','2','3']);
447  7   7:1  119           blank_area(45,0,100,100);
448  7   7:1  128           if answer ='1' then begin
449  7   7:3  133              if idno>18 then begin   {reach max. component number}
450  7   7:5  140                 align_cursor(45,34);
451  7   7:5  148                 writeln('no more room for');
452  7   7:5  168                 align_cursor(45,20);
453  7   7:5  175                 writeln('additional component');
454  7   7:5  195                 align_cursor(45,10);
455  7   7:5  202                 boolean_c:=space_wait(true);
456  7   7:4  207                 end
```

```
457  7   7:3  207                else begin          {accept another component}
458  7   7:5  209                    setviewport(0,5,49,49);
459  7   7:5  217                    fillscreen(0,0);
460  7   7:5  221                    setviewport(0,5,60,60);
461  7   7:5  229                    setwindow(0,0,60,60);
462  7   7:5  241                    drawbox;
463  7   7:5  243                    drawline;
464  7   7:4  245                    end;
465  7   7:2  245                end    {end of if answer='1'}
466  7   7:2  245
467  7   7:1  245              else if answer='2' then begin
468  7   7:4  252                  disp_comp;
469  7   7:4  254                  sc_clr_screen;
470  7   7:4  256                  clear_matrix(0);
471  7   7:4  261                  apply(0);
472  7   7:4  266                  setwindow(0,0,60,60);
473  7   7:4  278                  setviewport(0,5,60,60);
474  7   7:4  286                  drawbox;
475  7   7:4  288                  menu_continue;
476  7   7:3  290                  end          {end of else if answer='2'}
477  7   7:3  290
478  7   7:2  290              else c:='2';
479  7   6:0    0          end;         {end of procedure menu_continue}
480  7   6:0    0
481  7   6:d    1      procedure menu_line;
482  7   8:d    1          var
483  7   8:d    1              boolean_c:boolean; {local varialbe for use of space_wait function}
484  7   8:d    2              j:integer;         {local index}
485  7   8:d    3
486  7   8:0    0          begin
487  7   8:1    0              setviewport(45,0,100,100);
488  7   8:1    9              fillscreen(0,0);  {blank menu area}
489  7   8:1   13              setviewport(0,5,60,60);
490  7   8:1   21              align_cursor(45,43);
491  7   8:1   29              writeln('1. continue another line');
492  7   8:1   49              align_cursor(45,36);
493  7   8:1   57              writeln('2. component completed');
494  7   8:1   77              align_cursor(45,28);
495  7   8:1   84              writeln('3. erase last line');
496  7   8:1  104              align_cursor(45,20);
497  7   8:1  111              writeln('4. exit');
498  7   8:1  131              align_cursor(45,16);
499  7   8:1  138              sc_getc_ch(c,['1','2','3','4']);
500  7   8:1  151              setviewport(45,0,100,100);
501  7   8:1  160              fillscreen(0,0);
502  7   8:1  164              setviewport(0,5,60,60);
503  7   8:1  172              if c='1' then
504  7   8:2  178                  if nol>max_line then begin  {reach max number of line}
505  7   8:4  185                      align_cursor(45,16);
506  7   8:4  192                      writeln('can''t add any more lines!');
507  7   8:4  212                      align_cursor(45,12);
508  7   8:4  219                      boolean_c:=space_wait(true);
509  7   8:4  224                      menu_line;
510  7   8:3  226                      end
511  7   8:2  226                  else begin
512  7   8:4  228                      draw_line;
513  7   8:3  230                      end     {end of if nol>max}
```

```
514  7   8:3  230
515  7   8:1  230                else if c='2' then begin
516  7   8:4  238                        save_table;
517  7   8:4  240                        menu_continue;
518  7   8:3  242                        end
519  7   8:3  242
520  7   8:2  242                    else if c='3' then begin
521  7   8:5  250                        if nol>1 then erase;    {erase if there is at least}
522  7   8:5  259                        menu_line;              {one line}
523  7   8:4  261                        end;
524  7   6:0    0      end;        {end of procedure menu_line}
525  7   6:0    0
526  7   6:0    0  begin   {of drawline}
527  7   6:1    0      if c<>'2' then
528  7   6:2    7          begin
529  7   6:3    7              setviewport(45,0,100,100);
530  7   6:3   16              fillscreen(0,0);            {blank menu area}
531  7   6:3   20              setviewport(0,5,60,60);
532  7   6:3   28              align_cursor(45,43);
533  7   6:3   36              writeln('define a line by moving');
534  7   6:3   56              align_cursor(45,36);
535  7   6:3   64              writeln('the cursor to a starting');
536  7   6:3   84              align_cursor(45,28);
537  7   6:3   91              writeln('position with arrow keys,');
538  7   6:3  111              align_cursor(45,20);
539  7   6:3  118              writeln('and end with Z');
540  7   6:3  138
541  7   6:3  138              if nol=1 then begin {this is the first line}
542  7   6:5  144                  startx:=1;      {start the cursor at upper left corner}
543  7   6:5  150                  starty:=39;
544  7   6:4  157                  end
545  7   6:3  157                else begin
546  7   6:5  159                    startx:=endx;       {otherwise, start the cursor}
547  7   6:5  167                    starty:=endy;       {at the last position}
548  7   6:4  175                    end;
549  7   6:3  175            draw_cursor(startx,starty);
550  7   6:3  185            c_cursor:='X';
551  7   6:3  188            while ((c_cursor='Z')= false) do      {the signal of ending}
552  7   6:4  195                begin                             {is set <>'Z'}
553  7   6:5  195                    x:=startx;
554  7   6:5  201                    y:=starty;
555  7   6:5  207                    sc_getc_ch(c_cursor,['A','B','C','D','Z']);
556  7   6:5  218                    {'A' is the up key, 'B' is the down key,
557  7   6:5  218                     'C' is the right key, 'D' is the left key.}
558  7   6:5  218
559  7   6:5  218                    cursor_move(c_cursor,startx,starty);
560  7   6:5  227
561  7   6:5  227                    erase_cursor(x,y);
562  7   6:5  233                    draw_cursor(startx,starty);
563  7   6:5  243                    flag:=false; {set flag to false to indicate}
564  7   6:5  245                                 {a cursor is drawn}
565  7   6:4  245                end;
566  7   6:4  247
567  7   6:3  247            align_cursor(45,36);
568  7   6:3  255            write('the cursor to an ending');
569  7   6:3  268            endx:=startx; endy:=starty;
570  7   6:3  284            c_cursor:='X';
```

```
571  7   6:3  287          while ((c_cursor='Z')=false) do
572  7   6:4  294              begin
573  7   6:5  294                  x:=endx; y:=endy;
574  7   6:5  306                  sc_getc_ch(c_cursor,['A','B','C','D','Z']);
575  7   6:5  317                  cursor_move(c_cursor,endx,endy);
576  7   6:5  326                  if flag=true then erase_cursor(x,y);
577  7   6:5  336                  flag:=true;
578  7   6:5  338                  draw_cursor(endx,endy);
579  7   6:4  348              end;
580  7   6:4  350
581  7   6:3  350          if nol>1 then        {draw all previous line}
582  7   6:4  357              for j:=1 to nol-1 do
583  7   6:5  370                  with table_component[idno].lines[j] do begin
584  7   6:7  392                      startat(xypoint[1],xypoint[2]);
585  7   6:7  412                      lineto(xypoint[3],xypoint[4]);
586  7   6:6  432                      end;
587  7   6:6  437
588  7   6:3  437            startat (startx,starty);
589  7   6:3  447            lineto (endx,endy);
590  7   6:3  457            save_point;
591  7   6:3  459            menu_line;
592  7   6:2  461     end;       {end of if c<>2}
593  7   1:0    0     end;       {end of procedure drawline}
594  7   1:0    0
595  7   1:d    1 procedure menu_store;
596  7   1:d    1 {menu to see if components needs to be stored permanently on disk}
597  7   1:d    1
598  7   9:d    1     var
599  7   9:d    1         c:char;
600  7   9:0    0     begin
601  7   9:1    0         sc_clr_screen;
602  7   9:1    2         sc_goto_xy(5,17);
603  7   9:1    6         writeln('1. store new components in disk file, otherwise,');
604  7   9:1   26         sc_goto_xy(5,18);
605  7   9:1   30    writeln('   they will disappear once the machine is turned off');
606  7   9:1   50         sc_goto_xy(5,20);
607  7   9:1   54         writeln ('2. exit ');
608  7   9:1   74         sc_goto_xy(5,23);
609  7   9:1   78         sc_getc_ch(c,['1','2']);
610  7   9:1   89         if c='1' then store_file('component');
611  7   1:0    0     end;
612  7   1:0    0
613  7   1:d    1 procedure intro1;
614  7  10:d    1     var
615  7  10:d    1         boolean_c:boolean;                {dummy variable to use function}
616  7  10:0    0     begin
617  7  10:1    0         writeln;
618  7  10:1    7         writeln('    This thesis project creates and retrieves Chinese');
619  7  10:1   27         writeln('    characters using graphics techniques.');
620  7  10:1   47         writeln;
621  7  10:1   54         writeln('    A component is defined as a part of a Chinese');
622  7  10:1   74         writeln('    characters.  It contains up to 8 lines and is used');
623  7  10:1   94         writeln('    to compose Chinese characters.');
624  7  10:1  114         writeln;
625  7  10:1  121         writeln('    In this part of the program you may create your');
626  7  10:1  141         writeln('    own components (up to 18), or to redraw some of');
627  7  10:1  161         writeln('    the components already saved in a disk file');
```

```
628  7  10:1  181          sc_goto_xy(5,20);
629  7  10:1  185          boolean_c:=space_wait(true);
630  7   1:0    0          end;
631  7   1:0    0
632  7   1:d    1  procedure menu_1st;
633  7   1:d    1
634  7  11:d    1      var
635  7  11:d    1          c:char;         {user's response}
636  7  11:d    2
637  7  11:d    2      procedure delete_comp;
638  7  12:d    1          var
639  7  12:d    1              c:char;
640  7  12:d    2              i,row_no,column_no,delete_no:integer;
641  7  12:d    6              xs,ys,xt,yt:real;
642  7  12:d   14
643  7  12:0    0          begin
644  7  12:1    0              setviewport(0,0,105,55);
645  7  12:1    8              fillscreen(0,0);
646  7  12:1   12              setviewport(0,0,80,80);
647  7  12:1   20              find_no(leftmost_x,row_no,column_no,delete_no);
648  7  12:1   26                                                  {find which one to delete}
649  7  12:1   26              xs:=comp_xs;ys:=comp_ys;
650  7  12:1   38              xt:=comp_xt+12*(column_no-1);       {components are 12 unit apart}
651  7  12:1   49              yt:=comp_yt-(row_no-1)*10;          {10 units between lines      }
652  7  12:1   60
653  7  12:1   60              clear_matrix(0);
654  7  12:1   65              a_box(xs,ys,xt,yt); {draw a box around the one selected}
655  7  12:1   77              sc_goto_xy(0,19);
656  7  12:1   81              writeln('Is this the one you wish to delete?');
657  7  12:1  101              sc_goto_xy(1,20);
658  7  12:1  105              sc_getc_ch(c,['Y','N']);
659  7  12:1  116              if c='N' then begin
660  7  12:3  121                              erase_a_box(xs,ys,xt,yt);
661  7  12:3  133                              delete_comp;
662  7  12:2  135                              end
663  7  12:1  135                      else begin
664  7  12:3  137                      table_component[delete_no].id:=0;
665  7  12:3  148                      idno:=idno-1;
666  7  12:3  155                      for i:=delete_no to (idno-1) do
667  7  12:4  168                          table_component[i]:=table_component[i+1];
668  7  12:2  195                      end;
669  7  11:0    0          end;    {end of procedure delete_comp}
670  7  11:0    0
671  7  11:d    1      procedure menu_change;
672  7  13:d    1          var
673  7  13:d    1              c:char;
674  7  13:0    0          begin
675  7  13:1    0                   setviewport(0,0,105,55);
676  7  13:1    8                   fillscreen(0,0);
677  7  13:1   12                   setviewport(0,0,80,80);
678  7  13:1   20                   sc_goto_xy(5,19);
679  7  13:1   24                   writeln('1. add more component');
680  7  13:1   44                   sc_goto_xy(5,20);
681  7  13:1   48                   writeln('2. delete one component');
682  7  13:1   68                   sc_goto_xy(5,21);
683  7  13:1   72                   writeln('3. exit.');
684  7  13:1   92                   sc_goto_xy(5,23);
```

```
685  7  13:1   96                writeln('type your choice');
686  7  13:1  116                sc_getc_ch(c,['1','2','3']);
687  7  13:1  127                    if ((c='1') and (idno <  18)) then begin
688  7  13:3  140                                sc_clr_screen;
689  7  13:3  142                                nol:=1;
690  7  13:3  146                                clear_matrix(0);
691  7  13:3  151                                apply(0);
692  7  13:3  156                                setwindow(0,0,60,60);
693  7  13:3  168                                setviewport(0,5,60,60);
694  7  13:3  176                                drawbox;
695  7  13:3  178                                drawline;
696  7  13:3  180                                disp_comp;
697  7  13:3  182                                menu_change;
698  7  13:2  184                                end
699  7  13:1  184                           else if ((c='1') and (idno >= 18)) then begin
700  7  13:4  198                               sc_goto_xy(5,22);
701  7  13:4  202                               writeln('can not add more component');
702  7  13:4  222                               menu_change;
703  7  13:3  224                               end
704  7  13:2  224                           else if c='2' then begin
705  7  13:5  231                                        delete_comp;
706  7  13:5  233                                        disp_comp;
707  7  13:5  235                                        menu_change;
708  7  13:4  237                                        end;
709  7  13:4  237
710  7  11:0    0      end;        {end of menu_change}
711  7  11:0    0
712  7  11:0    0      begin       {of menu_1st}
713  7  11:1    0          sc_clr_screen;
714  7  11:1    2          sc_goto_xy(5,5);
715  7  11:1    6          writeln ('1. create all new components');
716  7  11:1   26          sc_goto_xy(5,7);
717  7  11:1   30          writeln ('2. review all existing components');
718  7  11:1   50          sc_goto_xy(5,9);
719  7  11:1   54          writeln ('3. exit');
720  7  11:1   74          sc_goto_xy(5,20);
721  7  11:1   78          writeln ('type your choice ');
722  7  11:1   98          sc_getc_ch(c,['1','2','3']);
723  7  11:1  109          if c='1' then begin
724  7  11:3  114              videomode(6);
725  7  11:3  120              setwindow(0,0,60,60);
726  7  11:3  132              setviewport(0,5,60,60);
727  7  11:3  140              drawbox;
728  7  11:3  142              zero_component;
729  7  11:3  144              s:='';       {set name to blank when creating new comp}
730  7  11:3  153              drawline;
731  7  11:3  155              menu_store;
732  7  11:3  157              menu_1st;
733  7  11:2  159              end
734  7  11:1  159            else if c='2' then begin
735  7  11:4  166                videomode(6);
736  7  11:4  172                idno:=1;
737  7  11:4  176                read_file('component');
738  7  11:4  183                disp_comp;
739  7  11:4  185                menu_change;
740  7  11:4  187                menu_store;
741  7  11:4  189                menu_1st;
```

```
742  7  11:3  191              end;
743  7   1:0    0     end;       {end of procedure manu_1st}
744  7   1:0    0
745  7   1:0    0  begin   {of segment procedure create_component}
746  7   1:1    0     videomode(2);
747  7   1:1    6     intro1;
748  7   1:1    8     menu_1st;
749  7   1:1   10     videomode(2);
750  7   1:0    0  end;
751  7   1:0    0  )       {coding of Part 1, 2 and 3 are in separated file     }
752  2   1:d    1  segment procedure Create_Character;
753  2   1:d    1
754  2   1:d    1  {This is part 2 of a thesis project -                                }
755  2   1:d    1  {A METHOD OF STORAGE AND DISPLAY OF CHINESE CHARACTERS AS GRAPHIC SYMBOLS.  }
756  2   1:d    1  {Part 2 demonstrates that a Chinese character can be created by transforming}
757  2   1:d    1  {and positioning several 'components'.  It also let the user to modify or   }
758  2   1:d    1  {delete existing character.                                           }
759  2   1:d    1  {                                                                     }
760  2   1:d    1  {                                                                     }
761  2   1:d    1  {                    Lo-yi Chung                                      }
762  2   1:d    1  {                    Sept. 10, 1985   rev. April 28, 1986            }
763  2   1:d    1
764  2   1:d    1
765  2   1:d    1  {*************************************************************************}
766 10   1:d    1     var
767 10   1:d    1
768 10   1:d    1         i,j:integer;
769 10   1:d    3         c,c_cursor:char;
770 10   1:d    5         cursor_key:boolean;
771 10   1:d    6         startx,starty,endx,endy,x,y: real;
772 10   1:d   18         change_char:boolean;    {true=a change to an existing character     }
773 10   1:d   19         comp_change:integer;    {the part no to be changed in an exiting ch }
774 10   1:d   20
775 10   1:d   20    procedure intro2;
776 10   2:d    1        var
777 10   2:d    1            boolean_c:boolean;           {a dummy variable to use function}
778 10   2:d    2
779 10   2:0    0        begin
780 10   2:1    0            writeln;
781 10   2:1    7            writeln('    This is part two of a thesis project which creates ');
782 10   2:1   27            writeln('    Chinese characters using graphics techniques.');
783 10   2:1   47            writeln;
784 10   2:1   54            writeln('    Part two shows how to compose a Chinese character from ');
785 10   2:1   74            writeln('    components already defined.');
786 10   2:1   94            sc_goto_xy(5,20);
787 10   2:1   98            boolean_c:=space_wait(true);
788 10   1:0    0            end;
789 10   1:0    0
790 10   1:d    1  procedure def_character;
791 10   1:d    1  {allow user to specify position of components in a character          }
792 10   3:d    1  var
793 10   3:d    1      x,y:real;
794 10   3:d    5      flag:boolean;       {flag=false when cursor is first drawn       }
795 10   3:d    6                          {flag=true when cursor is not the first one,  }
796 10   3:d    6                          {and needs to be erased.                      }
797 10   3:d    6      j:integer;
798 10   3:d    7
```

Missing Page

```
856  10   3:1  336          table_character[ch_id_no].id:=ch_id_no;
857  10   3:1  352          if change_char then     {just replace transformation info.character,  }
858  10   3:1  356                                 {do not replace ch_part_no                    }
859  10   3:2  356              with table_character[ch_id_no].info[comp_change] do
860  10   3:3  379                  begin
861  10   3:4  379                      sx:=(endx-startx)/40;
862  10   3:4  396                      sy:=(endy-starty)/40;
863  10   3:4  413                      tx:=startx;
864  10   3:4  421                      ty:=starty;
865  10   3:3  427                  end
866  10   3:1  427              else begin
867  10   3:1  429
868  10   3:3  429              table_character[ch_id_no].no_of_part:=ch_part_no;
869  10   3:3  450              with table_character[ch_id_no].info[ch_part_no] do
870  10   3:4  474              begin
871  10   3:5  474                  sx:=(endx-startx)/40;   {scale ratio of component in character}
872  10   3:5  491                  sy:=(endy-starty)/40;   {compared to 40x40 original comp.matrix}
873  10   3:5  508                  tx:=startx;
874  10   3:5  516                  ty:=starty;
875  10   3:4  522              end;    end;
876  10   1:0    0  end;
877  10   1:0    0
878  10   1:0    0
879  10   1:0    0
880  10   1:d    1  procedure change;
881  10   1:d    1  {allows modifications to already made character}
882  10   4:d    1      var i:integer;
883  10   4:d    2          c_set:SC_ChSet;      {SC_ChSet is set of ch def. in lib.}
884  10   4:d   18          answer:char;         {user's response}
885  10   4:d   19          j:integer;           {integer to pick up ith component  }
886  10   4:d   20          x_scale,y_scale, x_translate, y_translate:real;
887  10   4:d   28
888  10   4:d   28      procedure change_menu;
889  10   4:d    1
890  10   5:d    1              var answer:char;
891  10   5:d    2              x1,y1,x2,y2:integer;        {coordinates for viewport setting}
892  10   5:0    0          begin
893  10   5:1    0              blank_area(32,0,105,44);
894  10   5:1    9              clearline(16);
895  10   5:1   12              writeln('1. change to another component.');
896  10   5:1   32              clearline(17);
897  10   5:1   35              writeln('2. change the position.');
898  10   5:1   55              clearline(18);
899  10   5:1   58              writeln('3. no more changes.');
900  10   5:1   78              clearline(19);
901  10   5:1   81              sc_getc_ch(answer,['1','2','3']);
902  10   5:1   92
903  10   5:1   92                    {set viewport ready to blank the component area       }
904  10   5:1   92                    {in the character before new drawing                  }
905  10   5:1   92                    x1:=round(table_character[ch_id_no].info[comp_change].tx
906  10   5:1  116                          /aspect_ratio);
907  10   5:1  124                    y1:=round(table_character[ch_id_no].info[comp_change].ty);
908  10   5:1  149                    x2:=x1+round(table_character[ch_id_no].info[comp_change].
909  10   5:1  172                          sx*40/aspect_ratio);
910  10   5:1  187                    y2:=y1+round(
911  10   5:1  188                          table_character[ch_id_no].info[comp_change].sy*40);
912  10   5:1  220              if answer ='1' then begin
```

```
913  10   5:3  227                    blank_area (x1,y1,x2,y2);
914  10   5:3  233
915  10   5:3  233                    pick_component(centerx,j);
916  10   5:3  240                    table_character[ch_id_no].pointers[comp_change]:=j;
917  10   5:3  268
918  10   5:3  268                 {display the component again on row below, label it      }
919  10   5:3  268                 {as the order it was selected; for modification use.     }
920  10   5:3  268
921  10   5:3  268                 x_scale:=comp_xs;
922  10   5:3  275                 y_scale:=comp_ys;
923  10   5:3  282
924  10   5:3  282                 x_translate:=40 + 10*comp_change;     {show the component}
925  10   5:3  294                 y_translate:=50;                      {on separated line}
926  10   5:3  301                                                       {in a box}
927  10   5:3  301                 blank_area (round(x_translate/aspect_ratio),
928  10   5:3  311                             round(y_translate),
929  10   5:3  316                             round(((x_translate+x_scale*40)/aspect_ratio)),
930  10   5:3  335                             round(y_translate+y_scale*40));
931  10   5:3  351                             {blank selected components }
932  10   5:3  351
933  10   5:3  351                 scale(x_scale,y_scale,0);
934  10   5:3  364                 translate(x_translate,y_translate,0);  {with numerical label}
935  10   5:3  377                 apply(0);
936  10   5:3  382                 show_a_component(table_component[j]);
937  10   5:3  394                 drawbox;
938  10   5:3  396                 clear_matrix(0);
939  10   5:3  401                 apply(0);
940  10   5:3  406                    change_menu;
941  10   5:3  408                    draw_part(ch_id_no,comp_change);
942  10   5:3  418                      clear_matrix(0);
943  10   5:3  423                      apply(0);
944  10   5:2  428                    end
945  10   5:1  428              else if answer='2' then begin
946  10   5:4  435                       blank_area (x1,y1,x2,y2);
947  10   5:4  441                       change_char:=true;
948  10   5:4  445                       def_character;
949  10   5:4  447                       draw_part(ch_id_no,comp_change);
950  10   5:4  457                         clear_matrix(0);
951  10   5:4  462                         apply(0);
952  10   5:4  467                       change_menu;
953  10   5:3  469                       end;
954  10   5:3  469
955  10   4:0    0           end;       {end of change_menu}
956  10   4:0    0
957  10   4:0    0     begin      {begin of change}
958  10   4:1    0         blank_area(32,0,105,44);
959  10   4:1    9         clearline(12);
960  10   4:1   12         writeln('which of the above component you wish');
961  10   4:1   32         clearline(13);
962  10   4:1   35         writeln('to change?');
963  10   4:1   55
964  10   4:1   55         case table_character[ch_id_no].no_of_part of
965  10   4:1   70             1: c_set := ['1'];          {depending on how many component}
966  10   4:1   83             2: c_set := ['1','2'];      {there are, selection range is set}
967  10   4:1   96             3: c_set := ['1','2','3'];
968  10   4:1  109             4: c_set := ['1','2','3','4'];
969  10   4:1  122             5: c_set := ['1','2','3','4','5'];
```

```
 970  10    4:1  135            end;
 971  10    4:1  138
 972  10    4:1  138        sc_getc_ch(answer, c_set);
 973  10    4:1  145        case answer of
 974  10    4:1  149            '1':comp_change:=1;         {set current part no. in character}
 975  10    4:1  155            '2':comp_change:=2;
 976  10    4:1  161            '3':comp_change:=3;
 977  10    4:1  167            '4':comp_change:=4;
 978  10    4:1  173            '5':comp_change:=5;
 979  10    4:1  179            end;
 980  10    4:1  182
 981  10    4:1  182        change_menu;
 982  10    1:0    0    end;
 983  10    1:0    0
 984  10    1:0    0
 985  10    1:d    1  procedure more_component;
 986  10    1:d    1  {pick, scale, and draw up to 5 components for a character}
 987  10    1:d    1
 988  10    6:d    1  var  answer:char;       {user's response}
 989  10    6:d    2       j:integer;         {integer to pick up ith component in table_component}
 990  10    6:d    3       x_scale, y_scale, x_translate,y_translate:real;
 991  10    6:d   11       a,b,x,y:real;      {for calculation of position of label              }
 992  10    6:d   19
 993  10    6:d   19     procedure menu_component;
 994  10    7:0    0     begin
 995  10    7:1    0     blank_area(32,0,105,44);
 996  10    7:1    9     clearline(12);
 997  10    7:1   12     writeln('1. continue to pick another component.');
 998  10    7:1   32     clearline(14);
 999  10    7:1   35     writeln('2. this character is completed.');
1000  10    7:1   55     clearline(16);
1001  10    7:1   58     writeln('3. redefine one of the component.');
1002  10    7:1   78     clearline(18);
1003  10    7:1   81     writeln('4. exit.');        {exit=abort}
1004  10    7:1  101     clearline(20);
1005  10    7:1  104     sc_getc_ch(answer,['1','2','3','4']);
1006  10    7:1  117     if answer='1' then begin
1007  10    7:3  123                        if ch_part_no<max_part then
1008  10    7:4  130                            ch_part_no:=ch_part_no+1
1009  10    7:3  133                    else writeln('can''t add more component.');
1010  10    7:3  159                        more_component;
1011  10    7:2  161                        end
1012  10    7:1  161       else if answer='2' then begin
1013  10    7:4  172                             table_character[ch_id_no].id:=ch_id_no;
1014  10    7:4  188                             table_character[ch_id_no].no_of_part:=ch_part_no;
1015  10    7:4  209
1016  10    7:4  209                             if ch_id_no<max_char then begin
1017  10    7:6  216                               for j:=12 to 24 do clearline(j);
1018  10    7:6  240                               clearline(12);
1019  10    7:6  243                               writeln('do you want to do another one?');
1020  10    7:6  263                               clearline(13);
1021  10    7:6  266                               writeln('type y or n');
1022  10    7:6  286                               sc_getc_ch(answer,['Y','N']);
1023  10    7:6  299                               if answer = 'Y' then begin
1024  10    7:6  305                                   {clear previous drawing}
1025  10    7:8  305                                   blank_area (0,0,120,59);
1026  10    7:8  313                                   drawbox;
```

```
1027  10    7:8  315                                  ch_id_no:=ch_id_no+1;
1028  10    7:8  322                                  ch_part_no:=1;
1029  10    7:8  326                                  more_component;
1030  10    7:7  328                                  end;
1031  10    7:5  328                                end;
1032  10    7:3  328                               end
1033  10    7:2  328          else if answer='3' then begin
1034  10    7:5  336                                table_character[ch_id_no].id:=ch_id_no;
1035  10    7:5  352                                table_character[ch_id_no].no_of_part:=ch_part_no;
1036  10    7:5  373                                  change;
1037  10    7:5  375                                  menu_component;
1038  10    7:4  377                                  end;
1039  10    6:0    0      end;
1040  10    6:0    0
1041  10    6:0    0  begin   {begin of more_component}
1042  10    6:1    0          blank_area(32,0,105,44);
1043  10    6:1    9          pick_component(center_x,j);
1044  10    6:1   14          table_character[ch_id_no].pointers[ch_part_no]:=j;
1045  10    6:1   42
1046  10    6:1   42                     {display the component again on row below, label it       }
1047  10    6:1   42                     {as the order it was selected; for modification use.      }
1048  10    6:1   42
1049  10    6:1   42                     x_scale:=comp_xs;
1050  10    6:1   48                     y_scale:=comp_ys;
1051  10    6:1   53
1052  10    6:1   53                     x_translate:=40 + 10*ch_part_no;     {show the component}
1053  10    6:1   64                     y_translate:=50;                     {on separated line}
1054  10    6:1   69                                                          {in a box}
1055  10    6:1   69                     blank_area (round(x_translate/aspect_ratio),
1056  10    6:1   77                                 round(y_translate),
1057  10    6:1   80                                 round(((x_translate+x_scale*40)/aspect_ratio)),
1058  10    6:1   96                                 round(y_translate+y_scale*40));
1059  10    6:1  108                                 {blank selected components }
1060  10    6:1  108
1061  10    6:1  108                     scale(x_scale,y_scale,0);
1062  10    6:1  118                     translate(x_translate,y_translate,0);  {with numerical label}
1063  10    6:1  127                     apply(0);
1064  10    6:1  132                     show_a_component(table_component[j]);
1065  10    6:1  143                     drawbox;
1066  10    6:1  145                     clear_matrix(0);
1067  10    6:1  150                     apply(0);
1068  10    6:1  155                     a:=45 + 10*ch_part_no;
1069  10    6:1  167                     b:=47;       {a,b are xy position for numerical label}
1070  10    6:1  173                     worldtoscreen(a,b,x,y);
1071  10    6:1  185                     align_cursor(x,y);
1072  10    6:1  193                     write(ch_part_no);
1073  10    6:1  204
1074  10    6:1  204            def_character;
1075  10    6:1  206            draw_part(ch_id_no,ch_part_no);
1076  10    6:1  217            clear_matrix(0);
1077  10    6:1  222            apply(0);
1078  10    6:1  227          menu_component;
1079  10    6:1  229
1080  10    1:0    0  end;
1081  10    1:0    0
1082  10    1:d    1  procedure find_ch(left_x:integer; var row,column,N:integer);
1083  10    1:d    1  {calculate an index number N from row and column input}
```

```
1084  10    1:d    1  {left_x is where message starts from left of screen   }
1085  10    8:d    1  var
1086  10    8:d    1      i:integer;
1087  10    8:d    2      c:char;
1088  10    8:0    0  begin
1089  10    8:1    0      blank_area(left_x,0,100,39);
1090  10    8:1    8      sc_goto_xy(left_x,16);
1091  10    8:1   12      writeln('to specify the one you want');
1092  10    8:1   32      sc_goto_xy(left_x,17);
1093  10    8:1   36      writeln ('type row number from top of screen');
1094  10    8:1   56      sc_getc_ch(c,['1','2']);
1095  10    8:1   67      case c of
1096  10    8:1   70          '1':row:=1;
1097  10    8:1   75          '2':row:=2;
1098  10    8:1   80          end;
1099  10    8:1   83      blank_area(left_x,0,100,39);
1100  10    8:1   91      sc_goto_xy(left_x, 16);
1101  10    8:1   95      writeln('now type column number from');
1102  10    8:1  115      sc_goto_xy(left_x,17);
1103  10    8:1  119      writeln('left of the screen               ');
1104  10    8:1  139      sc_getc_ch(c,['1','2','3','4','5','6','7']);
1105  10    8:1  150      case c of
1106  10    8:1  153          '1':column:=1;
1107  10    8:1  158          '2':column:=2;
1108  10    8:1  163          '3':column:=3;
1109  10    8:1  168          '4':column:=4;
1110  10    8:1  173          '5':column:=5;
1111  10    8:1  178          '6':column:=6;
1112  10    8:1  183          '7':column:=7;
1113  10    8:1  188          end;
1114  10    8:1  191      N:=7*(row-1) + column;
1115  10    8:1  201      if N>ch_id_no then find_ch(left_x,row,column,N);
1116  10    8:1  215                                  {N can't not be > what is available}
1117  10    8:1  215                                  {on the screen                     }
1118  10    1:0    0  end;
1119  10    1:0    0
1120  10    1:d    1  procedure delete_char;
1121  10    1:d    1  {delete a character from existing table, by setting id_no=0}
1122  10    9:d    1  var c:char;     {user's response}
1123  10    9:d    2      i,row_no,column_no,delete_no:integer;
1124  10    9:d    6      xs,ys,xt,yt:real;          {xform factor}
1125  10    9:d   14
1126  10    9:0    0  begin
1127  10    9:1    0      blank_area(0,0,105,44);
1128  10    9:1    8      find_ch(leftmost_x,row_no,column_no,delete_no);
1129  10    9:1   14      xs:=char_xs; ys:=char_ys;
1130  10    9:1   26      xt:=char_xt+15*(column_no-1);
1131  10    9:1   37      yt:=char_yt-1-(row_no-1)*20;
1132  10    9:1   51      clear_matrix(Q);
1133  10    9:1   56      apply(Q);
1134  10    9:1   61      a_box(xs,ys,xt,yt);
1135  10    9:1   73      blank_area(0,0,104,44);
1136  10    9:1   81      sc_goto_xy(leftmost_x,16);
1137  10    9:1   85      writeln('Is this the one to delete?');
1138  10    9:1  105      sc_getc_ch(c,['Y','N']);
1139  10    9:1  116      if c='N' then begin
1140  10    9:3  121                    erase_a_box(xs,ys,xt,yt);
```

```
1141  10    9:3  133                  delete_char;
1142  10    9:2  135                  end
1143  10    9:1  135          else begin
1144  10    9:3  137              table_character[delete_no].id:=0;
1145  10    9:3  149              ch_id_no:=ch_id_no-1;
1146  10    9:3  156              for i:=delete_no to ch_id_no do
1147  10    9:4  168                     table_character[i]:=table_character[i+1];
1148  10    9:3  197              disp_character;
1149  10    9:2  199              end;
1150  10    1:0    0  end;
1151  10    1:0    0
1152  10    1:d    1  procedure mod_char;
1153  10    1:d    1  {modify a character by changing its component}
1154  10   10:d    1      var row_no,column_no,number:integer;
1155  10   10:d    4          xs,ys,xt,yt:real;              {xform factors}
1156  10   10:d   12          i:integer;
1157  10   10:d   13          a,b,x,y:real;                  {calculate position of numerical label}
1158  10   10:d   21
1159  10   10:0    0  begin
1160  10   10:1    0      blank_area(0,0,105,39);
1161  10   10:1    8      find_ch(leftmost_x,row_no,column_no,number);
1162  10   10:1   14      xs:=char_xs;  ys:=char_ys;
1163  10   10:1   25      xt:=char_xt+15*(column_no-1);
1164  10   10:1   36      yt:=char_yt-1-(row_no-1)*20;
1165  10   10:1   50      clear_matrix(Q); apply(Q);
1166  10   10:1   60      a_box(xs,ys,xt,yt);
1167  10   10:1   71      blank_area(0,0,105,39);
1168  10   10:1   79      sc_goto_xy(leftmostx,17);
1169  10   10:1   83      writeln('Is this the one you wish to modify?');
1170  10   10:1  103      sc_getc_ch(c,['Y','N']);
1171  10   10:1  116      if c='N' then begin
1172  10   10:3  122                    erase_a_box(xs,ys,xt,yt);
1173  10   10:3  133                    mod_char;
1174  10   10:2  135                    end
1175  10   10:1  135          else begin              {show each component on a separate line}
1176  10   10:1  138                                  {with numeric label, ready for change}
1177  10   10:3  138              disp_comp;
1178  10   10:3  140              drawbox;
1179  10   10:3  142              draw_character(number);
1180  10   10:3  145                  xs:=comp_xs;  ys:=comp_ys;
1181  10   10:3  156              for i:=1 to table_character[number].no_of_part do
1182  10   10:4  179              begin
1183  10   10:5  179                  xt:=40 + 10* i;  yt:=50;
1184  10   10:5  193                  blank_area (round(xt/aspect_ratio),     {blank area}
1185  10   10:5  201                              round(yt),
1186  10   10:5  204                              round(((xt+xs*40)/aspect_ratio)),
1187  10   10:5  220                              round(yt+ys*40));
1188  10   10:5  232                  clear_matrix(Q); apply(Q);
1189  10   10:5  242                  scale(xs,ys,Q);
1190  10   10:5  252                  translate(xt,yt,Q);
1191  10   10:5  261                  apply(Q);
1192  10   10:5  266        show_a_component(table_component[table_character[number].pointers[i]]);
1193  10   10:5  296                  drawbox;
1194  10   10:5  298                  clear_matrix(Q);
1195  10   10:5  303                  apply(Q);
1196  10   10:5  308                  a:=45+10*i;
1197  10   10:5  318                  b:=47;          {a,b are xy position for numerical label}
```

```
1198  10   10:5  324                worldtoscreen(a,b,x,y);
1199  10   10:5  336                align_cursor(x,y);
1200  10   10:5  344                write(i);
1201  10   10:4  353             end;
1202  10   10:3  360             ch_id_no:=number;
1203  10   10:3  364             change;
1204  10   10:2  366             end;
1205  10    1:0    0  end;
1206  10    1:0    0
1207  10    1:d    1  procedure option_menu;
1208  10   11:d    1     var answer:char;
1209  10   11:d    2         temp_ch_id:integer;     {hold ch_id_no to display all characters}
1210  10   11:d    3                                 {but allow ch_id_no to be used to modify}
1211  10   11:d    3                                 {components in a selected character.    }
1212  10   11:d    3
1213  10   11:0    0  begin
1214  10   11:1    0     blank_area (0,0,105,40);    {blank menu area}
1215  10   11:1    8     sc_goto_xy(1,17);
1216  10   11:1   12     writeln('You may choose one of the following:');
1217  10   11:1   32     writeln;
1218  10   11:1   39     writeln('1. add a character.');
1219  10   11:1   59     writeln('2. delete a character.');
1220  10   11:1   79     writeln('3. modify a character.');
1221  10   11:1   99     writeln('4. exit.');
1222  10   11:1  119     writeln;
1223  10   11:1  126     sc_getc_ch(answer,['1','2','3','4']);
1224  10   11:1  137     if answer ='1' then begin
1225  10   11:3  142                          sc_clr_screen;
1226  10   11:3  144                          videomode(6);
1227  10   11:3  150                          disp_comp;
1228  10   11:3  152                          drawbox;
1229  10   11:3  154                          clear_matrix(T);
1230  10   11:3  159                          if ch_id_no<max_char then begin
1231  10   11:5  166                            ch_id_no:=ch_id_no+1;
1232  10   11:5  173                            ch_part_no:=1;
1233  10   11:5  177                            change_char:=false;
1234  10   11:5  181                            more_component;
1235  10   11:5  183                            disp_character;
1236  10   11:4  185                            end
1237  10   11:3  185                            else writeln('can''t add more.');
1238  10   11:3  207                          option_menu;
1239  10   11:2  209                          end
1240  10   11:1  209        else if answer='2' then begin
1241  10   11:4  216                                delete_char;
1242  10   11:4  218                                option_menu;
1243  10   11:3  220                                end
1244  10   11:2  220            else if answer='3' then begin
1245  10   11:5  227                                    temp_ch_id:=ch_id_no;
1246  10   11:5  231                                    mod_char;
1247  10   11:5  233                                    ch_id_no:=temp_ch_id;
1248  10   11:5  237                                    disp_character;
1249  10   11:5  239                                    option_menu;
1250  10   11:4  241                                    end
1251  10   11:3  241                           else store_file('character');
1252  10    1:0    0  end;
1253  10    1:0    0
1254  10    1:d    1  procedure menu;
```

```
1255  10   12:d    1      var
1256  10   12:d    1          j:integer;      {local index}
1257  10   12:d    2          answer:char;    {user's response}
1258  10   12:d    3
1259  10   12:0    0      begin
1260  10   12:1    0          sc_clr_screen;
1261  10   12:1    2          writeln ('Please choose one of the following:');
1262  10   12:1   22          writeln;
1263  10   12:1   29          writeln;
1264  10   12:1   36          writeln('1. create a new file of character.');
1265  10   12:1   56          writeln;
1266  10   12:1   63          writeln('2. see an existing file of character.');
1267  10   12:1   83          writeln;
1268  10   12:1   90          writeln('3. exit.');
1269  10   12:1  110          writeln;
1270  10   12:1  117          sc_getc_ch(answer,['1','2','3']);
1271  10   12:1  128          if answer = '1' then begin
1272  10   12:3  135                                read_file('component');
1273  10   12:3  142                                videomode(6);
1274  10   12:3  148                                disp_comp;
1275  10   12:3  150                                drawbox;
1276  10   12:3  152                                s:='';  {not editing an exiting file}
1277  10   12:3  161                                clear_matrix(T);
1278  10   12:3  166                                ch_id_no:=1;    {initialize counter for table}
1279  10   12:3  170                                ch_part_no:=1;  {initialize no. of part}
1280  10   12:3  174                                for i:=1 to max_char do
1281  10   12:4  186                                    with table_character[i] do
1282  10   12:5  198                                    begin
1283  10   12:6  198                                        id:=0;
1284  10   12:6  201                                        change_char:=false;
1285  10   12:6  205                                        no_of_part:=0;
1286  10   12:5  213                                        end;
1287  10   12:3  221                                more_component;
1288  10   12:3  223                                if ch_id_no >0 then     {there is at least 1}
1289  10   12:4  230                                    begin
1290  10   12:5  230                                    disp_character;
1291  10   12:5  232                            sc_clr_screen;
1292  10   12:5  234                            sc_goto_xy(1,17);
1293  10   12:5  238                            writeln('Do you want to save them for later use?');
1294  10   12:5  258                                   writeln('type y or n.');
1295  10   12:5  278                                   sc_getc_ch(answer,['Y','N']);
1296  10   12:5  289                                   if answer='Y' then store_file('character');
1297  10   12:5  301                                   menu;
1298  10   12:4  303                                   end;
1299  10   12:2  303                         end
1300  10   12:1  303             else if answer='2' then begin
1301  10   12:4  310                                    read_file('component');
1302  10   12:4  317                                    read_file('character');
1303  10   12:4  324                                              {being edited}
1304  10   12:4  324                                    c_file:=s; {there is a character file b}
1305  10   12:4  333                                    disp_character;
1306  10   12:4  335                                    option_menu;
1307  10   12:4  337                                    menu;
1308  10   12:3  339                                  end;
1309  10    1:0    0      end;
1310  10    1:0    0
1311  10    1:0    0      begin       {of segment procedure create_character}
```

```
1312  10    1:1    0          videomode(2);
1313  10    1:1    6          intro2;
1314  10    1:1    8          menu;
1315  10    1:1   10          videomode(2);
1316  10    1:0    0      end;
1317  10    1:0    0  )        (because limited space in editor                    }
1318   2    1:d    1  segment procedure Sort_Character;
1319   2    1:d    1
1320   2    1:d    1  {this is part 3 of a thesis
1321   2    1:d    1   A METHOD OF STORAGE AND DISPLAY OF CHINESE CHARACTERS AS GRAPHIC SYMBOLS.
1322   2    1:d    1   Part 3 let the user pick a component, all Chinese characters containing
1323   2    1:d    1   the selected component will be displayed.
1324   2    1:d    1                                          Lo-yi Chung
1325   2    1:d    1                                          Sept. 10, 1985}
1326   2    1:d    1
1327   2    1:d    1  (*********************************************************************)
1328  11    1:d    1  const
1329  11    1:d    1      x1=0; y1=0; x2=105; y2=60;  {x and y are axis value for blank area   }
1330  11    1:d    1
1331  11    1:d    1
1332  11    1:d    1  procedure intro3;
1333  11    2:d    1        var
1334  11    2:d    1            i,j:integer;
1335  11    2:d    3            boolean_c:boolean;            {a dummy variable to use function}
1336  11    2:d    4
1337  11    2:0    0        begin
1338  11    2:1    0            writeln;
1339  11    2:1    7            writeln('    This is part three of a thesis project which creates ');
1340  11    2:1   27            writeln('    Chinese characters using graphics techniques.');
1341  11    2:1   47            writeln;
1342  11    2:1   54            writeln('    Part three shows you the sorting of Chinese characters');
1343  11    2:1   74            writeln('    by components in the character.') ;
1344  11    2:1   94
1345  11    2:1   94            {initialize character records}
1346  11    2:1   94            for i:=1 to max_char do
1347  11    2:2  103                with char_list[i] do begin
1348  11    2:4  114                    id:=0;
1349  11    2:4  117                    for j:=1 to max_part do begin
1350  11    2:6  126                        table_character[i].pointers[j]:=0;
1351  11    2:6  150                        pointers[j]:=0;
1352  11    2:6  165                        next_char[j]:=0;
1353  11    2:6  180                        next_part[j]:=0;
1354  11    2:5  195                        end;
1355  11    2:3  200                    end;
1356  11    2:1  205            sc_goto_xy(5,20);
1357  11    2:1  209            boolean_c:=space_wait(true);
1358  11    1:0    0            end;
1359  11    1:0    0
1360  11    1:0    0
1361  11    1:d    1  procedure put_comp;
1362  11    1:d    1  {set up components in comp_list after they are read into comp_table}
1363  11    1:d    1  {initialize pointer to next character to 0}
1364  11    3:d    1  var
1365  11    3:d    1      i,j:integer;
1366  11    3:d    3
1367  11    3:0    0  begin
1368  11    3:1    0      for i:=1 to max_comp do
```

```
1369  11    3:2    9      with comp_list[i] do begin
1370  11    3:4   20          id:=table_component[i].id;
1371  11    3:4   32          no_of_line:=table_component[i].no_of_line;
1372  11    3:4   49          for j:=1 to max_line do
1373  11    3:5   58              lines[j]:=table_component[i].lines[j];
1374  11    3:5   94
1375  11    3:4   94          next_char:=0;
1376  11    3:4  102          next_part:=0;
1377  11    3:3  110          end;
1378  11    1:0    0  end;
1379  11    1:0    0
1380  11    1:d    1  procedure put_char;
1381  11    1:d    1  {set up characters in char_list, after they are read into char_table,}
1382  11    1:d    1  {set up pointer to next character}
1383  11    4:d    1  var
1384  11    4:d    1      i,j:integer;
1385  11    4:d    3      N:integer;  {temp. storage for switching of link list pointer}
1386  11    4:d    4              boolean_c:boolean;
1387  11    4:d    5
1388  11    4:0    0      begin
1389  11    4:1    0          for i:=1 to ch_id_no do
1390  11    4:2   12              with char_list[i] do begin
1391  11    4:4   23              id:=table_character[i].id;
1392  11    4:4   36              no_of_part:=table_character[i].no_of_part;
1393  11    4:4   51              for j:=1 to no_of_part do begin
1394  11    4:6   62                  pointers[j]:=table_character[i].pointers[j];
1395  11    4:6   96                  info[j]:=table_character[i].info[j];
1396  11    4:6  128
1397  11    4:6  128                  if (pointers[j]>0) and (pointers[j]<=max_comp) then
1398  11    4:6  159                  {an valid comp. id}
1399  11    4:6  159
1400  11    4:7  159                      if comp_list[pointers[j]].next_char=0 then
1401  11    4:7  184                      {this is the 1st character for the component}
1402  11    4:8  184                        begin
1403  11    4:9  184                            comp_list[pointers[j]].next_char:=i;
1404  11    4:9  211                            comp_list[pointers[j]].next_part:=j;
1405  11    4:9  238                            next_char[j]:=0; {end of the list}
1406  11    4:9  253                            next_part[j]:=0;
1407  11    4:8  268                        end
1408  11    4:8  268
1409  11    4:7  268                        else if i<> comp_list[pointers[j]].next_char then
1410  11    4:7  296                                        {repetitive component in a char.}
1411  11    4:7  296                                        {only get to be pointed once}
1412  11    4:9  296                            begin       {there is some character the comp.}
1413  11    4:9  296                                        {points to,link to beginning of list}
1414  11    4:0  296                            next_char[j]:=comp_list[pointers[j]].next_char;
1415  11    4:0  332                            next_part[j]:=comp_list[pointers[j]].next_part;
1416  11    4:0  368                            comp_list[pointers[j]].next_char:=i;
1417  11    4:0  395                            comp_list[pointers[j]].next_part:=j;
1418  11    4:9  422                            end;
1419  11    4:5  422                  end;      {end of for j:=1 to max_part}
1420  11    4:3  428                  end;      {end of with char_list[i] do begin}
1421  11    1:0    0      end;
1422  11    1:0    0
1423  11    1:d    1  procedure sort;
1424  11    1:d    1  {prompt the user for a component, so that all characters containing that}
1425  11    1:d    1  {component can be shown on screen.  This procedure follows the pointer in}
```

```
1426  11    1:d    1  {the component record to pointers in character record until it - 0}
1427  11    1:d    1
1428  11    5:d    1  var
1429  11    5:d    1      answer:char;        {user's response}
1430  11    5:d    2      N,M,C_no,P_no:integer;          {the component selected}
1431  11    5:d    6      boolean_c:boolean;  {for use with space wait function}
1432  11    5:d    7      xs,ys,xt,yt:real;   {transformation factor}
1433  11    5:d   15      row,col:integer;
1434  11    5:d   17
1435  11    5:0    0  begin
1436  11    5:1    0      blank_area(x1,y1,x2,y2);
1437  11    5:1    8      pick_component(leftmost_x,N);
1438  11    5:1   12      if N<10 then begin
1439  11    5:3   17          row:=1; col:=N; end
1440  11    5:1   23          else begin
1441  11    5:3   25              row:=2; col:=N-9; end;
1442  11    5:3   33
1443  11    5:1   33      C_no:=comp_list[N].next_char;
1444  11    5:1   46      P_no:=comp_list[N].next_part;
1445  11    5:1   59          xs:=char_xs;
1446  11    5:1   65          ys:=char_ys;
1447  11    5:1   71          xt:=char_xt;
1448  11    5:1   77          yt:=char_yt-20;   {at 0.4 scale, lower y start after comp.}
1449  11    5:1   85      blank_area(0,0,105,59);     {blank menu area}
1450  11    5:1   93      if C_no= 0 then begin
1451  11    5:3   97                      blank_area(0, 0,105,59);
1452  11    5:3  105                      writeln('there is no character with this component');
1453  11    5:2  125                      end
1454  11    5:2  125
1455  11    5:1  125          else repeat
1456  11    5:3  128          clear_matrix(T);        {clear transformation for the character}
1457  11    5:3  133                                  {each time a character is done}
1458  11    5:3  133          apply(T);
1459  11    5:3  138          picture:=0;
1460  11    5:3  142               scale(xs,ys,T);
1461  11    5:3  153               translate(xt,yt,T);
1462  11    5:3  163               picture:=1;       {=1 indicate that there is character transf.}
1463  11    5:3  167                                 {matrix T and Q need to be concatenated.}
1464  11    5:3  167          draw_character(C_no);
1465  11    5:3  170                 xt:=xt+15;
1466  11    5:3  179               if xt>99 then begin
1467  11    5:5  188                   xt:=0;
1468  11    5:5  193                   yt:=yt-20;
1469  11    5:4  200                   end;
1470  11    5:3  200          picture:=0;
1471  11    5:3  204          sc_goto_xy(5,19);
1472  11    5:3  208          N:=C_no;
1473  11    5:3  210          M:=P_no;
1474  11    5:3  212          C_no:=char_list[N].next_char[P_no];
1475  11    5:3  233          P_no:=char_list[N].next_part[M];
1476  11    5:3  254
1477  11    5:2  254      until c_no=0;
1478  11    5:1  260          sc_goto_xy(5,20);
1479  11    5:1  264          boolean_c:=space_wait(true);
1480  11    1:0    0  end;
1481  11    1:0    0
1482  11    1:d    1  procedure sort_menu;
```

```
1483  11    1:d    1  {ask user to continue sort or exit'}
1484  11    6:d    1  var
1485  11    6:d    1      answer:char;        {user's response}
1486  11    6:d    2
1487  11    6:0    0  begin
1488  11    6:1    0      blank_area(x1,y1,x2,y2);
1489  11    6:1    8      sc_goto_xy(1,15);
1490  11    6:1   12      writeln;
1491  11    6:1   19      writeln('please choose one of the following');
1492  11    6:1   39      writeln;
1493  11    6:1   46      writeln('1. see characters by selected component.');
1494  11    6:1   66      writeln('2. exit.');
1495  11    6:1   86      sc_getc_ch(answer,['1','2']);
1496  11    6:1   97      if answer='1' then begin
1497  11    6:3  102          if disp_flag=0 then disp_component;
1498  11    6:3  110          disp_flag:=1;
1499  11    6:3  114          sort;
1500  11    6:3  116          sort_menu;
1501  11    6:2  118          end;
1502  11    1:0    0  end;
1503  11    1:0    0
1504  11    1:0    0
1505  11    1:0    0      begin       {of segment sort_character}
1506  11    1:1    0          videomode(2);
1507  11    1:1    6          intro3;
1508  11    1:1    8          read_file('component');
1509  11    1:1   15          read_file('character');
1510  11    1:1   22          put_comp;       {arrange pointers from component to}
1511  11    1:1   24          put_char;       {characters}
1512  11    1:1   26          videomode(6);
1513  11    1:1   32          disp_flag:=0;
1514  11    1:1   36          sort_menu;
1515  11    1:1   38          videomode(2);
1516  11    1:0    0      end;
1517  11    1:0    0  )
1518  11    1:0    0
1519  11    1:0    0  (*****************************************************************************)
1520  11    1:0    0
1521   2    1:d    1  procedure read_file;            {kind is 'component' or 'character' }
1522   2    4:d    1      var
1523   2    4:d    1          IO_error:integer;       {IO error code, =0 when no error}
1524   2    4:0    0      begin
1525   2    4:1    5          sc_clr_screen;
1526   2    4:1    7          sc_goto_xy(5,5);
1527   2    4:1   11          repeat
1528   2    4:1   11          (*$I-*)                 {deactivate the IO check abort}
1529   2    4:1   11
1530   2    4:2   11          writeln;
1531   2    4:2   16          writeln ('put your disk in the disk drive 1, and type name of the ');
1532   2    4:2   32          if kind='component' then write('COMPONENT') else write ('CHARACTER');
1533   2    4:2   65          writeln (' file, and press return key.');
1534   2    4:2   81          writeln;
1535   2    4:2   86          readln (s);
1536   2    4:2  101
1537   2    4:2  101          if kind='component' then
1538   2    4:3  110                  begin
1539   2    4:4  110                  idno:=1;
```

```
1540  2   4:4  114                reset (comp_file,s)
1541  2   4:3  126                end
1542  2   4:2  126            else begin
1543  2   4:4  128                ch_id_no:=1;
1544  2   4:4  132                reset(char_file,s);
1545  2   4:3  144                end;
1546  2   4:2  144            IO_error:=ioresult;
1547  2   4:2  148            if IO_error <> 0 then
1548  2   4:3  152               writeln ('unable to open file, try again.');
1549  2   4:1  168        until IO_error=0;
1550  2   4:0  174        (*$I+*);
1551  2   4:0  174
1552  2   4:1  174        if kind='component' then begin
1553  2   4:3  183          while eof(comp_file)= false do
1554  2   4:4  194              begin
1555  2   4:5  194                  table_component[idno]:=comp_file^;
1556  2   4:5  211                  get (comp_file);
1557  2   4:5  218                  idno:=idno+1;
1558  2   4:4  225              end;
1559  2   4:3  227          close (comp_file, lock);
1560  2   4:2  236          end
1561  2   4:1  236              else begin
1562  2   4:3  238                  while eof(char_file)   false do
1563  2   4:4  249                      begin
1564  2   4:5  249                          table_character[ch_id_no]:=char_file^;
1565  2   4:5  267                          get (char_file);
1566  2   4:5  274                          ch_id_no:=ch_id_no+1;
1567  2   4:4  281                          end;
1568  2   4:3  283                     close(char_file,lock);
1569  2   4:3  292                     ch_id_no:=ch_id_no-1;
1570  2   4:2  299                   end;
1571  2   1:0    0                   end;
1572  2   1:0    0
1573  2   1:d    1  procedure store_file;
1574  2   1:d    1  {depending of the parameter = 'component' or 'character', store components}
1575  2   1:d    1  {or character in a disk file}
1576  2   2:d    1  var
1577  2   2:d    1      i:integer;
1578  2   2:d    2      IO_error:integer;   {IO error code=0 when successful}
1579  2   2:d    3      boolean_c:boolean;
1580  2   2:d    4      answer:char;
1581  2   2:d    5      name:string;
1582  2   2:0    0  begin
1583  2   2:1    7      sc_clr_screen;
1584  2   2:1    9      videomode(2);
1585  2   2:1   15          repeat
1586  2   2:2   15              sc_goto_xy(0,5);
1587  2   2:2   19              if (s<>'') then begin
1588  2   2:4   30                  write('Your input file is '); writeln(s);writeln;
1589  2   2:3   69                  end;
1590  2   2:2   69              writeln('Put your disk in the disk drive and type file name');
1591  2   2:2   89              writeln('for output, and press return key');
1592  2   2:2  109              writeln;
1593  2   2:2  116              readln (c_file);
1594  2   2:2  135
1595  2   2:2  135            if c_file   s then       {save over the same file }
1596  2   2:2  147
```

```
1597  2    2:3  147                      if kind='character' then begin
1598  2    2:5  157                            reset(char_file,c_file);
1599  2    2:5  171                            close(char_file,purge);      {erase file first}
1600  2    2:5  180                            name  :=concat(c_file,'[4]');
1601  2    2:5  212                            rewrite(char_file,name  );
1602  2    2:5  224                            IO_error:=ioresult;
1603  2    2:4  228                       end
1604  2    2:3  228                       else begin
1605  2    2:5  230                            reset(comp_file,c_file);
1606  2    2:5  244                            close(comp_file,purge);
1607  2    2:5  253                            name:=concat(c_file,'[4]');
1608  2    2:5  285                            rewrite(comp_file,name);
1609  2    2:5  297                            IO_error:=ioresult;
1610  2    2:4  301                            end
1611  2    2:2  301         else begin      {a different name, meaning save over a new file     }
1612  2    2:4  303              name  :=concat(c_file,'[4]'); {allocate 4 blocks (512  byte x 4)
1613  2    2:4  335            (*$I-*)  {deactivate IO error abort}
1614  2    2:4  335              if kind='character' then rewrite (char_file,name  )
1615  2    2:4  355                   else rewrite (comp_file,name);
1616  2    2:4  367
1617  2    2:4  367              IO_error:=ioresult;
1618  2    2:4  371              sc_goto_xy(5,20);
1619  2    2:4  375              if IO_error <> 0 then
1620  2    2:5  379                writeln('unable to open file, try again.');
1621  2    2:3  395            end; {end of if c_file   s}
1622  2    2:1  395            until IO_error=0;
1623  2    2:1  401            (*$I+*)
1624  2    2:1  401
1625  2    2:1  401         if kind='character' then
1626  2    2:2  411         begin
1627  2    2:3  411         for i:=1 to (ch_id_no) do
1628  2    2:3  424
1629  2    2:4  424                if table_character[i].id<>0 then begin{id=0 means a delete item}
1630  2    2:6  438                   table_character[i].id:=i;
1631  2    2:6  450                   char_file^:=table_character[i];
1632  2    2:6  466                   put(char_file);
1633  2    2:5  473                   end;
1634  2    2:5  478
1635  2    2:3  478         close(char_file,lock);
1636  2    2:2  487         end
1637  2    2:1  487         else begin
1638  2    2:3  489              for i:=1 to (idno-1) do
1639  2    2:3  503
1640  2    2:4  503                    if table_component[i].id<>0 then begin
1641  2    2:6  516                        table_component[i].id:=i;
1642  2    2:6  527                        comp_file^:=table_component[i];
1643  2    2:6  542                        put(comp_file);
1644  2    2:5  549                        end;
1645  2    2:3  554                        close(comp_file,lock);
1646  2    2:2  563              end;
1647  2    1:0    0         end;
1648  2    1:0    0
1649  2    1:d    1  procedure cursor_move;
1650  2    1:d    1                 {simulate cursor movement on screen, directed by arrow keys}
1651  2    3:0    0                 begin
1652  2    3:0    0
1653  2    3:1    0                 if ((c='A') and (y<40.0))    {going left if not exceed box  }
```

```
1654  2   3:1  12                    then y:=y+1
1655  2   3:1  17                else if ((c='B') and (y>0))  {going right if not out of box }
1656  2   3:2  34                       then y:=y-1.0
1657  2   3:2  39                    else if ((c='C') and (x<40))  {go up if not out of box }
1658  2   3:3  58                           then x:=x+1.0
1659  2   3:3  63                       else if ((c='D') and (x>0)) {down if not out of box}
1660  2   3:4  81                              then x:=x-1.0;
1661  2   3:4  91
1662  2   1:0   0                end;
1663  2   1:d   1 procedure disp_character;
1664  2   1:d   1 {display all character in the table_character,except those id=0 which is dele.}
1665  2  13:d   1     var i,j,k:integer;
1666  2  13:d   4         xs,ys,xt,yt:real;       {transformation factor}
1667  2  13:d  12         boolean_c:boolean;
1668  2  13:0   0     begin
1669  2  13:1   0         videomode(6);
1670  2  13:1   6         sc_clr_screen;
1671  2  13:1   8         setwindow(0,0,80,80);
1672  2  13:1  20         setviewport(0,0,80,80);
1673  2  13:1  28         clear_matrix(T);
1674  2  13:1  33         apply(T);
1675  2  13:1  38         picture:=0;
1676  2  13:1  42         xs:=char_xs;    {display character at constant transformation}
1677  2  13:1  48         ys:=char_ys;
1678  2  13:1  53         xt:=char_xt;
1679  2  13:1  58         yt:=char_yt;
1680  2  13:1  63         for k:=1 to ch_id_no do
1681  2  13:2  75             if table_character[k].id<>0 then     {if =0, it is a deleted item}
1682  2  13:3  89             begin
1683  2  13:4  89               scale(xs,ys,T);
1684  2  13:4  99               translate(xt,yt,T);
1685  2  13:4 108               picture:=1;
1686  2  13:4 112               draw_character(k);
1687  2  13:4 115               clear_matrix(T);  {clear transformation for the character}
1688  2  13:4 120                                 {each time a character is done}
1689  2  13:4 120               xt:=xt+15;
1690  2  13:4 127               if xt>99 then begin
1691  2  13:6 135                   xt:=0;
1692  2  13:6 139                   yt:=yt-20;
1693  2  13:5 146                   end;
1694  2  13:3 146             end;
1695  2  13:1 151         clear_matrix(T);
1696  2  13:1 156         picture:=0;
1697  2  13:1 160         apply(T);
1698  2  13:1 165         sc_goto_xy(2,17);
1699  2  13:1 169         if ch_id_no >1 then begin
1700  2  13:3 176             write('there are ');write(ch_id_no);
1701  2  13:3 200             write(' characters in file ');
1702  2  13:2 213             end
1703  2  13:1 213           else begin
1704  2  13:3 215               write ('there is '); write(ch_id_no);
1705  2  13:3 239               write (' characters in file ');
1706  2  13:2 252               end;
1707  2  13:1 252         sc_goto_xy(5,19);
1708  2  13:1 256         boolean_c:=space_wait(true);
1709  2   1:0   0     end;
1710  2   1:0   0
```

```
1711  2   1:d   1  procedure show_a_component;
1712  2   1:d   1  (shows a component in table_component)
1713  2   1:d   1
1714  2   5:d   1        var j:integer;      {local index}
1715  2   5:0   0        begin
1716  2   5:1   0                    if C.id <>0 then begin
1717  2   5:3   5                        for j:=1 to C.no_of_line do
1718  2   5:4  15                            begin
1719  2   5:5  15                            startat(C.lines[j].xypoint[1],
1720  2   5:5  33                                  C.lines[j].xypoint[2]);
1721  2   5:5  53                            lineto (C.lines[j].xypoint[3],
1722  2   5:5  71                                  C.lines[j].xypoint[4]);
1723  2   5:4  91                            end;
1724  2   5:2  96                    end;          {end if id<>0, which is a deleted item}
1725  2   1:0   0        end;
1726  2   1:0   0
1727  2   1:d   1    procedure disp_comp;
1728  2   1:d   1        {display all components in the table_component on screen
1729  2   1:d   1         serve as review for what have in file }
1730  2   1:d   1
1731  2   6:d   1        var
1732  2   6:d   1            i,j: integer;
1733  2   6:d   3            boolean_c:boolean;
1734  2   6:d   4            x_scale,y_scale,x_translate,y_translate:real;
1735  2   6:d  12
1736  2   6:0   0        begin
1737  2   6:1   0            sc_clr_screen;
1738  2   6:1   2            setwindow(0,0,80,80);
1739  2   6:1  14            setviewport(0,0, 80, 80);
1740  2   6:1  22            clear_matrix(0);
1741  2   6:1  27            x_scale:=comp_xs;     {displays component at constant transformation}
1742  2   6:1  33            y_scale:=comp_ys;
1743  2   6:1  38            x_translate:=comp_xt;
1744  2   6:1  43            y_translate:=comp_yt;
1745  2   6:1  48            for i:=1 to (idno-1) do
1746  2   6:2  61                begin
1747  2   6:3  61                    scale(x_scale, y_scale,0);
1748  2   6:3  71                    translate(x_translate, y_translate,0);
1749  2   6:3  80                    apply(0);
1750  2   6:3  85
1751  2   6:3  85                    show_a_component(table_component[i]);
1752  2   6:3  96                    clear_matrix(0);
1753  2   6:3 101                    x_translate:=x_translate+12.0;
1754  2   6:3 109                    if x_translate>97 then begin
1755  2   6:5 117                        x_translate:=0;
1756  2   6:5 121                        y_translate:=y_translate-10;
1757  2   6:4 128                        end;
1758  2   6:2 128                `   end;  {end of for i:=1 to idno-1}
1759  2   6:2 133
1760  2   6:1 133        clear_matrix(0);
1761  2   6:1 138        apply(0);
1762  2   6:1 143        sc_goto_xy(2,17);
1763  2   6:1 147        i:=idno-1;
1764  2   6:1 152        if i>1 then begin
1765  2   6:3 157             write ('there are '); write (i); write (' components.       ');
1766  2   6:2 192             end
1767  2   6:1 192           else begin
```

```
1768  2   6:3  194            write('there is '); write (i); write(' component.       ');
1769  2   6:2  229            end;
1770  2   6:2  229
1771  2   6:1  229       sc_goto_xy (5,19);
1772  2   6:1  233       boolean_c:=space_wait(true);
1773  2   6:1  238       sc_goto_xy(2,17);
1774  2   6:1  242       writeln('                                                  ');
1775  2   6:1  262       sc_goto_xy(5,19);
1776  2   6:1  266       writeln('                                                  ');
1777  2   1:0    0       end;
1778  2   1:0    0
1779  2   1:0    0
1780  2   1:0    0
1781  2   1:0    0
1782  2   1:d    1   procedure introduction;
1783  2  14:d    1       var
1784  2  14:d    1           i,j:integer;
1785  2  14:d    3           boolean_c:boolean;           {a dummy variable to use function}
1786  2  14:d    4
1787  2  14:0    0       begin
1788  2  14:1    0           writeln;
1789  2  14:1    7           writeln('    This is a thesis project which creates ');
1790  2  14:1   27           writeln('    Chinese characters using graphics techniques.');
1791  2  14:1   47           writeln;
1792  2  14:1   54           writeln('    Part one let the user to create "part" or "component".');
1793  2  14:1   74           writeln;
1794  2  14:1   81           writeln('    Part two let the user to create Chinese characters');
1795  2  14:1  101           writeln('    using components generated in Part one.');
1796  2  14:1  121           writeln;
1797  2  14:1  128           writeln('    Part three shows you the sorting of Chinese characters');
1798  2  14:1  148           writeln('    by components in the character.') ;
1799  2  14:1  168
1800  2  14:1  168           sc_goto_xy(5,20);
1801  2  14:1  172           boolean_c:=space_wait(true);
1802  2   1:0    0           end;
1803  2   1:0    0
1804  2   1:0    0
1805  2   1:d    1     procedure a_box ;
1806  2   9:0    0         begin
1807  2   9:1    0         scale(xscale,yscale,0);
1808  2   9:1    9         translate(xtr,ytr,0);     {draw a box around the }
1809  2   9:1   18         apply(0);                 {selected component}
1810  2   9:1   23         drawbox;
1811  2   9:1   25         clear_matrix(0);
1812  2   9:1   30         apply(0);
1813  2   1:0    0         end;
1814  2   1:0    0
1815  2   1:d    1     procedure erase_a_box ;
1816  2  10:d    1         var x,y:real;
1817  2  10:0    0         begin
1818  2  10:1    0             scale(xscale,yscale,0);
1819  2  10:1   11             translate(xtr,ytr,0);
1820  2  10:1   20             apply(0);
1821  2  10:1   25             startat(0,0);
1822  2  10:1   31             pen_color(0);
1823  2  10:1   34             pen_mode(4);
1824  2  10:1   37             worldtoscreen(0,40,x,y);
```

```
1825  2  10:1  46            moveto(x,y);
1826  2  10:1  52            worldtoscreen(40,40,x,y);
1827  2  10:1  62            moveto(x,y);
1828  2  10:1  68            worldtoscreen(40,0,x,y);
1829  2  10:1  77            moveto(x,y);
1830  2  10:1  83            worldtoscreen(0,0,x,y);
1831  2  10:1  91            moveto(x,y);
1832  2  10:1  97            clear_matrix(Q);
1833  2  10:1 102            apply(Q);
1834  2   1:0   0            end;
1835  2   1:0   0
1836  2   1:d   1  procedure find_no;
1837  2   1:d   1  {calculate an index number N from row and column specified}
1838  2   7:d   1  var
1839  2   7:d   1        i:integer;
1840  2   7:d   2        c: char;        {user's response}
1841  2   7:d   3
1842  2   7:0   0  begin
1843  2   7:1   0        blank_area(left_x,0,100, 39);
1844  2   7:1   8        sc_goto_xy(left_x,16);
1845  2   7:1  12        writeln('to specify the one you want');
1846  2   7:1  32        sc_goto_xy(left_x,17);
1847  2   7:1  36        writeln('type row number from top of');
1848  2   7:1  56        sc_goto_xy(left_x,18);
1849  2   7:1  60        write('the screen.');
1850  2   7:1  73        sc_getc_ch(c,['1','2']);
1851  2   7:1  84        case c of
1852  2   7:1  87            '1': row:=1;
1853  2   7:1  92            '2': row:=2;
1854  2   7:1  97            end;
1855  2   7:1 100        blank_area(left_x,0,100, 39);
1856  2   7:1 108        sc_goto_xy(left_x,16);
1857  2   7:1 112        writeln('now type column number from ');
1858  2   7:1 132        sc_goto_xy(left_x,17);
1859  2   7:1 136        write('left of the scrren.');
1860  2   7:1 149        sc_getc_ch(c,['1','2','3','4','5','6','7','8','9']);
1861  2   7:1 160        case c of
1862  2   7:1 163            '1': column:=1;
1863  2   7:1 168            '2': column:=2;
1864  2   7:1 173            '3': column:=3;
1865  2   7:1 178            '4': column:=4;
1866  2   7:1 183            '5': column:=5;
1867  2   7:1 188            '6': column:=6;
1868  2   7:1 193            '7': column:=7;
1869  2   7:1 198            '8': column:=8;
1870  2   7:1 203            '9': column:=9;
1871  2   7:1 208            end;
1872  2   7:1 211        N:= 9*(row -1) + (column);
1873  2   7:1 221        if N>(idno-1) then find_no(left_x,row,column,N);
1874  2   7:1 236                                        {selected number can't be > }
1875  2   7:1 236                                        {what is available on screen }
1876  2   1:0   0  end;
1877  2   1:0   0
1878  2   1:d   1  procedure pick_component;
1879  2   1:d   1  {picks a component from table_component, and pass back the index}
1880  2   8:d   1      var
1881  2   8:d   1          i:integer;      {local index}
```

```
1882  2    8:d    2          a,b,x,y:real;        {to calculate position of numerical label}
1883  2    8:d   10                                 {for the picked component.}
1884  2    8:d   10          x_scale, y_scale:real;              {scale factors}
1885  2    8:d   14          x_translate, y_translate:real;      {translate factors}
1886  2    8:d   18          row_no, column_no:integer;       {position on screen}
1887  2    8:d   20          c:char;          {user response}
1888  2    8:d   21
1889  2    8:0    0      begin
1890  2    8:1    0          find_no(left_x,row_no,column_no,pick_no);
1891  2    8:1   10          clear_matrix(Q);
1892  2    8:1   15          x_scale:=comp_xs;
1893  2    8:1   21          y_scale:=comp_ys;
1894  2    8:1   27          x_translate:=((column_no)-1)*12.0;            {find lower left corner}
1895  2    8:1   38          y_translate:=comp_yt-(row_no -1)*10.0; {of the 1st character}
1896  2    8:1   53          a_box(x_scale,y_scale,x_translate,y_translate);
1897  2    8:1   67                                  {draw a box around the selected component}
1898  2    8:1   67          blank_area(left_x,0,100, 39);
1899  2    8:1   76          sc_goto_xy(left_x,16);
1900  2    8:1   81          writeln('Is it the component you want?');
1901  2    8:1  101          sc_goto_xy(left_x,17);
1902  2    8:1  106          write('type y or n');
1903  2    8:1  119          sc_getc_ch(c,['Y','N']);
1904  2    8:1  131          erase_a_box(x_scale,y_scale,x_translate,y_translate);
1905  2    8:1  145          if c='N' then
1906  2    8:2  151              pick_component(left_x,pick_no);
1907  2    8:2  157
1908  2    1:0    0       end;
1909  2    1:0    0
1910  2    1:0    0
1911  2    1:d    1  procedure draw_part;      {M is ch. id. no., N is part no.}
1912  2   11:d    1      var X:array[1..3,1..3] of real;     {X is a temporary holding matrix}
1913  2   11:d   19          x_scale,y_scale,x_translate,y_translate:real;
1914  2   11:d   27          i,j:integer;
1915  2   11:d   29
1916  2   11:d   29  {draw a component as a part of a character on screen}
1917  2   11:d   29
1918  2   11:0    0      begin
1919  2   11:1    0          x_scale:=table_character[M].info[N].sx;
1920  2   11:1   28          y_scale:=table_character[M].info[N].sy;
1921  2   11:1   56          x_translate:=table_character[M].info[N].tx;
1922  2   11:1   84          y_translate:=table_character[M].info[N].ty;
1923  2   11:1  110          scale(x_scale,y_scale,Q);
1924  2   11:1  121          translate(x_translate, y_translate,Q);
1925  2   11:1  132          if picture=1 then              {if there is char. xformation}
1926  2   11:2  140              begin
1927  2   11:3  140                clear_matrix(X);
1928  2   11:3  143                for i:=1 to 3 do          {concat Q - xform of parts and }
1929  2   11:3  157                                          {T   xform of character}
1930  2   11:4  157                   for j:=1 to 3 do
1931  2   11:5  171                   X[i,j]:=Q[i,1]*T[1,j]+Q[i,2]*T[2,j]+Q[i,3]*T[3,j];
1932  2   11:3  324               apply(X);
1933  2   11:2  327               end
1934  2   11:1  327                   else apply(Q);
1935  2   11:1  334          show_a_component(table_component[table_character[M].pointers[N]]);
1936  2   11:1  367
1937  2    1:0    0      end;
1938  2    1:0    0
```

```
1939  2   1:d   1  procedure draw_character;        {M indicate character id no}
1940  2   1:d   1  {draw the completed character composed of components}
1941  2   1:d   1
1942  2  12:d   1  var
1943  2  12:d   1     i:integer;
1944  2  12:d   2     j:integer;
1945  2  12:d   3
1946  2  12:0   0  begin
1947  2  12:1   0      for i:=1 to table_character[M].no_of_part  do
1948  2  12:2  20          begin
1949  2  12:3  20          clear_matrix(@);
1950  2  12:3  25          draw_part(M,i);
1951  2  12:2  32          end;
1952  2   1:0   0  end;
1953  2   1:0   0
1954  2   1:d   1  Procedure main_menu;
1955  2   1:d   1      {this menu allows user to branch to any one of the three parts}
1956  2  15:d   1  var
1957  2  15:d   1          answer:char;                    {user's response in main program     }
1958  2  15:d   2
1959  2  15:0   0  begin
1960  2  15:1   0      sc_clr_screen;
1961  2  15:1   2      writeln;
1962  2  15:1   9      writeln('  please choose one of the followings:');
1963  2  15:1  29      writeln;
1964  2  15:1  36      writeln;
1965  2  15:1  43      writeln('1. to create or review components.');
1966  2  15:1  63      writeln;
1967  2  15:1  70      writeln('2. to create or review characters.');
1968  2  15:1  90      writeln;
1969  2  15:1  97      writeln('3. to retrieve characters by components.');
1970  2  15:1 117      writeln;
1971  2  15:1 124      writeln('4. exit.');
1972  2  15:1 144      writeln;
1973  2  15:1 151      sc_getc_ch(answer,['1','2','3','4']);
1974  2  15:1 162      if answer='1' then begin
1975  2  15:3 167                         create_component;
1976  2  15:3 169                         main_menu;
1977  2  15:2 171                         end
1978  2  15:1 171             else if answer='2' then begin
1979  2  15:4 178                                     create_character;
1980  2  15:4 181                                     main_menu;
1981  2  15:3 183                                     end
1982  2  15:2 183                     else if answer='3' then begin
1983  2  15:5 190                                             sort_character;
1984  2  15:5 193                                             main_menu;
1985  2  15:4 195                                             end;
1986  2  15:4 195
1987  2   1:0   0  end;
1988  2   1:0   0
1989  2   1:0   0  (*****************************************************************************)
1990  2   1:0   0  begin   {of main program}
1991  2   1:1  22      videomode(2);
1992  2   1:1  28      introduction;
1993  2   1:1  30      main_menu;
1994  2   1:1  32      videomode(2);
1995  2    :0   0      end.
```

APPENDIX V

THE PROPOSAL

Rochester Institute of Technology
School of Computer Science and Technology

A proposal for
A Method of Storage and Display of
Chinese Characters as Graphic Symbols

by

Lo-yi Yang Chung

A thesis proposal, submitted to
The Faculty of the School of Computer Science and Technology,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Sciences.

Approved by: _____
Professor Guy Johnson

_____
Professor Evelyn Rozanski

_____
Professor Peter Anderson

June 8, 1984

Table of Contents

Introduction and Background

PROBLEM STATEMENTS

The majority of Chinese characters are picture-like symbols. Some of them were derived from the actual resemblance of nature scene or shape of objects; the others were abstract symbols of concepts, of similar pronunciation, and of composite of other symbols.

The lack of an alphabet-like structure in Chinese characters makes internal representation, input/output and storage out of the ordinary under the current computer environment.

This proposal treats the subject of storage and display of Chinese characters. Some basic components or forms, including "roots" are first defined. Chinese characters are created via the defined components and composite rules or methods. Computer graphics will be the vehicle for display, and a storage method will be discussed.

PREVIOUS WORK

In addition to the review of Computing Review Index, Quarterly Bibliography of Computers & Data Processing, Microcomputer Index, and Computer & Control Abstract in the Wallace Memorial Library, a computer search was requested. The computer search includes 65 entries.

The natioanal government of Taiwan, the Republic of China, has been working to achieve standardization in five areas: data interchange codes, input methods, character fonts, data communication protocol, and Chinese programming language. Chinese characters present several obstacles to automated information retrieval such as complex shapes, punctuation, homophones, pronunciation, and tone.

Most of the application systems require large storage and processing unit. Recently many Chinese systems apprear on microcomputer. A typical microcomputer system include a computer such as APPLE IIe with 64 K of memory, and a couple of circuit boards for storage of Chinese characters. One application system on APPLE IIe has 23,000 Chinese characters. There are about 40,000 Chinese characters for adequate publication and communication.

A Chinese Character Code for Infomation Intercharge (CCCII) was compiled and published in 1982. CCCII Vol. I consists of 4807 most frequently used Chinese characters. CCCII Vol. II is in two parts. The first part consists of a revision of Vol. I, with the addition of 16,197 less frequently used characters, the second part consists of 10,793 variant forms of Chinese characters.

Most of the storage of Chinese characters are in bit map format. A 16 x 16 matrix is thought to be low resolution, adequate

representation requires 24 x 24 or 40 x 40 matrix per character.

THEORETICAL AND CONCEPTUAL DEVELOPMENT

In the past 20 years, massive efforts were devoted to computerize Chinese language. At present, applications are visible in telephone directory look-up, electricity/water billing, library automation and income tax collection. However, because of the complexity of Chinese characters, major applications usually involve a main frame computer with enormous memory.

The hypothesis for this proposal is that it is possible to store Chinese characters as graphic symbols. And it may achieve saving in storage by storing similar parts (such as "roots") only once. The display mechanism involves traversing a data structrue so that parts of the character can be retrieved, scaled, transformed accordingly and then put together. The algorithm may be generalized in other graphic symbol processing.

GLOSSARY

basic component - a Chinese character or a root; it is an unique symbol, usually not easily generated with other symbols. The criteria used in this proposal is that a basic component should be no more than 6 lines, and it will probably take less time or storage to store and use it as a separate item than as a combination of the other basic components. The basic components are used to compose Chinese characters.

character - a Chinese character is a symbolic representation of Chinese language unit, usually conformed to a squarebox.

root - a part of a Chinese character. There are hundreds of roots, some of them are character themselves. Most of the roots are only one component of a character.

Chinese phonetic system - there are 21 consonants, 16 vowels and 5 tones. A Chinese character can be "spelled" by using:

- one vowel,
- two vowels,
- one consonant and one vowel,

or

- one consonant and two vowels.

One of the five tones is added to any of the above to further differentiate the sound of a character. There are many characters with the same pronunciation, and there are characters with more than one pronunciation.

Project Description

FUNCTIONAL SPECIFICATION

Functions Performed:
The proposed system will consist of three parts :

Part I - input and storage of basic components.

Part II - input and storage of characters using the basic components, including other display features, such as pronunciation, or English translation.

Part III - retrieval of characters by components, or browsing in some sequence.

function	Part I	Part II	Part III
graphic procedure: start at line to set window set viewport window to viewport aspect ratio transfer scale	x	x	x
display square box as boarder for a basic component or a character	x	x	x
draw lines segment	x	x	x
label square box (x,y) scale	x	x	x
save line segment as start (x,y) and end (x,y)	x		
write to file a component	x		
display components in some order	x	x	x
request for and lowerleft(x,y), upper right (x,y) for display in box		x	

draw component as requested, calculate transfer and scale factors	x		
save character as composite of basic components	x		
display character in sequence	x		x

Limitations and Restrictions:

In Part I and Part II the primary purpose is for someone familiar with the system to finish the front-end compilation. Character files will be created for use in Part III.

Part III is limited to retrieval only, will use the character files created in Part I and Part II. All process will be screen oriented, and interactive.

A pilot collection of about 30 basic components and 100 Chinese characters will be used to implement the system. Capability of the system to accept additional data will be demonstrated, however, maybe limited by the size of UCSD PASCAL system.

User Inputs:

The user of Part I and Part II of such a system needs to know the ideographic nature of Chinese characters and needs to know the purpose of Part I and Part II is to create the Chinese character data base. Specific skills required may include proficiency of Chinese language, knowledge of the defined basic components, the pre-defined precision (i.e. size of matrix for each character, 16 x 16, or 24 x 24, or 40 x 40).

Part III requires the user to identify components of a specific Chinese character. Based on the components, the system will display possible match of characters in the system.

User Outputs:

The display of Chinese characters will be on the monitor.

System Files:

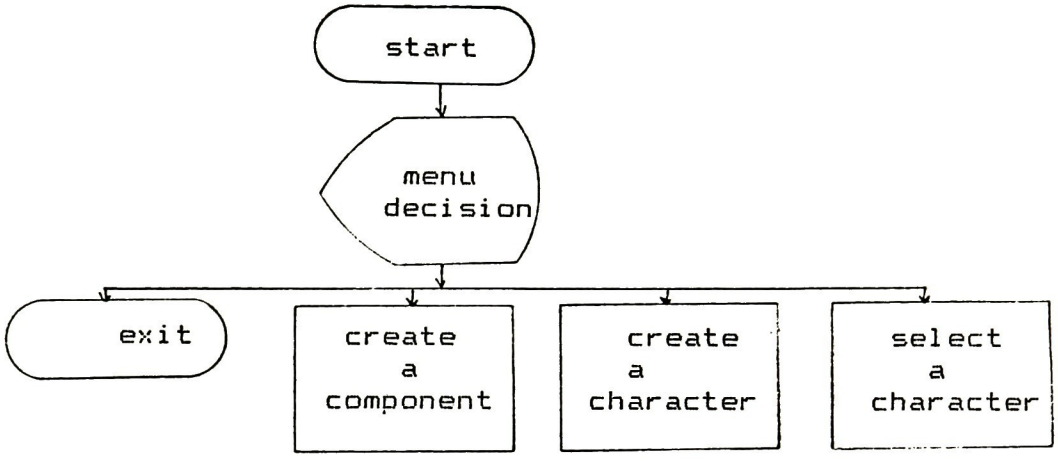
System files input, accessed and maintained by the system are:

1. the basic component file
2. the character specification file

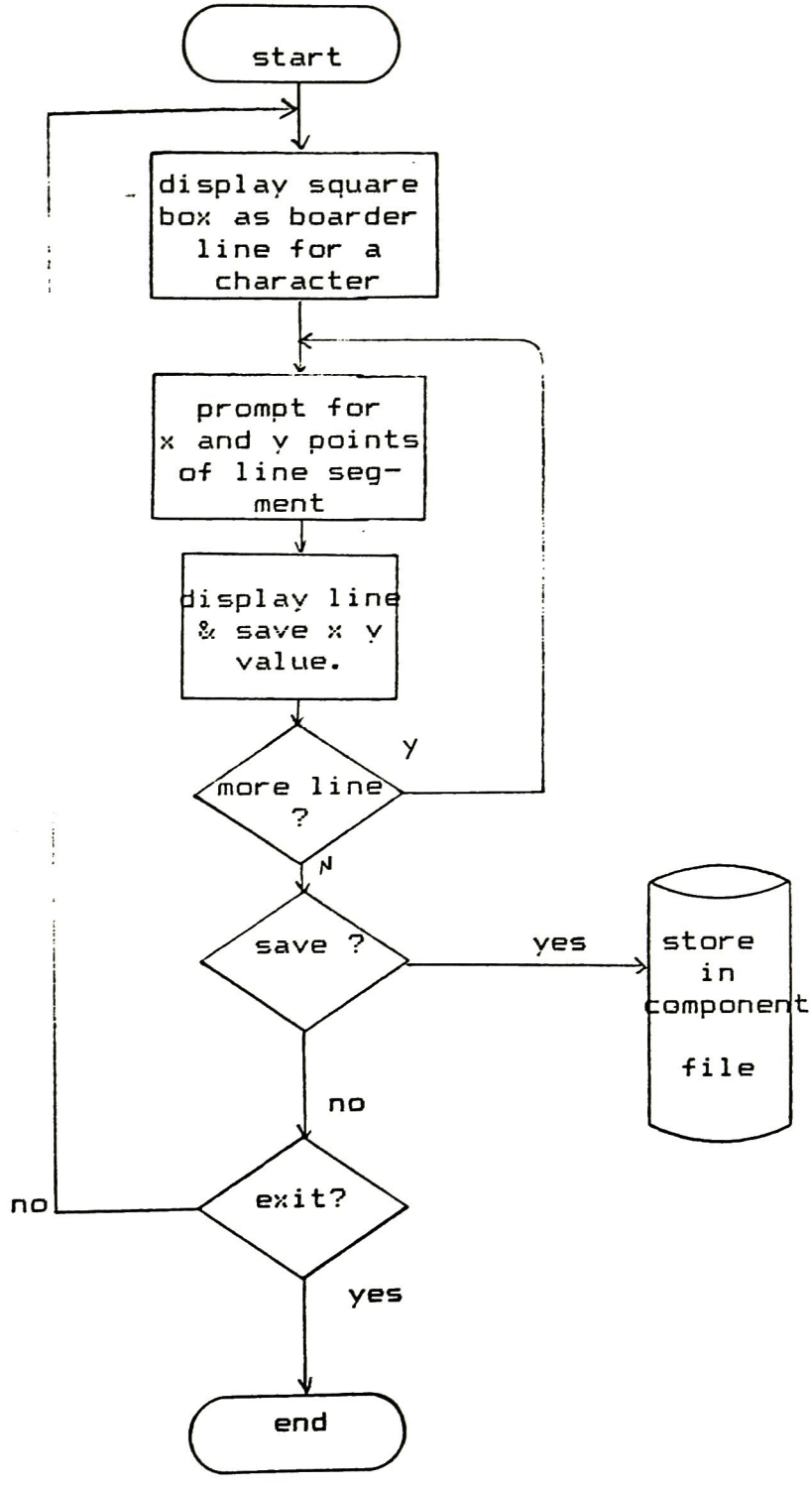
SYSTEM SPECIFICATION

System Organizational Chart

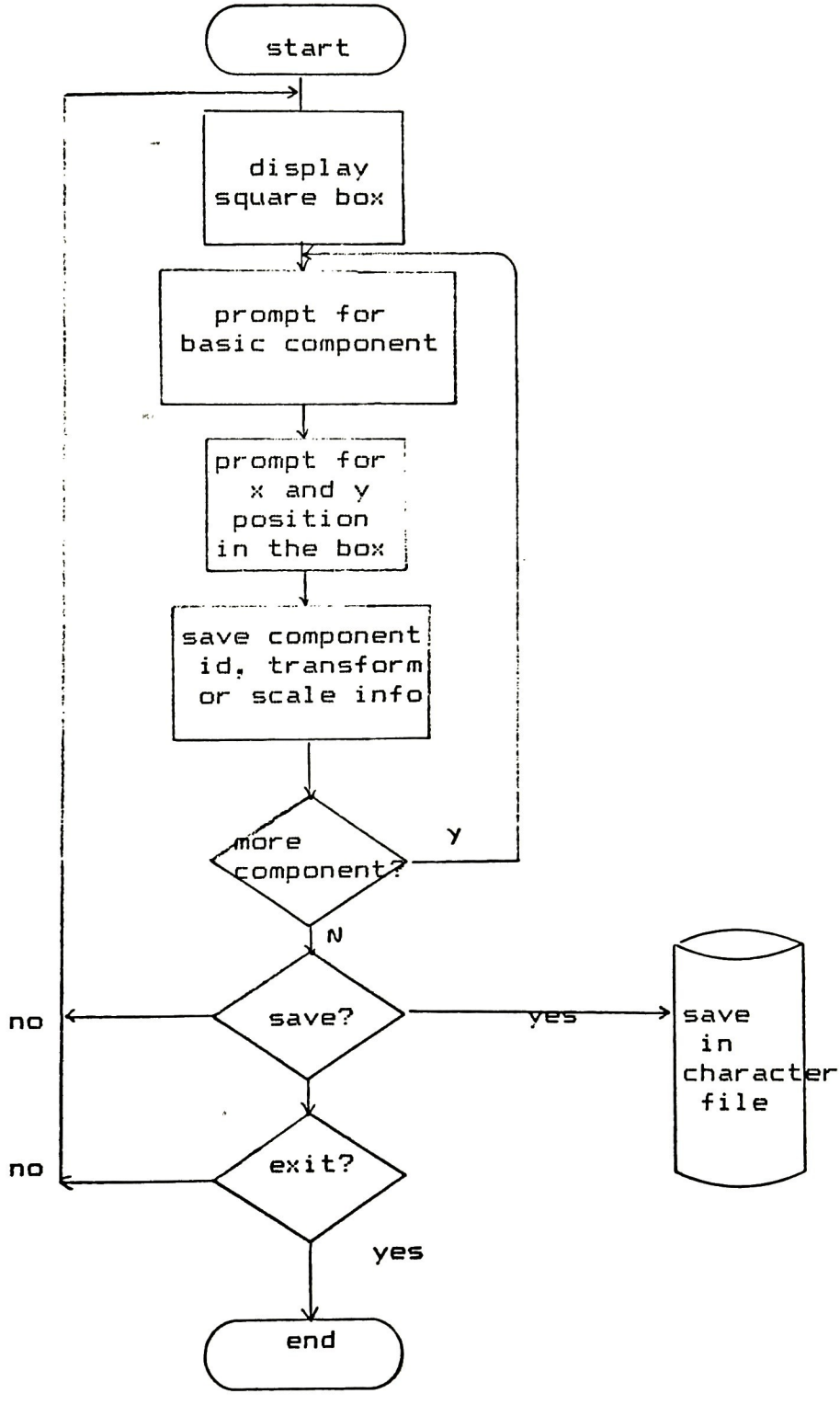
Overall flow chart



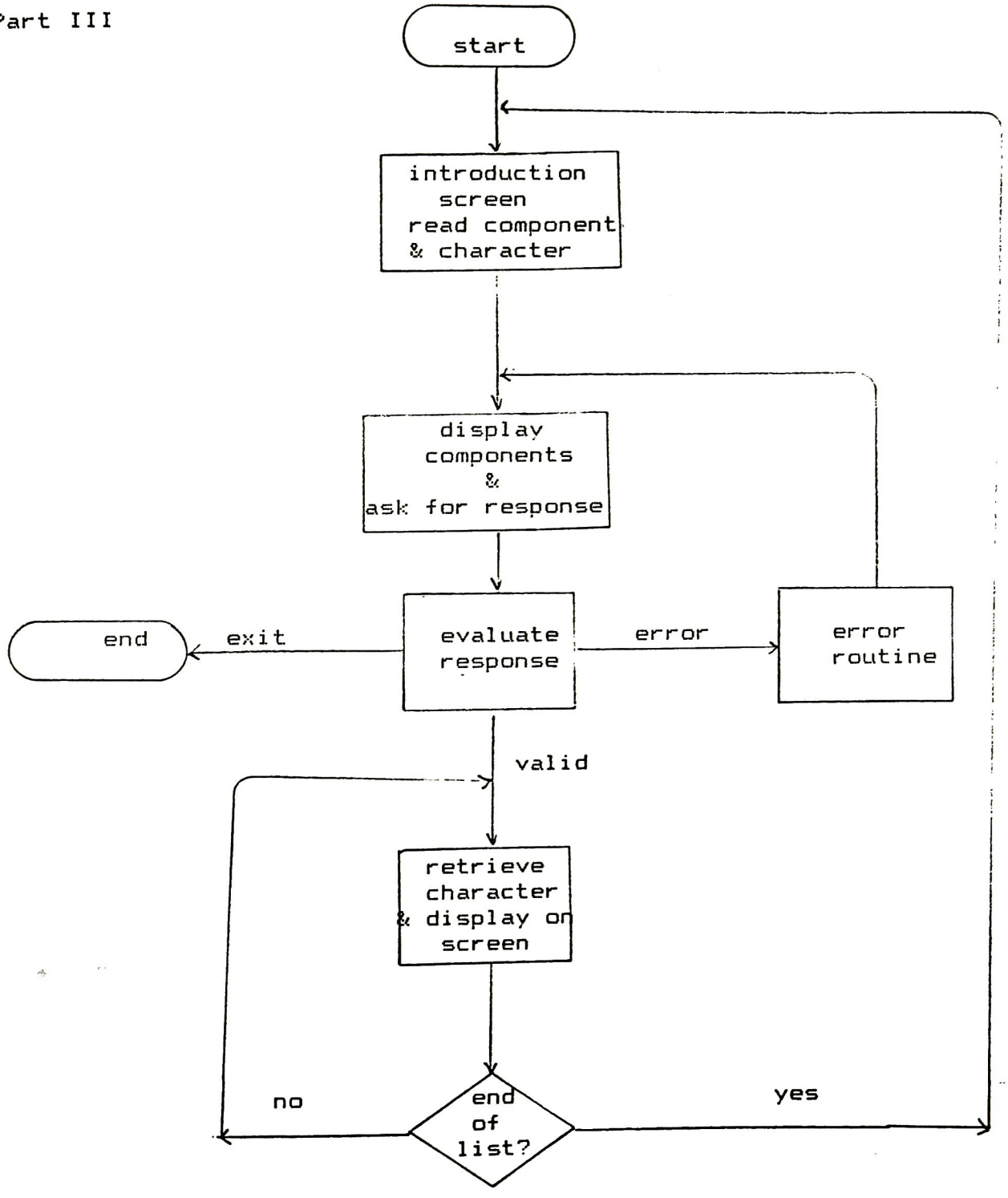
Part I


start
display square box as boarder line for a character
prompt for x and y points of line segment
display line & save x y value.
more line ?
y
N
save ?
yes
store in component file
no
exit?
no
yes
end

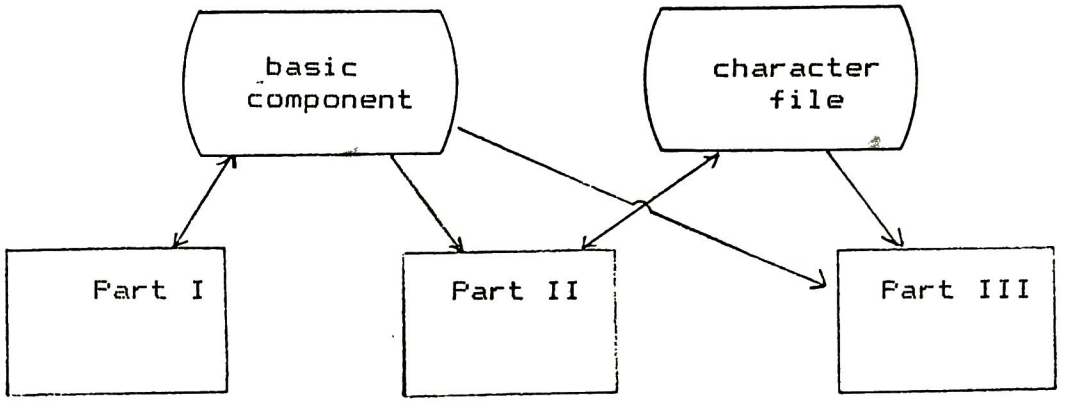
Part II


start
display square box
prompt for basic component
prompt for x and y position in the box
save component id, transform or scale info
more component?
Y
N
save?
yes
save in character file
no
exit?
no
yes
end

Part III


start
introduction screen read component & character
display components & ask for response
evaluate response
exit
end
error
error routine
valid
retrieve character & display on screen
end of list?
no
yes

System Data Flow Chart



Equipment Configuration
An IBM Personal Computer with 128k of memory and graphics capability will be used.

Implementation Tools
Because of its graphics capability, UCSD PASCAL will be used to implemented and demonstrate such a system on a microcomputer. Approximately 100 Chinese characters will be used to analyze and display for this study.

IMPLEMENTATION PLAN

Deliverable Items:

- proposal
- selection and listing of Chinese characters to be included
- diagram of data structure and file structure
- program listing
- display on monitor
- a final thesis

Milestone Identification

- proposal completion and approval
- display of all Chinese phonetic symbols
- completion of design of data structure and file structure
- display of Chinese characters
- a working system
- comparison of some other Chinese system on microcomputer
- completion of writing of thesis

Milestone Completion Criteria

- working
- documented

Milestone Schedule

activity	date
. proposal approved	June 8, 1984
. analysis of 100 Chinese characters	June 8, 1984
. design of data structure and file structure	June 10, 1984
. basic component file completed	June 30, 1984
. Chinese character file completed	July 15, 1984
. retrieval of character completed	July 30, 1984
. analysis of storage	November 1,1984
. comparison of other systems	September 1984 thru June 1985
. a final thesis	August 1985

Qualifications

Personal Background

I am a native Chinese, have completed my education through college in China. I received a Master degree in education in 1973 from Eastern Washington University.

My work experience included: serial librarian, planning analyst, and research associate in the areas of enrollment projection, retention and attrition. Experiences related to computing include need assessment of information system, time sharing on Sigma, PDP/UNIX and IBM/CMS, and applications on microcomputers.

Courses Taken

Statistics:
- Experiment Design I
- Experiment Design II
- Regression Analysis I
- Regression Analysis II

Programming:
- COBOL programming
- Programming Language Fortran
- Assembly Language programming
- Programming Language

Basic concept:
- Data Structure Analysis
- Foundation of Computing Theory
- Fundamentals of Computing
- Computer Architecture
- Software Architecture

Data Base:
- Data Base Concept
- Data Base System Implementation

Special interest courses:
- Discrete Simulation
- Information Storage and Retrieval
- Information System Design
- Microprocessors & Microcomputers
- Computer Graphics

Books and Articles Read

Program written

Previous Investigations and Projects

Bibliography

Hu, David Y.
Computer Processing of Chinese Library Materials in Taiwan. "Information Processing & Management" Vol. 19, no. 5, pp. 285-293, 1983.

Newman, William M. and Robert F. Sproull
"Principles of Interactive Computer Graphics" 2nd ed., New York, McGraw-Hill, c1979.

Smith, Peter
Minicomputer Vendors in Chinese Terminal Race, "Asian Computer Monthly" Oct 1983, p. 30.

Stroud, M.
Standardisation the Key to Chinese Language System. "Asian Computer Monthly" No. 74 Dec. 1983, p. 26,29.