โปรแกรมประมวลผลภาษาไทย/อังกฤษ

THAI/ENGLISH WORDPROCESSOR

โดย

นายศรัณย์ โพธิ์รุ่ง

นายสุวิชชา มุสิจรัล

นายอธิพร ลิ่มเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปีการศึกษา 2535

โครงงานโปรแกรมประมวลผลคำภาษาไทยอังกฤษ

โดย

นายศรัณย์ โพธิ์รุ่ง 321324

นายสุวิชชา มุสิจรัล 321399

นายอธิพร ลิ่มเจริญ 321412

อาจารย์ที่ปรึกษา

ดร.บุญธีร์ เครือตราชู

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมประมวลผลคำภาษาไทย/อังกฤษ

ผู้จัดทำ

1. นายศรัณย์ โพธิ์รุ่ง
2. นายสุวิชชา มุสิจรัล
3. นายอธิพร ลิ่มเจริญ

..... อาจารย์ที่ปรึกษา

(ดร. บุญธีร์ เครือตราชู)

โปรแกรมประมวลผลคำภาษาไทย/อังกฤษ

ศรัณย์ โพธิ์รุ่ง
สุวิชชา มุสิจรัล
อธิพร ลิ่มเจริญ

ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา
ปีการศึกษา 2535

บทคัดย่อ

โปรแกรมประมวลผลคำภาษาไทย/อังกฤษนี้ พัฒนาโดยใช้หลักการของพาราก์ราฟ คือข้อมูลแต่ละพารากราฟสามารถมีกั้นหน้ากั้นหลังเป็นของตัวเองได้ และมีความสามารถในพิมพ์ภาพสีออกทางเครื่องพิมพ์โดยสามารถอ่านภาพที่มีสีตั้งแต่ 16 สีถึง 256 สีแล้วทำการแปลงเป็นภาพ 16 gray scale แล้วส่งพิมพ์ไปยังเครื่องพิมพ์เป็นขาวดำอาศัยเทคนิคการทำ dithering เข้าช่วย มีความสามารถในการตัดคำกึ่งอัตโนมัติโดยอาศัยคุณสมบัติลักษณะของภาษาไทยในการหาขอบเขตและจุดการตัดคำ และสามารถพิมพ์ข้อความได้เหมือนกับโปรแกรมประมวลผลคำอื่นๆ

WORD PROCESSOR THAI/ENGLISH

Saran Porung
Suwitcha Musijaral
Atiporn Limcharoen

Dr. Boontee Kuretrachoo Advisor
1992

ABSTRACT

A Thai/English Word Processor developed and implemented based on the paragraph concept,each paragraph has self left-margin and right-margin. The program can print out picture or graphic, can read color picture 16 colors to 256 colors. The program will convert that picture into 16 gray scale picture and print it using dithering technics. The program has a thai syllable seperation. The analysis on thai syllable structures is carried out and consequently rules for syllable boundaries recongnition. and the program can print thai-english text same the other word processor.

คำนำ

โปรแกรมประมวลผลไทย-อังกฤษโดยทั่วไปไม่สามารถจะกำหนดกั้นหน้ากั้นหลังให้กับข้อมูลเพื่อแบ่งข้อมูลเป็นแต่ละพารากราฟได้ และยังไม่มีการพิมพ์รูปภาพสีออกทางเครื่องพิมพ์แบบขาวดำที่ดีพอ โครงงานโปรแกรมประมวลผลไทย-อังกฤษนี้จึงได้เกิดขึ้นเพื่อเป็นต้นแบบในการพัฒนาในขั้นตอนต่อไป

ขอบเขตของโครงงานนี้ได้ทำถึงการรับข้อมูลทางแป้นพิมพ์และพิมพ์ออกทางเครื่องพิมพ์ มีการตัดคำที่ถูกต้อง สามารถพิมพ์รูปภาพได้ และมีความสามารถพื้นฐานตามความสามารถของโปรแกรมประมวลผลคำเบื้องต้น

องค์ประกอบของปริญญานิพนธ์ฉบับนี้มีดังนี้

บทที่ 1 อธิบายโครงสร้างข้อมูลและฟังก์ชั่นการทำงานต่างๆที่เกี่ยวข้องกับข้อมูล ซึ่งเป็นพื้นฐานของโปรแกรมนี้

บทที่ 2 กล่าวถึงการนำข้อมูลในหน่วยความจำมาแสดงผลบนจอภาพโดยสัมพันธ์กันเพราะมิได้ใช้โครงสร้างข้อมูลแบบเป็นบรรทัดซึ่งจะแตกต่างกับโปรแกรมประมวลผลคำตัวอื่นๆ

บทที่ 3 กล่าวถึงรายละเอียดและลำดับของอักษรไทยตามมาตราฐานของ วทท.

บทที่ 4 การรับตัวอักษรทางแป้นพิมพ์ซึ่งจะต้องคอยตรวจสอบว่าใช่ปุ่มควบคุมหรือไม่ ตรวจสถานะปัจจุบันของแป้นพิมพ์ว่าเป็นภาษาไทยหรือภาษาอังกฤษ เพื่อจะจัดการกับข้อมูลให้ถูกต้องต่อไป

บทที่ 5 การพิมพ์ตัวอักษรภาษาไทย/อังกฤษ ออกทางเครื่องพิมพ์ บอกถึงการส่งรหัสควบคุมต่างๆเพื่อสั่งให้เครื่องพิมพ์ทำงานตามต้องการ

บทที่ 6 การตัดคำของอักษรไทยที่ใช้ในโปรแกรมนี้ อธิบายคุณสมบัติและลักษณะพิเศษของอักษรไทยแต่ละตัว และมีอัลกอริทึมอธิบายอยู่ด้วย

บทที่ 7 กล่าวถึงวิธีการแปลงรูปภาพสีเพื่อให้พิมพ์ออกทางเครื่องพมพ์ การศึกษาอัลกอริทึมต่างๆเพื่อให้ผลดีที่สุด

บทที่ 8 อธิบายอัลกอริทึมทั้งหมดของฟังก์ชั่นที่มีใช้ในโปรแกรมประมวลผลคำนี้

สารบัญ

บทที่ 1

DATA STRUCTURE

ลักษณะโครงสร้างข้อมูล

ลักษณะโครงสร้างข้อมูลที่ใช้เป็น double link list ซ้อนกันโดยชั้นแรกเป็น list ของแต่ละ paragraph โดยมี fields ต่าง ๆ ดังนี้

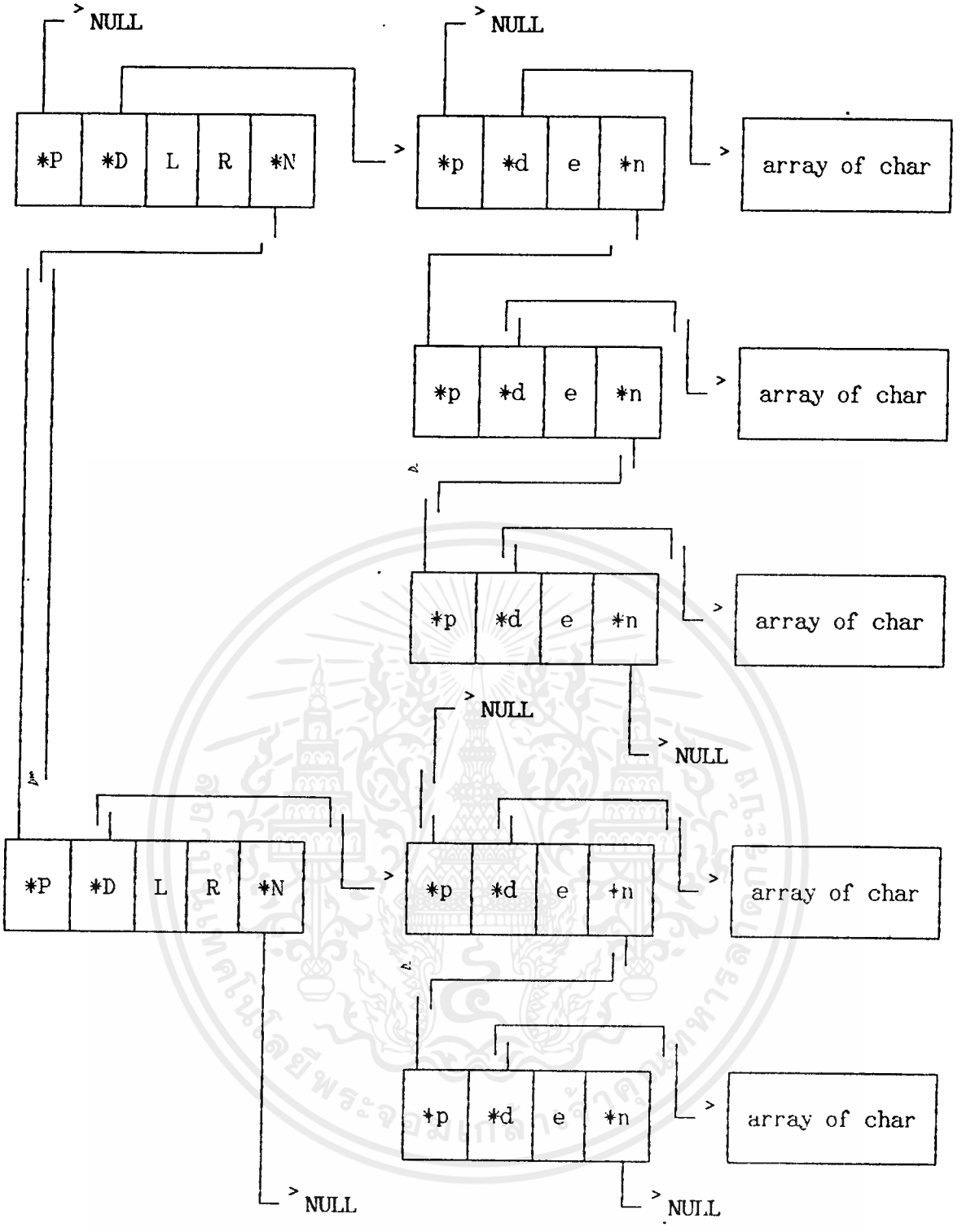
1. pointer ชี้ไปยัง paragraph ก่อนหน้านั้น
2. pointer ชี้ไปยัง list ของข้อมูลใน paragraph นั้น ๆ
3. ตำแหน่งกั้นหน้าซ้าย
4. ตำแหน่งกั้นหน้าขวา
5. pointer ชี้ไปยัง paragraph ถัดไป

ในชั้นที่สองที่เป็น double link list คือในส่วนของข้อมูลในแต่ละ paragraph จะเป็น list ของ array อีกชั้นหนึ่ง ทั้งนี้เนื่องจากการทำให้เป็น array ขนาดย่อย ๆ จะทำให้การเลื่อนข้อมูลใน array ทำได้เร็วขึ้น โดยยอมให้เกิด fragmentation ได้บ้าง ใน list ชั้นนี้จะประกอบด้วย fields ดังต่อไปนี้

1. pointer ชี้ไปยัง node ก่อนหน้านั้น
2. pointer ชี้ไปยัง array ของตัวอักษร ซึ่งมีขนาด 40 ตัวอักษร
3. ตำแหน่งของตัวอักษรตัวสุดท้ายใน array
4. pointer ชี้ไนยัง node ถัดไป

ข้อดีของโครงสร้างข้อมูลนี้ คือ ในแต่ละ paragraph เป็นอิสระต่อกัน การเปลี่ยนแปลงตำแหน่งกั้นหน้า และ กั้นหลังใน paragraph หนึ่งจะไม่ทำให้ paragraph อื่น ๆ เปลี่ยนแปลง ทำให้การจัดหน้ากระดาษ ทำได้ง่าย

ลักษณะของโครงสร้างข้อมูลแสดงไว้ดังรูปที่ 1



*P = pointer to previous paragraph
*D = pointer to datalist
*p = pointer to previous datalist
*d = pointer to charactor array
*N = pointer to next paragraph
L,R = (integer) Left/Right margin
*n = pointer to next datalist
e = (integer) position of '\x0'

รูปที่ 1 ลักษณะโครงสร้างข้อมูล

การแก้ไขข้อมูลใน data structure

การแก้ไขข้อมูลในหน่วยความจำแบ่งออกได้เป็น 3 กรณี คือ

1. แทรกตัวอักษร (Insert)
2. พิมพ์ทับ (Overwrite)
3. ลบตัวอักษร (Delete)

การแทรกตัวอักษร

แบ่งเป็นสองระดับคือ

1. การจัดการในระดับ ASCII Code เป็นการ insert ข้อมูลในระดับต่ำ จะไม่สนใจว่าตัวอักษรที่จะทำการ insert เป็นตัวอักษรอะไร จะทำการ insert โดย shiftข้อมูลที่มีอยู่จากตำแหน่ง cursor ไปทางขวา และ insert ข้อมูลตัวใหม่ลงที่ตำแหน่ง cursor เดิมและ update ตำแหน่ง cursor ฟังก์ชันที่ทำหน้าที่นี้คือฟังก์ชัน sub_insert แสดง flowchart ในรูปที่ 2

2. การจัดการในระดับที่จัดการกับตัวอักษรเท่านั้น โดยฟังก์ชันที่ทำงานนี้จะถือว่า input เป็นตัวอักษรและจะจัดการแทรก Control code ให้โดยอัตโนมัติถ้าตัวอักษรนั้นต้องการ Control code ฟังก์ชันที่ทำหน้าที่นี้คือฟังก์ชัน insert_char ทำงานดัง flowchart ในรูปที่ 3

การจัดการหน่วยความจำในการแทรกตัวอักษร : เนื่องจากในแต่ละNode มีข้อมูลเป็น String ซึ่งมีขนาดเท่ากับ MAXSTR การ Insert ตัว อักษรจึงทำได้ดังนี้

1. กรณีที่ String มีขนาดน้อยกว่า MAXSTR
   ให้เลื่อนข้อมูลตั้งแต่ตัวสุดท้ายของ String (รวม Null ด้วย) ถึงตำแหน่ง Cursor ไปทางขวา หนึ่งตำแหน่ง แล้วเติมข้อมูลที่ต้องการ Insert ที่ตำแหน่ง Cursor
2. กรณีที่ String มีขนาดเท่ากับ MAXSTR
   สามารถทำได้โดยเลื่อนข้อมูลทั้งหมดใน PARAGRAPH ไปทางขวา ซึ่งทำให้เสียเวลามาก จึงใช้วิธีสร้าง Node ใหม่ขึ้นมาแทน โดยจะมีการตรวจสอบและเลื่อนข้อมูลใน Node ต่อจาก Current Node ด้วย เพื่อป้องกันการสร้าง Node ซ้ำซ้อน ดังรูป

node 1	node 2	node 3	node 4	node 5
abcdefgh	ijklmno			
abcdXefg	h	ijklmno		
abcdXXef	g	h	ijklmno	
abcdXXXe	f	g	h	ijklmno

ปัญหาการสร้าง Node ซ้ำซ้อน

node 1	node 2	node 3
abcdefgh	ijklmno	
abcdXefg	hijklmno	
abcdXXef	g	hijklmno
abcdXXXe	fg	hijklmno
abcdXXXX	efg	hijklmno

แก้ไขโดยมีการตรวจสอบและเลื่อนข้อมูลใน Node ต่อไป

การพิมพ์ทับ

เป็นการลบตัวอักษรที่ตำแหน่ง Cursor แล้ว Insert ตัวอักษรใหม่ลงไปแทนโดยถ้าตำแหน่ง Cursor อยู่ที่ตัวอักษรท้ายสุดการ Overwrite จะเหมือนกับการ Insert ตามปกติ

เงื่อนไขในการพิมพ์ทับคือ

1. กรณีที่ตำแหน่ง Cursor ไม่ใช่ท้าย String (Null) สามารถเขียนข้อมูลลงที่ตำแหน่ง Cursor ได้เลยและทำการเลื่อนตำแหน่ง Cursor ไปทางขวาหนึ่งตำแหน่ง
2. กรณีที่ตำแหน่ง Cursor เป็นตำแหน่งท้าย String (Null) การเพิ่มเติมข้อมูลจะเป็นลักษณะเดียวกับ Insertmode เพราะ Null จะต้องอยู่ตำแหน่งสุดท้ายของ String เสมอ

การลบตัวอักษร

แบ่งเป็นสองระดับคือ

1. ลบในระดับ ASCII Code เป็นการลบ 1 charactor จากข้อมูลในโครงสร้างข้อมูลนี้ โดยทำการ shift ข้อมูลหลัง Cursor มาทางซ้ายให้แทนที่ข้อมูลที่ตำแหน่ง Cursor และทำการ

optimize ในกรณีที่ลบข้อมูลแล้ว Data node เกิดว่าง จะทำการลบ node นั้นออกไปเสีย ฟังก์ชันที่ทำงานนี้คือฟังก์ชัน sub_deletechar ทำงานดัง flowchart ในรูปที่ 4

2. ลบในระดับตัวอักษร จะจัดการลบตัวอักษรด้วยฟังก์ชัน sub_deletechar แล้วทำการตรวจสอบเกี่ยวกับ Control code และทำการลบ Control code ตามสมควร ฟังก์ชันนี้คือฟังก์ชัน delete_char มีการทำงานดัง flowchart ในรูปที่ 5

การเลื่อนตำแหน่ง Cursor ใน memory

การเลื่อนตำแหน่ง Cursor จะเกิดขึ้นเมื่อผู้ใช้กดปุ่มลูกศรหรือต้องมีการเลื่อนตำแหน่ง Cursor ในระหว่างการแก้ไขข้อมูล เช่นเดียวกับการ insert และ delete การเลื่อน Cursor จำเป็นต้องแบ่งออกเป็นหลายระดับด้วย ทั้งนี้เนื่องจากระบบ multifont ที่มี Control code เป็นตัวอักษรพิเศษ

1. การเลื่อน Cursor ไปยังตัวอักษรก่อนหน้า Cursor แบ่งออกเป็น 3 ฟังก์ชันคือ

1.1 previous มองเห็นเฉพาะตัวอักษรเมื่อเลื่อน Cursor จะเลื่อนข้าม Control code ไปและทำการ update ตัวแปรในการแสดงผลด้วย

1.2 sub1_prev มองเห็นเฉพาะตัวอักษรเช่นเดียวกับ previous แต่เมื่อเลื่อนตำแหน่งไปแล้วจะไม่ update ตัวแปรในการแสดงผล

1.3 sub2_prev มองเป็นรหัส ASCII การทำงานแต่ละครั้งจะเลื่อนไป 1 ตำแหน่งตัวอักษร ถ้าตำแหน่ง Cursor เดิมเป็นต้น data node หรือ paragraph ฟังก์ชันนี้จะเลื่อนตำแหน่ง Current node และ Current paragraph ด้วย ฟังก์ชันในข้อ 1.1 และ 1.2 เรียกใช้ฟังก์ชันนี้จึงทำให้เกิดการเปลี่ยน node และ paragraph อัตโนมัติด้วย

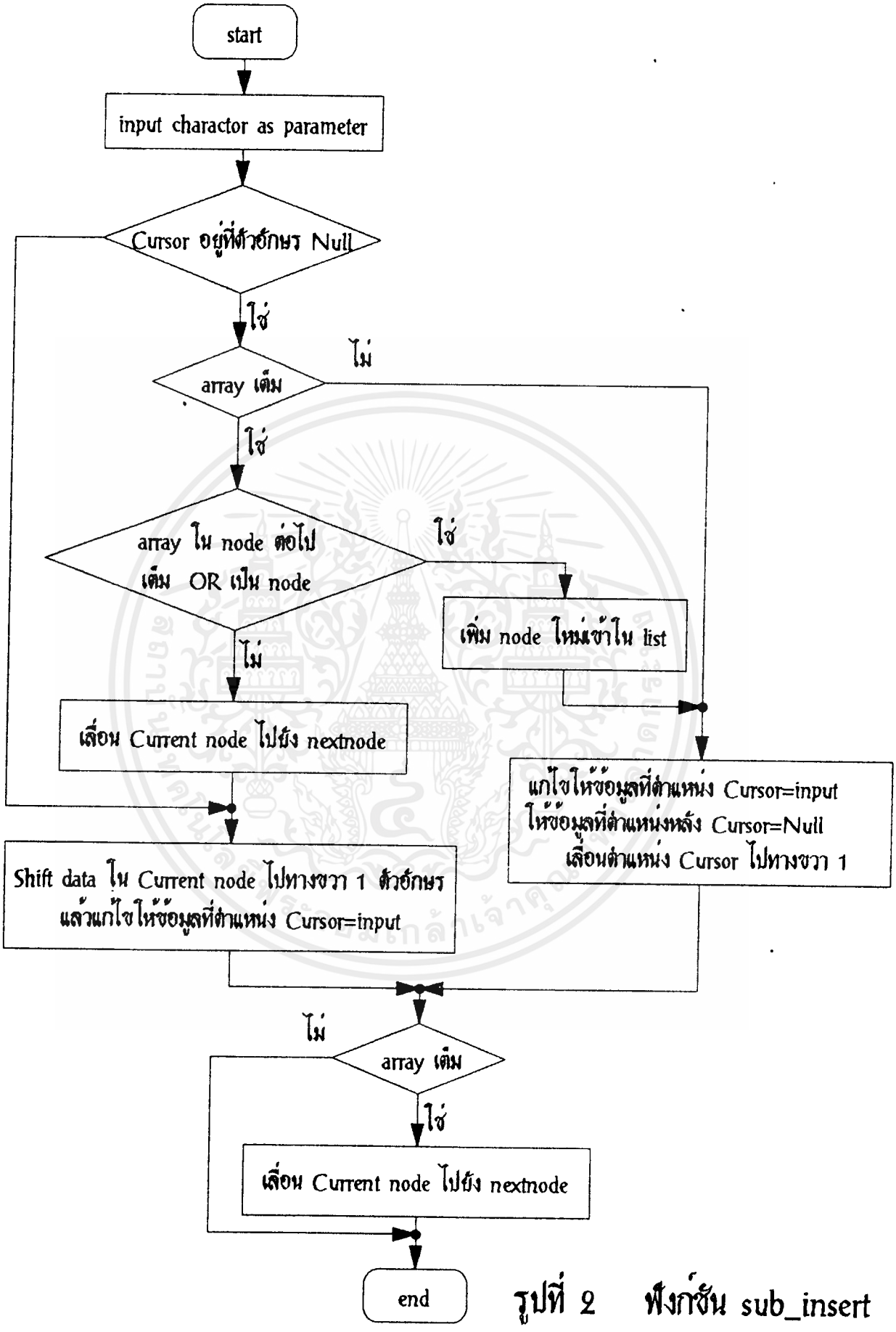
2. การเลื่อน Cursor ไปยังตัวอักษรถัดจาก Cursor เดิม แบ่งออกเป็น 3 ฟังก์ชันในลักษณะเดียวกันกับการเลื่อนแบบ previous

2.1 next มองเป็นตัวอักษร เลื่อนข้าม Control code และ update ตัวแปรแสดงผล

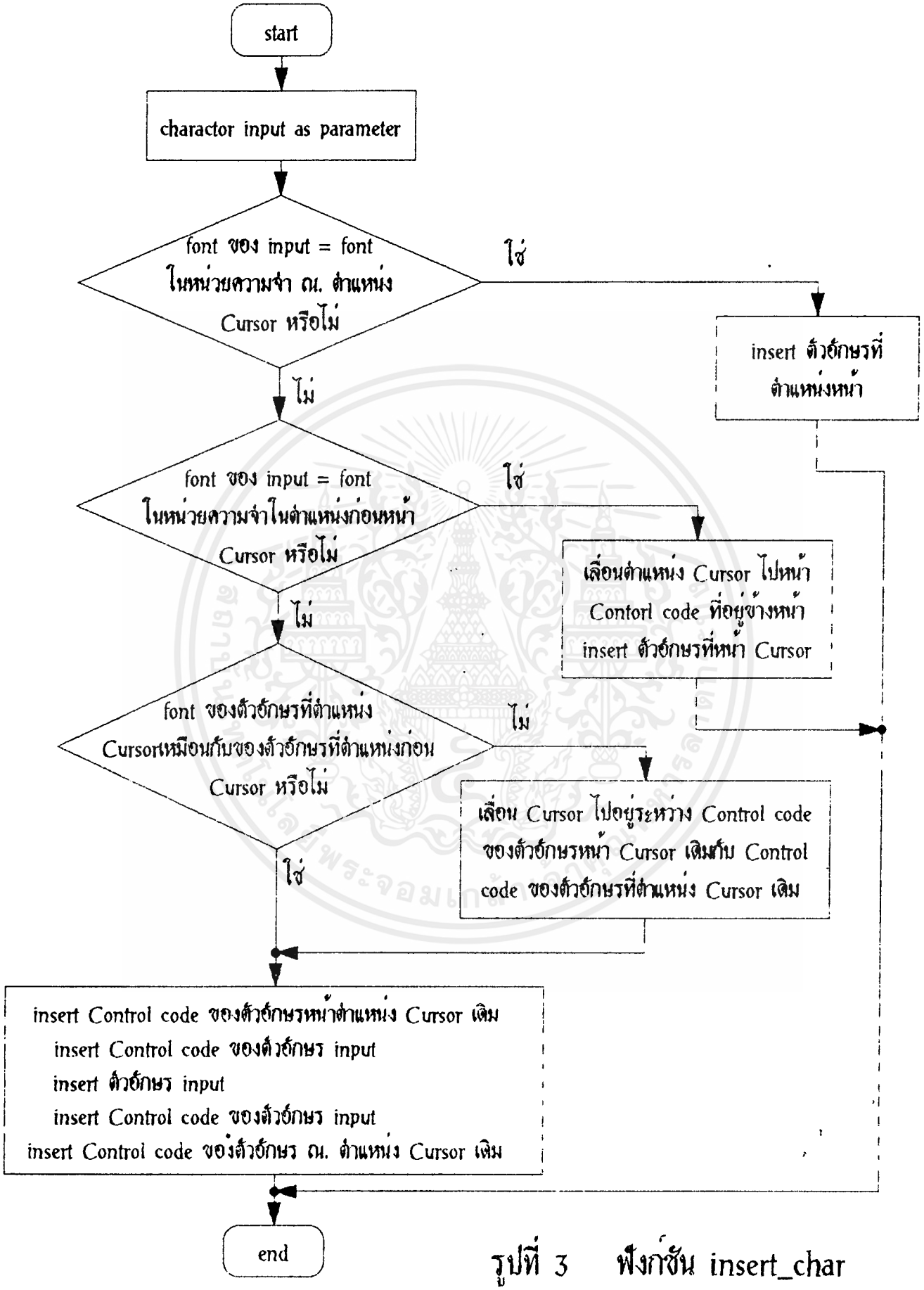
2.1 sub1_next ทำงานเหมือน next โดยไม่ update ตัวแปรแสดงผล

2.2 sub2_next มองเป็น ASCII ถ้าตำแหน่ง Cursor เดิมเป็นท้าย node หรือ paragraphจะทำการเลื่อนตำแหน่ง Current node และ Current paragraph ด้วย ฟังก์ชันในข้อ 2.1 และ 2.2 เรียกใช้ฟังก์ชันนี้จึงมีความสามารถในการเลื่อน Current node และ paragraph

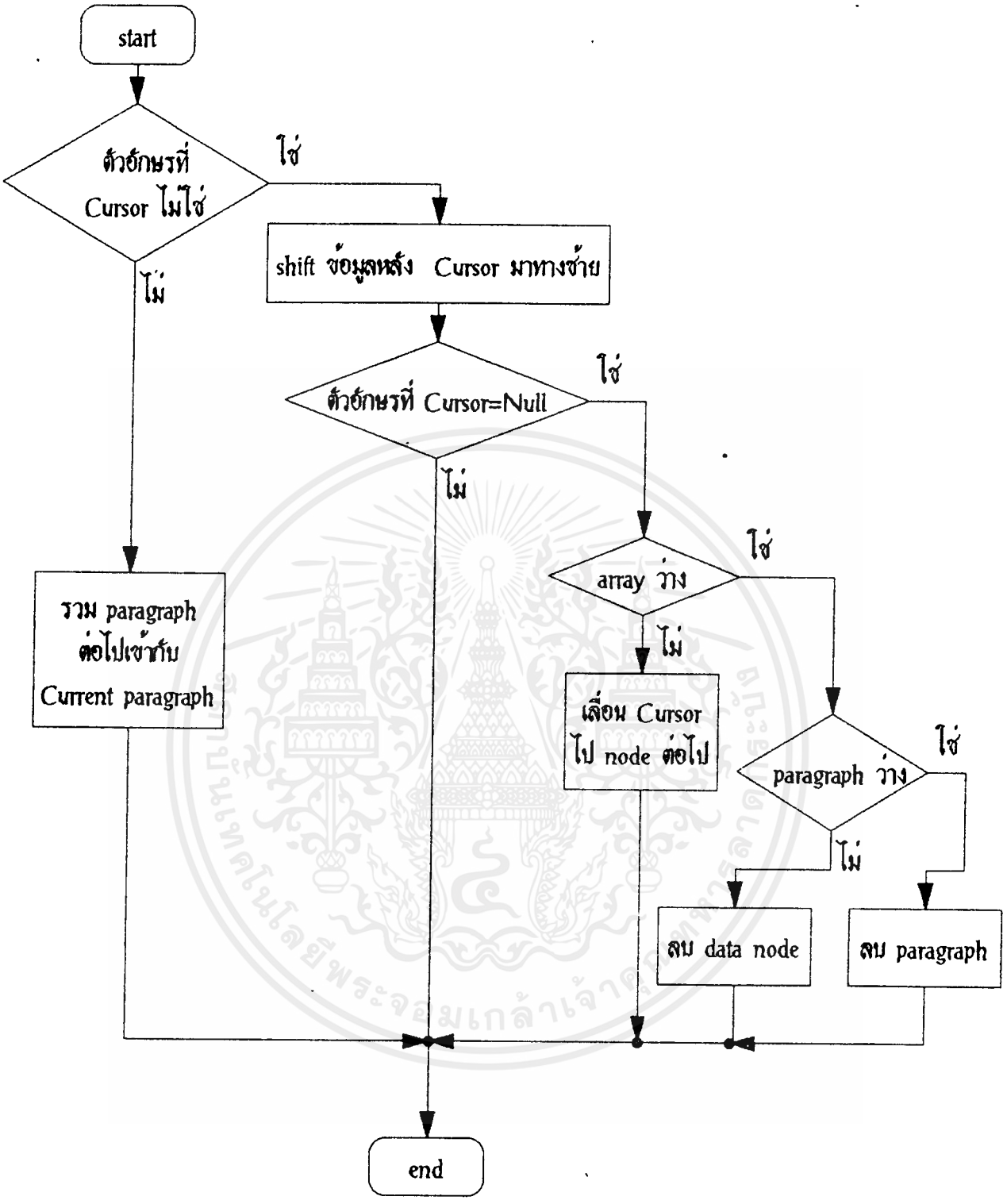
algorithm การทำงานของฟังก์ชันทั้ง 6 แสดงใน flowchart ในรูปที่ 6 ถึง 11



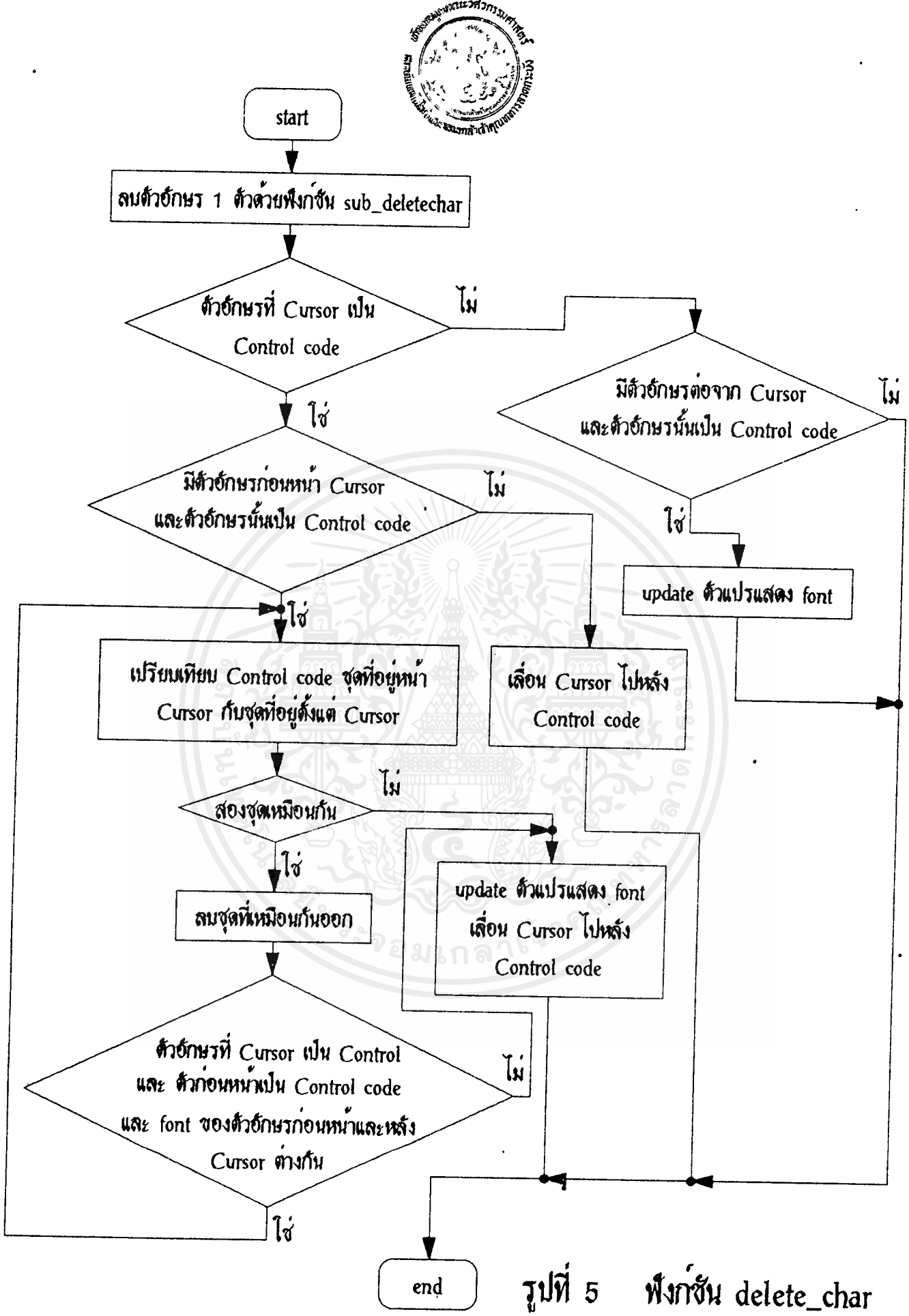
รูปที่ 2 ฟังก์ชัน sub_insert



รูปที่ 3 ฟังก์ชัน insert_char



รูปที่ 4 ฟังก์ชัน sub_deletechar



รูปที่ 5 ฟังก์ชัน delete_char

บทที่ 2
การแสดงผลตัวอักษรภาษาไทยและภาษาอังกฤษ

การแสดงผลตามปรกติของเครื่องคอมพิวเตอร์ในโหมดตัวอักษร จะใช้ตัวอักษรที่สร้างไว้ในรอมของการ์ดแสดงผลสำหรับแสดงผลทางจอภาพ สำหรับการแสดงผลตัวอักษรภาษาไทยมักจะมีสองแนวทางหลักๆที่ใช้ คือ แนวทางแรก จะแก้ไขที่ส่วนฮาร์ดแวร์ของการ์ดแสดงผลโดยตรงเพื่อให้สามารถแสดงผลภาษาไทยได้ วิธีนี้จะทำให้โปรแกรมภาษาไทยจะต้องผูกติดกับผู้ผลิตการ์ดนั้นๆ อีกวิธีหนึ่งคือใช้เทคนิคทางซอฟต์แวร์ในการจัดการแสดงผล โดยสร้างตัวอักษรเป็นรูปกราฟิกและแสดงผลเหมือนกับเป็นรูปกราฟิกธรรมดา ซึ่งวิธีนี้ใช้ในโครงงานนี้ สำหรับการ์ดแสดงผลตั้งแต่ VGA ขึ้นไปยังมีวิธีแสดงผลอีกรูปแบบหนึ่งที่กำลังนิยมใช้กันมากคือการเปลี่ยน Interrupt Vecter ที่ชี้ไปยังตารางตัวอักษรบนการ์ดแสดงผลใช้ชี้มายังตารางตัวอักษรที่เราออกแบบแทน วิธีนี้ยังมีข้อเสียตรงที่จะต้องใช้กับระบบแสดงผลวีจีเอขึ้นไปเท่านั้น

การแสดงผลตัวอักษรในโหมดกราฟิก

การแสดงผลในโหมดกราฟิกจะใช้การเขียนจุดลงไปบนจอภาพให้เป็นรูปของตัวอักษรขึ้นมา ตัวอักษรหนึ่งตัวก็จะใช้จุดจำนวน 8 x 20 สาเหตุที่ใช้ความสูงถึง 20 จุดก็เพื่อให้สามารถแสดงผลแบบสี่ระดับได้เลย รูปตัวอักษรที่เป็น Bit Map จะเก็บอยู่ในไฟล์ NORMAL.FON และไฟล์ ITALIC.FON ในหนึ่งตัวอักษรจะใช้ข้อมูล 20 ไบต์ โดยในไฟล์จะเก็บเรียงเป็นกลุ่มละ 20 ไบต์ตามรหัสตัวอักษรแบบสมอ. เช่น ข้อมูลในไฟล์ 20 ไบต์แรกก็จะเป็นภาพของตัวอักษรรหัส 00H ข้อมูล 20 ไบต์ต่อไปก็จะเป็นของตัวอักษรรหัส 01H จนกระทั่งครบ 256 ตัวอักษร ไฟล์จึงมีขนาด 5120 ไบต์

หน่วยความจำแสดงผลของจอภาพสี

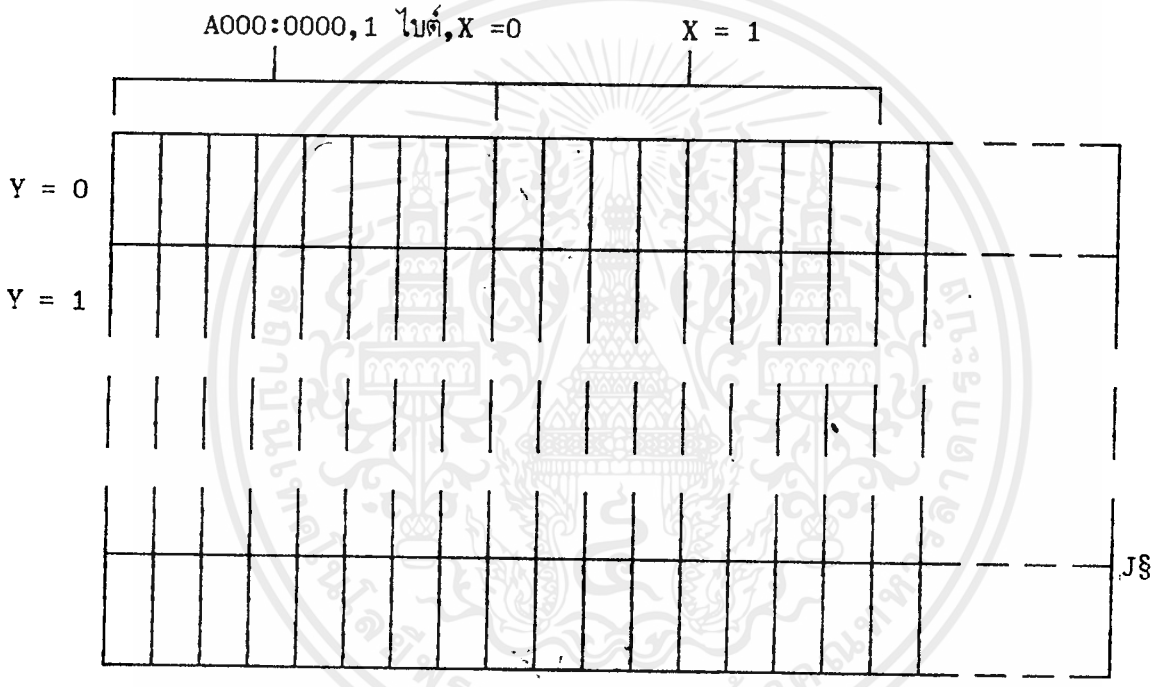
การแสดงภาพกราฟิกโดยทั่วไปแล้วจะช้ากว่าการแสดงผลในโหมดตัวอักษรมาก ดังนั้นจึงจำเป็นที่จะต้องใช้การติดต่อแสดงผลกับหน่วยความจำแสดงผลโดยตรงไม่ผ่านฟังก์ชันการจัดการของดอส เพื่อให้มีความเร็วสูงขึ้น หน่วยความจำแสดงผลของจอภาพจะมีตำแหน่งแอดเดรสต่างกันขึ้นอยู่กับชนิดของการ์ดแสดงผล ในจอภาพเฮอคิวลิสจะเริ่มที่ตำแหน่ง B000:0000 ในจอภาพวีจีเอจะเริ่มที่ A000:0000 ในส่วนของฟังก์ชันแสดงผลผ่านหน่วยความจำแสดงผลโดยตรงนี้ ได้พัฒนาโปรแกรมให้สามารถแสดงผลได้ทั้งจอภาพแบบวีจีเอ อีจีเอและเฮอคิวลิส แต่ในช่วงแรกนี้ใช้เฉพาะส่วนของจอวีจีเอขึ้นไปเท่านั้น หน่วยความจำแสดงผลของจอ

ภาพแบบวีจีเอจะเริ่มที่แอดเดรส A000:0000 และเรียงต่อกันไปเรื่อยๆ ฟังก์ชันแสดงผลนี้ชื่อว่า vscreen โดยมีอาร์กิวเมนต์เป็นตำแหน่ง X,Y บนจอภาพและข้อมูลที่ต้องการแสดงขนาด 1 ไบต์ (ตำแหน่ง X บนจอภาพจะเท่ากับ getmaxx()/8) สำหรับ mode จะใช้กำหนดว่าจะให้แสดงผลแบบ XOR หรือ FILL ซึ่งแบบหลังจะเป็นการเขียนข้อมูลทับเลย แต่แบบแรกจะเป็นการทำ eXclusive Or กับตัวอักษรเดิมจะใช้สำหรับการแสดงตัวพยัญชนะกับสระบน,วรรณยุกต์และสระล่าง

```
void vscreen(int x,int y,int data,int mode)
{
   char far *v_ram =NULL;                    /* this function start from*/
   long Row_v_ram = 0;                       /* position 0,0 on monitor*/
   long dis_p = 0;
   int  test_scan;
   int  maxx;

  maxx = getmaxx();
  /*  แปลงค่า X,Y เป็นตำแหน่งแอดเดรสในหน่วยความจำแสดงผล */
  if(TypeScreen)
  { /* กรณีที่เป็นจอภาพแบบ เฮอคิวลิส */
     v_ram = (char far *)0xB0000000L;
     test_scan = y % 4;                      /* test No. scan */
     if( test_scan == 0 )                    /* the first line */
        Row_v_ram += 0x6000 + ((int)(y /4) -1)*0x5A;
     else
        Row_v_ram += (test_scan -1)*0x2000 + ((int)(y/4))*0x5A;
     dis_p = (long)Row_v_ram +(long)(x);
  }else{ /* จอภาพสี */
     v_ram = (char far *)0xA0000000L;
     dis_p = (long)((maxx+7)/8)*(long)(y)+(long)(x);
```

```
    }
    if(mode) /* เลือกโหมดการแสดงผล*/
       *(char far *)((long)v_ram +dis_p) ^= data; /* XOR */
    else
       *(char far *)((long)v_ram +dis_p)  = data; /* FILL */
}
```



รูปที่ 1 ตำแหน่งในหน่วยความจำเทียบกับตำแหน่งบนจอภาพ

การแสดงผลตัวอักษรแบบหลายรูปแบบ

การแสดงแบบตัวอักษรหลายรูปแบบมีดังต่อไปนี้

1. ตัวหนา
2. ตัวขยาย
3. ตัวขีดเส้นใต้หนึ่งเส้น
4. ตัวขีดเส้นใต้สองเส้น

การแสดงผลตัวอักษรตัวหนา

แบบตัวอักษรที่ใช้ในโปรแกรมประมวลผลคำนี้เป็นตัวอักษรแบบ Bit Map และมีขนาดของตัวอักษรคงที่ คือ ขนาด 8x20 จุด การแสดงผลตัวอักษรตัวหนึ่งๆ จะแสดงโดยการเขียนข้อมูลลงหน่วยความจำแสดงผลครั้งละ 1 ไบต์จำนวน 20 ครั้ง ในการปรับให้มีการแสดงผลเป็นแบบตัวหนานี้จะมีการเปลี่ยนแปลงขั้นตอนการแสดงผลเดิมจากที่เขียนลงโดยตรง เป็นการนำข้อมูลมาทำการเลื่อนไปทางขวา 1 บิตแล้ว นำไป OR กับข้อมูลเดิมที่ไม่มีการเลื่อนข้อมูล แล้วจึงนำไปแสดงผลทางจอภาพ การแสดงตัวอักษรตัวหนานี้จะทำวิธีเดียวกันนี้ในแบบตัวอักษรอื่นที่มีการแสดงตัวหนา ตัวอย่างเช่น ตัวอักษรปกติประกอบด้วยข้อมูลต่อไปนี้

0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H
0 0 1 1 1 1 0 0 ค่าในฐาน16เป็น 3CH
0 1 0 0 0 0 1 0 ค่าในฐาน16เป็น 42H
0 0 1 0 0 0 1 0 ค่าในฐาน16เป็น 22H
0 1 0 0 0 0 1 0 ค่าในฐาน16เป็น 42H
0 1 0 0 0 0 1 0 ค่าในฐาน16เป็น 42H
0 1 0 0 0 0 1 0 ค่าในฐาน16เป็น 42H
0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H
0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H

ทำให้เป็นตัวหนาโดยการเลื่อนบิตไปทางขวาและนำไป OR จะได้ข้อมูลเป็น

0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H
0 0 1 1 1 1 1 0 ค่าในฐาน16เป็น 3EH
0 1 1 0 0 0 1 1 ค่าในฐาน16เป็น 63H
0 0 1 1 0 0 1 1 ค่าในฐาน16เป็น 33H
0 1 1 0 0 0 1 1 ค่าในฐาน16เป็น 63H
0 1 1 0 0 0 1 1 ค่าในฐาน16เป็น 63H
0 1 1 0 0 0 1 1 ค่าในฐาน16เป็น 63H
0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H
0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H

*หมายเหตุ ขนาดตัวอักษรที่ใช้ในโปรแกรมจริง จะมีขนาดทั้งหมด 8x20 จุด
แบบตัวอักษรตัวเอียงที่ใช้อยู่ในโปรแกรมตอนนี้ ในลักษณะตัวปกติจะมีการใช้ข้อมูลในส่วนของ บิตสุดท้ายอยู่แล้วเมื่อทำให้เป็นตัวหนาโดยการเลื่อนข้อมูลไปทางขวา ทำให้ข้อมูลบางส่วนสูญเสียไป ทางแก้คือ การออกแบบตัวอักษรตัวเอียงใหม่ โดยให้มีที่ว่างอย่างน้อย หนึ่งบิต ที่ขอบซ้าย หรือ ขวาของตัวอักษร ซึ่งอีกวิธีที่จะทำได้คือ ขยายขนาดการแสดงผลบนจอภาพจากความกว้าง 8 จุดในตัวอักษรทุกแบบ เป็นเมื่อมีการพิมพ์ตัวอักษรตัวเอียงจะขยายขนาดออกเป็นขนาด 9 จุด ซึ่งการจัดการแสดงผลจะซับซ้อนมากกว่าเดิม

การแสดงผลตัวอักษรตัวขยาย

การแสดงผลตัวอักษรตัวขยายจะเป็นการเพิ่มขนาดของตัวอักษร เพื่อให้ตัวอักษรมีขนาดใหญ่ เพื่อใช้ในการเน้นข้อความหรือเป็นหัวข้อ การขยายตัวอักษรนี้จะขยายตัวอักษร ในแนวแกน X ออกหนึ่งเท่า โดยที่ขนาดในแนวแกน Y จะคงที่ วิธีการทำงานจะต่างจากการ แสดงผลปกติโดยแต่เดิมจุดหนึ่งจุดจะแสดงหนึ่งจุดบนจอภาพ การแสดงตัวขยายจะนำจุดหนึ่งจุดมาขยายเป็น 2 จุดในแนวแกน X บนจอภาพ โดยที่ในแนวแกน Y ยังคงแสดงเพียงจุดเดียวเหมือนปกติ ตัวอย่างการแสดงผลเป็นดังนี้

0 0 0 0 0 0 0 0	ค่าในฐาน16เป็น 00H
0 0 1 1 1 1 0 0	ค่าในฐาน16เป็น 3CH
0 1 0 0 0 0 1 0	ค่าในฐาน16เป็น 42H
0 0 1 0 0 0 1 0	ค่าในฐาน16เป็น 22H
0 1 0 0 0 0 1 0	ค่าในฐาน16เป็น 42H
0 1 0 0 0 0 1 0	ค่าในฐาน16เป็น 42H
0 1 0 0 0 0 1 0	ค่าในฐาน16เป็น 42H
0 0 0 0 0 0 0 0	ค่าในฐาน16เป็น 00H
0 0 0 0 0 0 0 0	ค่าในฐาน16เป็น 00H

เมื่อทำการแสดงผลตัวขยายข้อมูลจะถูกขยายเป็น 2 ไบต์ดังนี้

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H 00H

0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 ค่าในฐาน16เป็น 0FH F0H

0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 ค่าในฐาน16เป็น 30H 0CH

0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 ค่าในฐาน16เป็น 0CH 0CH

0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 ค่าในฐาน16เป็น 30H 0CH

0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 ค่าในฐาน16เป็น 30H 0CH

0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 ค่าในฐาน16เป็น 30H 0CH

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H 00H

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ค่าในฐาน16เป็น 00H 00H

ฟังก์ชันที่ใช้ในการขยายตัวอักษรชื่อ Expand2Byte โดยจะรับอาร์กิวเมนต์สามตัวคือ Source เป็นตัวอักษรปรกติ และจะส่งค่ากลับโดยตัวแปร HByte และ LByte จำนวนสองไบต์

```
void Expand2Byte(unsigned char Source,unsigned char *HByte,
                 unsigned char *LByte)
{
   register   int i;
   unsigned char mark =0xC0;

   *HByte &= 0;
   *LByte &= 0;
   for( i =7; i >=4; i--)        /* ขยาย 4 บิตแรกเป็นหนึ่งไบต์ */
   {
      if(TestBit(i,Source))
         *HByte |=mark;
      mark >>= 2;
   }
  mark = 0xC0;
```

```
    for( i =3; i >=0; i--)        /* ขยาย 4 บิตสุดท้ายเป็นหนึ่งไบต์ */
    {
       if(TestBit(i,Source))
          *LByte |=mark;
       mark >>= 2;
    }
}
```

การแสดงผลตัวขีดเส้นใต้

การแสดงผลขีดเส้นใต้นั้นทำได้โดยการนำข้อมูล FFH ไป OR ที่ข้อมูลตัวอักษรก่อนที่จะแสดงผล โดยที่ตัวขีดเส้นใต้ตัวเดียวจะนำข้อมูล OR ที่ตำแหน่ง ไบต์ที่ 17 ของการแสดงผลตัวอักษร และการแสดงผลตัวขีดเส้นใต้สองเส้น จะ OR ที่ตำแหน่ง 16 และ 18 การขีดเส้นใต้ตัวอักษรนี้จะขีดเส้นตัวอักษรเฉพาะกลุ่มตัวอักษรที่มีการ เลื่อนตำแหน่งเมื่อมีการแสดงผล เช่น ตัวพยัญชนะ สระหน้า หรือ สระหลัง เป็นต้น

ฟังก์ชันส่วนแสดงผลตัวอักษร

ฟังก์ชัน PutC จะมีหน้าที่แสดงผลตัวอักษรที่ตำแหน่ง X,Y บนจอภาพตามแอทริบิวและโหมดที่กำหนด ฟังก์ชันนี้จะถูกเรียกทุกครั้งที่มีการแสดงผลตัวอักษรทางจอภาพกราฟิก โดยจะเรียกใช้ฟังก์ชัน vscreen อีกต่อหนึ่ง ในส่วนของฟังก์ชัน PutC นี้จะจัดการเกี่ยวกับการแสดงผลของตัวอักษรแอทริบิวต่างๆด้วย เช่นหากเป็นตัวหนาก็ต้องทำการเลื่อนบิตไปทางขวาก่อนแล้วจึงแสดงผล ฟังก์ชันในโปรแกรมเป็นดังนี้

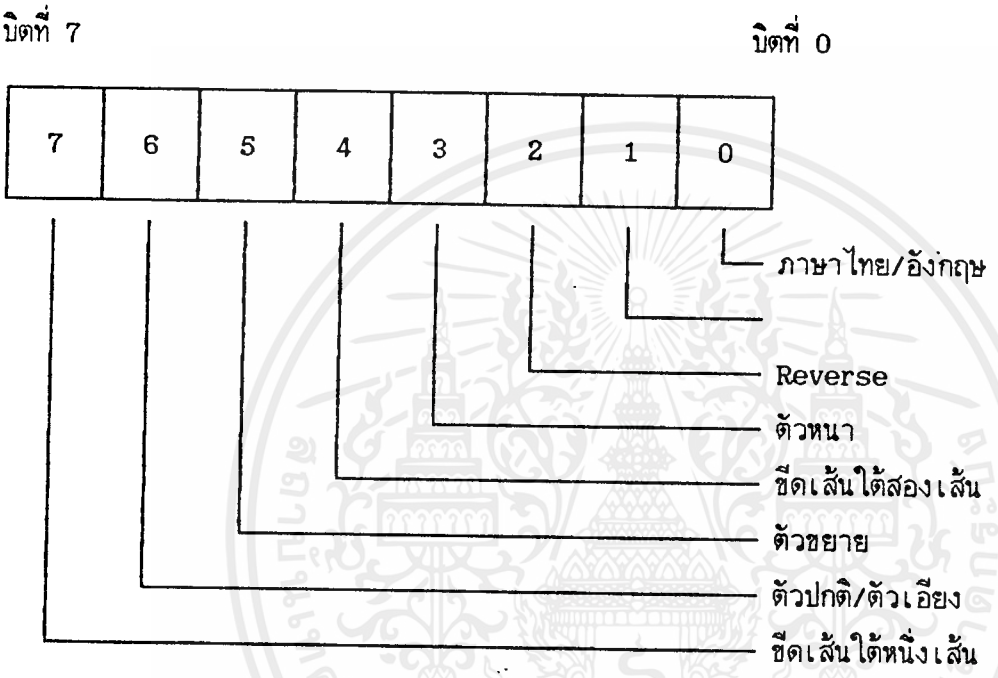
```
void PutC(unsigned char asciicode,int X,int Y,int mode,unsigned char attrb)
{
   register int i;
   int YExpand;
   unsigned char code;
   unsigned char temp;
   unsigned char HighByte;
```

```
unsigned char LowByte;

YExpand = Y*CharLong;                    /*ระยะระหว่างตัวอักษรในแกน Y */
for( i =0; i <20; i++)                   /* วนลูปจนแสดงครบทั้ง 20 เส้น*/
{   /* อ่านค่า Bitmap ที่จะต้องนำไปแสดง*/
   code =((attrb&ITALIC)!=0)? AlphI[asciicode][i]:AlphN[asciicode][i];
   if((attrb&EMPHASIZED)!= 0)
   {
      temp = code >> 1;                  /* ตัวหนา */
      code |= temp;
   }
 /* ตัวขีดเส้นใต้สองเส้นจะอยู่ที่เส้นที่ 16 กับ 18 */
 /* ตัวขีดเส้นใต้หนึ่งเส้นจะอยู่ที่เส้น 17 */
  if((((attrb&DUNDERLINE)!=0&&(i==16||i==18))||
  ((attrb&UNDERLINE)!=0&&i==17))&&TACchtype(asciicode) <=FV3)
     code = 0xFF;                        /* ตัวขีดเส้นใต้ */
  if((attrb&REVERSE) != 0)               /*ตัวรีเวิร์ส*/
     code = ~code;
  if((attrb&EXPAND)!=0)
  {                 /* ตัวขยาย */
     Expand2Byte(code,&HighByte,&LowByte);
     vscreen(X,i+YExpand,HighByte,mode);
     vscreen(X+1,i+YExpand,LowByte,mode);
  }else
     vscreen(X,i+YExpand,code,mode);          /* ตัวอักษรปรกติ */
}
}
```

ตัวแปรควบคุมสถานะการแสดงผลตัวอักษร

ในการแสดงผลนั้นจะมีตัวแปร ตัวหนึ่งกำหนดลักษณะของแบบตัวอักษรที่จะแสดงผลโดยตัวแปรนี้มีขนาด 1 ไบต์ และใช้การเซ็ต หรือ รีเซ็ตบิต ในตัวแปรนี้บอกลักษณะของการแสดงผล ความหมายของแต่ละบิต มีดังรูปที่ 2



รูปที่ 2 ความหมายแต่ละบิตของแอทริบิวต์

ตัวแปรตัวนี้ชื่อ CharAttrb โดยที่การจัดการข้อมูลจะเป็นแบบบิต ทุกบิตจะสามารถเซ็ตหรือ รีเซ็ตค่าโดยที่ไม่ขึ้นกับบิตใดๆ ยกเว้นบิตที่ 5 กับ 4 ที่จะเซ็ตพร้อมกันไม่ได้ ขนาดของตัวแบบ 1 ไบต์นั้นไม่พอที่จะเก็บลักษณะของข้อมูลได้ครบ เนื่องจากว่าแบบตัวอักษรยังขาด ตัวยกขึ้น ตัวห้อย ดังนั้นอาจมีการขยายขนาดเป็น 2 ไบต์ หรือ อาจนำบางบิตออกไป เช่น บิตภาษาไทย/อังกฤษ แต่ในโครงงานนี้ยังใช้ขนาด 1 ไบต์อยู่โดยใช้แต่ละบิตตามที่กล่าวข้างต้น

บทที่ 3
อักขรวิถีภาษาไทยในคอมพิวเตอร์

อักขรวิถีภาษาไทยหมายถึงการตรวจสอบลำดับการป้อนข้อความภาษาไทยและการจัดระดับการแสดงผลภาษาไทยโดยที่จะใช้หลักอักขรวิถีภาษาไทยสำหรับคอมพิวเตอร์ ตามร่างมาตรฐาน วทท. 2.0 โดย คณะทำงานร่างข้อกำหนดร่วมเพื่อการเขียนโปรแกรมซึ่งแสดงผลเป็นภาษาไทย (Thai API Consortium) ร่างในโครงงานโปรแกรมประมวลผลคำนี้ได้ใช้แนวทางจากร่างมาตรฐานนี้เป็นหลักในการพัฒนาโปรแกรมในทุกส่วนที่เกี่ยวข้องกับภาษาไทย ในร่างมาตรฐานนี้ได้กล่าวครอบคลุมเรื่องที่เกี่ยวกับภาษาไทยเกือบทั้งหมดทั้งการจัดระดับการแสดงผล การแปลงรหัสมาตรฐานเป็นรหัสอื่นๆ และรหัสภาษาไทยสำหรับเครื่องพิมพ์ต่างๆ เป็นต้น ส่วนของร่างมาตรฐานที่นำมาใช้ในโครงงานโปรแกรมประมวลผลคำนี้จะประกอบด้วยหัวข้อต่อไปนี้

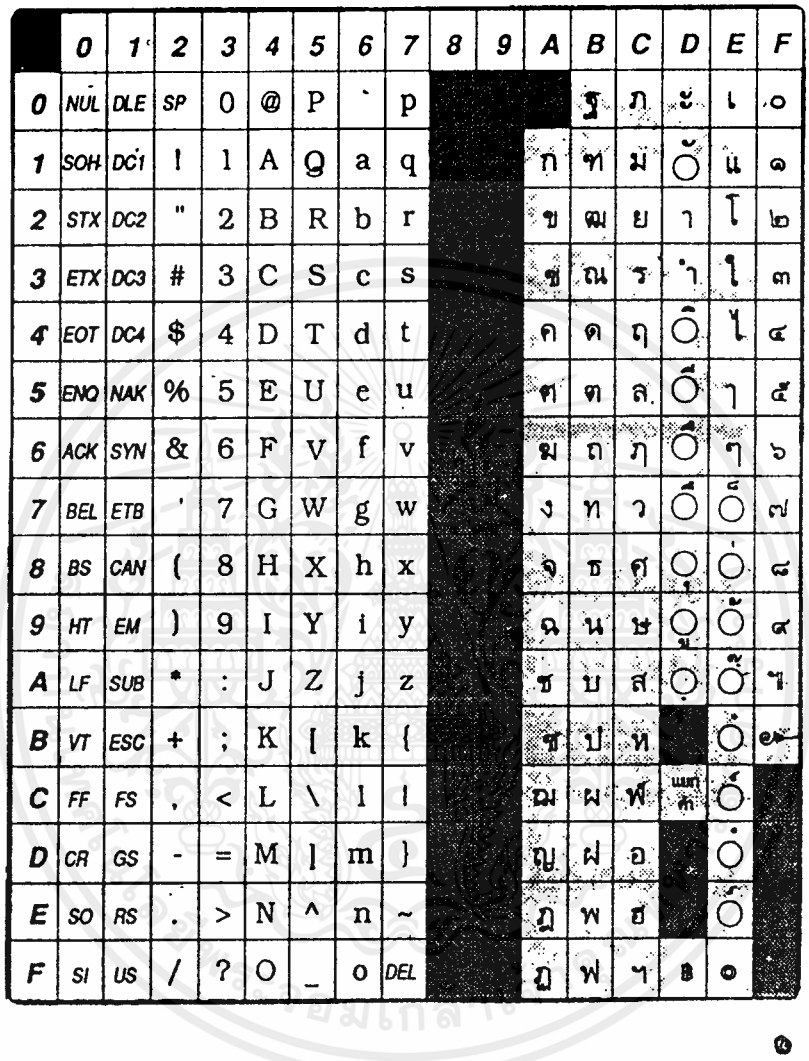
1. ชุดอักขระ
2. การตรวจสอบลำดับการป้อนข้อความภาษาไทย

ชุดอักขระ

ชุดอักขระที่ใช้ในโปรแกรมนี้เป็นตามร่างมาตรฐาน วทท. โดยที่ชุดอักขระจะมีดังตารางที่ 1 ในการใช้งานจริงจะมีการนำตำแหน่งรหัสที่ว่างบนตารางอักขระมาใช้เป็นรหัสควบคุมต่างๆ เช่น รหัสแสดงการตัดคำ 8DH และนำมาใช้เป็นรหัสตาราง ตารางชุดอักขระพื้นฐานนี้ เป็นมาตรฐานที่ใช้ในการติดต่อสื่อสารข้อมูลไม่ได้ครอบคลุมถึงรหัสที่ใช้ในโปรแกรม ดังนั้นเราจึงสามารถนำตำแหน่งบางตำแหน่งมาใช้ได้สำหรับตาราง มีข้อสังเกตุคือในตารางชุดอักขระพื้นฐานนี้ จะไม่มีการใช้สระหรือวรรณยุกต์ผสม เช่น ไม้หันอากาศกับไม้โท แต่จะพบว่ามีการผสมรหัสระหว่างสองตัวอักษรในตารางรหัสสำหรับการพิมพ์ภาษาไทยทางเครื่องพิมพ์ และในตารางมีการวางตำแหน่งของรหัสที่ไม่ใช้บ่อยนัก เช่น ตัวฟองมัน สามาเหตุที่มีการนำอักษรเหล่านี้รวมอยู่ในตารางด้วยก็เพื่อให้เกิดความสมบูรณ์มีตัวอักษรภาษาไทยครบถ้วนสำหรับการสร้างเอกสาร

การตรวจสอบลำดับการป้อนข้อความภาษาไทย

การตรวจลำดับตัวอักษรขณะที่ทำการพิมพ์ จะมีประโยชน์ในการช่วยเตือนผู้ใช้โปรแกรมเมื่อมีการพิมพ์ข้อมูลผิด เช่น อาจพิมพ์สระบนติดกันสองตัวซึ่งเป็นไปไม่ได้ และการตรวจสอบลำดับตัวอักษรจะทำให้ลำดับ



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p				ฐ	ภ	ะ	เ	๐
1	SOH	DC1	!	1	A	Q	a	q			ก	ฑ	ม	ั	แ	๑
2	STX	DC2	"	2	B	R	b	r			ข	ฒ	ย	า	โ	๒
3	ETX	DC3	#	3	C	S	c	s			ฃ	ณ	ร	ำ	ใ	๓
4	EOT	DC4	$	4	D	T	d	t			ค	ด	ฤ	ิ	ไ	๔
5	ENQ	NAK	%	5	E	U	e	u			ฅ	ต	ล	ี	ๅ	๕
6	ACK	SYN	&	6	F	V	f	v			ฆ	ถ	ฦ	ึ	ๆ	๖
7	BEL	ETB	'	7	G	W	g	w			ง	ท	ว	ื	็	๗
8	BS	CAN	(	8	H	X	h	x			จ	ธ	ศ	ุ	่	๘
9	HT	EM	)	9	I	Y	i	y			ฉ	น	ษ	ู	้	๙
A	LF	SUB	*	:	J	Z	j	z			ช	บ	ส	ฺ	๊	ฯ
B	VT	ESC	+	;	K	[	k	{			ซ	ป	ห		๋	๚
C	FF	FS	,	<	L	\	l	|		ฌ	ผ	ฬ	็	์	
D	CR	GS	-	=	M	]	m	}			ญ	ฝ	อ		๎	
E	SO	RS	.	>	N	^	n	~			ฎ	พ	ฮ		๏	
F	SI	US	/	?	O	_	o	DEL			ฏ	ฟ	ฯ	ๆ	๐	

ตารางที่ 1 รหัสของตัวอักขระตามร่างมาตรฐาน วทท. 2.0

การป้อนข้อมูลเข้ามีรูปแบบที่แน่นอนไม่ต้องสร้างส่วนสำหรับการจัดเรียงลำดับข้อมูลใหม่ ลักษณะการจัดเรียงลำดับข้อมูลหมายถึง การจัดเรียงของตัวอักษรในหน่วยความจำ เช่น คำว่า "อ้อยควั่น" จะต้องมีการจัดลำดับในหน่วยความจำดังนี้ อ-้-อ-ย-ค-ว-ั-่-น ในโปรแกรมประมวลผลคำอื่นๆที่ไม่ได้ตรวจสอบลำดับการป้อนข้อมูลตาม วทท. 2.0 ผู้ใช้สามารถที่จะพิมพ์คำว่า "ควั่น" โดยจะพิมพ์ ไม้หันอากาศหรือไม้เอกก่อนก็ได้ ซึ่งผู้พัฒนาโปรแกรมจะเขียนโปรแกรมเพิ่มสำหรับการจัดเรียงลำดับของรหัสให้ถูกต้องไว้แล้ว แต่ถ้าได้กำหนดลำดับการพิมพ์ข้อมูลไว้แล้ว ส่วนจัดลำดับข้อมูลใหม่ก็ไม่จำเป็นต้องใช้

ตัวอย่างการตรวจสอบลำดับการป้อนข้อความภาษาไทย เช่น ต้องการพิมพ์คำว่า

มาที่นา

โดยพิมพ์ตามลำดับต่อไปนี้

ม - า - ท - ่ - ี - น - า

จะเห็นว่าลำดับการป้อนข้อมูลผิดหลักการสะกดคำภาษาไทย คือพิมพ์ไม้เอก (วรรณยุกต์) ก่อนที่จะพิมพ์ตัวอักษรสระอี เมื่อมีลำดับการป้อนข้อมูลที่ผิดในลักษณะนี้ จะต้องมีการเตือนผู้ใช้ให้ทราบเพื่อที่จะทำการแก้ไข ในในโปรแกรมประมวลผลคำภาษาไทย/อังกฤษนี้ เมื่อมีข้อผิดพลาดในลักษณะข้างต้นเกิดขึ้น โปรแกรมก็จะใช้เสียงเตือนให้ผู้ใช้ทราบและไม่รับการกดปุ่มที่ผิดนั้นๆ

ส่วนตรวจสอบลำดับตัวอักษร

ในส่วนตรวจสอบลำดับการป้อนตัวอักษร ในขั้นแรกก็จะแบ่งชุดอักขระออกเป็น 6 กลุ่มดังนี้

1.อักขระควบคุม (Control Character หรือ CONS) ได้แก่อักขระที่ไม่แสดงผลแต่จะใช้ในการควบคุมการทำงานของโปรแกรม และการแสดงผลต่างๆ

2.พยัญชนะไทย

3.สระไทย สระจะแบ่งเป็นกลุ่มย่อย ดังนี้

3.1 สระนำ คือสระที่นำหน้าพยัญชนะต้น (leading vowels) หรือ LV มี 5 ตัว ได้แก่ เ แ ไ ใ โ

3.2 สระตาม คือสระที่อยู่หลังพยัญชนะต้น (forward vowels) มี 6 ตัว ได้แก่ สระตามทั่วไป 4 ตัวคือ ะ า ำ ๅ และสระตามพิเศษ 2 ตัว คือ ฤ ฤๅ (รือ)

3.3 สระล่าง คือสระที่อยู่ใต้พยัญชนะต้น(below vowels หรือ BV) มี2 ตัว ได้แก่ อุ อู

3.4 สระบน คือสระที่อยู่เหนือพยัญชนะต้น (above vowels หรือ AV) มี 5 ตัว ได้แก่ อิ อี อึ อื อั

4. เครื่องหมายวรรณยุกต์(tone mark)หรือ TONE มี 4 รูปแบบคือ อ่ อ้ อ๊ อ๋

5. เครื่องหมายกำกับเสียง(diacritics หรือ -D) ประกอบด้วยกลุ่มสัญลักษณ์ดังนี้

- เครื่องหมายกำกับเสียงเหนือพยัญชนะหรือตัวสะกด (Above diacritics หรือ AD) มี 4 ตัว ได้แก่ นิคหิต ทัณฑฆาต ไม้ไต่คู้ ยามักการ

- เครื่องหมายกำกับเสียงใต้ตัวสะกดหรือตัวควบ(below diacritic หรือ BD) มีหนึ่งตัว คือพินทุ

6. สัญญลักษณ์อื่นๆ (non-composible หรือ NON) ได้แก่สัญญลักษณ์ที่นอกเหนือจากกลุ่มที่กล่าวมาทั้งหมด มีคุณสมบัติเหมือนอักขรวิถีแบบอังกฤษทั่วไปคือ จะใช้เนื้อที่การแสดงผลตัวละหนึ่งเซล

เพื่อให้สามารถบรรยายข้อกำหนดภาษาไทยได้สมบูรณ์จึงมีการแยกกลุ่มของรหัสข้อมูลเป็นกลุ่มย่อยๆ ดังตารางที่ 3

ประเภท	จำนวน	รายละเอียด
CTRL	66	กลุ่มอักขระควบคุม
NON	119	อักขระตามมาตรฐาน ISO 646-1983 อักขระพิเศษตามมาตรฐาน มอก.620-2533 และอักขระที่ไม่มีการกำหนดมาตรฐานใดๆได้แก่ รหัส 219,221,222,252,253,254
CONS	44	พยัญชนะไทย
LV	5	เ แ โ ใ ไ
FV1	3	ะ า ำ
FV2	1	า
FV3	2	ฤ ฤ (รือ)
BV1	1	ุ
BV2	2	ู

BD	1	พินทุ
TONE	4	(ก่) เอก (ก้) โท (ก๊) ตรี (ก๋) จัตวา
AD1	2	นิคหิต ทัณฑฆาต
AD2	1	ไม้ไต่คู้
AD3	1	ยามักการ
AV1	1	อิ
AV2	2	อั อิ
AV3	2	อี อึ

ตารางที่ 2 ตารางแสดงกลุ่มของตัวอักษร

การกำหนดประเภทอักขระย่อยนี้จะใช้ประโยชน์ในการตรวจลำดับการป้อนข้อมูลของตัวอักษรโดยที่กลุ่มต่างๆที่แบ่งนั้นจะแทนด้วยค่าคงที่จำนวนเต็มบวกหนึ่งค่าเรียกว่า"เลขบอกประเภท" เพื่อใช้ในการเขียนโปรแกรม โดยที่ค่าเหล่านี้มีการกำหนดไว้ในร่างมาตรฐานภาษาไทย วทท. 2.0 ดังนี้

ชื่อประเภท	เลขบอกประเภท
CTRL	0
NON	1
CONS	2
LV	3
FV1	4
FV2	5
FV3	6
BV1	7
BV2	8
BD	9
TONE	10

AD1	11
AD2	12
AD3	13
AV1	14
AV2	15
AV3	16

ตารางที่ 3 เลขบอกประเภทของตัวอักษรกลุ่มต่างๆ

กฎในการตรวจสอบลำดับตัวอักษร

การตรวจสอบลำดับตัวอักษรจะเป็นการตรวจสอบลำดับของข้อมูลที่เข้ามาว่าถูกต้องตามลำดับตัวอักษรที่กำหนดหรือไม่ โดยพิจารณาจากตัวอักษรที่จะป้อนเข้าเรียกตัวตัวตาม และตัวอักษรก่อนหน้าเรียกตัวนำ โดยหลักการพิจารณาเป็นดังตารางที่ 6

ผล	ความหมาย	ลักษณะการแสดงผลของระบบแสดงผล
A	accept	ในเซลถัดไป
C	accept	ในเซลเดียวกับตัวตั้ง
R	reject	ในเซลถัดไป
X	accept	ไม่แสดงผล

ตารางที่ 4 ตารางความหมายของสัญลักษณ์ในตารางที่ 5

	ตัวตาม															
ตัวนำ	1	2	3	4	5	6	0	7	8	9	10	11	12	13	14	15	16
1 NON	A	A	A	R	R	A	X	R	R	R	R	R	R	R	R	R	R
2 CONS	A	A	A	A	R	A	X	C	C	C	C	C	C	C	C	C	C
3 LV	R	A	R	R	R	R	X	R	R	R	R	R	R	R	R	R	R
4 FV1	R	A	R	A	R	A	X	R	R	R	R	R	R	R	R	R	R
5 FV2	A	A	A	A	R	A	X	R	R	R	R	R	R	R	R	R	R
6 FV3	A	A	A	R	A	R	X	R	R	R	R	R	R	R	R	R	R
0 CTRL	A	A	A	A	A	A	X	R	R	R	R	R	R	R	R	R	R
7 BV1	A	A	A	A	R	A	X	R	R	R	C	C	R	R	R	R	R
8 BV2	A	A	A	R	R	A	X	R	R	R	C	R	R	R	R	R	R
9 BD	A	A	A	R	R	A	X	R	R	R	R	R	R	R	R	R	R
10 TONE	A	A	A	A	A	A	X	R	R	R	R	R	R	R	R	R	R
11 AD1	A	A	A	R	R	A	X	R	R	R	R	R	R	R	R	R	R
12 AD2	A	A	A	R	R	A	X	R	R	R	R	R	R	R	R	R	R
13 AD3	A	A	A	R	R	A	X	R	R	R	R	R	R	R	R	R	R
14 AV1	A	A	A	R	R	A	X	R	R	R	C	C	R	R	R	R	R
15 AV2	A	A	A	R	R	A	X	R	R	R	C	R	R	R	R	R	R
16 AV3	A	A	A	R	R	A	X	R	R	R	C	R	C	R	R	R	R

ตารางที่ 5 ตารางสำหรับตรวจสอบลำดับของตัวอักษร

การประยุกต์พัฒนาในโปรแกรมประมวลผลคำ

การตรวจสอบประเภทตัวอักษร

ฟังก์ชั่นที่ใช้ในการตรวจสอบประเภทของตัวอักษรคือ

```
int TACchtype(unsigned char ch);
```

ค่า argument ที่ผ่านให้ฟังก์ชั่นคือค่ารหัสของตัวอักษรที่จะตรวจสอบ โดยที่จะส่งค่าเลขบอกประเภทกลับมา ชื่อของฟังก์ชันและการประกาศโปรโตไทป์ได้มีการกำหนดไว้ในร่าง วทท. 2.0 แล้ว แต่ตัวโปรแกรม API ยังไม่มีการพัฒนาและเผยแพร่ ในโครงงานนี้จึงได้พัฒนาโปรแกรมในส่วนนี้โดยอาศัยเทคนิคการเปรียบเทียบและคำสั่ง if ดังนี้

```
int TACchtype(unsigned char ch)
{
   if(BETWEEN(ch,0x00,0x20)||ch==0x7F||ch==0xFF||ch==0x00)
      return(CTRL);
   /*  หมายความว่ารหัสตั้งแต่  01H ถึง 19H รหัส 7FH,รหัส FFH  และรหัส  00H  เป็นรหัสควบคุม
       (CTRL) */
   if(BETWEEN(ch,0x1F,0x7F)||ch==0xA0||ch==0xCF||ch==0xDC||
      ch==0xDF||ch==0xE6||BETWEEN(ch,0xEE,0xFC)||ch==0xDB||
      ch==0xDD||ch==0xDE||BETWEEN(ch,0x7F,0xA0)||BETWEEN(ch,0xFB,0xFF))
      return(NON);
     .              .              .
     .              .              .
     .              .              .
   if(ch==0xD4)
      return(AV1);
   if(ch==0xD6||ch==0xD1)
      return(AV2);
```

```
    if(ch==0xD5||ch==0xD7)
       return(AV3);
    return(-1);
}
```

สำหรับ BETWEEN มีการกำหนดไว้ดังนี้

```
#define BETWEEN(value,low,high) ((value > low) && (value < high))
```

การตรวจสอบลำดับการป้อนข้อความ

ในโปรแกรมประมวลผลคำ การตรวจสอบลำดับการป้อนข้อมูลจะทำหลังจากที่มีการกดแป้นพิมพ์ใด และใช้ตัวอักษรที่กดเป็นตัวตาม และอักษรตัวสุดท้ายที่โปรแกรมกำลังชี้อยู่จะกลายเป็นตัวนำ หากตรวจสอบแล้วว่าถูกต้องตามลำดับการป้อนข้อมูล ก็จะนำข้อมูลไปใส่ในหน่วยความจำ หากพบว่าผิดลำดับ ก็จะเรียกโปรแกรมย่อยที่สำหรับสร้างเสียงเตือน ใช้เสียงเตือนผู้ใช้ถึงการป้อนข้อมูลผิดลำดับ และไม่รับข้อมูลที่พิมพ์ใหม่และรอรับการพิมพ์ข้อมูลตัวต่อไป สำหรับตารางที่ใช้ตรวจสอบลำดับการป้อนข้อมูลข้างต้นนี้ ในโปรแกรมได้ประยุกต์เป็นเมตริกโดยใช้อาร์เรย์สองมิติ โดยที่ตำแหน่งคอลัมน์จะเป็นค่าตัวตามตำแหน่งแถวจะเป็นตัวนำ และจะแทนค่า Accept ด้วยค่า 1 และ Reject ด้วยค่า 0 และเราสามารถกำหนดไม่ให้มีการตรวจสอบลำดับการป้อนข้อมูลก็ได้ โดยเคลียค่าตัวแปร CheckSequence ซึ่งอาจจะกำหนดไว้ในเมนูให้ผู้ใช้เลือกได้

```
int CheckSequence(unsigned char key)
{
   unsigned char Prevchar;          /* ตัวแปรเก็บค่าตัวนำ*/
   register int i,count =0;         /* ตัวแปรสำหรับให้พอยน์เตอร์กลับมาชี้ตำแหน่งเดิม*/
   int flag;                        /* ใช้ตรวจสอบว่าเป็นการพิมพ์ตัวอักษรตัวแรกหรือไม่*/

   /* อ่านตัวอักษรตัวนำ */
   do{
      if((flag=previous())==0)
         break;
      count++;
```

```
    }while(DCurnt->data[I]==WWRAP_CHAR); /* จะต้องข้ามรหัสตัดคำไป*/
    Prevchar = (flag)? DCurnt->data[I]: 0x00;
    /* Restore พอยน์เตอร์ให้กลับมาชี้ตำแหน่งเดิม */
    for( i =0; i <count; i++)
       next();
    return(TACcomposible(key,Prevchar));
}
```

ฟังก์ชัน TACcomposible จะแปลงรหัสของตัวนำและตัวตามให้เป็นเลขบอกประเภทแล้วนำมาตรวจสอบลำดับระหว่างตัวนำและตัวตามโดยใช้ Accept Matrix

```
int TACcomposible(unsigned char CurrChar,unsigned char PrevChar)
{
    int SelfType,FrontType;
    /* ตรวจสอบประเภทของตัวอักษร */
    SelfType = TACchtype(CurrChar);
    FrontType= (PrevChar==0x00)? 2: TACchtype(PrevChar);
    /* ส่งคืนค่าการตรวจสอบโดยอาศัย Accept Matrix*/
    return(AcceptMatrix[FrontType-1][SelfType-1]);
}
```

ตัวอย่างของ Accept Matrix ซึ่งสร้างขึ้นจากข้อกำหนดตามมาตรฐาน วทท. 2.0 โดยคอลัมน์แทนตัวตามและแถวแทนตัวนำ

```
int AcceptMatrix[16][16] ={1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,
                           .       .       .
                           .       .       .
                          1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,
                          1,1,1,1,1,1,0,0,0,1,0,1,0,0,0,0};
```

ตำแหน่งของส่วนตรวจสอบลำดับตัวอักษรจะอยู่ในไฟล์ PRJ6.C ฟังก์ชัน GetKeyboard ในส่วนหลังจากที่รับการกดปุ่มมาแล้วและจะนำไปใส่ไว้ในหน่วยความจำ

```
void GetKeyboard()
{
   while(bioskey(1) ==0)          /* วนลูปรอการกดแป้นพิมพ์*/
   {
   }
   key = bioskey(0);        /* อ่านรหัสแป้นพิมพ์จากคีย์บอร์ดบัฟเฟอร์*/
   switch(key){
      case DOWN:            /* ตรวจสอบเงื่อนไขของปุ่มต่างๆ*/
      case HOME:
      case END:
      case PGUP:
      default:              /* กรณีที่เป็นปุ่มตัวอักษร */
         scancode = (key&0xFF00)>>8; /* อ่านเฉพาะรหัสแสกน*/
         ch = key&0xFF;              /* อ่านค่ารหัสแอสกี */
         ch=Numeric(scancode,ch);    /* ตรวจสอบแป้นพิมพ์ตัวอักษรสำหรับแป้นพิมพ์ 101*/
         if(Sequence&&!CheckSequence(ch))/* ตรวจสอบลำดับการป้อนข้อมูล*/
         {
            Beep();       /*ลำดับการป้อนข้อมูลผิด*/
            break;        /* กลับไปเริ่มที่ Loop while ใหม่*/
         }
         DispChar(ch,CharAttrb);   /* ลำดับการป้อนข้อมูลถูกต้อง*/
         /*แสดงผลตัวอักษรและนำไปใส่ในหน่วยความจำ*/
```

บทที่ 4
การจัดการแป้นพิมพ์

การจัดการแป้นพิมพ์

ตัวอักขระที่ได้จากแป้นพิมพ์แบบ 101 ปุ่มของเครื่องคอมพิวเตอร์มีหลายประเภทแบ่งได้ 3 ประเภท ตามวิธีในการจัดการดังนี้

1. ปุ่มตัวอักขระที่สามารถอ่านค่าได้โดยใช้วิธีปกติและจะมีค่าหนึ่งค่าเป็นตัวแปรแบบ character ขนาดหนึ่งไบต์ สามารถใช้ฟังก์ชั่น getch() ของภาษาซีอ่านค่าได้โดยตรง ปุ่มตัวอักขระในลักษณะนี้มีทั้งหมด 100 อักขระดังต่อไปนี้

` 1 2 3 4 5 6 7 8 9 0 - = \ ~ ! @ # $ % ^ & * ( ) _ + | q w e r t y u i o p [ ] a s d f g h j k l ; ' z x c v b n m , . / Q W E R T Y U I O P { } A S D F G H J K L : " Z X C V B N M < > ? ESC ENTER ,Tab, Space, Backspace

2. ปุ่มตัวอักขระที่เป็นฟังชั่นสามารถอ่านค่าได้ตามปกติโดยค่าที่อ่านได้จะมีขนาด 2 ไบต์ โดยที่ ไบต์แรกจะมีค่าเป็น 00H มี 22 ปุ่ม ดังต่อไปนี้

F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 Insert, Delete, Home, End, Pageup, PageDown, Cursor Left, Cursor Right, Cursor Up, Cursor Down

3. ปุ่มสถานะที่ไม่สามารถอ่านค่าได้โดยวิธีปกติ การอ่านค่าปุ่มเหล่านี้ต้องใช้ฟังก์ชั่น bioskey() อ่านค่าสถานะของแป้นพิมพ์แล้วนำมาตีความหมายซึ่งค่าสถานะที่อ่านได้จะมีขนาด 2 ไบต์ แต่จะใช้เพียงไบต์แรกในการตีความหมาย โดยที่แต่ละบิตของไบต์แรกมีความหมายดังตารางที่ 1

เนื่องจากฟังก์ชั่น bioskey() เป็นฟังก์ชั่นที่เรียกการทำงานอินเทอร์รัพต์ของดอสโดยตรงและสามารถอ่านค่าของปุ่มทั้งหมดบนแป้นพิมพ์ได้

	ค่าเลขฐานสิบหก	ความหมาย
บิตที่ 7	80	Insert on
บิตที่ 6	40	Caps on
บิตที่ 5	20	Numlock on
บิตที่ 4	10	Scroll Lock on
บิตที่ 3	08	Alt pressed
บิตที่ 2	04	Ctrl pressed
บิตที่ 1	02	Left Shift pressed
บิตที่ 0	01	Right Shift pressed

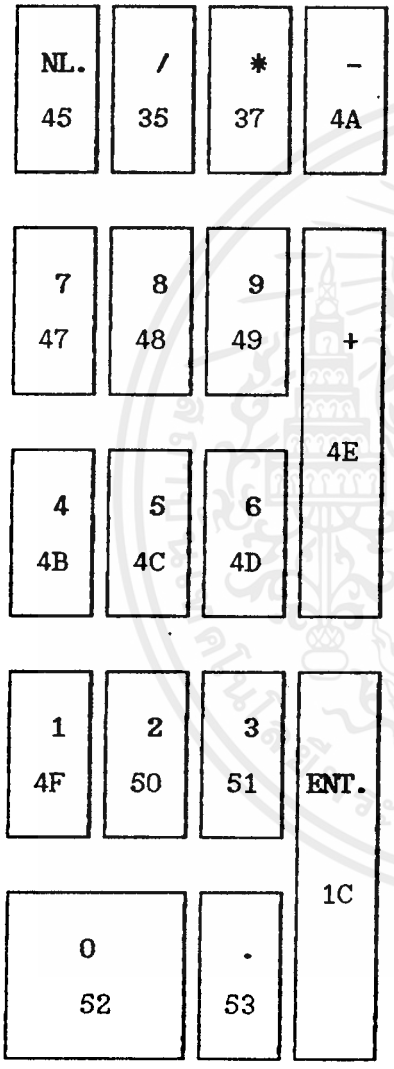
ตารางที่ 1 ความหมายในแต่ละบิตของค่าสถานะของแป้นพิมพ์

Algorithm สำหรับการอ่านการกดแป้นพิมพ์โดยใช้ฟังก์ชั่น bioskey()

```
while(bioskey(1)  ==0)          /*  โปรแกรมวนลูปจนกระทั่งมีการกดปุ่ม
{                                   ประเภทที่ 1 หรือ 2 */
     if(bioskey(2)&ALTPRESSED)  /* ตรวจสอบดูว่ามีการกดปุ่ม  Alt
     {                              หรือไม่*/
     /* สลับแป้นพิมพ์ภาษาไทยและภาษาอังกฤษ */
     }
}
/* เริ่มต้นส่วนจัดการแป้นพิมพ์ประเภทที่หนึ่งหรือประเภทที่สอง */
key = bioskey(0); /* อ่านค่ารหัสจากคีย์บอร์ดบัฟเฟอร์ */
/* นำค่า scan code และรหัส ASCII ไปประมวลผล */
แยกส่วนแสกนโค้ดและรหัสแอสกี
if(ถ้าอยู่ในโหมดภาษาไทย)
     แปลงรหัสภาษาอังกฤษให้เป็นภาษาไทย
```

การจัดการแป้นพิมพ์ตัวอักษร

ในแป้นพิมพ์แบบ 101 ปุ่มในเครื่องคอมพิวเตอร์แบบ AT จะมีแป้นพิมพ์ตัวอักษรอีกชุดหนึ่ง เพื่อความสะดวกสำหรับผู้ใช้ที่มักจะต้องใช้ปุ่มตัวเลขบ่อยๆ ลักษณะการจัดเรียงของแป้นพิมพ์ตัวเลขและค่ารหัส Scan Code เป็นดังรูปที่ 1



รูปที่ 1 แสดงแป้นพิมพ์ตัวเลขพร้อมทั้ง Scan Code

ค่าของแป้นพิมพ์ตัวอักษรสามารถที่จะอ่านได้โดยใช้ฟังก์ชัน getch() ธรรมดาโดยไม่ต้องอาศัยการติดต่อกับไบออสโดยตรงเพื่ออ่านค่า Scan Code ในร่างมาตรฐาน วทท. 2.0 ไม่ได้กล่าวรวมถึงแป้นพิมพ์ตัวเลขเอาไว้ ดังนั้นในโครงงานโปรแกรมประมวลผลคำนี้จะกำหนดหน้าที่ของแป้นพิมพ์ตัวเลขนี้ไว้ดังนี้

1.แป้นพิมพ์ตัวเลขจะมีหน้าที่การทำงานแตกต่างกัน ขึ้นอยู่กับปุ่มสถานะสองปุ่มต่อไปนี้คือ NumLock และปุ่ม Scroll Lock และขึ้นกับว่าในขณะนั้นโปรแกรมเป็นภาษาไทยหรือภาษาอังกฤษ

2.ความหมายของปุ่มแต่ละปุ่มจะเป็นตามตารางที่ 2

แป้นพิมพ์	สถานะ			ความหมาย
	Num Lock	ภาษา	Scroll Lock	
1	ON	อังกฤษ	OFF	1
	ON	ไทย	OFF	๑
	ON	ไทย/อังกฤษ	ON	⌊
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม END
2	ON	อังกฤษ	OFF	2
	ON	ไทย	OFF	๒
	ON	ไทย/อังกฤษ	ON	⌊
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่มลูกศรลง
3	ON	อังกฤษ	OFF	3
	ON	ไทย	OFF	๓
	ON	ไทย/อังกฤษ	ON	⌋
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม PgDown
4	ON	อังกฤษ	OFF	4

แป้นพิมพ์	สถานะ			ความหมาย
	Num Lock	ภาษา	Scroll Lock	
	ON	ไทย	OFF	๔
	ON	ไทย/อังกฤษ	ON	├
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่มลูกศรซ้าย
5	ON	อังกฤษ	OFF	5
	ON	ไทย	OFF	๕
	ON	ไทย/อังกฤษ	ON	│
	OFF	ไทย/อังกฤษ	ON/OFF	๕/5
6	ON	อังกฤษ	OFF	6
	ON	ไทย	OFF	๖
	ON	ไทย/อังกฤษ	ON	┤
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่มลูกศรขวา
7	ON	อังกฤษ	OFF	7
	ON	ไทย	OFF	๗
	ON	ไทย/อังกฤษ	ON	┌
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม HOME
8	ON	อังกฤษ	OFF	8
	ON	ไทย	OFF	๘
	ON	ไทย/อังกฤษ	ON	┬

แป้นพิมพ์	สถานะ			ความหมาย
	Num Lock	ภาษา	Scroll Lock	
9	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่มลูกศรขึ้น
	ON	อังกฤษ	OFF	9
	ON	ไทย	OFF	๙
	ON	ไทย/อังกฤษ	ON	]
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม PgUp
0	ON	อังกฤษ	OFF	0
	ON	ไทย	OFF	๐
	ON	ไทย/อังกฤษ	ON	๐/0
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม Insert
.	ON	อังกฤษ	OFF	.
	ON	ไทย	OFF	.
	ON	ไทย/อังกฤษ	ON	.
	OFF	ไทย/อังกฤษ	ON/OFF	ปุ่ม Delete

ตารางที่ 2 ความหมายของแป้นพิมพ์ตัวเลข

Algorithm การจัดการแป้นพิมพ์ตัวเลข

เนื่องจากแป้นพิมพ์ตัวเลขในการกดปุ่มแต่ละครั้งจะมีความหมายขึ้นกับปสถานะของโปรแกรมในตอนนั้น ดังนั้นในส่วนการรับค่าจากแป้นพิมพ์จึงต้องมีการขยายการทำงานดังต่อไปนี้

```
.    .    .    .
.    .    .    .
switch(key){
   case LEFT:
   case RIGHT:
   case UP:
   case DOWN:
   .    .    .    .
   .    .    .    .
   case K:   /* Block service (CTRL-K *)*/
   default:
      scancode = (key&0xFF00)>>8;
      ch = key&0xFF;
      /* เพิ่มส่วนตรวจสอบสำหรับแป้นพิมพ์ตัวอักษร */
      ch=Numeric(scancode,ch);
      if(Sequence&&!CheckSequence(ch))
      .    .    .    .    .
      .    .    .    .    .
```

ในส่วนการตรวจสอบแป้นพิมพ์ตัวเลขของฟังก์ชัน Numericเป็นดังนี้

```
unsigned char Numeric(unsigned char ScanCode,unsigned char ch)
{
      int status;
```

```
        status = bioskey(2);     /* อ่านค่าสถานะของแป้นพิมพ์ */
        switch(ScanCode) {
            case 0x47:           /* ปุ่มเลข 7 */
                if(status&SCROLLLOCKON) /* ตรวจสอบสถานะ Scroll */
                    ch =0x98;           /* เป็นเส้นกรอบตาราง */
                else{
                    if((CharAttrb&THAI)!=0)  /* ตรวจสอบภาษา */
                        ch =0xF7;     /* เป็นเลข 7 ไทย */
                    else
                        ch =0x37;     /* เป็นเลข 7 อารบิก */
                }
            break;
        case 0x48:
            if(status&SCROLLLOCKON)
        .    .    .    .
        .    .    .    .
 return(ch);
}
```

บทที่ 5
การพิมพ์เอกสารภาษาไทยและอังกฤษ

ส่วนการจัดพิมพ์เอกสารเป็นอีกส่วนหนึ่งที่สำคัญของโปรแกรมประมวลผลคำ เนื่องจากหากเอกสารที่สร้างขึ้นมาไม่สามารถพิมพ์ได้ก็ไม่มีประโยชน์อย่างใด ข้อกำหนดทางการพิมพ์ภาษาไทยในปัจจุบันนี้ยังไม่มีการยอมรับมาตรฐานอย่างใดอย่างหนึ่งแน่นอน แม้ว่าจะมีประกาศของ สมอ.ออกมานานแล้วก็ตาม เครื่องพิมพ์ของบริษัทต่างๆก็ใช้รหัสภาษาไทยต่างกัน ใช้เทคนิคการพิมพ์ต่างกัน ทำให้การเขียนโปรแกรมสำหรับเครื่องพิมพ์ทุกแบบทำได้ลำบาก

การพิมพ์ข้อความบนเครื่องพิมพ์ที่มีรหัสภาษาไทยหลายรหัส

ส่วนจัดการพิมพ์เอกสารได้พัฒนาบนเครื่องพิมพ์ Epson LX-800 ใช้รหัสภาษาไทยคือรหัสเกษตรหรือชื่อที่อ้างใน วทท. 2.0 คือ TAPIC Code ID 42 เนื่องจากในร่างมาตรฐาน วทท. ไม่มีการระบุชื่อของรหัสต่างๆ ดังนั้นจึงไม่มีชื่อของรหัสเกษตรหรือรหัส สมอ. หรือชื่อใดๆอยู่ ทั้งนี้ก็เนื่องมาจากก่อนที่จะมีร่างมาตรฐาน วทท. ออกมาเกี่ยวกับรหัสภาษาไทยบนเครื่องพิมพ์ มีรหัสมากมายหลายแบบบนเครื่องพิมพ์แบบต่างๆ แม้แต่รหัสเกษตรก็ยังมีเกษตรหลายแบบ ไม่สามารถใช้ได้อย่างครบถ้วนสมบูรณ์ระหว่างเครื่องพิมพ์ต่างยี่ห้อกัน การจะเรียกเป็นรหัสเกษตรก็ยังคงมีความสับสนอยู่ จึงได้มีการกำหนดชื่อของรหัสต่างๆเป็นรหัสไว้โดยำรหัสที่นิยมใช้กันอยู่มาจดทะเบียนไว้ การจดทะเบียนรหัสหลายๆ รูปแบบไว้ทำให้มาตรฐาน วทท. 2.0 นี้สามารถพิมพ์ได้กับเครื่องพิมพ์ที่ผู้ผลิตกำหนดรหัสภาษาไทยต่างกันได้ โดยอาศัยการใช้ตารางแปลงรหัสที่จดทะเบียนไว้ เมื่อรหัสภาษาไทยที่ วทท. กำหนดเป็นที่นิยมมากขึ้นแล้ว น่าจะทำให้รหัสภาษาไทยของเครื่องพิมพ์ทั้งหมดเหมือนกันเพื่อที่ผู้พัฒนาโปรแกรมจะได้ไม่ต้องทำการแปลงรหัสต่างๆอีก ร่างมาตรฐาน วทท. ได้กำหนดส่วนที่เกี่ยวกับการพิมพ์เอกสารบนเครื่องพิมพ์ 3 ส่วนคือ

1. Printer ID
2. Code ID
3. Conversion Table (ตารางแปลงรหัส)

Printer ID และ Code ID

Printer ID หรือ รหัสของเครื่องพิมพ์เป็นการกำหนดชื่อเรียกเครื่องพิมพ์รุ่นต่างๆ และยี่ห้อต่างๆ เพื่อใช้เป็นมาตรฐานในการอ้างถึงเครื่องพิมพ์เครื่องนั้นๆ ส่วนของ Printer ID จะประกอบด้วยข้อมูล 3 ส่วนดังนี้

ชื่อยี่ห้อ	ชื่อรุ่น	รหัสของชุดอักขระที่ใช้

ในเครื่องพิมพ์ที่ใช้ทำการทดสอบคือเครื่องพิมพ์ Epson LX-800 มีรหัสเครื่องพิมพ์ตามที่ได้จดทะเบียนไว้เป็น EPL3 สำหรับส่วนของรหัสชุดอักขระที่ใช้ไม่ได้มีแจ้งไว้ ดังนั้นจึงได้เขียนโปรแกรมขึ้นมาโปรแกรมหนึ่งสำหรับใช้ตรวจสอบรหัสของเครื่องพิมพ์รุ่นต่างๆ จากผลการทดสอบพบว่าเครื่องพิมพ์ Epson LX-800 ใช้รหัส TAPIC Code ID 42 ตามที่จดทะเบียนไว้ใน วทท. ดังนั้นชื่อรหัสเต็มของเครื่องพิมพ์ LX-800 คือ **EPL342**

ส่วน Code ID หรือ TAPIC Code ID จะเป็นส่วนที่ให้ผู้ผลิตหรือผู้จัดจำหน่ายเครื่องพิมพ์มาลงทะเบียนรหัสอักขระของเครื่องพิมพ์ที่ใช้ โดยที่ค่าของเลขรหัสจะมีค่าสองไบต์ มีค่าเป็นเลขฐานสิบหก

Conversion Table (ตารางแปลงรหัส)

ในการพิมพ์ข้อความภาษาไทย ขั้นตอนแรกที่จะต้องทำคือการแปลงรหัสของตัวอักษรให้ตรงกับรหัสของเครื่องพิมพ์เพื่อให้สามารถพิมพ์ข้อความได้อย่างถูกต้อง การแปลงรหัสนี้จากแปลงเพียงตัวอักขระเพียงบางตัวหรืออาจต้องแปลงรหัสทั้งหมด ก่อนที่จะมีการกำหนดร่างมาตรฐาน วทท. เมื่อมีการพัฒนาโปรแกรมประมวลผลคำขึ้นมาแต่ละครั้งผู้พัฒนาโปรแกรมก็วางรูปแบบและวิธีการแปลงรหัสต่างๆเอง ซึ่งในบางครั้งเมื่อมีเครื่องพิมพ์ลักษณะใหม่ๆ หรือรุ่นใหม่ โปรแกรมที่เขียนไว้แล้วก็ไม่สามารถทำงานได้ เมื่อมีการกำหนดร่างมาตรฐาน วทท. ขึ้นมาจึงได้กำหนดมาตรฐานสำหรับตารางแปลงรหัสด้วย โดยได้กำหนดส่วนบอกลักษณะของรหัสเข้าไปในตารางด้วย ดังนี้

1. รหัสมีการใช้รหัสผสม เช่น รวมไม้เอกกับสระอี เข้าด้วยกันเป็นหนึ่งตัวอักษร ใช้รหัสว่า CC

2.รหัสมีการใช้รหัสตารางตามมาตรฐาน มอก. 988-2533 ใช้รหัสว่า LG

3.มีการใช้รหัสผสมของสระอำกับวรรณยุกต์ ใช้รหัสว่า UM

4.รหัสของสระโอ สระไอ และ สระใอ ใช้การต่อหัว ใช้รหัสว่า EX

5.มีตัวเลขไทย ใช้รหัสว่า TN

6.มีอักขระพิเศษอย่างอื่นที่นอกจากมาตรฐาน ใช้รหัสว่า SP

7.รหัสที่ไม่สามารถแปลงได้จะใช้รหัส 3F ('?') แทน

```
CC -- UM -- TN SP      ! TAC11X13.COD 16 AUG 1991
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
81 82 83 84 8B 8C 8D 8E 8F 90 91 92 93 95 96 97
98 9A 9B 9C 9D 86 87 88 89 3F 3F 3F 3F 3F 3F 3F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA 3F 3F 3F 3F 3F
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB 3F 3F 3F 3F
```

รูปที่ 1 ตัวอย่างไฟล์แปลงรหัสภาษาไทย

รูปที่ 1 เป็นตัวอย่างตารางแปลงรหัสสำหรับแปลงจาก TAPIC Code ID 11 เป็น TAPIC Code ID 13 (ส่วนขยายของชื่อตารางแปลงรหัสจะเป็น .COD) สำหรับค่าในบรรทัดแรกจะเป็นค่าคุณสมบัติของชุดตัวอักษรตามที่ได้กล่าวมาแล้วข้างต้น เครื่องพิมพ์ Epson LX-800 มีรหัสภาษาไทยคือ TAPIC Code ID 42 ดังนั้นตารางแปลงรหัสที่ใช้คือตารางชื่อ TAC11X42.COD ฟังก์ชันสำหรับการอ่านตารางแปลงรหัสจากไฟล์นี้จะอยู่ในไฟล์ THAI.C ฟังก์ชัน LoadFont(); ดังนี้

```
if((fp=fopen("TAC11X42.COD","r"))==NULL){}
do{
   n=getc(fp);          /* อ่านจนกระทั่งข้ามบรรทัดแรก */
}while(n!='\n');
for( i =0; i <256; i++)     /* อ่านข้อมูลจากไฟล์ทั้งหมด 256 ตัวอักษร */
{
   for( j =0; j <2; j++)
      code[j] = getc(fp);
   getc(fp); /*อ่านข้อมูลเป็นตัวอักษรมาแล้วแปลงเป็นเลขฐานสิบหก*/
   CONVERT[i] =(unsigned char)atox(code);
}
```

สำหรับเครื่องพิมพ์ที่ได้มีการจดทะเบียนรหัสไว้แล้วสามารถนำข้อมูลในส่วนนี้มาใช้ได้ทันที แต่สำหรับเครื่องพิมพ์ที่มีจำหน่ายโดยทั่วไปผู้ขายมักจะไม่แจ้งถึงรหัสของเครื่องพิมพ์ตามมาตรฐานให้ทราบดังนั้น จึงจำเป็นที่จะต้องทดสอบเองสำหรับเครื่องพิมพ์รุ่นอื่นที่จะนำมาทดสอบ และมักจะพบปัญหาที่ว่ารหัสภาษาไทยของเครื่องพิมพ์ที่นำมาทดสอบมีรหัสที่ไม่ตรงกับที่ได้ลงทะเบียนไว้ในมาตรฐาน วทท. 2.0 เช่นเครื่องพิมพ์ Star NX-1001

ขั้นตอนการพิมพ์เอกสาร

การพิมพ์เอกสารที่สร้างขึ้นจากโปรแกรมประมวลผลคำนั้นมีขั้นตอนการจัดพิมพ์เอกสารดังนี้คือ

1.นำข้อมูลที่จะทำการพิมพ์มาทำการตรวจสอบระดับการพิมพ์ เนื่องจากว่าการพิมพ์นั้นจะประกอบกับ 3 ระดับคือ ระดับกลาง เช่น พยัญชนะ ระดับ บน เช่น วรรณยุกต์ ระดับล่าง เช่น สระอุ ในการตรวจสอบข้อ

มูลนั้นจะทำการอ่านข้อมูลมาครั้งละ 1 บรรทัดโดยอาศัยรหัสตัดคำช่วยในการแยกบรรทัด และแยกข้อมูลตามระดับการพิมพ์ต่างๆ ซึ่งจะรวมส่วนการควบคุมแบบตัวอักษร เช่นตัวหนาหรือตัวเอียง ด้วย และยังต้องมีการรวมและแยกสระหรือวรรณยุกต์บางตัวด้วยเนื่องจากเครื่องพิมพ์บางรุ่นมีการกำหนดรหัสผสมสำหรับอักขระบางคู่โดยเฉพาะเช่น ไม้หันอากาศกับไม้เอก เป็นต้น เมื่อแยกระดับข้อมูลได้แล้วจึงส่งไปสู่ส่วนการพิมพ์ข้อมูลซึ่งจะพิมพ์ครั้งละหนึ่งบรรทัดสามระดับจนครบ

2.การพิมพ์ข้อมูลจะมี สองลักษณะคือ พิมพ์โดยใช้แบบตัวอักษรของเครื่องพิมพ์ กับการพิมพ์โดยการสร้างแบบตัวอักษรเองโดยโปรแกรมและสั่งพิมพ์โดยโปรแกรม การเลือกวิธีการพิมพ์ทั้งสองแบบนี้มีข้อดี ข้อเสียคือ

การพิมพ์โดยใช้แบบตัวอักษรของเครื่องพิมพ์

ข้อดี

1.ความเร็วในการพิมพ์จะสูงกว่า

2.ถ้าเครื่องพิมพ์มีแบบตัวอักษรที่สวย และมีให้เลือก ก็สามารถพิมพ์ตัวอักษรได้สวยกว่า

ข้อเสีย

1.เครื่องพิมพ์นั้นจะต้องมีการสร้างแบบตัวอักษรภาษาไทยไว้แล้ว

การพิมพ์โดยใช้แบบตัวอักษรของโปรแกรม

ข้อดี

1.เครื่องพิมพ์ไม่จำเป็นต้องมีแบบตัวอักษรภาษาไทย

2.ถ้าแบบตัวอักษรที่สร้างในเครื่องพิมพ์มีน้อย จะทำให้สามารถพิมพ์ได้มากแบบกว่า

ข้อเสีย

1.การทำงานจะช้ากว่าวิธีแรก

ในโครงงานนี้ใช้วิธีการพิมพ์ในโหมดตัวอักษร การแก้ไขให้สามารถทำการพิมพ์ในโหมดกราฟิกทำได้โดยแก้ไขส่วนล่างสุดที่สั่งพิมพ์ ในการพิมพ์จะสั่งให้พิมพ์ระดับบนสุดก่อนแล้วจึงขึ้นบรรทัดใหม่และสั่งให้พิมพ์ระดับที่สองคือระดับของพยัญชนะแล้วจึงพิมพ์ระดับของรหัสล่างเป็นชุดสุดท้าย

ชุดคำสั่งที่ใช้ในการกำหนดค่าต่างๆของเครื่องพิมพ์

เครื่องพิมพ์แบบดอตเมทริกซ์โดยทั่วไปเราสามารถสั่งให้เครื่องพิมพ์ พิมพ์ตัวอักขระที่เราต้องการได้โดยส่งรหัสของตัวอักขระนั้นๆ ไปให้เครื่องพิมพ์ แต่ในการจัดพิมพ์เอกสาร เราจำเป็นที่จะต้องมีการจัดหน้า

ทั้งกั้นซ้าย จำนวนบรรทัดต่อหน้า เพื่อให้เอกสารดูสวยงาม นอกจากนี้สำหรับการพิมพ์ตัวอักษรแบบหลายๆแบบตัวอักษร เราจำเป็นที่จะต้องส่งรหัสสำหรับเลือกแบบตัวอักษรให้กับเครื่องพิมพ์ด้วย ชุดคำสั่งที่ใช้ในโครงงานนี้เป็นชุดคำสั่ง ESC ซึ่งเป็นคำสั่งควบคุมเครื่องพิมพ์ที่ค่อนข้างเป็นมาตรฐาน ในโครงงานนี้ใช้คำสั่งบางส่วนที่เกี่ยวกับการควบคุมเครื่องพิมพ์ดังนี้

รหัส ASCII	เลขฐานสิบ	ความหมาย
ESC @	64	เริ่มต้นเครื่องพิมพ์ใหม่
CR	13	ขึ้นบรรทัดใหม่
FF	12	ขึ้นหน้าใหม่
LF	10	เลื่อนบรรทัด
ESC A n	65	ตั้งการเว้นบรรทัดขนาด n/72 นิ้ว
ESC l n	108	ตั้งจุดเริ่มต้นขอบกระดาษด้านซ้ายที่ตำแหน่ง n
ESC Q n	81	ตั้งจุดจบขอบด้านขวาที่ตำแหน่ง n
ESC x n	120	เลือกแบบการพิมพ์ 0 - draft 1 - NLQ
ESC ! n	33	เลือกลักษณะการพิมพ์ ค่าของ n จะได้จากการนำค่าต่อไปนี้มาบวกกัน pica: 0 12 cpi: 1 ตัวบีบ: 4 ตัวเน้น: 8 ตัวพิมพ์ 2 ครั้ง: 16 ตัวกว้างสองเท่า: 32 ตัวเอน: 64 ตัวขีดเส้นใต้: 128

รหัส ASCII	เลขฐานสิบ	ความหมาย
DC4	20	ยกเลิกโหมดความกว้างสองเท่า
ESC F	70	ยกเลิกโหมดตัวเน้น
ESC H	72	ยกเลิกโหมดตัวพิมพ์สองครั้ง
ESC T	84	ยกเลิกโหมดตัวบน/ตัวล่าง
ESC - 1/0	45	เลือกหรือไม่เลือกโหมดขีดเส้นใต้
ESC 5	53	ยกเลิกโหมดตัวเอน

ตารางที่ 1 คำสั่ง ESC ที่ใช้ในโปรแกรมประมวลผลคำ

การตรวจสอบข้อผิดพลาดของเครื่องพิมพ์

การส่งข้อมูลออกไปทางเครื่องพิมพ์ใช้ฟังก์ชัน biosprint() ซึ่งฟังก์ชันนี้จะส่งค่ากลับทุกครั้งหลังจากใช้คำสั่งให้ส่งข้อมูลไปยังเครื่องพิมพ์แล้วค่าที่ส่งคืนมาจะมีความหมายในแต่ละบิตดังนี้

บิตที่ 0 Device time out
บิตที่ 3 I/O error
บิตที่ 4 Selected
บิตที่ 5 Out of paper
บิตที่ 6 Acknowledge
บิตที่ 7 Not busy

สำหรับเครื่องพิมพ์ LX-800 ค่าแต่ละบิตที่ส่งกลับมาเมื่อเกิดข้อผิดพลาดต่างๆ มีดังนี้

ข้อผิดพลาด	ค่าที่ส่งกลับ
ไม่เปิดเครื่องพิมพ์	I/O error,Acknowledge,Not busy
ไม่เปิด On-line	I/O error,Selected
กระดาษหมด	I/O error,Selected,Out of papere

ตารางที่ 2 แสดงบิตที่เช็ตกับข้อผิดพลาดที่เกิดขึ้นของเครื่องพิมพ์

การตรวจสอบข้อผิดพลาดอย่างง่ายๆก็คือตรวจสอบว่ามี I/O error หรือไม่ ถ้ามีก็แจ้งให้ผู้ใช้ทราบ สาเหตุที่ไม่ใช้ค่าที่ส่งกลับทั้งหมดเพราะว่าสำหรับเหตุการณ์เดียวกัน ในเครื่องพิมพ์คนละแบบก็จะให้ค่าที่ส่งกลับต่างกัน

Algorithm ในการพิมพ์เอกสาร

```
คำนวนระยะห่างระหว่างบรรทัดจากจำนวนบรรทัดต่อหน้ากับความยาวกระดาษ
สั่งให้เครื่องพิมพ์อยู่ในสภาวะเริ่มต้น;
กำหนดให้พอยน์เตอร์ของข้อมูลชี้ที่ต้นข้อมูล
เลือกโหมดการพิมพ์ NLQ/draft
กำหนดค่ากั้นขวา
กำหนดค่าเริ่มต้นของบัฟเฟอร์ทั้งสามระดับเป็น 0
do{
    if(ขึ้น paragraph ใหม่)
        กำหนดค่ากั้นซ้าย
    if(อ่านตัวอักษรจนจบบรรทัด)
    {
        สั่งพิมพ์ข้อความหนึ่งบรรทัดสามระดับ
        สั่งให้เลื่อนบรรทัดหนึ่งบรรทัด
```

```
        กำหนดค่าเริ่มต้นของบัฟเฟอร์ทั้งสามระดับเป็น 0
        continue;
    }
    อ่านค่าแอทริบิวต์ของตัวอักษรเขียนลงในบัฟเฟอร์
    อานข้อมูลตำแหน่งปัจจุบัน
    ถ้าข้อมูลตำแหน่งปัจจุบันแสดงผลระดับ TOP หรือ ABOVE
    {
        ใส่บัฟเฟอร์ที่หนึ่ง
        ถ้าหากว่าตำแหน่งนี้มีข้อมูลอยู่แล้ว เปิดตารางเพื่อหารหัสของตัวอักษรผสม
    }
    ถ้าข้อมูลตำแหน่งปัจจุบันแสดงผลระดับ BASE ใส่บัฟเฟอร์ที่สอง
    ถ้าข้อมูลตำแหน่งปัจจุบันแสดงผลระดับ BELOW ใส่บัฟเฟอร์ที่สาม
    เลื่อนพอยน์เตอร์ของบัฟเฟอร์ไปยังตำแหน่งต่อไป
}while(ยังสามารถเลื่อนไปยังข้อมูลตำแหน่งต่อไป);
พิมพ์ข้อความบรรทัดที่ยังเหลือค้างในบัฟเฟอร์
ส่งคำสั่ง Form Feed;
}
```

บทที่ 6
การตัดคำภาษาไทย

ในการเขียนภาษาไทย แต่ละคำจะติดกันโดยไม่มีช่องว่าง ดังนั้นขอบเขตของคำจึงไม่สามารถรู้ได้ ซึ่งทำให้เกิดปัญหาในการใช้คอมพิวเตอร์เพื่อประมวลภาษาไทยกรณีที่มีประโยคซึ่งยาวกว่าบรรทัด และเช่นเดียวกันถ้าไม่สามารถแยกแต่ละคำได้ ก็จะสามารถใช้คอมพิวเตอร์เพื่อจะวิเคราะห์คำได้ วิธีแก้อย่างตรงไปตรงมาคือให้มนุษย์ใส่ช่องว่างเอาเอง ซึ่งก็มีโอกาศที่จะทำผิดพลาดได้ และก็ต้องสิ้นเปลืองเวลาทำให้ไม่สามารถใช้ข้อมูลได้ทันที ในปี พ.ศ.2522 Lorchirachoonkul ได้พัฒนาอัลกอริทึมในการแยกคำภาษาไทย โดยแยกเป็นกลุ่มตัวอักษรที่น้อยที่สุดที่สามารถอ่านออกได้ และไม่ได้ใช้คุณสมบัติลักษณะของภาษาไทย อย่างไรก็ตามการศึกษาของเขาได้เก็บเป็นความลับ ไม่เปิดเผยข้อมูลอื่นๆ

และยังมีกลุ่มวิจัยกลุ่มอื่นๆได้ทำการศึกษาการตัดคำภาษาไทยนี้ แต่ก็ไม่ได้รับการเปิดเผย ยกเว้นของ Y. Thairatananon ซึ่งได้ทำการวิจัยเพื่อเป็นวิทยานิพนธ์ระดับปริญญาโท สาขาวิทยาการคอมพิวเตอร์ ที่สถาบันเทคโนโลยีแห่งเอเชีย ซึ่งเป็นงานวิจัยชิ้นเดียวที่เผยแพร่สู่สาธารณะ มีจุดมุ่งหมายเพื่อให้การตัดคำภาษาไทยเป็นไปอย่างมีประสิทธิภาพและถูกต้อง

วิธีการตัดคำ

วิธีการตัดคำภาษาอังกฤษสามารถนำมาใช้ได้ในการตัดคำภาษาไทย มีอยู่ 2 วิธีในการตัดคำภาษาไทยโดยคอมพิวเตอร์

วิธีแรกคือใช้การตัดคำตามพจนานุกรม โดยในพจนานุกรมจะเก็บจุดที่สามารถตัดได้ของคำไว้ โปรแกรมจะตรวจคำไปเปรียบเทียบกับคำในพจนานุกรมซึ่งมีอยู่มากกว่า 100,000 คำ และผู้ใช้สามารถเพิ่มเติมคำลงในพจนานุกรมได้ วิธีนี้เรียกว่า SCRIPSIT Spelling and Hyphenation Dictionary

อีกวิธีคือการใช้กฎตัดตามลักษณะของคำ เรียกว่า hyphenation algorithm ซึ่งจะมีการทำงานน้อยกว่าวิธีแรก มีรายละเอียดดังนี้

- ตรวจดูว้าเป็นคำยกเว้นในพจนานุกรม โดยจะมีพจนานุกรมซึ่งเก็บเฉพาะคำยกเว้นไว้
- ถ้าไม่ปรากฏในพจนานุกรม ก็จะใช้อัลกอริทึมพิเศษเพื่อหาจุดตัดที่ดีที่สุด

ประเภทของอักษรไทย

Thairatananond (1981) ได้จัดประเภทตัวอักษรของภาษาไทยออกเป็น 5 กลุ่มและกลุ่มย่อยๆของแต่ละกลุ่มอีก ดังนี้

1. ก่ลุ่มพยัญชนะ แบ่งเป็น 5 กลุ่มย่อยคือ
   - เป็นอักษรนำเสมอ
   - มีโอกาสเป็นอักษรนำมาก
   - เป็นตัวสะกด
   - เป็นพยัญชนะพิเศษ
   - อื่นๆ
2. กลุ่มสระ แบ่งเป็น 4 กลุ่มย่อยคือ
   - สระที่ไม่ต้องการตัวสะกด
   - เป็นสระนำเสมอ
   - มีโอกาสที่จะมีตัวสะกดมาก
   - อาจจะมีหรือไม่มีตัวสะกดก็ได้
3. กลุ่มวรรณยุกต์
4. กลุ่มตัวเลข
5. กลุ่มพิเศษ เป็นกลุ่มตัวอักษรพิเศษที่มักจะอยู่ต้นคำหรือท้ายคำ

วิธีการตัดคำนั้นจะพิจารณาตัวอักษรที่อยู่ ณ ตำแหน่งกั้นขวาก่อน โดยถ้าเป็นตัวอักษรที่อยู่ในกลุ่มที่เป็นกฎที่แน่นอน เช่น กลุ่มย่อยที่ 1 ของกลุ่มพยัญชนะ หรือ กลุ่มย่อยที่ 1 และ 2 ของกลุ่มสระ หรือ กลุ่มพิเศษ ก็จะสามารถระบุจุดตัดได้ทันที ส่วนในกรณีที่อักษร ณ ตำแหน่งกั้นขวาไม่อยู่ในกลุ่มดังกล่าวข้างต้น จะยังไม่สามารถระบุจุดตัดได้ จะต้องใช้ backward checking เพื่อหาสระที่ใกล้ที่สุดทางซ้ายของกั้นขวาแล้วจะพิจารณาตามจุดที่จะตัดตามคุณสมบัติของสระนั้น

วิธีการตัดคำในลักษณะนี้ได้ผลเป็นที่น่าพอใจ แต่ยังมีข้อบกพร่องอยู่บ้างดังนี้

- กฎของตัวอักษรยังไม่ครอบคลุมได้ครบถ้วน
- ยังต้องแก้ไขในส่วนของคำที่มีการยกเว้นในแต่ละกฎอีก
- ควรจะสามารถให้ผู้ใช้เพิ่มเติมกฎของคำยกเว้นบางคำได้ด้วยตัวเอง

ตัวอย่างการตัดคำบางคำ

ตัดผิดตำแหน่ง	ตัดถูกตำแหน่ง
โ*ต ้ะ	*โต๊ะ
จะต*้อง	จะ*ต้อง
อ*ันม ี	อ ัน*ม ี

- จากการเขียนโปรแกรมตามอัลกอริทึม พบว่ายังไม่สามารถตอบสนองกฎ และข้อยกเว้นต่างๆได้ถูกต้องสมบูรณ์ทั้งหมด ยังมีตัดผิดอยู่บ้างบางคำ
- เนื่องจากว่าในการตัดคำ การพิจารณาตัดตามสระและลักษณะอักษรให้มีการ FAIL ได้ ดังนั้นในบางกรณี การตัดคำจะเลื่อนไปตัดที่คำทางซ้ายแทนถ้าสามารถหาจุดที่แน่ใจได้

คุณสมบัติของอักษรไทย

พยัญชนะ

ในปัจจุบัน อักษรไทยที่มีใช้อยู่จริงมีอยู่ 42 ตัวและสามารถทำหน้าที่ได้แตกต่างกันออกไปตามลักษณะการใช้ เช่น เป็นอักษรนำ, เป็นตัวสะกด, เป็นสระ, เป็นส่วนหนึ่งของสระ เป็นต้น

จากการศึกษาสามารถแบ่งพยัญชนะออกได้เป็น 5 กลุ่มย่อยโดยแบ่งตามการใช้งานปกติได้ดังนี้

- ฉ ผ ฝ ฮ พยัญชนะเหล่านี้จะปรากฏเป็นอักษรนำเสมอ
- ห ภ ม ฟ ช ท พยัญชนะเหล่านี้โดยมากจะใช้เป็นอักษรนำ
- อ ว ร พยัญชนะเหล่านี้สามารถใช้เป็นสระได้ และถ้า ร จะทำตัวเป็นสระแล้วจะอยู่ในรูปของ รร
- ศ ณ ญ ษ ฐ ฎ ฎ ฒ ฬ ฌ พยัญชนะเหล่านี้โดยมากจะปรากฏเป็นตัวสะกด ยกเว้นในบางคำ
- ก ข ค ง จ ช ด พยัญชนะเหล่านี้สามารถเป็นได้ทั้งอักษรนำและทั้งตัวสะกด

สระ

มีสระทั้งหมด 17 ตัวที่สามารถจัดประเภทได้ดังนี้

- เ แ ไ โ ใ สระเหล่านี้ปกติแล้วจะปรากฏอยู่หน้าพยัญชนะ

- ะ ำ สระเหล่านี้ก่อเสียงแล้วจะปรากฏเป็นอักษรตัวสุดท้ายของพยางค์ และไม่จำเป็นต้องมีตัวสะกด
- ั ิ ี ึ ื สระเหล่านี้จะต้องมีตัวสะกดเสมอ
- า ็ ุ ู สระเหล่านี้สามารถปรากฏทั้งมีตัวสะกดหรือไม่ก็ได้
- ฤ สระนี้มีคุณสมบัติพิเศษ

วรรณยุกต์

มีอักษรอยู่เพียง 4 ตัวเท่านั้น และสามารถจะอยู่ ณ ตำแหน่งใดๆของพยางค์ก็ได้ จึงไม่จำเป็นต้องแยกกลุ่มของอักษรเหล่านี้ อักษร 4 ตัวนี้ได้แก่ ่ ้ ๊ ๋ และมีคุณสมบัติที่น่าสนใจดังนี้

- จะอยู่หลังสระ ั ิ ี ึ ื ุ ู เสมอ
- จะอยู่ก่อนสระ ะ า ำ อ ว
- สระ เ แ ไ โ ใ จะต้องตามด้วยพยัญชนะอย่างน้อย 1 ตัวเสมอ และถึงจะตามด้วยวรรณยุกต์
- สระ ฤ ็ และสัญญลักษณ์ ์ ไม่เกิดขึ้นพร้อมกับวรรณยุกต์

สัญญลักษณ์

มีสัญญลักษณ์ 14 ตัวที่ใช้ในการพิมพ์ภาษาไทย คือ

์ ๆ ฯ ( ) _ " ! ? : ; - , .

ตัวเลข

มีคุณสมบัติที่น่าสนใจคือ

- ในกลุ่มตัวเลข สามารถมี , (comma) และ . (full stop) ได้ แต่ต้องไม่มีช่องว่าง
- สามารถแยกคำที่มีตัวเลขประกอบอยู่ออกได้เช่น

  ปตท. , ก.ม.5

วิเคราะห์การตัดคำ

(ก) ลักษณะการใช้ของตัวอักษรที่พิเศษ

แบ่งได้เป็น 3 กลุ่มคือ

(1) rigid case (กรณีที่แน่นอน)

เช่น สระ ◌ิ จะไม่ปรากฏในรูปแบบ c◌ิ และ c◌ิss ดังนั้นสามารถจะสร้างกฎได้โดยง่ายดังนี้ "สระ ◌ิ ต้องมีตัวสะกดเพียงหนึ่งตัว และไม่ขึ้นอยู่กับวรรณยุกต์"

(2) ambiguity case (กรณีที่คลุมเคลือ)

เช่น สระ ◌ิ สามารถปรากฏได้ทั้งในรูปแบบ c◌ิ , c◌ิs , c◌ิss จึงไม่สามารถจะตัดสินตำแหน่งตัดคำได้โดยไม่พิจารณาอักษรข้างเคียง

(3) exceptional case (กรณียกเว้น)

ตัวอย่างคำยกเว้น

สาส์น ธาตุ ชาติ มาตร มหรสพ

(ข) พิจารณาขอบเขต

- พิจารณาขอบเขตของคำโดยไม่สนใจคำที่คลุมเคลือ
- แบ่งขอบเขตออกเป็นขอบหน้าและขอบหลัง
- ดูรูปแบบของขอบเขตที่แตกต่างไปจากกฎที่เก็บไว้ โดยขอบเขตหน้าและขอบเขตหลังจะแยกเก็บในแต่ละรูปแบบ
- พิจารณาคำยกเว้นของรูปแบบนั้นๆ
- ดูว่าเป็นคำควบกล้ำหรือไม่ โดยดูจากอักษรควบของสระแต่ละตัว

สรุป

1. การพิจารณาขอบเขตโดยมากจากแน่นอน
2. โดยมากในการหาตำแหน่งการตัดต้องอาศัยอักษรข้างเคียงมาพิจารณาด้วย
3. การหาขอบเขตหน้าสามารถหาได้จากการพิจารณารูปแบบคำควบกล้ำของคำนั้นๆ
4. คำยกเว้นโดยมากจะสามารถหาจุดตัดได้จากการหาขอบหลัง

กฎการตัดคำ

(1) กฎขอบหน้า

กฎที่ 1

สระ ◌ั ◌ิ ◌ี ◌ึ ◌ุ ◌ู ◌็ ◌ะ ◌ำ ◌ั◌ และ วรรณยุกต์ ◌่ ◌้ ◌๊ ◌๋ ต้องมีพยัญชนะนำอย่างน้อยหนึ่งตัวเสมอ

กฎที่ 2

โดยปกติ สระ เ แ ไ โ จะนำหน้าอักษรอื่นเสมอ ยกเว้นคำยกเว้นบางคำที่จะมีพยัญชนะนำหน้าสระข้างต้น

กฎที่ 3

สระ ใ จะนำหน้าอักษรอื่นเสมอโดยไม่มีข้อยกเว้น

กฎที่ 4

พยัญชนะ ฉ ผ ฝ ฮ จะนำหน้าอักษรอื่นเสมอ ยกเว้นจะมี เ แ ไ โ ใ นำหน้า

กฎที่ 5

มีคำ 9 คำในภาษาไทยที่มีสระ ฤ เป็นพยัญชนะตัวแรกของคำ นอกจากนี้แล้ว ฤ จะต้องมีพยัญชนะหนึ่งตัวนำเสมอ ยกเว้นคำว่า อมฤต

กฎที่ 6

พยัญชนะ ห โดยมากจะเป็นพยัญชนะนำ ยกเว้นในบางคำ

(2) กฎขอบหลัง

กฎที่ 1

พยัญชนะ ศ ณ ญ ษ ฐ ฏ ฎ ฒ ฬ ฌ โดยมากจะเป็นตัวสะกด ยกเว้นที่ปรากฏเป็นตัวควบ เช่น ศก ศร ศง ศพ ษก ฐก พิส ญวน ณรงค์ ศตวรรษ

กฎที่ 2

สระ ั จะต้องมีตัวสะกดอย่างน้อยหนึ่งตัวเสมอ

กฎที่ 3

สระ ิ จะมีตัวสะกดเพียงหนึ่งตัวเท่านั้น ยกเว้น ร ิ ห ิ อ ิ

กฎที่ 4

สระ ะ และ ำ จะต้องเป็นอักษรตัวสุดท้ายเสมอ

กฎที่ 5

สัญญลักษณ์ ์ โดยมากจะปรากฏเป็นตัวสุดท้าย ยกเว้นบางคำเช่น ฟาร์ม

กฎที่ 6

สระ ั ิ ี ึ ถ้ามีวรรณยุกต์ จะมีตัวสะกดเพียงตัวเดียวเท่านั้น

กฎที่ 7

ถ้าสระ ี ปรากฏพร้อมกับวรรณยุกต์อื่นที่ไม่ใช่ ี คำๆนั้นมีความเป็นไปได้ที่จะไม่ต้องการตัวสะกด

รูปแบบของภาษาไทย

ความหมายของสัญญลักษณ์

c : อักษรนำ

s : ตัวสะกด

t : วรรณยุกต์

g : การันต์

* : ซอบเขต

.. : อักษรใดๆ

สถานะของรูปแบบ

R : รูปแบบตายตัว

N : รูปแบบไม่ตายตัว

X : รูปแบบยกเว้น

สระ า	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c า	กา บาร์	N
	* c c า	มหา สถาน	N
	* c t า	น้ำ ย้าย	N
	* c c t า	หน่าย คล้าย	N

ตารางอักษรควบกล้ำของสระ า

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ท ห ส ข ร
ร	ก อ ท ต ส ป ค พ ช ฆ
อ	ส
ง	ห ส ผ
ม	ห ส
ย	อ ห ส พ ช
ว	ก ท ห ส ค ข ผ ถ
ล	ก อ ท ต ห ส ป ค พ ช ผ ถ ฉ
ท	ป
ต	ส

ห	ม ท ส
บ	ส ช
พ	ส
ถ	ส
ญ	ห
ภ	บ ส
ฏ	ช

สระ ิ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ิ	ค ิด จ ิบ	N
	* c c ิ	หย ิ่ง ปฏ ิ	N
	* c c c ิ	อมร ินทร์ สปร ิง อคต ิ	X,N

ตารางอักษรควบกล้ำของสระ ิ

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ว ห ส ช ถ
ร	ก อ ห ป บ ค พ ภ
ง	ห
ม	อ ท ห ส
ย	ห
ว	อ ท ห ส ค ช

ล	ก อ ต ห ส ป พ ผ
ต	ส
ห	อ
ช	ว
ถ	ส
ณ	ค พ
ธ	อ
ญ	ห
ภ	อ
ฏ	ป

สระ ี	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ี	ด ี ต ี ฝ ี	N
	* c c ี	วล ี หน ี ศร ี	N
	* c c c ี	สตร ี	N

ตารางอักษรควบกล้ำของสระ ็

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ห
ร	ก ท ต ห ส ป ค ศ ฟ
ก	ส
ม	ห
ย	ห
ว	ก ท ห ส ด
ล	ก ว ห ป ด พ ผ ถ
ด	บ ค
บ	ก
พ	ร
ณ	ม

สระ ุ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ุ	ข ุ ด จ ุ ด ห ุ น	N
	* c c ุ	สม ุ ด ปล ุ ก หน ุ ่ม	N

ตารางอักษรควบกล้ำของสระ ุ

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ม ห ส
ร	ก อ ว ส ป ค ผ
ก	อ ม ส ศ
อ	ช
ง	อ
ม	ห ส
ย	อ ห ส พ
ว	ห
ล	ป
ท	ส ผ
ด	ส ผ
ส	อ พ
ถ	ส
ธ	ม

สระ ุ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ุ	สุ้ ดุ ชุ	N
	* c c ุ	หนุ อยุ สบุ	N
	* c c c ุ	สกรุ	X,N

ตารางอักษรควบกล้ำของสระ ู

อักษรตาม	อักษรนำที่เป็นไปได้
น	ม ห ธ
ร	ห ป ค
ม	ห
ย	อ ห
ว	ห
ล	อ ห ป ฝ
ต	ส
ท	พ
บ	ส
ณ	อ

สระ ั	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ั	คั ด จั บ รั ก	N
	* c c ั	หยั ่ ง สงั ด หวั ด	N
	* c c c ั	มฆวั น	X

ตารางอักษรควบกล้ำของสระ ั

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ห ส ป พ ผ ฉ
ร	ก อ ท ต ห ส ป บ จ ค พ ฝ
ก	ส
ง	ห ส ผ
ม	ห ส
ย	ห ข
ว	ร ต ห ส ค ข ธ
ล	ก อ ต ห ส ป ค พ ผ
ท	ห
ด	ส
ห	ร อ ม
ส	อ ว
บ	ฉ
ภ	อ

สระ ั	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ั	ค ั ก จ ั ง ถ ั ง	N
	* c c ั	หน ั ่ ง สล ั ง หม ั ก	N

ตารางอักษรควบกล้ำของสระ ั

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ห ค ผ
ร	ต ป ค พ
ม	ห
ล	ก อ ส ค ผ ถ

สระ ี	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ี	ม ี ด ี น	N
	* c c ี	หม ี น กล ี น หร ี อ	N

ตารางอักษรควบกล้ำของสระ ี

อักษรตาม	อักษรนำที่เป็นไปได้
น	ห
ร	ห ป ค
ม	ห
ว	ห
ล	ก ห ป ค

สระ ◌็	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ◌็	ล◌็อค ช◌็อป ด◌็อก	N
	* c c ◌็	พล◌็อต สม◌็อค พล◌็อม	N

ตารางอักษรควบกล้ำของสระ ◌็

อักษรตาม	อักษรนำที่เป็นไปได้
ม	ส
ล	พ

สระ ะ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ะ	จะ ระ	N
	* c c ะ	สระ กระ ชยะ	N
	* c t ะ	ค่ะ จ๊ะ	N
	* c c c ะ	สรณะ อมตะ รตนะ	X,N

ตารางอักษรควบกล้ำของสระ ะ

อักษรตาม	อักษรนำที่เป็นไปได้
น	ช
ร	ก ท ต ห ส ป พ ข
ว	ห ส
ล	ห ส ค ผ
ณ	ค ณ ษ

สระ ำ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* c ำ	ดำ จำ นำ	N
	* c c ำ	หยำ ตรำ	N
	* c t ำ	ค่ำ น้ำ ย้ำ	N
	* c c t ำ	ปล้ำ สม่ำ หน่ำ	N

สระ เ	รูปแบบและขอบเขต	ตัวอย่าง	สถานะ
ขอบเขตหน้า	* เ	โดยมากจะเกิดกรณีนี้	N
	* c เ	พเนจร อเวจี (มีกรณีที่เกิดน้อย)	N

ตารางอักษรควบกล้ำของสระ เ

อักษรตาม	อักษรนำที่เป็นไปได้
น	อ ห
ร	ก ท ห ค ศ
ง	ห
ย	ห
ว	ห ค ช ศ
ล	ห พ ผ
ษ	ก

สระ เ-า เ-าะ

ตารางอักษรควบกล้ำของสระ เ-า

อักษรตาม	อักษรนำที่เป็นไปได้
ร	ป ค พ ศ
ง	ห
ม	ห ช
ย	ห ช
ว	ห
ล	ก ห ป ค พ ช

ตารางอักษรควบกล้ำของสระ เ-าะ

อักษรตาม	อักษรนำที่เป็นไปได้
ร	ก ป ค พ
ม	ห
ย	ห
ล	ห ป พ ผ

บทที่ 7
ส่วนการทำงานเกี่ยวกับภาพกราฟฟิก

ความสามารถหนึ่งของโปรแกรมนี้คือ การเก็บชื่อแฟ้มข้อมูลภาพกราฟฟิก และขนาดการพิมพ์ไว้พร้อมกับตัวอักษร เพื่อนำไปพิมพ์ได้ทันที โดยจะทำการอ่านแฟ้มข้อมูลและพิมพ์ภาพกราฟฟิกซึ่งเก็บในรูปแบบของแฟ้มข้อมูลภาพกราฟฟิกที่กำหนดไว้ได้ จะพิมพ์เป็นกรอบสี่เหลี่ยมโดยมีชื่อแฟ้มข้อมูลกำกับไว้เพื่อให้ผู้ใช้ทราบว่าเป็นภาพใด

การทำงานของส่วนที่เกี่ยวกับภาพกราฟฟิกแบ่งเป็น 2 ส่วนดังนี้

1. ส่วนที่ทำงานโดยให้ผู้ใช้กำหนดชื่อแฟ้มข้อมูล และขนาดภาพ ใช้งานในการแก้ไขแฟ้มข้อมูลของ wordprocessor นี้ จะเรียกว่า **ส่วนกำหนดภาพ**
2. ส่วนที่ทำงานเมื่อพิมพ์แฟ้มข้อมูล จะเรียกว่า **ส่วนพิมพ์ภาพ**

ส่วนกำหนดภาพ

การใช้งานของผู้ใช้ สามารถเลือกใช้งานได้จากเมนู "แก้ไขข้อมูล" โดยแบ่งเป็น "ติดรูป" และ "จัดรูป"

การทำงาน

ส่วนของการติดรูป เมื่อเลือเมนูติดรูป จะมีการทำงานต่อไปนี้คือ

1. รับชื่อแฟ้มข้อมูล โดยยังไม่จัดเก็บเข้าใน data structure ทั้งนี้เพื่อว่าถ้าผู้ ใช้ยกเลิกคำสั่งนี้กลางครันจะได้ไม่ต้องทำการเก็บและลบข้อมูล
2. หลังจากรับชื่อแฟ้มข้อมูลแล้วจะรับตำแหน่งของภาพได้แก่ ขอบด้านซ้าย ขอบด้านขวา และความยาวของภาพ โดยรับเป็นจำนวนตัวอักษร ส่วน interface กับผู้ใช้ในส่วนนี้จะเป็นกรอบสี่เหลี่ยมซึ่งให้ผู้ใช้ ลากแล้วกด Enter ไม่ต้องใส่เป็นตัวเลข
3. เมื่อผู้ใช้กด Enter ครั้งสุดท้ายเป็นการยอมรับการติดรูปภาพในข้อมูลแล้ว จะทำการเพิ่ม paragraph ใหม่ที่ตำแหน่ง cursor เพื่อให้เป็น paragraph ของภาพ เก็บข้อมูลตัวอักษรพิเศษคือ 0xFC เพื่อระบุว่า paragraph นี้เป็นภาพกราฟฟิก ไม่ใช่ตัวอักษร ตามด้วยชื่อ แฟ้มข้อมูล ,กรอบด้านซ้าย ,กรอบด้ายขวา และ ความยาว ของภาพ ตามลำดับ

ส่วนของการจัดรูป ในระหว่างที่ผู้ใช้แก้ไขข้อมูลแล้วต้องการแก้ไขขนาดของรูป หรือชื่อแฟ้มข้อมูลของรูป สามารถทำได้โดยเลือกเมนูจัดรูป การทำงานเป็นดังนี้

1. เมื่อผู้ใช้เลือกเมนู จะอ่านข้อมูลของภาพกราฟฟิกภาพนั้นเข้าในหน่วยความจำก่อนเพื่อทำการแก้ไข ทั้งนี้เพื่อว่าถ้าผู้ใช้ยกเลิกกลางครัน จะไม่ต้องแก้ไขข้อมูลใน data structure
2. ให้ผู้ใช้แก้ไขชื่อแฟ้มข้อมูล และขนาดของภาพตามลำดับ เหมือนการติดรูป
3. เมื่อผู้ใช้ยอมรับการแก้ไขแล้ว จึงเขียนข้อมูลของภาพนั้นกลับลงใน data structure

การแสดงผลทางหน้าจอ เพื่อความรวดเร็วของการแสดงผลและ เพื่อประหยัดหน่วยความจำ จึงแสดงผลเป็น กรอบสี่เหลี่ยมซึ่งมีชื่อภาพกำกับอยู่ด้วย ทั้งนี้ตีความว่าผู้ใช้ทราบอยู่แล้วว่าภาพที่จะติดลงในแฟ้มข้อมูลนี้เป็นภาพอะไร มีอัตราส่วนด้านกว้าง ด้านยาว เป็นเท่าไร การทำงานเป็นดังนี้

ขั้นตอนการติดรูป เมื่อผู้ใช้เลือกทำการติดรูป และใส่ชื่อแฟ้มข้อมูลภาพเรียบร้อยแล้ว

1. ทำการ scroll หน้าจอจากตำแหน่ง cursor ลง
2. จากข้อมูลตำแหน่ง ซ้าย ขวา และความยาว วาดกรอบสี่เหลี่ยม และ ใช้ฟังก์ชัน RefreshScr เพื่อเขียนส่วนที่เหลือของจอภาพต่อไป
3. เมื่อกรอบเกินจอภาพ จะแสดงผลเฉพาะส่วนของภาพ

การแสดงผลปรกติ เมื่อแสดงผลมาถึงส่วนที่เป็นภาพกราฟฟิก คือเมื่อตรวจพบตัวอักษรพิเศษ 0xFC แล้ว จะทำการแสดงผลโดยสร้างกรอบสี่เหลี่ยมขึ้นตามข้อมูลของภาพ และ พิมพ์ชื่อแฟ้มข้อมูลกำ กับไว้ด้วย

ส่วนพิมพ์ภาพ

การทำงาน

เมื่อทำการพิมพ์แฟ้มข้อมูลและทำการพิมพ์มาถึงส่วนของภาพกราฟฟิก จะมีการทำงานดังต่อไปนี้

1. ถ้าเดิมพิมพ์ในโหมดตัวอักษร จะเปลี่ยนโหมดการพิมพ์เป็นโหมดกราฟฟิก ถ้าเป็นโหมดกราฟฟิกอยู่แล้วก็ไม่ต้องเปลี่ยน
2. ทำการอ่านภาพกราฟฟิก ถ้าไม่พบแฟ้มข้อมูลภาพกราฟฟิก หรือรูปแบบการเก็บข้อมูลไม่เป็นไปตาม รูปแบบของ .PCX จะพิมพ์กรอบสี่เหลี่ยมซึ่งมีชื่อแฟ้มข้อมูลกำกับอยู่แทน
3. เมื่อทำการอ่านภาพกราฟฟิกได้แล้ว ต้องทำการแปลง scale สีของภาพให้เป็นภาพ grayscale ที่มี 10 ระดับ เพื่อให้สามารถทำการพิมพ์ทางเครื่องพิมพ์ได้ แล้วจึงส่งภาพออกทางเครื่องพิมพ์

การอ่านภาพกราฟฟิก

รูปแบบการเก็บข้อมูลของไฟล์ .PCX

การเก็บข้อมูลภาพเป็นไฟล์มีหลายวิธี วิธีหนึ่งที่แพร่หลายคือเก็บในลักษณะ .PCX File ตามมาตรฐานของบริษัท ZSoft ซึ่งมีรูปแบบการเก็บข้อมูลดังนี้

ไฟล์จะถูกแบ่งออกได้เป็น 3 ส่วนคือ

- Header 128 bytes เพื่อระบุรายละเอียดต่าง ๆ ของภาพนั้น ๆ
- Data เป็นข้อมูลภาพเก็บแบบ Run Length Encoding
- Color Map สำหรับภาพที่มีสีมากกว่า 16 สีจะเก็บเป็น 3xจำนวนสี bytes ส่วนภาพที่มีสี 16 สีจะเก็บข้อมูลของสีไว้ใน Header แล้ว

ส่วน Header

ส่วนต้นของแฟ้มข้อมูลภาพแบบ .PCX จำนวน 128 bytes เป็นรายละเอียดเกี่ยวกับภาพที่เก็บและวิธีการเก็บ เรียกว่า Header มีความหมายดังในตารางต่อไปนี้

Byte	Size(bytes)	Name	Description
0	1	Password	ไฟล์ .PCX ของบริษัท ZSoft จะมีค่า Password เป็น 0Ah
1	1	Version	Version ของโปรแกรม PC Paintbrush 0 = version 2.5 2 = version 2.8 พร้อมข้อมูล palette 3 = version 2.8 ไม่มีข้อมูล palette 5 = version 3.0
2	1	Encoding	1 = ใช้วิธี Run Length แบบของ .PCX
3	1	Bit per pixel	จำนวนบิทที่ใช้แทนจุด 1 จุดใน 1 plane
4	8	Window Dimension	เป็นค่า integer 4 จำนวนแทนจุดซ้ายบนและ จุดขวาล่าง ของส่วนที่จะแสดงภาพ

มีต่อ

ตาราง header (ต่อ)

Byte	Size(bytes)	Name	Description
			เรียงลำดับ เป็น x1,y1,x2,y2
12	2	Horizontal Resolution	ความละเอียดของอุปกรณ์แสดงผลในแนวนอน
14	2	Vertical Resolution	ความละเอียดของอุปกรณ์แสดงผลตามแกนตั้ง
16	48	Color Map	สำหรับภาพที่มีสีไม่เกิน 16 สี
64	1	Reserved	
65	1	Number of planes	จำนวน plane ที่ใช้ในการแสดงผล
66	2	Bytes per line	จำนวน byte ใน 1 scan line
68	2	Palette description information	วิธีแปล palette

ส่วนข้อมูล

ใช้วิธี Run Length Encoding สามารถ Decode ได้ดังนี้

1. ข้อมูลที่มีค่ามากกว่า C0h จะเป็นจำนวนจุดของข้อมูลตัวต่อไป ให้นำข้อมูลตัวปัจจุบันมา XOR กับค่า C0h จะได้จำนวนจุด แล้วจึงอ่านค่าข้อมูลที่ติดต่อกันจากข้อมูลตัวต่อไปหนึ่ง byte ซึ่งอาจจะมีค่า มากกว่า C0h อีกแต่จะเป็นค่าของสีไม่ใช่จำนวนนับ
2. ข้อมูลที่มีค่าน้อยกว่า C0h เป็นค่าสีของจุด

ตัวอย่างข้อมูลที่ทำ Run Length Encoding

ข้อมูลจริง คือ	3	3	3	3	3	3	3	3	3	3	3	3	191	192	64	64	64	64	64	2	2	2
จำนวนนับ และ ค่าข้อมูล คือ													13	3	7	1	191	1	192	5	64	3	2
เป็นเลขฐาน 16 คือ													0D	03	07		BF	01	C0	05	40	03	02
ข้อมูลที่ Encode แล้วคือ													CD	03	C7		BF	C1	C0	05	40	03	02

ลักษณะของภาพที่เก็บจะแบ่งได้เป็น 2 ลักษณะ คือ

1. เก็บแบบ 1 plane เป็นการเก็บที่สนับสนุนการแสดงผลแบบ VGA ปรกติ ส่วนมากมักจะเก็บเป็น 8 bit ต่อหนึ่งจุด จะเก็บภาพต่อเนื่องกันไปแบบ Run Length Encoding เมื่อ Decode แล้วสามารถใช้สีนั้นเป็นจุดที่จะแสดงผลได้ทันที
2. เก็บแบบหลาย plane ส่วนมากจะเป็น 4 plane ซึ่งสนับสนุนการแสดงผลแบบ EGA มักจะเก็บเป็น 1 bit ต่อหนึ่งจุด โดยจะเก็บข้อมูลของแถวแรกของ plane แรก และตามด้วยข้อมูลของแถวแรกของ plane ต่อ ๆ มา จนครบทุก plane แล้วจึงเก็บแถวต่อไปจนครบ เมื่อ Decode แล้วจะได้ แต่ละ byte เป็น pattern ของจุดที่ติดกัน 8 จุด ซึ่งสีจะกำหนดได้จาก plane ต่าง ๆ

ส่วน Color Map

กรณีที่เป็นการเก็บภาพแบบ 4 planes ซึ่งแสดงสีได้ 16 สี ส่วนของ color map จะอยู่ใน header แต่ในกรณีที่เก็บภาพแบบ 8 bit/pixel หรือมากกว่าซึ่งแสดงสีได้ 256 สีขึ้นไปส่วน color map จะอยู่ท้ายแฟ้มข้อมูลโดยจะเก็บต่อจากข้อมูลภาพไป

การเก็บข้อมูลใช้ 3 bytes ต่อ 1 สีโดยเก็บเรียงลำดับแม่สีของแสง แดง,เขียว,น้ำเงิน ตามลำดับ

<u>การพิมพ์ภาพกราฟฟิกจากไฟล์ .PCX ทางเครื่องพิมพ์ dot metrix</u>

dither pattern

จากภาพกราฟฟิก .PCX ซึ่งเป็นได้ทั้งภาพสีและขาวดำ และสามารถมีได้ถึง 256 สี เมื่อจะทำการพิมพ์ภาพออกทางเครื่องพิมพ์จะต้องทำการแปลงจากภาพ 256 สีให้เป็นภาพขาวดำเท่านั้น และเพื่อให้คุณภาพของภาพที่ได้ยังใกล้เคียงของเดิมมากที่สุดจึงต้องมีการแบ่ง scale ของสีให้มากกว่าสอง คือ แบ่งเป็น grayscale และ ใช้ pattern ของจุดขาว-ดำ แทนสีเทาระดับต่าง ๆ pattern เหล่านี้เรียกว่า dither pattern ซึ่งสามารถสร้าง pattern ที่เหมาะสมได้จากสิ่งที่เรียกว่า dither matrice โดยภาพที่ได้จะมี grayscale = nxn+1 ระดับ สำหรับ pattern ขนาด nxn

การสร้าง dither matrices สำหรับ pattern nxn ซึ่ง n เป็นเลขคู่ใด ๆ สร้างได้จากสมการดังนี้

$$D_n = \begin{bmatrix} 4D_{n/2} & 4D_{n/2}+2U_{n/2} \\ 4D_{n/2}+3U_{n/2} & 4D_{n/2}+4U_{n/2} \end{bmatrix}$$

โดยที่ D_n คือ dither matrice ขนาด nxn

U_n คือ matrice ที่มีสมาชิกทุกตัวเป็น 1

จากสมการจะสร้าง dither matrice 4x4 ได้ดังนี้

ให้ $D_1 = \begin{bmatrix} 0 \end{bmatrix}$ เนื่องจากถ้าใช้ D_1 จะมีภาพได้ 2 ระดับคือ 0 = ขาว และ 1 = ดำเท่านั้น

จาก D_1 จะได้ $D_2 = \begin{bmatrix} 4*0 & 4*0 + 2*1 \\ 4*0 + 3*1 & 4*0 + 4*1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$

จาก D_2 จะได้ $D_4 = \begin{bmatrix} 4*\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} & 4*\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}+2*\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ 4*\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}+3*\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} & 4*\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}+4*\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \end{bmatrix}$

$$D_4 = \begin{bmatrix} 0 & 8 & 0+2 & 8+2 \\ 12 & 4 & 12+2 & 4+2 \\ 0+3 & 8+3 & 0+1 & 8+1 \\ 12+3 & 4+3 & 12+1 & 4+1 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

จาก dither matrice D_4 จะสามารถสร้าง pattern ได้ 4x4+1 = 17 ระดับ ดังรูปที่ 6

การแปลงภาพสีเป็น 17 grayscale

เนื่องจากในภาพหนึ่งภาพจะประกอบด้วยสี 256 สีซึ่งแต่ละสีประกอบด้วยแม่สี 3 สีคือ แดง เขียว และน้ำเงิน สามารถแปลงสีแต่ละสีเป็นค่าความสว่างได้โดย ถือว่าค่าความสว่างของแสงสีแดงเป็น 30% สีเขียว 59% สีน้ำเงิน 11% เมื่อเทียบกับแสงขาว ดังนั้นจากภาพสี 256 สี จะสามารถแปลงเป็นภาพ 256 grayscale ได้โดยใช้ค่าความสว่างของแต่ละจุดเป็นค่า grayscale

จากภาพ 256 grayscale คัดเอา 17 scale ที่เหมาะสมมาเป็นค่าตัวแทนซึ่งจะใช้กับเครื่องพิมพ์ โดยจุดใดที่มีค่าไม่ตรงกับค่าทั้ง 17 ปัดเข้าสู่ค่าที่เหมาะสม

การคัดค่า grayscale ที่เหมาะสม

การคัดค่า grayscale จากการทดลองพบว่า

1. การคัดค่า grayscale ที่มีการใช้งานมากที่สุด 17 ค่ามาใช้เป็นค่าตัวแทน ใช้ได้ดีกับภาพที่สีทั้ง 17 มีค่าความสว่างแตกต่างกันมาก ๆ ถ้าค่าความสว่างอยู่ใกล้เคียงกันมาก ๆ แล้ว จะทำให้บางส่วนของภาพมืด หรือสว่างเกินไป ทำให้รายละเอียดของภาพขาดหายไป

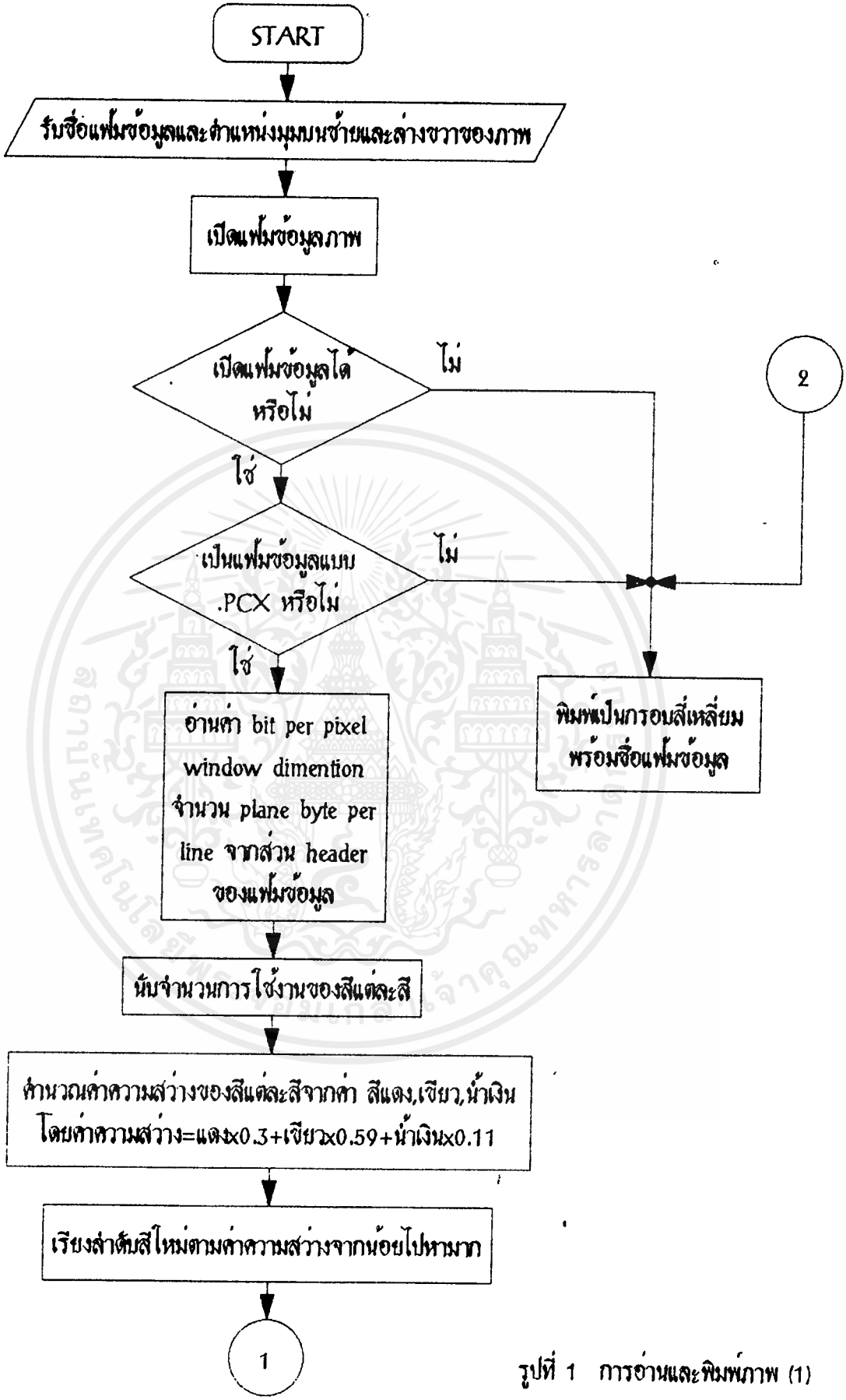
2. การคัดโดยแบ่งช่วงสีทั้งหมดจากสีที่มีค่าความสว่างต่ำสุดถึงสูงสุดด้วยอัตรส่วนที่เท่า ๆ กัน ใช้ได้ดีกับภาพที่มีการกระจายการใช้สีค่อนข้างเท่ากันและทั่วถึง ในกรณีที่มีการใช้สีที่ค่าความสว่างใดมาก ๆ จะทำให้ภาพที่ได้ขาดรายละเอียด

3. วิธีที่เลือกใช้คือ นับจำนวนจุดทั้งหมด แล้วแบ่งด้วยอัตราส่วนเท่า ๆ กัน (แบ่งจำนวนการใช้ ไม่ใช่แบ่งตามความสว่าง) และหาค่าความสว่างโดยเริ่มค่าความสว่างแรกที่ความสว่างต่ำสุดของภาพ และนับจำนวนจุดที่ใช้ค่าความสว่างเพิ่มขึ้นตามลำดับเมื่อจำนวนจุดถึงช่วงที่แบ่งไว้ให้ใช้ค่าความสว่างนั้นเป็นค่าตัวแทน วิธีนี้ยังมีข้อบกพร่องในกรณีที่มีการใช้สีเดียวเป็นปริมาณมาก ๆ จะมีการคัดค่าความสว่างซ้ำ แก้ไขโดยเมื่อพบว่าค่าความสว่างใดใน 17 ค่าซ้ำกันจะทำการ shift ข้อมูลมาทางซ้าย ซึ่งจะทำให้ข้อมูลที่ซ้ำเป็นค่าความสว่างสูงสุด แล้วทำการแบ่งค่าความสว่างสูงสุดกับค่าความสว่างรองออกเป็นส่วนเท่า ๆ กันใช้แทนค่าที่ซ้ำนั้น ปรากฏว่าโดยส่วนใหญ่แล้วผลการแปลงออกมาเป็นที่น่าพอใจ

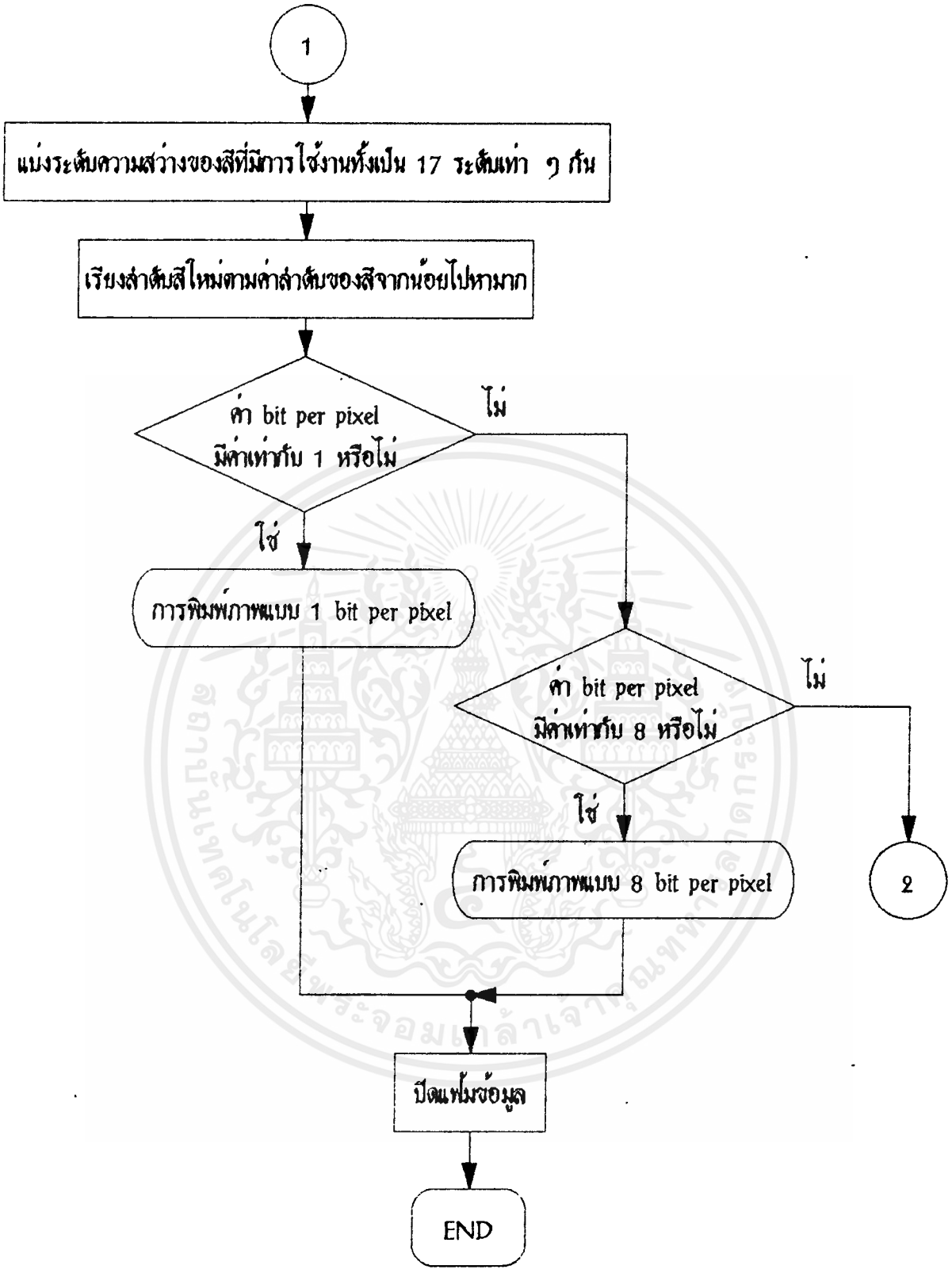
การนำภาพส่งให้เครื่องพิมพ์

จากภาพซึ่งเก็บเป็นรูปแบบของจอภาพคือเก็บข้อมูลตามแนวแกน x จุดละ 1 byte เมื่อส่งให้เครื่องพิมพ์แบบ dot metrix ซึ่งจะรับข้อมูลทีละ 1 byte แทนจุด 8 จุดตามแนวแกน y แต่ทุก ๆ 8 จุดจะเรียงลำดับในแนวแกน x จึงต้องมีการแปลง โดย

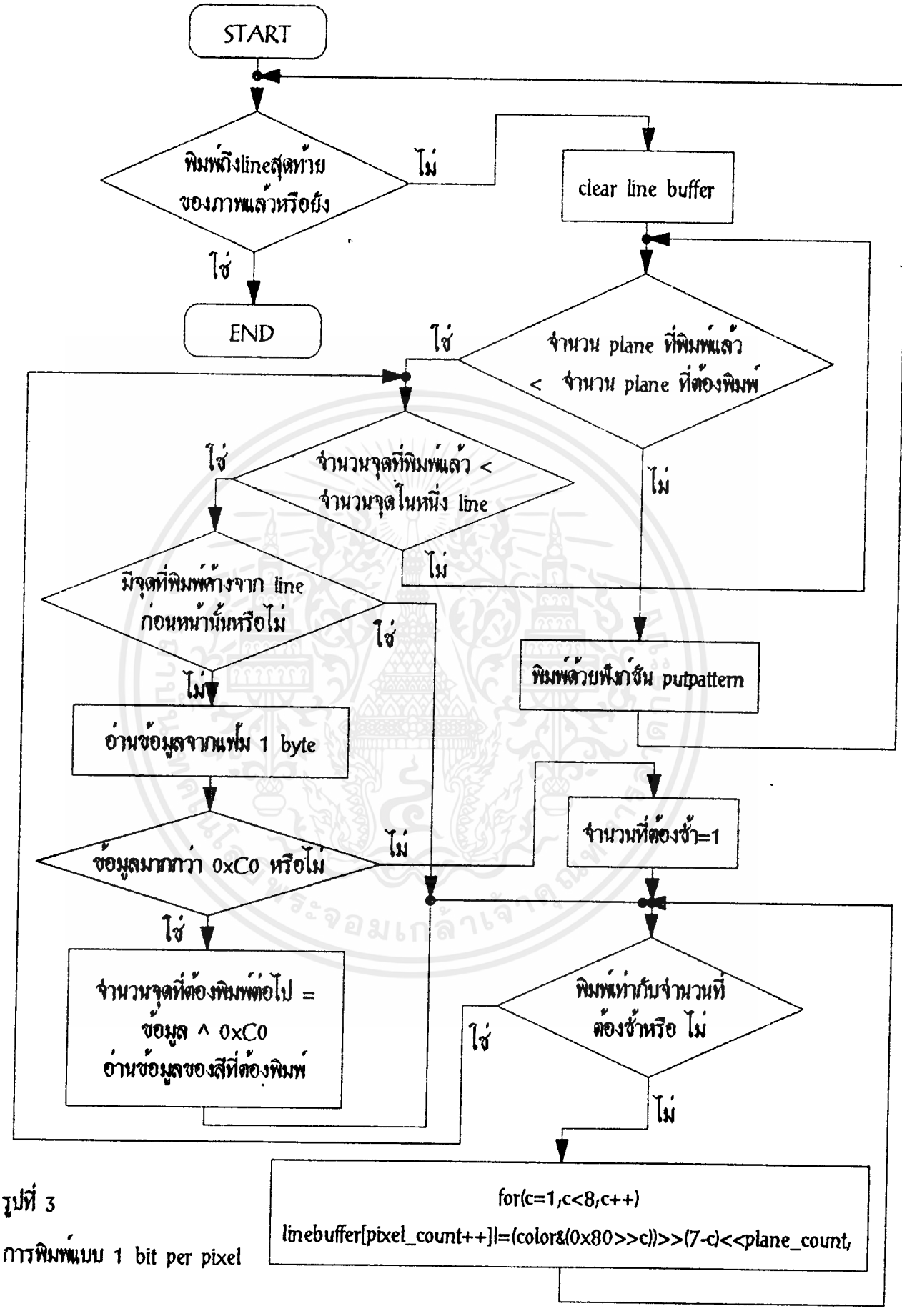
จากข้อมูลภาพจุดที่ x=1 y=1 ให้เป็น bit 7 ของ byte ที่ 1 ของข้อมูลที่จะส่งออกเครื่องพิมพ์ โดยข้อมูลจะทำ buffer เป็น array ไว้ จุดที่ x=1 y=2 จะเป็น bit 6 ของ byte ที่ 1 ของข้อมูลเครื่องพิมพ์ เมื่อ buffer เต็มก็นำข้อมูลส่งออกเครื่องพิมพ์แล้วทำซ้ำจนหมดข้อมูลภาพ



รูปที่ 1 การอ่านและพิมพ์ภาพ (1)

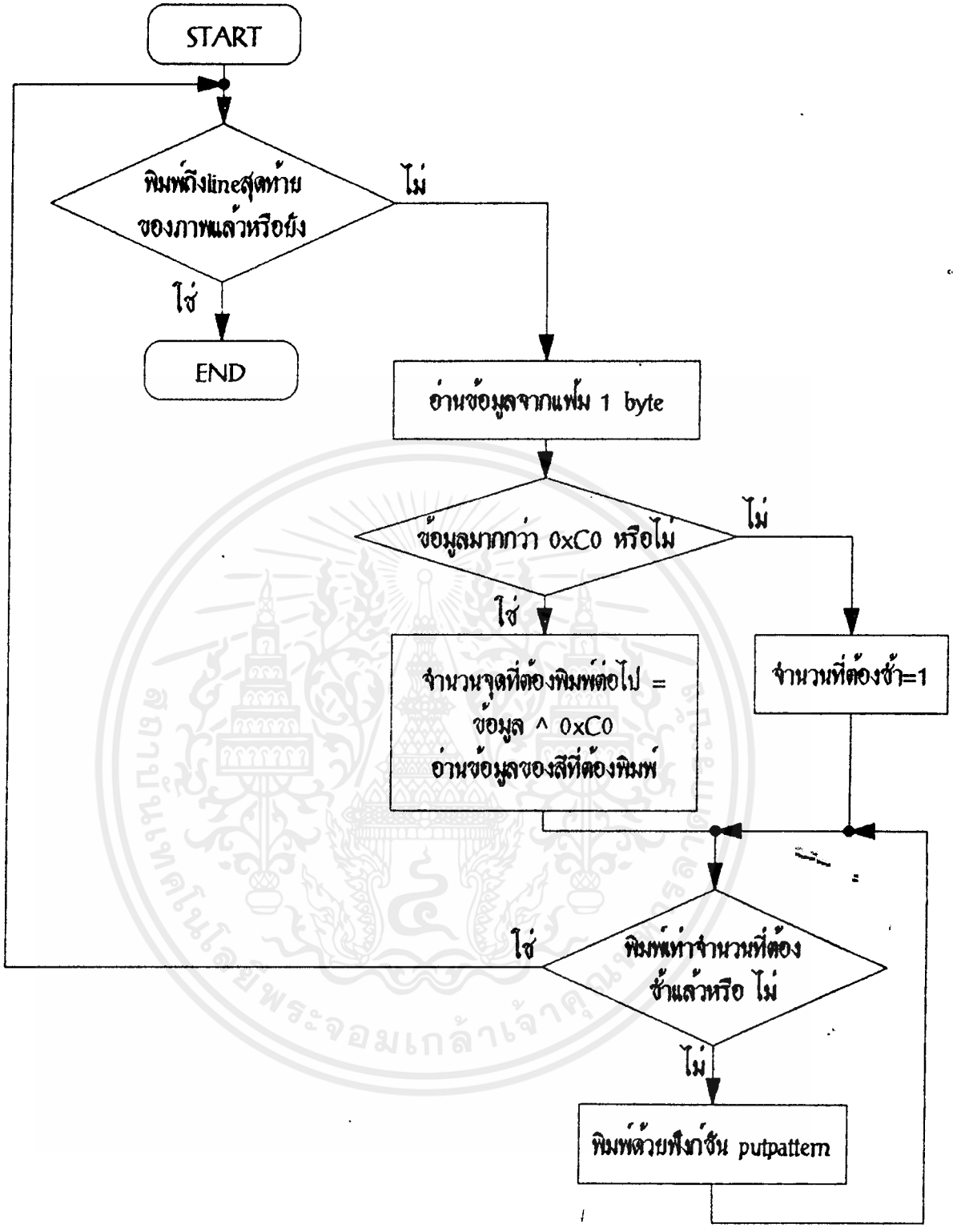


รูปที่ 2 การอ่านและพิมพ์ภาพ (2)

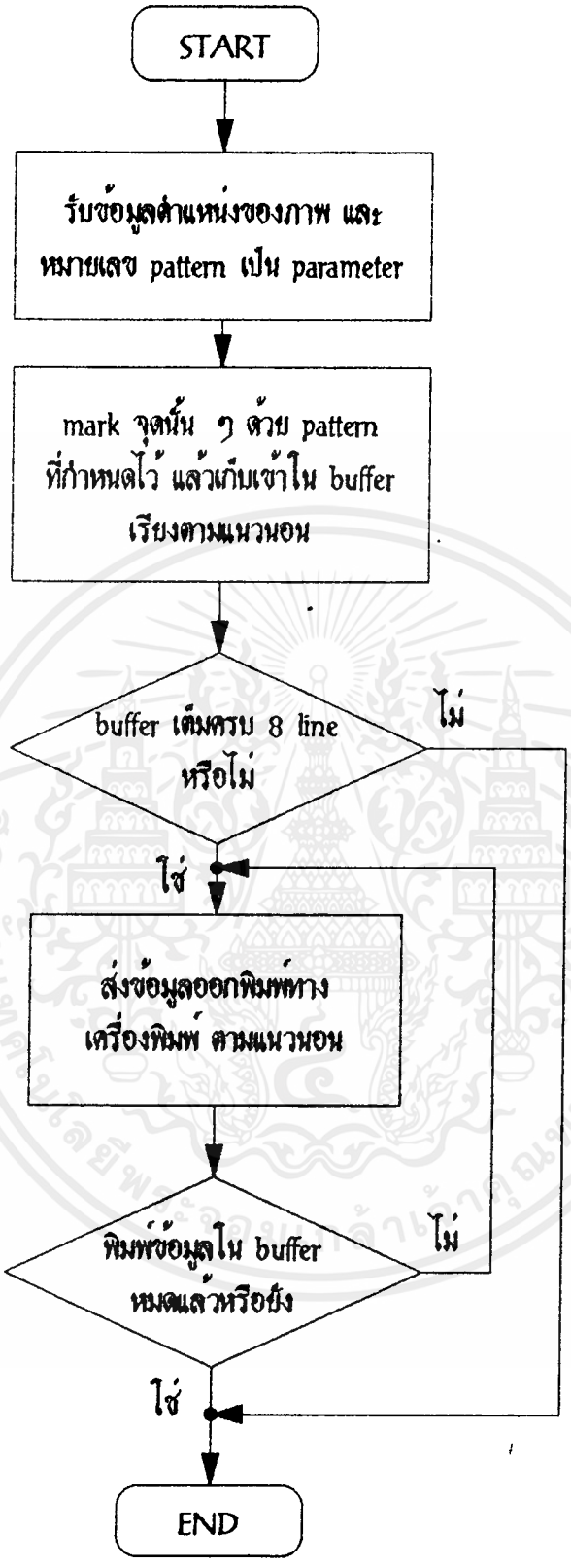


รูปที่ 3

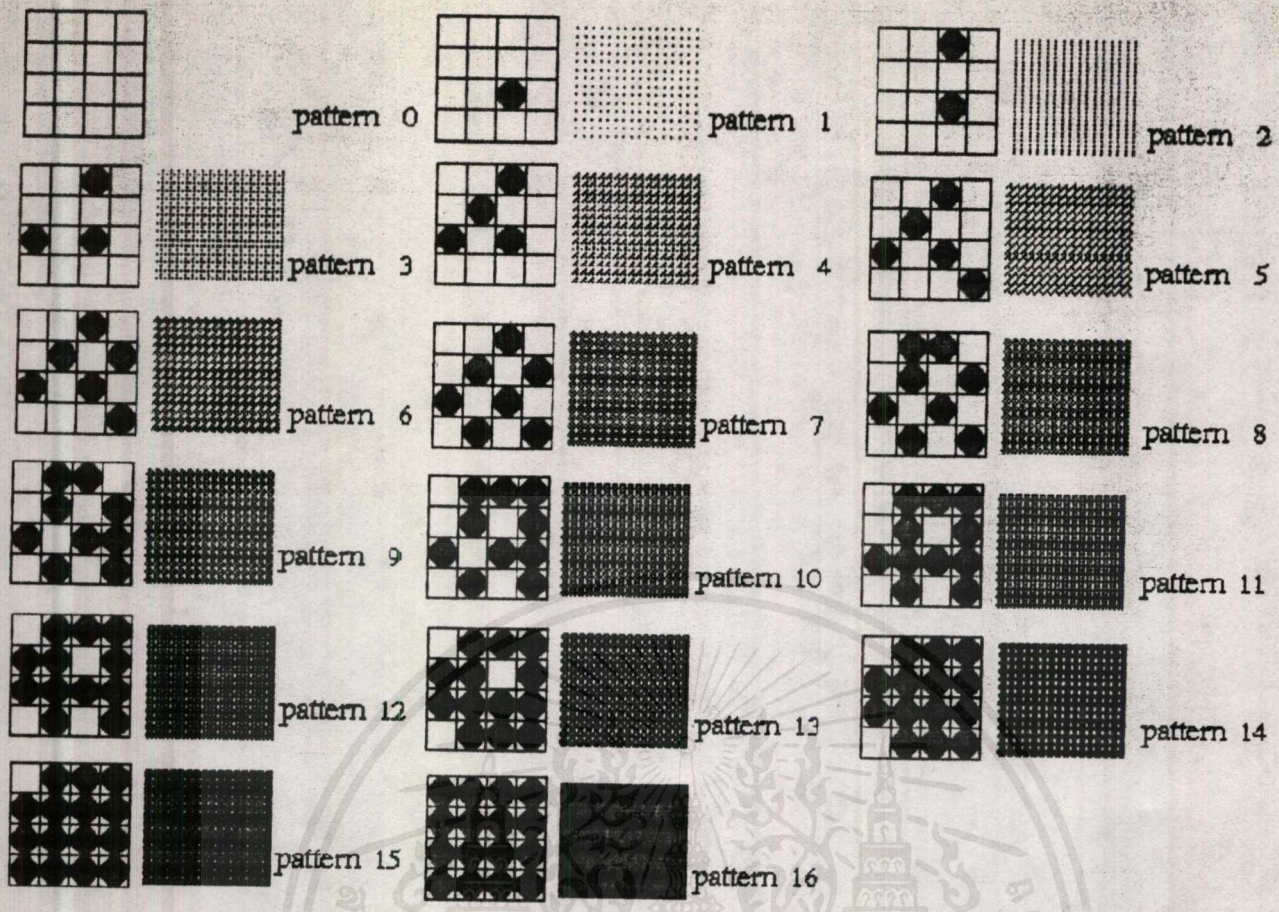
การพิมพ์แบบ 1 bit per pixel



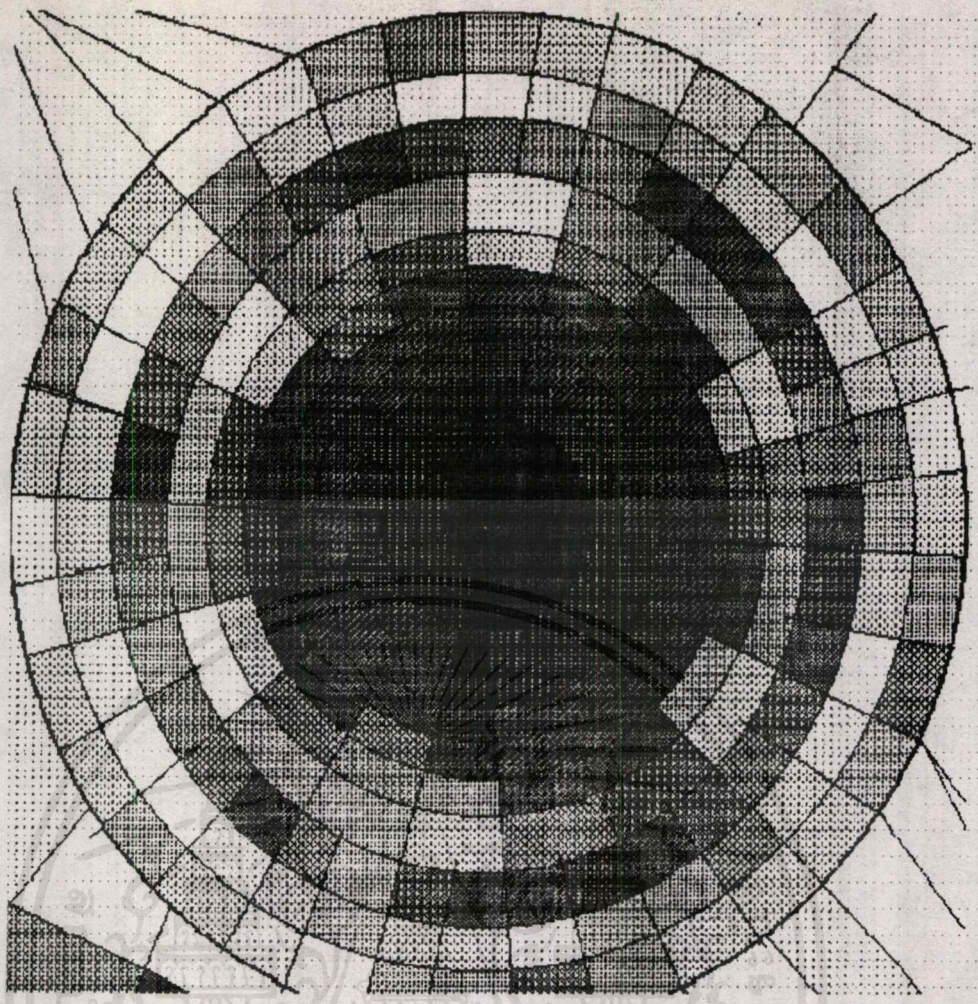
รูปที่ 4 การพิมพ์แบบ 8 bit per pixel



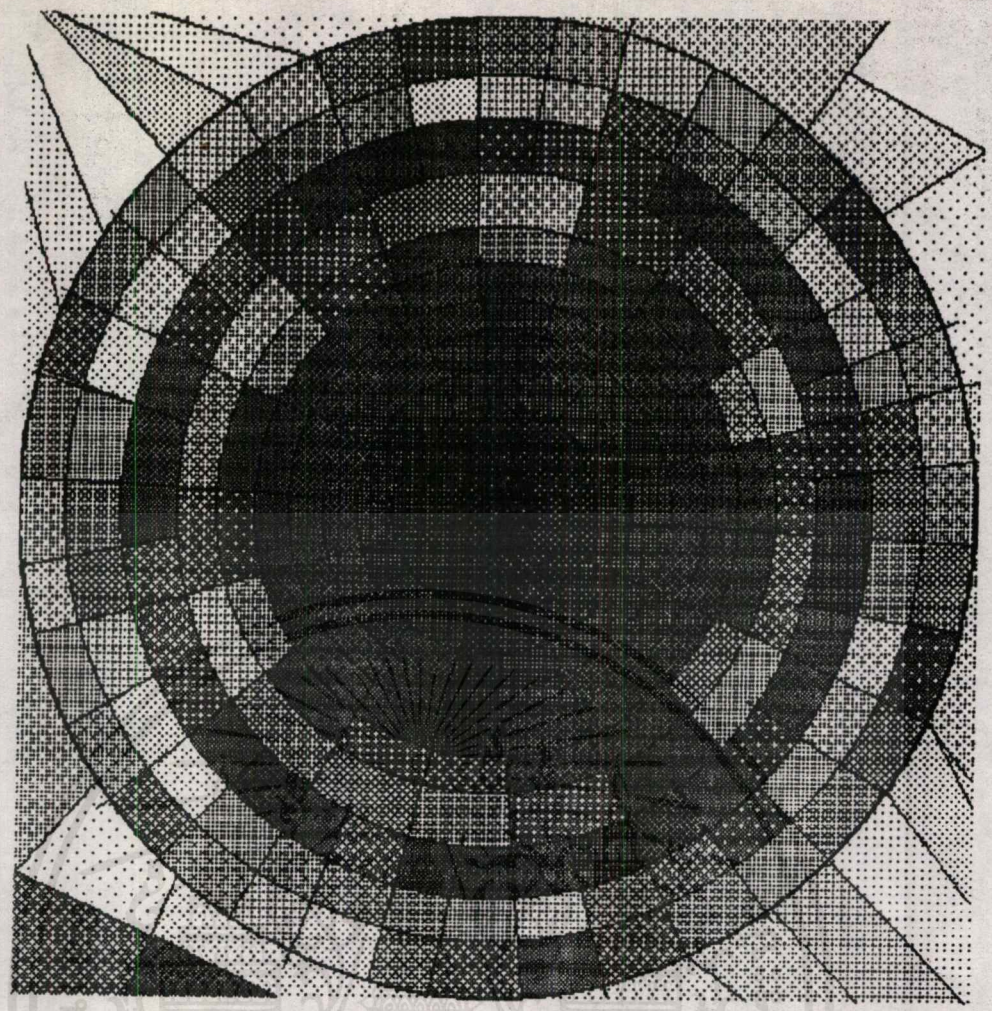
รูปที่ 5 ฟังก์ชัน putpattern



รูปที่ 6 pattern ที่ได้จาก dither matrice 4x4



ภาพวงกลม 256 สี พิมพ์ด้วยโปรแกรม pcx2prn



ภาพวงกลม 256 สี พิมพ์ด้วยโปรแกรม publisher paintbrush

ภาพตัวอย่าง ซึ่งพิมพ์ด้วยโปรแกรม pcx2prn

รายละเอียดของเครื่องพิมพ์ที่ใช้ในการทดสอบ

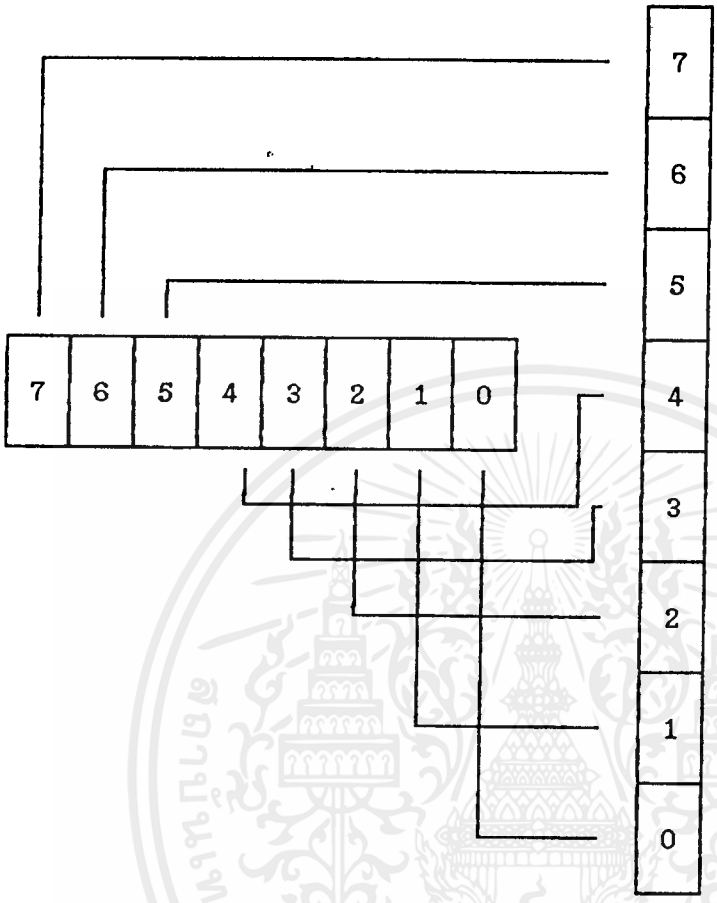
เครื่องพิมพ์ที่ใช้ในการทดสอบคือเครื่องพิมพ์ Epson LX-800 ซึ่งสามารถพิมพ์ในโหมดตัวอักษรและโหมดกราฟิก ซึ่งโหมดในการพิมพ์กราฟิกมีดังนี้

โหมดการพิมพ์	จำนวนเข็มที่ใช้ในการพิมพ์ (เข็ม)	จำนวนจุดในแนวนอน (จุดในแนวนอนต่อนิ้ว)
Single-density	8	60
Double-density	8	120
High-speed Double-density	8	120
Quadruple-density	8	240
CRT I	8	40
Plotter	8	80
CRT II	8	90

โหมดการทำงานที่เลือกคือทำงานในโหมด Double-density เพราะว่าในโหมดอื่นๆที่มีความละเอียดสูงกว่านี้ จะไม่สามารถพิมพ์จุดที่อยู่ติดกันได้ซึ่งอาจจะมีปัญหาในการพิมพ์ภาพกราฟิกหรือในการพิมพ์ตัวอักษร

ขั้นตอนการทำงานของการพิมพ์ในโหมดกราฟิก

การทำงานในการพิมพ์โหมดกราฟิกจะเป็นขั้นตอนในการส่งคำสั่งและข้อมูลไปยังเครื่องพิมพ์เพื่อให้เครื่องพิมพ์พิมพ์ภาพที่ต้องการ ลักษณะของคำสั่งจะเป็นคำสั่ง ในการเลือกโหมดการพิมพ์และคำสั่งส่วนหัวของการส่งข้อมูลไปพิมพ์ ข้อมูลที่จะส่งไปพิมพ์นี้จะได้จากการคำนวนโดยโปรแกรมหรือผู้ใช้โดยจะมีขนาดหนึ่งไบต์หัวเข็มของเครื่องพิมพ์ LX-800 ที่ใช้ในการพิมพ์กราฟิกจะใช้ 8 เข็ม โดยที่การสั่งให้เข็มแต่ละอันทำงานจะขึ้นกับข้อมูลไบต์ที่ส่งไปยังเครื่องพิมพ์ซึ่งแต่ละบิตจะตรงกับเข็มในเครื่องพิมพ์ดังนี้



ตัวอย่างเช่น หากต้องการให้พิมพ์โดยกดหัวเข็มที่ 7 และ 1 ทำได้โดยส่งข้อมูล 1000 0010 (82H)
ไปที่เครื่องพิมพ์

Algorithm ในการพิมพ์กราฟิก

1. กำหนดค่าความกว้างระหว่างบรรทัด

ในเครื่อง Epson LX-800 กำหนดให้มีขนาด 8/72นิ้ว โดยใช้คำสั่ง ESC A 8

2.ส่งข้อมูลกำหนดโหมดการพิมพ์ (Double-density) และจำนวนไบต์ที่ต้องการส่งไปพิมพ์
ใช้คำสั่ง ESC * และส่งขนาดของข้อมูลที่ต้องการพิมพ์

3.ส่งข้อมูลที่กำหนดออกไปพิมพ์

บทที่ 8
โปรแกรมประมวลผลคำภาษาไทยและภาษาอังกฤษ

ฟังก์ชัน main()

ฟังก์ชันหลักของโปรแกรมประมวลผลคำ การทำงานหลักที่สำคัญของฟังก์ชันคือการอ่านไฟล์ตัวอักษร ไฟล์แปลงรหัสภาษาไทย การเข้าสู่ภาษาไทยในโหมดกราฟิกส์และการสร้างโครงสร้างข้อมูลเริ่มต้น และทำหน้าที่สุดท้ายในการออกจากโปรแกรมประมวลผลคำคือจัดการปล่อยหน่วยความจำที่จองไว้และเปลี่ยนโหมดกลับมาอยู่ในโหมดตัวอักษรภาษาอังกฤษเหมือนเดิม

Algorithm ฟังก์ชัน main()

```
กำหนดค่าเริ่มต้น อ่านไฟล์ข้อมูลต่างๆ และเข้าสู่ภาษาไทยโหมดกราฟิก
สร้างโครงสร้างข้อมูลชุดแรกและกำหนดค่าให้ตัวแปรที่ชี้ขอบเขตเริ่มต้นของการแสดงผล
while(MainLoop)
{
    เปิดเครื่องหมายเคอร์เซอร์
    วนลูปรับการกดแป้นพิมพ์
    และหากมีการกดปุ่ม ESC ก็เข้าสู่ลูปเมนู
    /*ออกจาก while ลูปได้จากการกำหนดค่าตัวแปรในฟังก์ชัน menu()*/
}
คืนหน่วยความจำที่จองไว้สำหรับเก็บข้อมูลที่พิมพ์เข้า
เปลี่ยนโหมดการแสดงผลกลับมาสู่โหมดตัวอักษร
}
```

ฟังก์ชัน GetKeyboard()

เป็นฟังก์ชันหลักที่ทำงานอยู่เป็นส่วนใหญ่ของการรันโปรแกรม โดยฟังก์ชันนี้จะคอยรอรับการกดแป้นพิมพ์ต่างๆและให้บริการตามแป้นพิมพ์ที่กดนั้น หากมีการกดปุ่ม ESC ก็จะออกจากฟังก์ชันนี้ไปสู่ลูปหลักของฟังก์ชัน

main เพื่อเข้าสู่ฟังก์ชันเมนู แต่หากเป็นการกดแป้นพิมพ์อื่นๆ การทำงานก็จะอยู่ในฟังก์ชันนี้ตลอด

Algorithm ฟังก์ชัน GetKeyboard()

```
while(bioskey(1) ==0)     /* วนลูปรอจนกว่าจะมีการกดแป้นพิมพ์ปรกติ*/
{
     if(bioskey(2)&ALTPRESSED) /* ถ้าหากมีการกดปุ่ม ALT */
          สลับโหมดการแสดงผลระหว่างภาษาไทยและอังกฤษ
}
/* มีการกดแป้นพิมพ์ปรกติ */
key = bioskey(0);        /* อ่านค่าจากคีย์บอร์ดบัฟเฟอร์ */
ปิดเคอร์เซอร์
switch(key){
     case LEFT: กดปุ่มลูกศรซ้าย
     เรียกฟังก์ชัน CursorLeft();
 case RIGHT: กดปุ่มลูกศรขวา
     เรียกฟังก์ชัน CursorRight();
 case UP: กดปุ่มลูกศรขึ้น
    เรียกฟังก์ชัน CursorUp();
 case DOWN: กดปุ่มลูกศรลง
    เรียกฟังก์ชัน CursorDown();
 case HOME:     กดปุ่ม HOME
     เลื่อนเคอร์เซอร์กลับมายังอักษรตัวแรกของบรรทัด
     โดยเลื่อนเคอร์เซอร์ถอยหลังมาเรื่อยๆ จนเจอตัวอักษร WWRAP_CHAR
     เลื่อนเคอร์เซอร์เดินหน้า จะเป็นตำแหน่งอักษรตัวแรกของบรรทัด
 case END:      กดปุ่ม END
     เลื่อนเคอร์เซอร์ไปยังตัวอักษรตัวสุดท้ายของบรรทัด
     โดยเลื่อนเคอร์เซอร์เดินหน้าไปเรื่อยๆ จะเจอเครื่องหมาย WWRAP_CHAR
     เลื่อนเคอร์เซอร์ถอยหลังจนเจอตัวอักษรที่เป็นระดับ BASE ก็จะเป็นอักษรตัวสุดท้าย
```

case INSRT: กดปุ่ม Insert

สลับโหมดพิมพ์แทรก/พิมพ์ทับ

case DEL: กดปุ่ม Delete

ถ้าตัวอักษรที่ลบเป็นตัวอักษรมุมบนซ้ายสุดของจอภาพ

หลังจากลบตัวอักษรแล้วให้เก็บตำแหน่ง BeginDisp ใหม่

เรียกใช้ฟังก์ชัน Delete();

ถ้ากั้นซ้ายหรือขวามีการเปลี่ยนแปลงให้แสดงเครื่องหมายที่ตำแหน่งใหม่

case F1: กดปุ่ม F1

สลับโหมดตัวเอียง/ตัวปรกติ

case F5: กดปุ่ม F5

สลับโหมดตัวหนา

case F6: กดปุ่ม F6

สลับโหมดตัวขยาย

case F7: กดปุ่ม F7

สลับโหมดตัวขีดเส้นใต้

case F8: กดปุ่ม F8

สลับโหมดตัวขีดเส้นใต้สองเส้น

case F10: กดปุ่ม F10

ตัดคำ

case BACKSP: กดปุ่ม Backspace

ถ้าตัวอักษรที่ลบเป็นตัวอักษรมุมบนซ้ายสุดของจอภาพ

หลังจากลบตัวอักษรแล้วให้เก็บตำแหน่ง BeginDisp ใหม่

เรียกใช้ฟังก์ชัน BackSpace();

ถ้ากั้นซ้ายหรือขวามีการเปลี่ยนแปลงให้แสดงเครื่องหมายที่ตำแหน่งใหม่

case ENTER: กดปุ่ม ENTER

แทรกรหัสจบ paragraph และรหัสตัดคำ

แสดงเครื่องหมาย END_PARAG

สร้าง node สำหรับ paragraph ใหม่

```
      if(สร้าง paragraph ใหม่จาก paragraph เดิม)
          ย้ายข้อมูลที่อยู่หลังเครื่องหมาย END_PARAG มายัง paragraph ใหม่
      else
          กำหนดค่าแอทริบิวต์เริ่มต้น
      นำ node ที่สร้างใหม่เข้าไปรวมในโครงสร้างข้อมูล
      กำหนดค่าพอยน์เตอร์ให้ชี้ที่ paragraph ใหม่
      แสดงผลข้อความที่ตัดออกเป็น paragraph ใหม่
  case ESC:    กดปุ่ม ESC
      เซ็ตตัวแปรสำหรับเรียกเมนู
  case K:      ตรวจสอบสำหรับกรณีที่กด Ctrl+K
     if(ถ้ามีการกดปุ่ม Ctrl ร่วมกับ K)
     {
        if(ถ้าตามด้วย B เป็นการกำหนดต้นบล็อก)
           mark_begin_block();  /* Mark begin block */
        if(ถ้าตามด้วย K เป็นการกำหนดปลายบล็อก)
        {
           mark_end_block();       /* Mark end block */
           copy_to_clip();         /* คัดลอกข้อมูลเข้าคลิปบอร์ด*/
        }
        if(ถ้าตามด้วย C เป็นการคัดลอกข้อมูลจากคลิปบอร์ดเข้าหน่วยความจำ)
           clip_to_list();
        if(ถ้าตามด้วย T เป็นการย้ายบล็อก)
           clear_block();
        break;
     }
  default:      สำหรับกรณีที่กดปุ่มตัวอักษรทั่วไป
     scancode = (key&0xFF00)>>8;     แยกส่วนเป็นค่าแสกนโค้ด
     ch = key&0xFF;                  และแอสกีโค้ด
```

```
ch=Numeric(scancode,ch);
if(Sequence&&!CheckSequence(ch)) ตรวจสอบลำดับการป้อนข้อความ
      ผิดลำดับ ส่วเสียงเตือนและกลับไปรอรับคีย์ใหม่
 แสดงตัวอักษรที่พิมพ์เข้าไปโดยฟังก์ชัน DispChar();
 if(มีการเซ็ต WordWrapFlag)
 {
      if(ถ้ามีการแทรกรหัสตัดคำไว้แล้ว)
      {
            เลื่อนเคอร์เซอร์ไปยังอักษรตัวต่อไป
            และลบเครื่องหมายตัดคำตัวต่อไปให้หมดเพื่อตัดคำข้อความที่เหลือใหม่
      }else
            if(ถ้ายังไม่มีการแทรกเครื่องหมายตัดคำ)
            {
                  เลื่อนเคอร์เซอร์ไปจนเจอตัวอักษรระดับ BASE
                  ลบเครื่องหมายตัดคำตัวต่อไปให้หมดเพื่อตัดคำข้อความที่เหลือใหม่
                  แทรกรหัสตัดคำ
                  แสดงเครื่องหมายตัดคำ
            }
      WordWrapFlag =0;
}
if(กรณีที่มีการพิมพ์ทับ)
      ลบตัวอักษรปัจจุบันเพื่อให้แทรกตัวอักษรใหม่ได้
insert_char(ch); /*แทรกตัวอักษรใหม่ */
แสดงข้อความที่เหลือจนสุดจอภาพ
if(ถ้ามีการเปลี่ยนแปลงที่ตัวอักษรตัวแรกของจอภาพ)
      เก็บค่าตัวอักษรแรกของจอภาพใหม่
}
แสดงตำแหน่ง X,Y
```

เปิดเคอร์เซอร์

}

ฟังก์ชัน DeleteKey()

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับลบตัวอักษรที่ตำแหน่งเคอร์เซอร์ โดยจะลบตัวอักษรจำนวนหนึ่งเซลโดยที่ในหนึ่งเซลนั้นอาจจะมีตัวอักษรอยู่หลายระดับรวมกันก็ได้ เมื่อลบตัวอักษรจนหมด paragraph เคอร์เซอร์ก็จะเลื่อนไปอยู่ยัง paragraph ต่อไป

Algorithm ฟังก์ชัน DeleteKey()

```
/* นับจำนวนตัวอักษรที่ต้องลบ */
while(!IsMoveChar(DCurnt->data[I]))
{
     next();
     count++;
}

เลื่อนตำแหน่งไปยังท้าย paragraph
if(ตำแหน่งตัวอักษรปัจจุบันไม่ใช่ 0x00)
{
     เลื่อนเคอร์เซอร์ไปยังเซลสุดท้ายของ paragraph
     if(เคอร์เซอร์อยู่ที่ท้ายบรรทัดหรือท้าย paragraph)
     {
          กำหนดตัวแปรให้ต้องแสดงข้อมูลใหม่ในทุก paragraph
          while(next()!=0)
          {
               ทดลองเพิ่มตำแหน่ง X,Y ไปเรื่อยๆเหมือนการแสดงผลปรกติ
               แต่ไม่มีการแสดงผล
```

```
            }
            if(ตำแหน่งสุดท้ายที่จะแสดงผลไม่เกินจำนวนบรรทัดในจอภาพ)
                ClipLong =RghtMar-LftMar+1;
            else
                ClipLong =1;
        }else
            ClipLong =1;
    }else
        ClipLong =1;
/* ClipLong คือจำนวนของตัวอักษรบนจอภาพที่จะต้องลบหลังจากการทำงานเสร็จแล้ว*/

/* ลบตัวอักษรทั้งหมดในเซลที่อยู่ปัจจุบัน */
for( i =0; i <count; i++)
    delete_char();

if(ลบตัวอักษรจนกระทั่งหมด paragraph )
    if(ReadNext()  ==  WWRAP_CHAR)
        /* ใน paragraph เหลือ  WWRAP_CHAR  และ END_PARAG*/
        if((Temp=ReadPrev()) == WWRAP_CHAR||Temp == 0x00||Temp==ERROR)
        {
            if(Temp == 0x00 || Temp == ERROR )
            {
                ลบ END_PARAG
                ลบ WWRAP_CHAR
                เลื่อนเคอร์เซอร์ไปยังท้ายของ paragraph ใหม่
            }else{
                /* มีแต่ WWRAP_CHAR เพียงตัวเดียว */
                เลื่อนเคอร์เซอร์ไปยัง WWRAP_CHAR
```

```
                เรียกฟังก์ชัน delete_char();
            }
            เลื่อนเคอร์เซอร์ไปยังตำแหน่งเครื่องหมายตัดคำ(WWRAP_CHAR)

            /* เลื่อนเคอร์เซอร์ไปยังตำแหน่งแรกของบรรทัด */
            while(DCurnt->data[I]==WWRAP_CHAR||DCurnt->data[I]==0x00)
                next();
            กำหนดตำแหน่ง X,Y ที่จุดต้นบรรทัด
            แสดงข้อความบรรทัดนั้นใหม่จนหมดบรรทัด
        }
    แสดงข้อมูลส่วนที่เหลือจากตำแหน่งเคอร์เซอร์
}
```

ฟังก์ชัน BackspaceKey()

ฟังก์ชันนี้จะทำการลบตัวอักษรทางซ้ายมือหนึ่งตัวอักษร ซึ่งต่างจากการลบตัวอักษรครั้งละหนึ่งเซลเนื่องจากหนึ่งตัวอักษรอาจจะไม่ใช่หนึ่งเซลก็ได้ เช่นอาจใช้ฟังก์ชัน BackspaceKey() ลบวรรณยุกต์เอกเพียงตัวเดียว สำหรับข้อมูลหลาย paragraph เมื่อเราใกดปุ่ม Backspace ในขณะที่เราอยู่ที่ตำแหน่งแรกของ paragraph จะเป็นการรวม paragraph เข้าด้วยกัน โดยจะมีค่าการกำหนด paragraph ตาม paragraph แรก

Algorithm ฟังก์ชัน BackspaceKey()

```
/* คำนวนจำนวนตัวอักษรที่จะต้องลบจากจอภาพหลังจากที่ลบตัวอักษรและแสดงผลแล้ว */
เลื่อนตำแหน่งไปยังท้าย paragraph
if(ตำแหน่งตัวอักษรปัจจุบันไม่ใช่ 0x00)
{
    เลื่อนเคอร์เซอร์ไปยังเซลสุดท้ายของ paragraph
    if(เคอร์เซอร์อยู่ที่ท้ายบรรทัดหรือท้าย paragraph)
```

```
{
     ต้องแสดงข้อมูลใหม่ในทุก paragraph
     while(next()!=0)
     {
          ทดลองเพิ่มตำแหน่ง X,Y ไปเรื่อยๆเหมือนการแสดงผลปรกติ
          แต่ไม่มีการแสดงผล
     }
     if(ตำแหน่งสุดท้ายที่จะแสดงผลไม่เกินจำนวนบรรทัดในจอภาพ)
          ClipLong =RghtMar-LftMar+1;
     else
          ClipLong =1;
}else
     ClipLong =1;
}else
     ClipLong =1;
/* ClipLong คือจำนวนของตัวอักษรบนจอภาพที่จะต้องลบหลังจากการทำงานเสร็จแล้ว*/
/* เริ่มส่วนลบตัวอักษร */
เลื่อนเคอร์เซอร์ไปยังตัวอักษรตัวก่อนหน้า
เลื่อนเคอร์เซอร์แสดงผลไปยังตำแหน่งก่อนหน้า
ลบตัวอักษรบนจอภาพหนึ่งตัวอักษร
if(ตัวอักษรที่ลบไม่ใช่ตัวอักษรที่ความกว้าง 1 เซล)
     เพิ่มตำแหน่ง X หนึ่งตำแหน่ง
เรียกฟังก์ชัน backspace();
แสดงข้อความส่วนหลังจากการแก้ไข
}
```

ฟังก์ชัน CursofLeft()

ฟังก์ชันนี้จะทำงานในลักษณะของการกดปุ่มลูกศรไปทางซ้ายหนึ่งเซล แสดงผล หากว่าเคอร์เซอร์อยู่ที่บรรทัดแรกของจอภาพและยังมีข้อมูลอยู่ก่อนหน้านี้อีก ก็จะเป็นการ Scroll จอภาพลงและแสดงข้อมูลบรรทัดก่อนหน้าอีกหนึ่งบรรทัด ในการเลื่อนเคอร์เซอร์แต่ละครั้งถ้าหากว่าเป็นการเลื่อนข้าม paragraph ก็จะต้องมีการเปลี่ยนเครื่องหมายกำหนดกั้นซ้ายขวาด้วย หลังจากเลื่อนเคอร์เซอร์ไปทางซ้ายแล้ว เคอร์เซอร์จะอยู่บนตัวอักขระที่มีระดับการแสดงผลเป็น BASE เท่านั้น

Algorithm ฟังก์ชัน CursorLeft()

```
if(ยังมีข้อมูลตัวก่อนหน้านี้)
{
    ลดตำแหน่งแกน X ลงหนึ่งตำแหน่ง
    if(ถ้าตำแหน่ง X น้อยกว่ากั้นซ้าย)
    {
        เลื่อนเคอร์เซอร์ไปจนกระทั่งชี้ที่ตัวอักษรตัวสุดท้ายของบรรทัดบน
        if(ลดตำแหน่งแกน Y ลงหนึ่งตำแหน่งแล้วน้อยกว่า 1)
        {
            Scroll จอภาพลง
            แสดงข้อความบรรทัดก่อนหน้านี้หนึ่งบรรทัด
        }
        ปรับตำแหน่งเคอร์เซอร์บนจอภาพให้ชี้ถูกตำแหน่ง
    }
    while(เคอร์เซอร์ชี้ที่ตัวอักษรที่ไม่ใช่ BASE )
        เลื่อนเคอร์เซอร์ไปยังอักษรตัวต่อไป
}
ถ้าหากว่ามีการเปลี่ยนแปลงกั้นซ้าย/ขวา
ลบเครื่องหมายกั้นซ้ายขวาเดิมและวาดใหม่ที่ตำแหน่งใหม่
}
```

ฟังก์ชัน CursorRight()

ฟังก์ชันจะเลื่อนตำแหน่งเคอร์เซอร์ทั้งในหน่วยความจำและบนจอภาพไปทางขวาหนึ่งตำแหน่ง ถ้าหากเคอร์เซอร์อยู่ที่ท้าย paragraph ก็จะเป็นการเลื่อนไปยัง paragraph ใหม่และต้องวาดเครื่องหมายกั้นซ้าย/ขวาใหม่ด้วย ในกรณีที่เคอร์เซอร์อยู่บรรทัดสุดท้ายและตัวอักษรตัวสุดท้าย เมื่อเรียกฟังก์ชันนี้ หากยังมีข้อมูลต่อไปอีกก็จะเป็นการ Scroll จอภาพขึ้นและแสดงข้อความบรรทัดต่อไปหนึ่งบรรทัด

Algorithm ฟังก์ชัน CursorRight()

```
เลื่อนเคอร์เซอร์ไปยังตัวอักษรต่อไปที่มีระดับการแสดงผลเป็น BASE
ถ้าหากว่าเลื่อนเคอร์เซอร์ไปไม่ได้เพราะไม่มีข้อมูลอีกแล้ว
     return;

if(เคอร์เซอร์ชี้อยู่ที่ WWRAP_CHAR)
{
     เลื่อนเคอร์เซอร์ต่อไปจนกระทั่งเจอตัวอักษร
     if(เพิ่มตำแหน่งในแกน Y ถ้าหากว่าเลยจอภาพ)
     {
          Scroll จอภาพขึ้นหนึ่งบรรทัด
          แสดงข้อความต่อมาอีกหนึ่งบรรทัด
     }
     ให้เคอร์เซอร์ชี้ที่อักษรตัวแรกของบรรทัด
}else /* ไม่มีการข้าม paragraph */
     เลื่อนตำแหน่งในแกน X หนึ่งตำแหน่ง
ถ้ากั้นซ้าย/ขวามีการเปลี่ยนแปลง
     เลื่อนเครื่องหมายกั้นซ้าย/ขวาไปตำแหน่งใหม่
}
```

ฟังก์ชัน CursorUp()

ฟังก์ชันนี้จะเลื่อนเคอร์เซอร์ขึ้นหนึ่งบรรทัดโดยที่ตำแหน่งในแนวแกน X จะคงที่ ถ้าหากว่าเคอร์เซอร์อยู่ในบรรทัดบนสุดของจอภาพก็จะเป็นการเลื่อนจอภาพลงหนึ่งบรรทัดและแสดงข้อความบรรทัดก่อนหน้าบรรทัดบนสุดเดิมหนึ่งบรรทัด ถ้าหากในแนวแกน X ไม่มีตัวอักษรที่ตรงกันอยู่เลยก็จะไม่สามารถเลื่อนเคอร์เซอร์ได้ ผลของการเลื่อนเคอร์เซอร์ข้าม paragraph และมีการเปลี่ยนกั้นซ้าย/ขวาก็จะต้องเปลี่ยนเครื่องหมายกั้นซ้าย/ขวาเช่นเดียวกับฟังก์ชันอื่น

Algorithm ฟังก์ชัน CursorUp()

เลื่อนเคอร์เซอร์ขึ้นหนึ่งบรรทัดโดยเรียกใช้ฟังก์ชั่น ChangeLine();
ถ้าหากว่าเป็นบรรทัดแรก
    Scroll จอภาพลงและแสดงข้อความบรรทัดก่อนหน้า
ถ้ากั้นซ้ายหรือกั้นขวาเปลี่ยนไปจากเดิม
    แสดงเครื่องหมายกั้นซ้ายและขวาที่ตำแหน่งใหม่
}

ฟังก์ชัน CursorDown()

ฟังก์ชันนี้จะเลื่อนเคอร์เซอร์ลงหนึ่งบรรทัดโดยที่ตำแหน่งในแนวแกน X จะคงที่ ถ้าหากว่าเคอร์เซอร์อยู่ในบรรทัดล่างสุดของจอภาพก็จะเป็นการเลื่อนจอภาพขึ้นหนึ่งบรรทัดและแสดงข้อความบรรทัดต่อจากบรรทัดล่างสุดเดิมหนึ่งบรรทัด ถ้าหากในแนวแกน X ไม่มีตัวอักษรที่ตรงกันอยู่เลยก็จะไม่สามารถเลื่อนเคอร์เซอร์ได้ ผลของการเลื่อนเคอร์เซอร์ข้าม paragraph และมีการเปลี่ยนกั้นซ้าย/ขวาก็จะต้องเปลี่ยนเครื่องหมายกั้นซ้าย/ขวาเช่นเดียวกับฟังก์ชันอื่น

Algorithm ฟังก์ชัน CursorDown()

เลื่อนเคอร์เซอร์ลงหนึ่งบรรทัดโดยเรียกใช้ฟังก์ชั่น ChangeLine();
ถ้าหากว่าเป็นบรรทัดสุดท้าย
    Scroll จอภาพขึ้นและแสดงข้อความบรรทัดต่อไปหนึ่งบรรทัด
ถ้ากั้นซ้ายหรือกั้นขวาเปลี่ยนไปจากเดิม

แสดงเครื่องหมายกั้นซ้ายและขวาที่ตำแหน่งใหม่

ฟังก์ชัน ChangeLine()

เป็นฟังก์ชันที่ทำหน้าที่ในการเลื่อนเคอร์เซอร์ขึ้นไปยังบรรทัดต่อไป โดยที่ตำแหน่งในแนวแกน X ยังเหมือนเดิม มีอาร์กิวเมนต์ที่จะผ่านไปคือ ค่าตำแหน่งปัจจุบัน(PosX) เพื่อใช้ในการคำนวนหาตำแหน่งที่จะอยู่ในบรรทัดใหม่ สาเหตุที่ต้องเขียนฟังก์ชันนี้ขึ้นมาเพราะในโปรแกรมประมวลผลคำนี้เป็นแบบ paragraph ซึ่งแต่ละ paragraph สามารถมีกั้นซ้ายหรือกั้นขวาที่ต่างกันได้ การเลื่อนเคอร์เซอร์ที่ข้าม paragraph จึงจะต้องมีการคำนวนตำแหน่งใหม่ให้ถูกต้อง ตำแหน่งใหม่ในที่นี้หมายถึงตำแหน่งในหน่วยความจำ สำหรับตำแหน่งบนจอภาพก็เพียงแต่เลื่อน PosY ขึ้นหรือลงเท่านั้น สำหรับค่าอาร์กิวเมนต์ที่เป็น ลบ จะหมายถึงการเลื่อนเคอร์เซอร์ไปยังบรรทัดบน สำหรับค่าบวกจะหมายถึงเลื่อนเคอร์เซอร์ไปบรรทัดถัดไป

Algorithm ฟังก์ชัน ChangeLine(int DispX)

```
    if(DispX > 0)
    {
        เลื่อนเคอร์เซอร์ไปยังบรรทัดถัดไป
        เลื่อนเคอร์เซอร์ไปยังท้ายบรรทัด
        เลื่อนเคอร์เซอร์ไปยังต้นบรรทัดใหม่
    }else{ /* เลื่อนเคอร์เซอร์ไปยังบรรทัดบน */
        เลื่อนเคอร์เซอร์ในหน่วยความจำไปยังตัวอักษรตัวสุดท้ายของบรรทัดบน
        เลื่อนเคอร์เซอร์ไปยังตัวอักษรตัวแรกของบรรทัดบน
    }
    if(ถ้าสามารถเลื่อนเคอร์เซอร์ขึ้นหรือลงได้)
        เลื่อนเคอร์เซอร์ไปชี้ตำแหน่งข้อมูลที่มีตำแหน่งของ X บนจอภาพตรงกับบรรทัดที่แล้ว
    else
        return(0);      /* ไม่สามารถเลื่อนเคอร์เซอร์ได้ */
  return(1); /* การทำงานสำเร็จ */
}
```

ฟังก์ชัน DispChar()

ฟังก์ชัน DispChar นี้เป็นฟังก์ชันที่อยู่เหนือฟังก์ชัน PutC() หนึ่งระดับ โดยฟังก์ชันนี้จะเรียกใช้ฟังก์ชัน PutC เพื่อสั่งให้แสดงผลตัวอักษรที่ตำแหน่ง PosX,PosY และมีหน้าที่เปลี่ยนตำแหน่ง PosX หรือ PosY หลังจากการแสดงผล และมีหน้าที่ตรวจสอบและจัดระดับการแสดงผลสำหรับตัวพยัญชนะหรือสระด้วย ในโครงงานนี้ได้สร้างส่วนจัดการขึ้นบรรทัดใหม่อย่างง่ายไว้ คือเมื่อผู้ใช้แสดงผลตัวอักษรมาจนเกินกั้นขวาแล้ว WordWrapFlag ก็จะถูกเซ็ต เมื่อออกจากฟังก์ชันนี้ไปก็จะมีฟังก์ชันอื่นคอยตรวจสอบแฟลกนี้และแทรกรหัสสำหรับขึ้นบรรทัดใหม่ให้ หน้าที่ของฟังก์ชันนี้นอกจากการแสดงตัวอักษรปรกติแล้วยังมีหน้าที่ในการแสดงผลตัวอักษรพิเศษ เช่นตัวอักษร WWRAP_CHAR ด้วย

algorithm ฟังก์ชัน DispChar()

```
ถ้ารหัสที่ต้องการแสดงผลเป็น WWRAP_CHAR
    ขึ้นบรรทัดใหม่
ถ้าตัวอักษรที่จะแสดงผลเป็นสระหรือวรรณยุกต์ที่แสดงผลโดยไม่ต้องเลื่อนเซล
{
    ถอยหลังกลับไปหนึ่งตำแหน่งเพื่อแสดงผล
    เลือกโหมดการแสดงผลเป็น XOR
}
if(ถ้าหากว่าตำแหน่งที่จะพิมพ์ใหม่เกินจากกั้นขวาแล้ว)
{
    ถ้าหากมีเครื่องหมาย END_PARAG ต้องลบเครื่องหมายนี้ก่อน
    เซ็ต WordWrapFlag เพื่อให้มีการแทรก WWRAP_CHAR
    ขึ้นบรรทัดใหม่
}
เรียกใช้ฟังก์ชัน PutC() สำหรับแสดงตัวอักษร
เพิ่มตำแหน่ง X หนึ่งเซล
}
```

ฟังก์ชัน DispStr()

ฟังก์ชันนี้จะใช้สำหรับการแสดงผลข้อความในหน่วยความจำโดยจะแสดงข้อมูลตั้งแต่ตำแหน่งที่เคอร์เซอร์ชี้อยู่ไปจนถึงจบ paragraph หรือจนสุดจอภาพแล้วแต่โหมดที่ผู้ใช้กำหนด ฟังก์ชันนี้จะมีส่วนของการตัดคำเข้ามาเกี่ยวข้องด้วย โดยจะตรวจสอบและตัดคำเมื่อมีการแสดงข้อความที่เกินกั้นขวา

Algorithm ฟังก์ชัน DispStr()

```
ลบเครื่องหมายตัดคำที่อยู่ในข้อความเดิม
ถ้า mode เท่ากับ 1
     ลบเฉพาะ paragraph เดียว
มิฉะนั้น
     ลบจากข้อมูลทั้งหมด
ถ้าหากเคอร์เซอร์ชี้ที่รหัสควบคุม
     เลื่อนเคอร์เซอร์ไปยังตำแหน่งต่อไป
do{
     if(มีการข้ามไปยัง Paragraph ใหม่)
     {
          if(mode) /* ถ้าต้องการแสดงผลเฉพาะหนึ่ง Paragraph */
          {
               เลื่อนเคอร์เซอร์กลับไปยัง paragraph เดิม
               break;
          }
          เปลี่ยนตำแหน่งเคอร์เซอร์แสดงผลไปยังบรรทัดใหม่ที่อักษรตัวแรก
          ลบข้อความในบรรทัดใหม่ทั้งบรรทัด
     }
     if(แสดงผลถึงบรรทัดสุดท้าย)
          EndLine =1;
     if(ถ้าข้อตัวอักษรที่จะแสดงไม่ได้เป็น 0x00)
```

```
    {
        if(เป็นตัวอักขระดับการแสดงผล TOP หรือ ABOVE หรือ BELOW)
        {
            ลงตำแหน่ง X หนึ่งตำแหน่ง
            เลือกโหมดการแสดงผลเป็น XOR
        }else
            เลือกโหมดการแสดงผลเป็น FILL;
        if(ถ้าข้อความที่แสดงยาวเกินกั้นขวา)
        {
            if(ข้อความที่แสดงเป็นบรรทัดสุดท้าย)
            {
                แทรกรหัสตัดคำ WWRAP_CHAR;
                break;
            }
            ลบเครื่องหมาย END_PARAG
            แสดงเครื่องหมายตัดคำ
            แสดงตัวอักษรหนึ่งจนครบหนึ่งเซล
            if(ตำแหน่งปัจจุบันไม่ใช่ WWRAP_CHAR)
                แทรกรหัสตัดคำ(WWRAP_CHAR)
        }
        แสดงตัวอักษรและเลื่อนตำแหน่ง X
    }
}while(เลื่อนไปยังตัวอักษรตัวต่อไป);
/* ลบข้อความเดิมที่จะต้องลบทิ้งเนื่องจากความยาวในการแสดงผลสั้นลงจากเดิม*/
if(ไม่มีการขึ้น paragraph ใหม่)
    if(ความยาวของข้อความที่จะลบไม่เกินหนึ่งบรรทัด)
        ลบข้อความจนครบที่กำหนดไว้
    else{
```

ลบตัวอักษรจนถึงกั้นขวา
คำนวนความยาวที่จะต้องลบต่อไป
ลบตัวอักษรจนครบที่คำนวนได้

}

}

การ Directory files

เมื่อต้องการ Load หรือ Save ข้อมูลและให้ผู้ใช้ตั้งชื่อไฟล์ต้องทำการตรวจหาไฟล์ซึ่งปรากฏอยู่แล้วบนแผ่น disk โดยฟังก์ชัน getfilename จะทำการรับชื่อไฟล์รวมทั้ง drive และ path ด้วย โดยการเรียกใช้ฟังก์ชันในการรับ input string อีกต่อหนึ่ง หลังจากได้ input string แล้วจะนำไปแยกออกเป็น drive,path และ filename แล้วทำการเปลี่ยน drive,directory เพื่อค้นหาชื่อไฟล์ที่ตรงกับ filename การค้นหาเป็นฟังก์ชันชื่อ fsearch ให้ผลลัพธ์เป็น filename ที่ไม่ใช่ wildcard การทำงานของฟังก์ชัน getfilename และ fsearch เป็นดังรูปที่ 1 และ 2

การอ่านแฟ้มข้อมูล

เมื่อเปิดแฟ้มข้อมูลได้แล้วจะทำการอ่านรหัสซึ่งจะแสดงว่าเป็นรูปแบบการเก็บข้อมูลของโปรแกรมนี้ ถ้ารูปแบบการเก็บตรงกันจะทำการอ่าน กั้นซ้าย,กั้นขวา และข้อมูลตามลำดับ เมื่อครบหนึ่ง paragraph ก็อ่านกั้นซ้าย,กั้นขวา และข้อมูลของ paragraph ต่อ ๆ ไปจนหมดทั้งแฟ้มข้อมูล

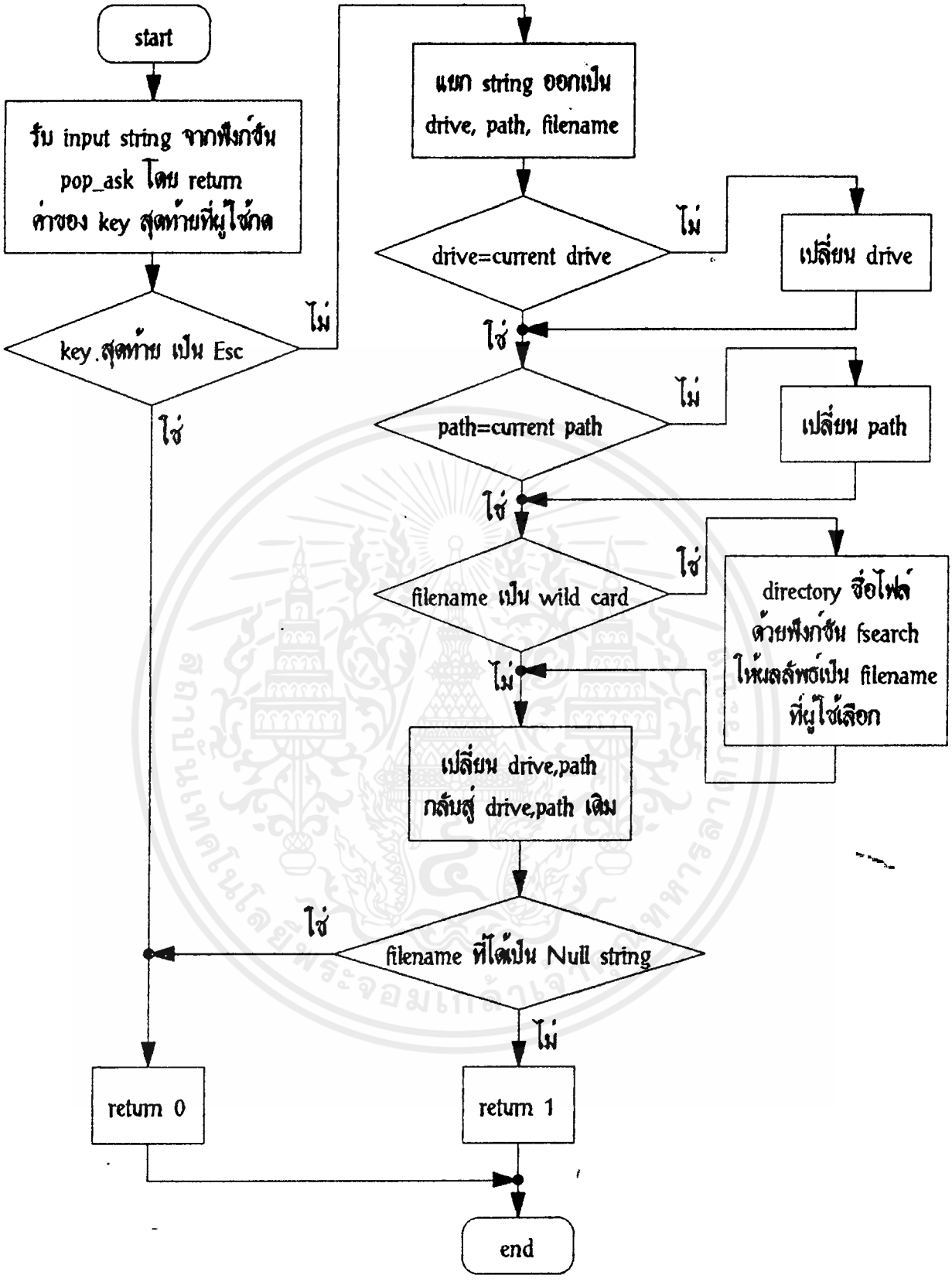
ถ้ารูปแบบการเก็บข้อมูลไม่ตรงกับรูปแบบของโปรแกรมนี้ จะถามคำยืนยันจากผู้ใช้ว่าจะให้อ่านข้อมูลหรือไม่ถ้ายืนยันให้อ่านจะตั้งกั้นซ้าย,กั้นขวา เป็น 1 และ 75 และอ่านข้อมูลเป็น paragraph เดียวโดยตลอด

ในรูปที่ 3 แสดง flow chart ของการอ่านแฟ้มข้อมูล

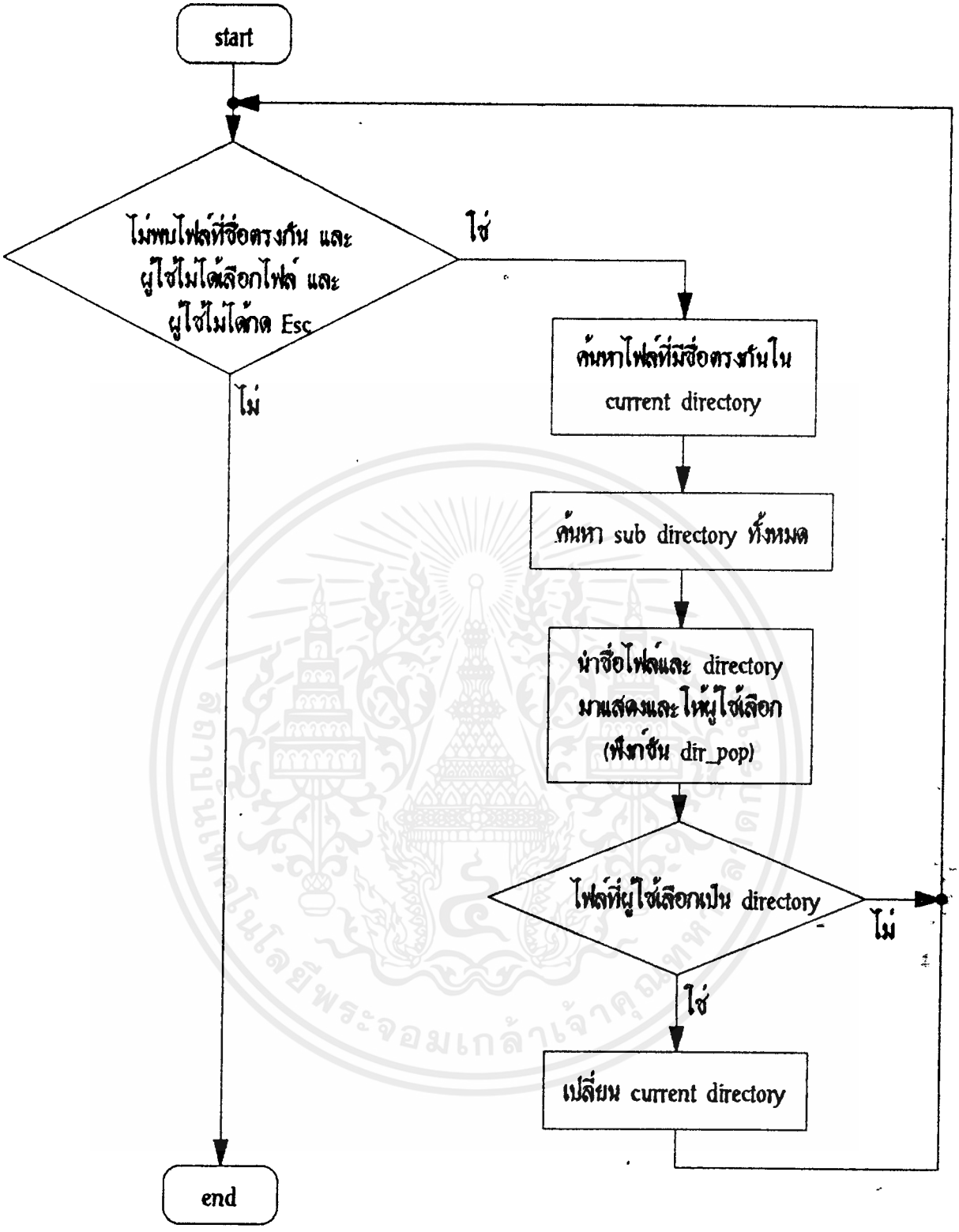
การจัดเก็บแฟ้มข้อมูล

เมื่อเก็บแฟ้มข้อมูลจะเก็บรหัสซึ่งแสดงว่าเป็นรูปแบบการเก็บข้อมูลของโปรแกรมนี้ ขึ้นบรรทัดใหม่ เก็บค่ากั้นซ้ายเป็นสตริง ขึ้นบรรทัดใหม่ เก็บค่ากั้นขวา ขึ้นบรรทัดใหม่ แล้วเก็บค่าข้อมูลใน paragraph นั้นต่อเนื่องกันไปจนหมด paragraph แล้วขึ้นบรรทัดใหม่และเริ่มเก็บตั้งแต่กั้นซ้ายของ paragraph ต่อไป จนหมดแฟ้มข้อมูล

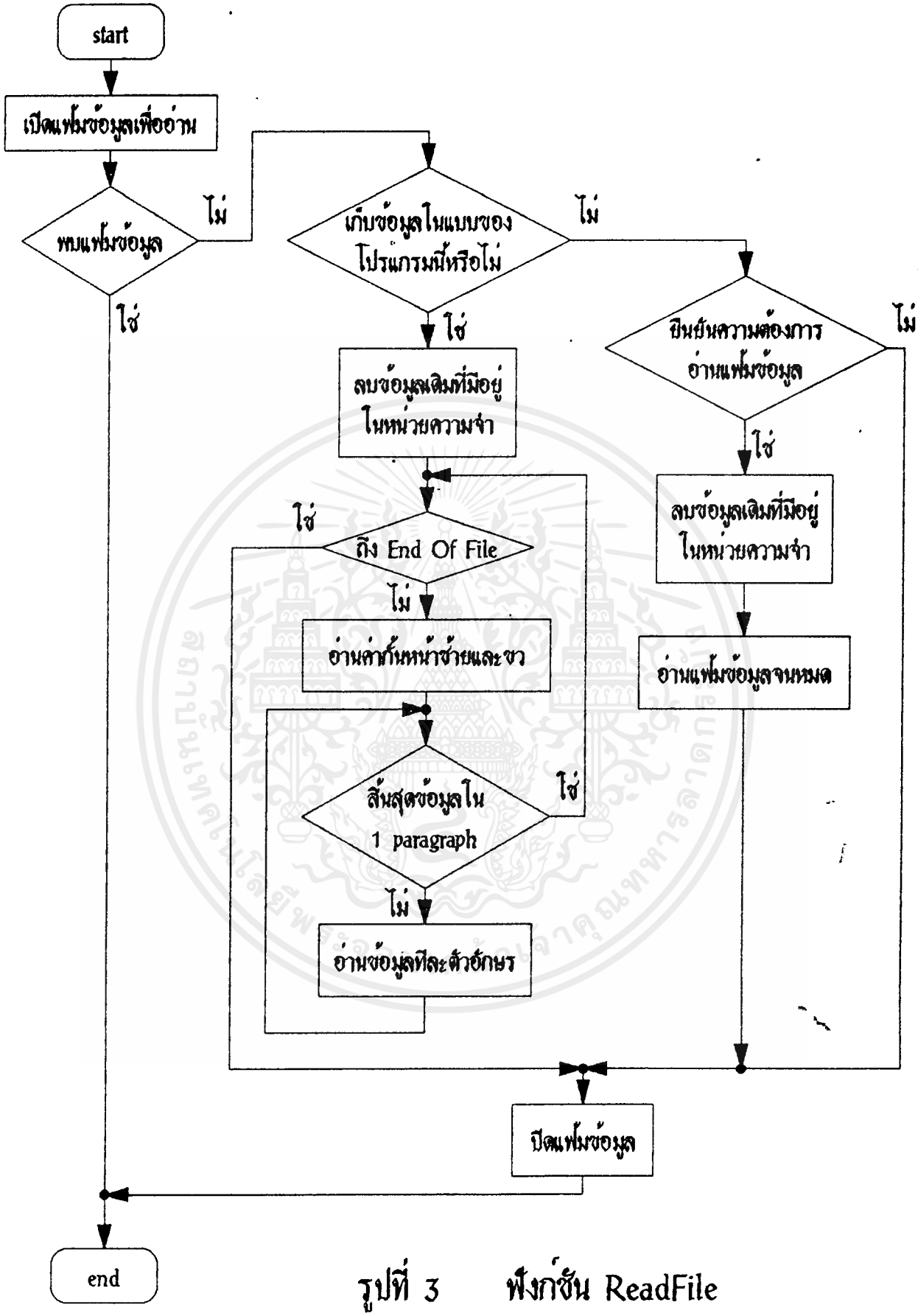
ในรูปที่ 4 แสดง flow chart ของการเก็บแฟ้มข้อมูล



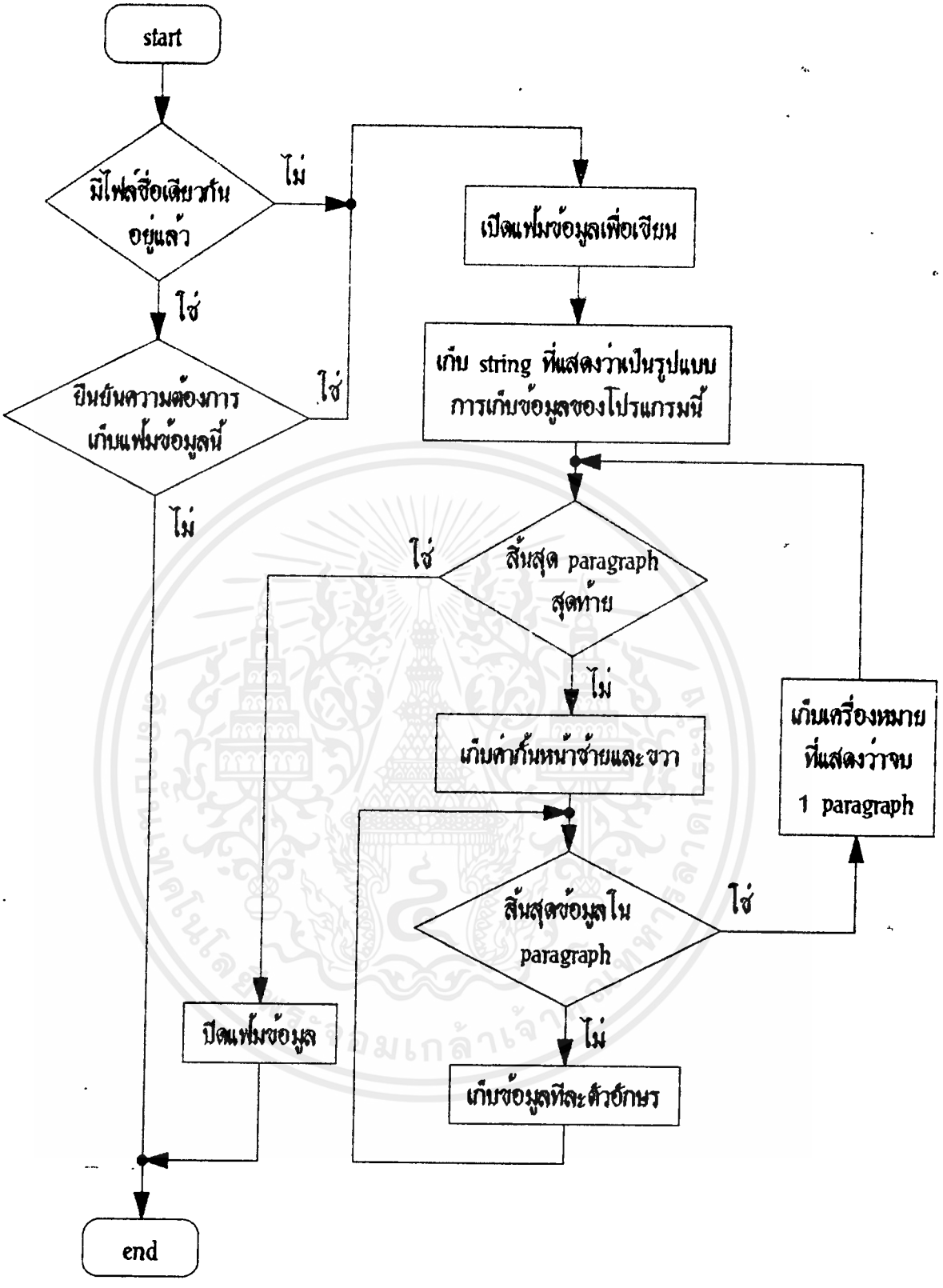
รูปที่ 1 ฟังก์ชัน getfilename



รูปที่ 2 ฟังก์ชัน fsearch



รูปที่ 3 ฟังก์ชัน ReadFile



รูปที่ 4 ฟังก์ชัน SaveFile

กิตติกรรมประกาศ

ทางผู้จัดโครงงานโปรแกรมประมวลผลคำภาษาไทยขอขอบพระคุณอาจารย์ที่ปรึกษา ดร.บุญธีร์ เครือตราชู ที่ให้คำแนะนำที่มีประโยชน์และผลักดันโครงงานนี้อย่างต่อเนื่อง ขอขอบคุณ ดร. ศุภมิตร จิตตะยโสธร ที่แนะนำวิทยานิพนธ์การตัดคำภาษาไทย

ขอขอบคุณบิดามารดาของผู้จัดทำที่ให้กำลังใจในการทำงานนี้ตลอดมา และขอขอบคุณ คุณเทอดศักดิ์ ลิ่วหาทอง ผู้เอื้อเฟื้อข้อมูลและบุคคลอื่นๆที่มีส่วนช่วยให้โครงงานนี้เป็นไปได้ด้วยดี

บรรณานุกรม

TERRY DETTMANN, DOS PROGRAMMER'S REFERENCE 3rd EDITION QUE CORPORATION ค.ศ.1992

KENT PORTER, STRETCHING TURBO C PRENTICE HALL TRADE ค.ศ.1989

GEORGE SUTTY, AND STEVE BLAIR, PROGRAMMEN'S GUIDE TO THE EGV/VGA PRENTICE HALL TRADE ค.ศ. 1988

BORLAND INTERNATIONAL COROPERATION, TURBO C 2.0 REFERENCE AND USER'S GUIDE

ทวีศักดิ์ กออนันตกูล, คอมพิวเตอร์กับภาษาไทย ศูนย์เทคโนโลยีอิเล็คทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) กระทรวงวิทยาศาสตร์เทคโนโลยีและการพลังงาน บริษัท เอ.อาร์.อินฟอร์เมชัน แอนด์ พับลิเคชัน จำกัด พิมพ์ครั้งที่ 1 2534

บริษัท สหวิริยาอินเตอร์แนชชั่นแนลคอมพิวเตอร์จำกัด, คู่มือการใช้งานเครื่องพิมพ์ 9 เข็มพิมพ์ของเอปสัน บริษัท เอ.อาร์.อินฟอร์เมชัน แอนด์ พับลิเคชัน จำกัด พิมพ์ครั้งที่ 2 2533

พงษ์ระพี เตชพาหพงษ์, แอดวานซ์ เอ็มเอสดอส สำนักพิมพ์ซีเอ็ดยูเคชัน พ.ศ.2533

F.S. HILL, JR, COMPUTER GRAPHICS MAGMILLAN PUBLISHING COMPANY ค.ศ. 1990

ROGER T. STEVENS, ADVANCE FRACTAL PROGRAMMING M&T PUBLISHING INC. ค.ศ. 1990

DON PEARSON, IMAGE PROCESSING Mc GROWHILL ค.ศ. 1991

SURIN CHARNYAPORNPONG, A THAI STYLLABLE SEPARATION ALGORITHM ASIAN INSTITUTE OF TECHNOLOGY BANGKOK THAILAND ค.ศ.1983