การแปลงอักษรเบรลล์

BRAILLE TRANSLATOR

โดย

นายทวนทอง สุวรรณหงษ์

อาจารย์ที่ปรึกษา

รศ.ดร. ครรชิต ไมตรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

การแปลงอักษรเบรลล์

Braille Translate

ผู้จัดทำ

นายทวนทอง สุวรรณหงษ์

.....อาจารย์ที่ปรึกษา

(รศ.ดร. ครรชิต ไมตรี)

การแปลงอักษรเบรลล์

ทวนทอง สุวรรณหงษ์
รศ.ดร.ครรชิต ไมตรี อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

โปรแกรมแปลงอักษรปกติ เป็นอักษรเบรลล์ ได้ศึกษาการอ่านและการเขียนอักษรเบรลล์ของคนตาบอด เพื่อให้สามารถที่จะแปลงข้อมูลจากอักษรปกติ เป็นอักษรเบรลล์ได้อย่างถูกต้อง ตามกฎการอ่านและเขียนของอักษรเบรลล์มากที่สุด โครงงานนี้ได้ศึกษาการประสมคำในรูปแบบต่างๆ ของอักษรเบรลล์ ซึ่งแตกต่างจากอักษรที่จะใช้มากที่สุด ได้มีการสร้าง DICTIONARY ขึ้นมาเพื่อเปรียบเทียบแปลงให้ถูกต้องยิ่งขึ้น โดยจุดประสงค์หลักก็เพื่อให้คนทั่วไปที่ไม่มีความรู้ เกี่ยวกับอักษรเบรลล์ สามารถที่จะพิมพ์หนังสือสำหรับคนตาบอดได้ และนำข้อมูลที่มีอยู่แล้วมาแปลงเป็นอักษรเบรลล์ได้

BRAILLE TRANSLATOR

Tauntong Suwannahong
Associate Professor Kanchit Maitree Advisor
1994

Abstract

The program in transferring print to braille studied the reading and writing of blind people in order to be able to transfer print to braille correctly according to the braille lingustic rule. This project studied various word formation specefic form, and execptional form of braille. The directory was established to correctly campare this transfer. The principal objectives were to enable general person who have no knowledge of braille to type braille books for the blind and transfer existed information in to braille.

คำนำ

ในปัจจุบัน การศึกษานับเป็นสิ่งสำคัญสำหรับมนุษย์ เนื่องจากเป็นเครื่องมือสำคัญในการประกอบอาชีพ การดำรงชีวิตอยู่ในสังคม และสร้างความเจริญก้าวหน้าให้กับตัวเอง การศึกษาตามปกติ ผู้คนส่วนใหญ่จะนึกถึงแต่ตัวอักษรธรรมดา หนังสือทั่วไป สำหรับคนที่สายปกติเท่านั้น ความจริงแล้วมีบุคคลอีกประเภทหนึ่งที่อยู่รวมในสังคม แต่ไม่มีสายตาที่จะใช้เป็นเครื่องช่วยในการอ่าน เขียนตัวอักษรเช่นเดียวกับคนปกติได้ แต่ต้องใช้ประสาทสัมผัสที่มือและนิ้วแทนสายตา เพื่อใช้ในการอ่านและเขียน ซึ่งเรียกอักษรที่ใช้อ่านและเขียนว่าอักษรเบรลล์ สำหรับหนังสือที่เป็นอักษรเบรลล์นั้น มีให้ศึกษาไม่มากนัก สาเหตุเนื่องมาจากขั้นตอนการพิมพ์ที่ยุ่งยากซับซ้อนกว่าอักษรปกติทั่วไป และบุคลากรที่มีความรู้เกี่ยวกับอักษรเบรลล์มีน้อย

โปรแกรมการแปลงอักษรธรรมดา เป็นอักษรเบรลล์นั้น มีประโยชน์อย่างยิ่งสำหรับคนตาบอด เพราะว่าคนทั่ว ๆ ไปที่ไม่มีความรู้เกี่ยวกับอักษรเบรลล์ ก็สามารถที่จะพิมพ์หนังสือสำหรับคนตาบอดได้ หรือสามารถนำข้อมูลของอักษรปกติทั่วไปแปลงเป็นอักษรเบรลล์โดยใช้เวลาเพียงไม่กี่นาที ทำให้ลดขีดจำกัดในการศึกษาความรู้เทคโนโลยีใหม่ๆ และข้อมูลที่ทันสมัยสำหรับคนตาบอดอีกต่อไป เป็นการลดภาระของสังคม และสร้างโอกาสให้กับคนพิการทางตาอีกทางหนึ่งด้วย

20 มีนาคม 2538
ทวนทอง สุวรรณหงษ์
ผู้จัดทำ

สารบัญ

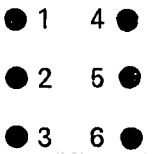
บทที่ 1
บทนำ

อักษรเบรลล์ คือ ระบบการเขียนหนังสือสำหรับคนตาบอด อ่านโดยการสัมผัสด้วยปลายนิ้วมือระบบการอ่านเขียนหนังสือสำหรับคนตาบอดนี้ได้คิดประดิษฐ์โดย หลุยส์ เบรลล์ (Braile 1924)

หลุยส์ เบรลล์ เกิดเมื่อวันที่ 4 มกราคม ค.ศ. 1809 (พ.ศ. 2352) ในเมือง Coupray ในประเทศฝรั่งเศส เขาตาบอดโดยอุบัติเหตุเมื่ออายุ 3 ขวบ ตอนแรกเขาเรียนร่วมในโรงเรียนปกติของคนตาดี แล้วต่อมาเรียนที่สถาบันคนตาบอดแห่งชาติที่ปารีส (L'Institution Nationale des Jeunnes Aveugles) และเมื่อสำเร็จการศึกษาเขาได้ทำงานเป็นครูสอนคนตาบอดที่นั่น เขามีความรู้สึกว่าหากคนตาบอดไม่มีอักษรสำหรับบันทึกข้อความแล้ว การศึกษาจะเป็นไม่ได้ดี ต่อมาเขาได้นำความคิดมาจาก กัปตันชาร์ลส์ บาบิแอร์ นายทหารแห่งกองทัพบกฝรั่งเศส ซึ่งได้นำวิธีการส่งข่าวสาร ทางการทหารในเวลากลางคืน ลองมาใช้กับคนตาบอดดู ระบบนี้ให้รหัส จุด - ขีด - นูน เขียนลงบนกระดาษแข็ง ซึ่งเรียกว่า โซโนกราฟี (Sonography) แม้ระบบนี้ค่อนข้างจะยุ่งยากแต่ หลุยส์เบรลล์ เห็นคุณค่าตั้งแต่นั้นมาจนถึงปัจจุบัน โดยที่ยังไม่เคยมีใครดัดแปลงให้เหมาะแก่การสัมผัสด้วยปลายนิ้วมือ หรือพัฒนาขึ้นอีกเลย นอกจากนำจุดอักษรนั้นมาปรับปรุงดัด แปลงให้เป็นระบบภาษาของตัวเอง เช่น การเขียนอักษรนูนเป็นภาษาไทย จึงนับว่า หลุยส์ เบรลล์ เป็นบุคคลสำคัญอย่างยิ่งที่เป็นผู้เริ่มบุกเบิกวิเริ่มการศึกษาของคนตาบอด ทำให้คนตาบอดได้มีโอกาสได้ศึกษาเล่าเรียน ทั้งในโรงเรียนและศึกษาด้วยตนเองเช่นเดียวกับเด็กปกติ เพื่อเป็นเกียรติแก่ท่าน อักษรนูนที่คนตาบอดใช้ จึงเรียกว่า อักษรเบรลล

ลักษณะและตำแหน่งของจุดในอักษรเบรลล์

อักษรเบรลล์ ซึ่งเป็นอักษรที่คนตาบอดใช้ในการเขียนและอ่านนั้น ประกอบขึ้นด้วยจุด ๖ จุด เรียงกันเป็นสอบแถวตามแนวตั้งตามลำดับ ดังนี้



การอ่านจุดทั้ง ๖ นี้ จุดจะเรียงแถวละ ๓ จุด ๒ แถว การอ่านจะนับจากข้างบนลงมาข้างล่าง ซึ่งมีผลในการฝึกการเขียนอ่านเบรลล์ ในตอนต่อไป

● 1 4 ●

● 2 5 ● แถวซ้ายเรียงจากบนลงล่างเรียก จุด 1,2,3

● 3 6 ● แถวขวาเรียงจากบนลงล่างเรียก จุด 4,5,6

การนำจุดต่างๆ นี้มาจัดกลุ่มกันเป็นรหัสเช่นเดียวกับในวิชาคณิตศาสตร์ที่เราเรียกการจัดกลุ่มเช่นนี้ว่า คอมบิเนชั่น (Combination) จากการใช้สูตรทางคณิตศาสตร์หรือจากการนำจุดหก จุดนี้มาจัดกลุ่มจริง ๆ เราจะได้ถึง ๖๓ กลุ่ม ซึ่งสามารถนำไปใช้แทนตัวอักษรของภาษาไทย สัญญลักษณ์ ทางคณิตศาสตร์ วิทยาศาสตร์ ดนตรี และเครื่องหมายต่างๆ ได้

ในระยะแรก ๆ จากการศึกษาของอักษรเบรลล์ อาจจะสับสนกันอยู่บ้าง แต่ ถ้าได้มีโอกาสเขียนหรืออ่านบ่อย ๆ จะช่วยให้เรียนรู้ได้เร็วขึ้นและจะไม่รู้สึกยากนัก

● 1 4 ●

● 2 5 ●

● 3 6 ●

จากภาพที่แสดงข้างบนนี้ จะเห็นลักษณะและตำแหน่งของจุดต่างๆ ซึ่งมีรายละเอียดดังต่อไปนี้

จุดที่ 1 คือ จุดบนซ้าย

จุดที่ 2 คือ จุดกลางซ้าย

จุดที่ 3 คือ จุดล่างซ้าย

จุดที่ 4 คือ จุดบนขวา

จุดที่ 5 คือ จุดกลางขวา

จุดที่ 6 คือ จุดล่างขวา

จุด 6 จุดของอักษรเบรลล์นี้ เรียกว่า 1 กลุ่ม

จุด 6 จุดของอักษรเบรลล์นี้มี 2 ชุด เรียกว่า 2 กลุ่ม

ดังนั้น ในการเขียนอักษรเบรลล์ต่อไปนี้จะเรียก "กลุ่ม" แทนจุด 5 จุด

เมื่อได้ทราบถึงตำแหน่งของจุดทั้ง 6 จุดนี้แล้ว ก็ใช้ประกอบกันเข้าทำให้เกิดเป็นตัวพยัญชนะ สระ รวมทั้งเครื่องหมายต่างๆ ที่ใช้ในภาษาไทย โดยยึดข้อแตกต่างของจำนวนจุด และตำแหน่งของจุดเหล่านี้เป็นสำคัญ

ในการเขียนหรืออ่านอักษรเบรลล์ จำเป็นต้องเรียงตามลำดับก่อนหลังจากตำแหน่งของจุดต่างๆ ตามที่กำหนดไว้ ทำให้ไม่เกิดการสับสน

บทที่ 2
การอ่านเขียนอักษรเบรลล์

กำเนิดอักษรเบรลล์ไทย

นางสาวเจเนวีฟ คอลฟิลด์ สุภาพสตรีตาบอดชาวอเมริกัน ซึ่งเป็นคนก่อตั้งโรงเรียนสอนคนตาบอดขึ้นในประเทศไทย เมื่อ พ.ศ.2482 เป็นผู้ให้คำแนะนำในการปรับปรุงและดัดแปลงการเขียนอักษรเบรลล์ไทยครั้งแรก โดยอาศัยรากฐานการเขียนเบรลล์ ในภาษาอังกฤษที่มีเสียงเหมือนกันทั้งภาษาไทยและภาษาอังกฤษไว้ดังนี้

ก - G

ข - K

จ - J

ด - D

ถ - T

น - N

ผ - P

ม - M

ย - Y 

ร - R 

ล - L 

ว - W 

ส - S 

ห - H 

อ - O

จากตัวอย่างข้างบนนี้ เป็นการแสดงให้เห็นที่มา ของการเขียนอักษรเบรลล์ไทย ซึ่งได้รับอิทธิพลมาจากการเขียนเบรลล์ในภาษาอังกฤษด้วย ดังนั้น จึงทำให้มีปัญหาบางประการในการเขียนอักษรเบรลล์ไทย เพราะอักษรเบรลล์ไทยไม่สามารถเขียนสระหรือวรรณยุกต์ที่อยู่ข้างบนหรือล่าง ของตัวอักษรตามอักษรของอักษรไทยตามปกติได้ เพราะถูกจำกัดด้วย เนื้อที่ของวิธีการเขียนเบรลล์ และไม่สะดวกต่อการอ่านของคนตาบอดที่ต้องสัมผัส เพราะการอ่านเบรลล์ เป็นการอ่านไปตามแนวนอน จากซ้ายไปขวา เช่นเดียวกับการอ่านหนังสือของคนปกติ แต่ก็เป็นปัญหาในการเขียนและอ่านเบรลล์สำหรับคนตาบอดที่จะอ่านหนังสือคนปกติ คนตาบอดจึงใช้หนังสือของคนปกติไม่ได้ต้องใช้อ่านและเขียนด้วยเบรลล์แทน

หนังสือสำหรับเด็กปกตินั้น ในที่นี้จะใช้คำที่ตรงกับภาษาอังกฤษว่า inkprinted หรือ Printed เพื่อมิให้เกิดการสับสนในการกล่าวถึง การเขียนเบรลล์กับตัวพิมพ์ของคนปกติ

การประสมคำ

1. เมื่อเด็กเขียนได้ทั้ง พยัญชนะ สระ วรรณยุกต์แล้ว จึงจะประสมคำง่ายๆ การประสมคำจะต้องประกอบด้วย พยัญชนะ สระ ตัวสะกด วรรณยุกต์ ตามหลักภาษาศาสตร์ ใช้ดังนี้

พยัญชนะต้น ใช้คำย่อว่า C (consonant)

สระ ใช้คำย่อว่า V (vowel)

ตัวสะกด ใช้คำย่อว่า C (consonant)

วรรณยุกต์ ใช้คำย่อว่า T (tone)

ดังนั้น เมื่อประสมคำแล้วจะเป็น CVCT

1.1 พยัญชนะต้น + สระ

เช่น มา 
ม า

1.2 พยัญชนะต้น + สระ + ตัวสะกด

เช่น กิน 
ก อิ น

1.3 พยัญชนะต้น + สระ + ตัวสะกด + วรรณยุกต์

เช่น อิ่ม 
อ อิ อ่ ม

2. สระต่อไปนี้ จะเขียนอยู่หลังพยัญชนะเสมอ ได้แก่ -ะ , -า, อิ, อี, อึ, อื, อุ, อู, เ-ะ, แ-ะ, โ-ะ , เ-าะ, -อ, อัวะ, อัวฐ เอียะ, เอีย, เอื่อะ, เออ, เ-อ, อำ, เ-า และเมื่อคำที่ใช้สระเหล่านี้มรูปวรรณยุกต์จะเขียนไว้หลังสระ ยกเว้น สระอา กับ สระออ ที่ใช้รูปวรรณยุกต์ก่อนสระ

ตัวอย่าง

มะลิ 
ม ะ ล อิ

ปูนา 
ป อู น า

สีดำ 
ส อี ด ำ

เสาผุ 
ส เ-า ผ อุ

ข้อยกเว้นสระ อา กับสระออ

จำ จ อ ำ

กว่า ก ว อ่ า

3. สระต่อไปนี้จะเขียนอยู่หน้าพยัญชนะ ได้แก่ เ, แ, โ, ใ, ไ

การเขียนเบรลล์จะวางรูปคำเหมือนกับการเขียนปกติ

ตัวอย่าง

เวลา เ ว ล า

โต โ ต

กาแฟ ก า แ ฟ

ใบไม้ ใ บ ไ ม อ้

ใช้งาน ใ ช อ้ ง า น

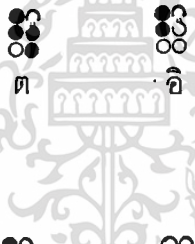
แม่พ่อ แ ม อ่ พ อ่ อ

ข้อสังเกต ถ้าเราดูเผิน ๆ จะเห็นได้ว่า มีสระบางตัวมีรูปเหมือนกันจะต่างกันที่ตำแหน่งเท่านั้น เช่น

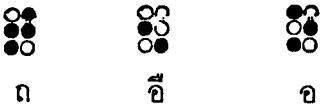
สระอิ 

สระอี 

สัญญลักษณ์ เหล่านี้เกิดการสับสนได้ง่าย ดังนั้น สระของการเขียนยเบรลล์ที่ยกตัวอย่างมาจะไม่เขียนเดี่ยวๆ เพราะทำให้เกิดการสับสนได้ง่าย แต่เมื่อนำไปประสมคำเราสามารถเปรียบเทียบกับตัวข้างเคียงได้ เช่น

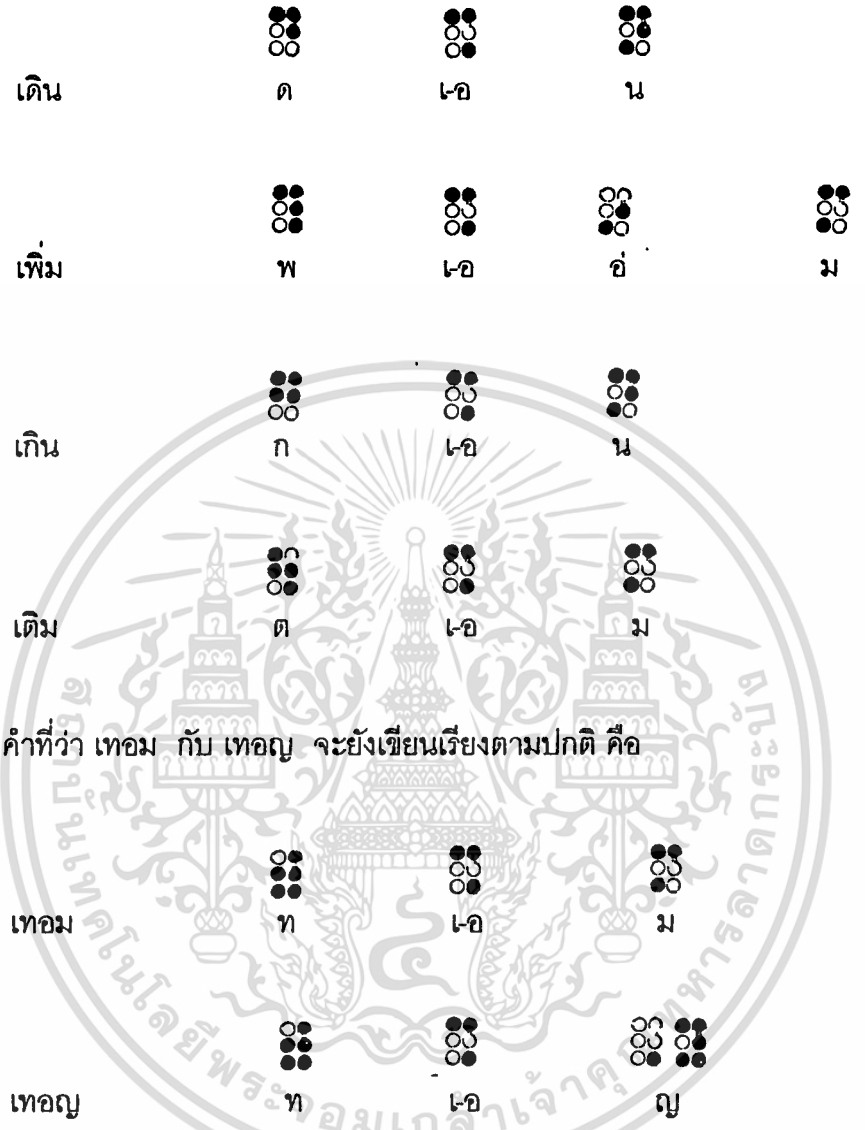


4. เราจะวาง -อ ไว้หลัง อี เมื่อคำนั้น ไม่มีตัวสะกด เช่น



5. คำที่ใช้ เ-อ และมีตัวสะกดอยู่ด้วย ซึ่งโดยทั่วไปจะเปลี่ยนรูป เป็น เอิ- แต่ในอักษรเบรลล์ยังคงรูป สระ เ-อ ไว้เช่นเดิม เช่น

ปกติ อักษรเบรลล์



ยกเว้น คำที่ว่า เทอม กับ เทอญ จะยังเขียนเรียงตามปกติ คือ

ข้อสังเกต อักษรเบรลล์ มีลักษณะการผสมที่อาจจะแตกต่างจากอักษรปกติหลายประการด้วยเหตุนี้ จึงควรสอนการประสมคำในอักษรปกติให้แก่เด็กที่เรียนรู้อักรเบรลล์จนแตกฉานแล้วควบคู่ไปด้วย เพื่อเป็นพื้นฐานในการเรียนพิมพ์ดีดสัมผัสของคนปกตินั้น จะได้เป็นประโยชน์ และจำเป็นอย่างมากสำหรับการศึกษาในระดับสูงของเด็กตาบอด ซึ่งปัจจุบันมีเด็กตาบอดเรียนร่วมกับเด็กปกติ จบระดับปริญญาตรีแล้วจำนวนหลายคน

เนื่องจากเด็กตาบอดไม่สามารถมองเห็นเว้นวรรค หรือเขียนแสดงความรู้สึกได้เช่นเดียวกับเด็กปกติ จึงต้องใช้เครื่องหมายต่างๆ แทน

เครื่องหมายต่าง ๆ ในภาษาไทย

เครื่องหมายปกติ การเขียนโดยใช้จุดต่างๆ จะเขียนเป็นเบรลล์ได้ดังนี้

ๆ ไม้ยมก จุด 2 

อ์ การันต์ จุด 3 - 4 - 6 

. มหัพภาค จุด 2 - 5 - 6 

! อัศเจรีย์ จุด 2 - 3 - 5 

คือเครื่องหมายตกใจ ตัวอย่าง อุ๊ย !

? ปรัศนี คือ จุด 2 - 3 - 6 

เครื่องหมายคำถาม ตัวอย่าง ใคร?

" บุดสัญญา คือ จุด 5 - 2 

เครื่องหมาย ละ ตัวอย่าง อาวรณ์ แปลว่า ห่วงใย

(-) นขลิขิต คือ จุด 2 - 3 - 5 - 6 , 2 - 3 - 5 - 6 

วงเล็บ ตัวอย่าง อาวุโส (อายุ) 

"....." อัญประกาศ คือ จุด 2 - 3 - 6 , 2 - 3 - 6 

ฯ ไปยาลน้อย จุด 5 - 6 , 2 - 3 

ฯลฯ ไปยาลใหญ่ จุด 5 - 6 , 1 - 2 - 3 

/ เครื่องหมายทับ จุด 3 - 4 

- ยติภังค์ จุด 3 - 6 

จะเห็นว่าในการใช้เครื่องหมายบางตัวอาจจะซ้ำกับสระก็มี เช่น สระเอา กับ เครื่องหมายอัศเจรีย์ ! ที่ใช้จุดเดียวกัน จึงทำให้เกิดการสับสนได้ง่าย ดังนั้น เพื่อมิให้เกิดการสับสน จึงจะต้องมีหลักการใช้เครื่องหมายต่างๆ ดังนี้

1. เมื่อต้องการใช้เครื่องหมายอัศเจรีย์ ! หรือปรัศนีย์ ? เราจะเขียนครื่อง หมายติดกับประโยค โดยใช้จุด 4 - 5 - 6 นำหน้าเครื่องหมายอัศเจรีย์ และปรัศนีย์ เสมอ และให้เว้นหนึ่งช่องเพื่อขึ้นประโยค ใหม่ เพื่อให้มองเห็นได้ชัดเจน ขอยก ตัวอย่างในการใช้เครื่องหมายต่างๆ ดังนี้

2. เครื่องหมาย ไปยาลน้อย ฯ ให้เขียนต่อกับคำที่ละไว้ ตัวอย่าง กรุงเทพฯ

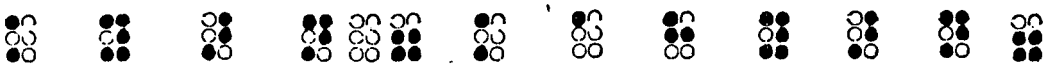
3. เครื่องหมาย ไปยาลใหญ่ ฯลฯ ก่อนใช้เครื่องหมายนี้ ต้องเว้นเครื่องหมาย หนึ่งช่อง และเว้นหลังเครื่องหมายหนึ่งช่อง

หมายเหตุ จุด 6 จุด แสดงการเว้นวรรค

4. เครื่องหมายวงเล็บ (...) และเครื่องหมายคำพูด "...." ก่อนใช้ เครื่องหมายนี้ต้องเว้นหน้าเครื่องหมายหนึ่งช่อง และเว้นหลังเครื่องหมายหนึ่งช่อง ข้อความในวงเล็บ หรือเครื่องหมายคำพูดให้เขียนติดกับเครื่องหมายได้เลย

ตัวอย่าง "ตัวเรากับการกิน"

ตัวอย่าง ขยัน (ขะหยัน)



5. ส่วนเครื่องหมายยติภังค์ - และ / เราไม่เว้นช่องหน้าหลังเครื่องหมาย เช่นเดียวกับ (-) และ "...." ถ้ามีตัวเลขระหว่างเครื่องหมายตัวเลขที่อยู่หลังเครื่องหมายนี้ ไม่ต้องมีเครื่องหมายนำเลข (3 - 4 - 5 - 6)นำหน้า แต่ถ้าอยู่คนละบรรทัดก็จำเป็นต้องใส่เครื่องหมายนำเลข

ตัวอย่าง 25 - 26

ใช้เลขต่ำ 2 5 - 2 6

ใช้เลขสูง 2 5 - 2 6

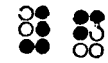
สัญญลักษณ์เบรลล์ทางคณิตศาสตร์

จำนวนเลข

ตัวเลขที่เราใช้กันโดยทั่วไป มีตั้งแต่เลข 1 - 9 และ 0 เราได้นำตัวเลขเหล่านี้มาผสมผสานได้จำนวนเลขมากมาย ในการเขียนจำนวนตัวเลขในอักษเบรลล์ เราต้องนำด้วยเครื่องหมายนำเลขคือ

จุด 3-4-5-6 

นำหน้าตัวเลขที่มีลักษณะคล้ายกับตัวอักษรภาษาอังกฤษ จึงจะได้จำนวนเลขได้แก่

จำนวนเลข	ในการเขียนใช้จุด	อักษรเบรลล์
1. การเขียนตัวเลขทั่วไปที่มีระดับสูง		
1	จุด 3-4-5-6,1	
2	จุด 3-4-5-6,1-2	
3	จุด 3-4-5-6,1-4	
4	จุด 3-4-5-6,1-4-5	
5	จุด 3-4-5-6,1-5	
6	จุด 3-4-5-6,1-2-4	

7 จุด 3-4-5-6,1-2-4-5 

8 จุด 3-4-5-6,1-2-5 

9 จุด 3-4-5-6,2-4 

0 จุด 3-4-5-6,2-4-5 

ตัวอย่าง เลขตั้งแต่ 2 หลักขึ้นไปจะต้องมีเครื่องหมายนำเลข แล้วเลขหลักต่างๆ เขียนได้ดังนี้

15 จุด 3-4-5-6,1,1-5 

เครื่องหมายนำเลข 1 5

208 จุด 3-4-5-6,1-2,2-4-5,1-2-5 

เครื่องหมายนำเลข 2 0 8

1080 จุด 3-4-5-6,1,2-4-5,2-4-5 

เครื่องหมายนำเลข 1 0 8 0

ในทางคณิตศาสตร์ เราจะใช้ตัวเลขที่แตกต่างจากข้างบนเพื่อไม่ให้เกิดการสับสนในการเรียนคณิตศาสตร์ชั้นสูง ซึ่งจะต้องมีเรื่องตัวหนังสือ ตัวอักษรภาษาอังกฤษเครื่องหมายที่ปนอยู่กับตัวเลข ลักษณะของตัวเลขที่ใช้ในทางคณิตศาสตร์เฉพาะจะมีรูปเหมือนเลขที่กล่าวมาแล้ว แต่จะอยู่ในระดับที่ต่ำกว่า ดังนี้

เลขในระดับต่ำ

จำนวนเลข	ในการเขียนใช้จุด	อักษรเบรลล์
1	จุด 3-4-5-6,2	
2	จุด 3-4-5-6,2-3	
3	จุด 3-4-5-6,2-5	
4	จุด 3-4-5-6,2-5-6	
5	จุด 3-4-5-6,2-6	
6	จุด 3-4-5-6,2-3-5	
7	จุด 3-4-5-6,2-3-5-6	
8	จุด 3-4-5-6,2-3-6	
9	จุด 3-4-5-6,3-5	
0	จุด 3-4-5-6,3-5-6	

ในการเขียนตัวจำนวนหลักมาก เราจะต้องมีการใช้เครื่องหมายจุด 6 กั้นตัวเลขไว้ จะเห็นได้ว่า การเขียนตัวเลข ไม่ใช่เว้นวรรคเหมือนปกติเพราะเด็กมองไม่เห็น จึงใช้เครื่องหมายจุลภาคเป็นจุดที่ 6 แทน

ตัวเลข	อักษรเบรลล์
1,2,3	 1 , 2 , 3
15,234	 1 5 , 2 3 4

เครื่องหมายทางคณิตศาสตร์

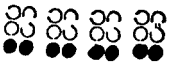
1. เครื่องหมายบวก ใช้จุด 3-4-6 + หลังเครื่องหมายบวก ตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง 2+3

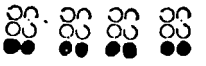

2 + 3

2 

+3 

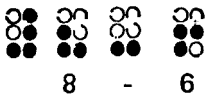
—— 

5 

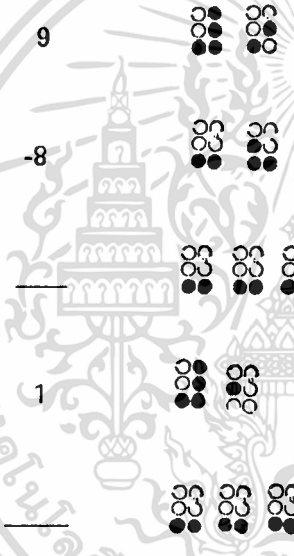
—— 

2. เครื่องหมายลบ ใช้จุด 3-6 ⠤ หลังเครื่องหมายลบ ตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง 8-6

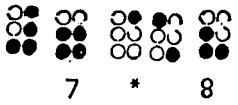


ตัวอย่าง 9-8



3. เครื่องหมายคูณ ใช้จุด 4,1-6 ⠔ หลังเครื่องหมายคูณตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง 7*8



7

*8

——

56

——

4. เครื่องหมายหาร ใช้จุด 4-6,3-4 / หลังเครื่องหมายหาร ตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง 16/2

1 6 / 2

ตัวอย่าง 142/14

1 4 2 / 1 4

หมายเหตุ เครื่องหมาย) ใช้จุด 1-3-5 หลังเครื่องหมาย ตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง

5

5)25

25

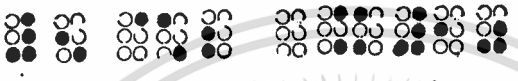
——

5. เครื่องหมายเท่ากับ จุด 4-6,1-3  ต้องเว้นช่องข้างหน้าและหลังเครื่องหมาย

ตัวอย่าง 8+7=15

8 + 7 = 15

ตัวอย่าง 5*2=10



5 * 2 = 10

6. เครื่องหมายมากกว่า > จุด 4-6,2  ต้องเว้นข้างหน้าและหลังเครื่องหมาย

ตัวอย่าง 7>2

7 > 2

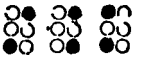
7. เครื่องหมายน้อยกว่า < จุด 5,1-3  ต้องเว้นช่องหน้าและหลังเครื่องหมาย

ตัวอย่าง 2<7

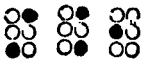
2 < 7

8. ส่วนเครื่องหมายปฏิเสธ คือ "ไม่" ให้ใช้จุด 3-4  นำหน้าเครื่องหมายนั้น ๆ เสมอ เช่น

≠ เครื่องหมายไม่เท่ากัน

ใช้จุด 3-4,1-3 

เครื่องหมายไม่มากกว่า

ใช้จุด 3-4,4-6,2 

เครื่องหมายไม่น้อยกว่า

ใช้จุด 3-4,5,1-3 

วิธีใช้ต้องเว้นช่องข้างหน้าและหลังเครื่องหมาย

ตัวอย่าง 2+3 9

2 + 3 ≠ 9

9. เครื่องหมายเพราะฉะนั้น ใช้จุด 6,1-6 

เวลาใช้ต้องเว้นวรรคข้างหน้าและหบังเครื่องหมาย

10. เครื่องหมายถ้าเช่นนั้น ใช้จุด 4,3-4 

เวลาใช้ต้องเว้นวรรคข้างหน้าและหลังเครื่องหมาย

11. เครื่องหมาย ทศนิยม . ใช้จุด 4-6 

หลังเครื่องหมายนี้ตัวเลขไม่ต้องมีเครื่องหมายนำเลข

ตัวอย่าง 3.14

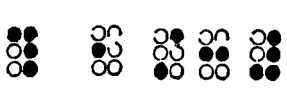
3 . 1 4

12. เครื่องหมาย เศษส่วน เครื่องหมายนำเศษ จะใช้จุด 1-4-5-6 

และเครื่องหมายนำส่วนคือ จุด 3-4  หลังเครื่องหมายนี้ ตัวเลขไม่ต้อง

มีเครื่องหมายนำเลข และจบลงด้วย จุด 3-4-5-6 

ตัวอย่าง 1/3 เขียนเศษส่วนได้ดังนี้



เครื่องหมายเศษ 1 ส่วน 3 เครื่องหมายปิดเศษส่วน

ส่วนการเขียนเศษส่วนในรูปจำนวนคละ จะต้องเป็นเลขจำนวนเต็ม แล้วตามด้วยรูปเศษส่วน แต่ก่อนจะนำด้วยรูปเศษส่วน เราจะนำด้วยเครื่องหมาย จุด 4-5-6 แล้ว  จึงต่อด้วยเครื่องหมายนำเศษ เครื่องหมายนำส่วน และจบด้วย

จุด 4-5-6,3-4-5-6 

ตัวอย่าง 4 3/5

จำนวนเต็ม 4 เศษ 3 ส่วน 5 เครื่องหมายปิดเศษส่วนคละ

ได้พัฒนาโปรแกรมให้สามารถแสดงผลได้ทั้งจอภาพแบบวีจีเอ อีจีเอ และ เฮอคิวลิส แต่ในช่วงแรกนี้ใช้เฉพาะส่วนของจอวีจีเอขึ้นไปเท่านั้น หน่วยความจำแสดงผลของจอภาพแบบวีจีเอจะเริ่มที่แอดเดรส A000:0000 และเรียงต่อกันไปเรื่อยๆ ฟังก์ชั่นแสดงผลนี้ชื่อว่า vscreen โดยมีอาร์กิวเมนต์เป็นตำแหน่ง X,Y บนจอภาพและข้อมูลที่ต้องการแสดงขนาด 1 ไบต์ (ตำแหน่ง X บนจอภาพจะเท่ากับ getmaxx()/8)สำหรับ Mode จะใช้กำหนดว่าจะให้แสดงผลแบบ XOR หรือ FILL ซึ่งแบบหลังจะเป็นการเขียนข้อมูลทับเลย แต่แบบแรกจะเป็นการทำ exclusive or กับตัวอักษรเดิมจะใช้สำหรับการแสดงตัวพยัญชนะกับสระบนวรรณยุกต์และสระล่าง

```
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 += (test_scan -1) *0x2000 +((int)(y/4))+0x5A;
    dis_p = (long) Row_v_ram +(long)(x);
}else{ /*จอภาพสี */
```

บทที่ 3
การแสดงผลตัวอักษรภาษาไทย-อังกฤษและอักษรเบรลล์

การแสดงผลตามปกติของเครื่องคอมพิวเตอร์ในโหมดตัวอักษร จะใช้ตัวอักษรที่สร้างไว้ในรอมของการ์ดแสดงผลสำหรับแสดงผลทางจอภาพ สำหรับการแสดงผลตัวอักษรภาษาไทยมักจะมีสองแนวทางหลักๆ ที่ใช้คือ แนวทางแรก จะแก้ไขที่ส่วนฮาร์ดแวร์ของการ์ดแสดงผลโดยตรงเพื่อให้สามารถแสดงผลภาษาไทยได้ วิธีนี้จะทำให้โปรแกรมภาษาไทยจะต้องผูกติดกับผู้ผลิตการ์ดนั้น ๆ อีกวิธีหนึ่งคือใช้เทคนิคทางซอฟต์แวร์ในการจัดการแสดงผล โดยสร้างตัวอักษรเป็นรูปกราฟฟิกและแสดงผลเหมือนกับเป็นรูปกราฟฟิกธรรมดา ซึ่งวิธีนี้ใช้ในโครงงานนี้ สำหรับการ์ดแสดงผลตั้งแต่ VGA ขึ้นไปยังมีวิธีแสดงผลอีกรูปแบบหนึ่งที่กำลังนิยมใช้กันมากคือการแปลี่ยน Interrupt Vecter ที่ชี้ไปยังตารางตัวอักษรบนการ์ดแสดงผลใช้ชี้มายังตารางตัวอักษรที่เราออกแบบแทน วิธีนียังมีข้อเสียตรงที่จะต้องใช้กับระบบแสดงผลวีจีเอขึ้นไปเท่านั้น

การแสดงผลตัวอักษรในโหมดกราฟฟิก

การแสดงผลในโหมดกราฟิกจะใช้การเขียนจุดลงไปบนจอภาพให้เป็นรูปของตัวอักษรขึ้นมา ตัวอักษรหนึ่งตัวก็จะใช้จุดจำนวน 8*20 สาเหตุที่ใช้ความสูงถึง 20 จุดก็เพื่อให้สามารถแสดงผลแบบสี่ระดับได้เลยรูปตัวอักษที่เป็น Bit Map จะเก็บอยู่ในไฟล์ NORMAL.FON และไฟล์ Braille ในหนึ่งตัวอักษรจะใช้ข้อมูล 20 ไบต์ โดยไฟล์จะเก็บเรียงเป็นกลุ่มละ 20 ไบต์ตามรหัสตัวอักษรแบบ สมอ. เช่น ข้อมูลในไฟล์ 20 ไบต์แรกก็จะเป็นภาพของตัวอักษรรหัส OOH ข้อมูล 20 ไบต์ต่อไปก็จะเป็นของตัวอักษรรหัสO1H จนกระทั่งครบ 256 ตัวอักษร ไฟล์จึงมีขนาด 5120 ไบต์

หน่วยความจำแสดงผลของจอภาพสี

การแสดงภาพกราฟิกโดยทั่วไปแล้วจะช้ากว่าการแสดงผลในโหมดตัวอักษรมาก ดังนั้นจึงจำเป็นที่จะต้องใช้การติดต่อแสดงผลกับหน่วยความจำแสดงผลโดยตรงไม่ผ่านฟังก์ชันการจัดการของดอสเพื่อให้มีความเร็วสูงขึ้น หน่วยความจำแสดงผลของจอภาพจะมีตำแหน่งแอดเดรสต่างกัน ขึ้นอยู่กับชนิดของการ์ดแสดงผลในจอภาพเฮอคิวลิสจะเริ่มที่ตำแหน่ง B000:0000 ในจอภาพวีจีเอจะเริ่มที่ A000:0000 ในส่วนของฟังก์ชั่นแสดงผลผ่านหน่วยความจำแสดงผลโดยตรงนี้

```
    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 *)((lohg)v_ram +dis_p) = data; /* FILL */
}
```

การแสดงผลภาษาไทย

การแสดงผลภาษาไทยในที่นี้จะกล่าวถึงการแสดงผลใน GRAPHIC MODE ซึ่งมีข้อดีคือตัวอักษรนิ่งและไม่ต้องอาศัยอุปกรณ์เพิ่ม เช่นการ์ดภาษาไทย จะมีปัญหาอยู่ที่การแสดงผลจะช้าจะต้องมีขั้นตอนในการแสดงผลที่จะทำให้สามารถแสดงผลได้เร็ว หรือไม่รู้สึกว่าช้ามาก เช่นการติดต่อกับ Video RAM โดยตรง หรือแสดงผลการเปลี่ยนแปลงของจอภาพเฉพาะบางช่วง โครงงานนี้จุดประสงค์คือ เพื่อที่จะศึกษาระบบภาษาไทย และพัฒนาเพื่อให้สามารถแสดงผลในเครื่องไมโครคอมพิวเตอร์ได้ โดยมีเนื้อหาโดยสรุปดังนี้

รหัสภาษาไทย

รหัสของตัวอักษรภาษาไทยนั้นมีอยู่ด้วยกันหลายแบบ เนื่องจากช่วงแรกที่มีการพัฒนาระบบภาษาไทยขึ้นมานั้นยังไม่มีมาตรฐานที่แน่นอน รหัสต่างๆ จะขึ้นอยู่กับผู้ออกแบบ และวิธีการออกแบบ ภายหลังจึงมีการกำหนดมาตรฐานขึ้นมาคือ รหัส สมอ. ในปัจจุบันนี้บนเครื่องไมโครคอมพิวเตอร์ส่วนใหญ่จะเป็น รหัส สมอ. กับเกษตร ในโครงงานนี้จะเลือกใช้รหัสของ สมอ. เป็นหลัก เนื่องจากเป็นมาตรฐาน ข้อแตกต่างที่เห็นได้ชัดของรหัส สมอ. กับ เกษตร คือ รหัส สมอ. จะมีการรวมตัว ขอขวด กับ คอคน ไว้ในตารางตามลำดับตัวอักษรเลย ตารางรหัส สมอ. มีดังนี้

การเก็บรหัส สมอ. ในไฟล์ตารางตัวอักษร

ในโครงงานนี้ใช่การแสดงผลแบบ BITMAP ซึ่งจะต้องเก็บข้อมูล BITMAP ของตัวอักษรไว้เพื่อใช้ในการแสดงผล เช่นโปรแกรมเวิร์ดจุฬา โดยจะเก็บข้อมูล BITMAP ของตัวอักษรในไฟล์ *.FON ซึ่งการเก็บ จะเรียงลำดับตัวอักษรตามรหัส สมอ. ตัวอักษรมีขนาด

8*20 โดยจะเก็บข้อมูลขนาด 1 ไบต์ จำนวน 20 ตัวต่อ หนึ่งตัวอักษร ตัวอักษรจำนวน 256 ตัว ดังนั้นไฟล์จะมีขนาด 5120 ไบต์ โดยที่หากเราต้องการตัวอักษรแบบใดก็ จะต้องออกแบบตัวอักษรแบบนั้น ๆ แล้วเก็บไว้ในไฟล์ หรือใน หน่วยความจำเมื่อจะแสดงตัวอักษรตัวใดก็อ่านจากตารางแล้วนำมาแสดงผล ทีละไบต์จนครบ 20 ไบต์

การเก็บตัวอักษรแบบ BITMAP นั้น มีข้อดีคือ เป็นวิธีที่ง่ายและเร็วไม่ต้องอาศัยการคำนวนมาก แต่ข้อเสียคือหากเราต้องการตัวอักษรแบบใดก็จะต้องมีการเก็บตัวอักษรแบบนั้น ๆ ไว้ทำให้เปลืองหน่วยความจำ และจะมีข้อจำกัดในการแสดงผลมาก มีลักษณะตัวอักษรอีกลักษณะหนึ่งคือ แบบ Vectorโดยจะเก็บจุดเริ่มต้นและทิศทางของเส้นตัวอักษรว่าทิศทางใด จะมีข้อดีคือตัวอักษรจะมีได้หลายแบบ สามารถขยาย หรือย่อได้ แต่จะต้องอาศัยการคำนวนมากทำให้อาจช้า

การแสดงผลภาษาไทยจะใช้ข้อมูลจาก font ใน file normal.fon ซึ่งได้ map เก็บไว้แล้ว ลักษณะการแสดงผลก็จะมีการตรวจสอบก่อนว่าเป็นสระบนสระล่างหรือไม่ ถ้าใช่ก็จะไม่เลื่อนตำแหน่ง pointer การแสดงผล และ set การแสดงผลเป็นแบบ XOR คือการแสดงผลทับข้อมูลเดิม

Alogorithm ในการแสดงผลภาษาไทย

รับข้อมูลจาก key board 1 ตัวอักษร
เปลี่ยนจากรหัส ascii ภาษาอังกฤษ เป็นรหัส ascii ภาษาไทยโดยใช้ตาราง tcode [ ]
if (ตรวจสอบว่าเป็นพยัญชนะหรือไม่) เลื่อนตำแหน่งการแสดงผลนั้น 1 ตำแหน่ง
else แสดงผลออกทางจอภาพ
setmod = XOR
ไม่เลื่อนตำแหน่งการแสดงผลโดยนำข้อมูลแสดงผลออกจอภาพเลย

การแสดงผลภาษาอังกฤษ ก็จะใช้ font ชุดเดียวกับการแสดงผลภาษาไทย เมื่อรับข้อมูลมาแล้วก็สามารถแสดงผลได้เลย โดยเลื่อนตำแหน่งการแสดงผลขึ้นครั้งละตำแหน่ง

การเก็บข้อมูลแบบ BITMAP ของตัวอักษรภาษาไทย

ลักษณะการเก็บจะเป็นดังนี้ จำนวน 20 ไบต์ ตัวอักษร ก-

	0	1	2	3	4	5	6	7	
0									00H
1									00H
2									00H
3									00H
4									00H
5									00H
6									00H
7									00H
8									3cH
9									42H
10									22H
11									22H
12									22H
13									22H
14									22H
15									22H
16									00H
17									00H
18									00H
19									00H

การเก็บข้อมูลแบบ BITMAP ของตัวอักษรภาษาอังกฤษ

ลักษณะการเก็บจะเป็นดังนี้ จำนวน 20 ไบต์ ตัวอักษร a

	0	1	2	3	4	5	6	7	
0									00H
1									00H
2									00H
3									00H
4									00H
5									00H
6									00H
7									00H
8									00H
9									00H
10									78H
11									04H
12									04H
13									7cH
14									84H
15									84H
16									7aH
17									00H
18									00H
19									00H

การแสดงผลอักษรเบรลล์

การแสดงผลอักษรเบรลล์ในโปรแกรมจะต้องแสดงผลใน Graphic Mode โดยจะใช้ Function การแสดงผลเดียวกับการแสดงผลภาษาไทย แต่ใช้ชุดอักษรที่แสดงคนละชุดกัน

โดยจะเก็บข้อมูล BITMAP ของตัวอักษร Braille ไว้ในไฟล์ Braill .Fon ซึ่งการเก็บ จะเรียงลำดับตัวอักษรตามรหัส สมอ. ตัวอักษรมีขนาด 8*20 โดยจะเก็บข้อมูลขนาด 1 ไบต์ จำนวน 20 ตัวต่อหนึ่งตัวอักษร ตัวอักษรจำนวน 256 ตัว ดังนั้น ไฟล์จะมีขนาด 5120 ไบต์ โดยเราออกแบบตัวอักษรต่างๆ ของอักษรเบรลล์ เก็บไว้ ใน File เช่นอักษร a เราก็จะออกแบบเป็น เก็บเอาไว้ อักษรตัวอื่น ก็เช่นกัน

ลักษณะการเก็บ Font ของอักษรเบรลล์ก็จะเก็บตามตำแหน่งตัวอักษรภาษาอังกฤษ ตามอักษรเบรลล์ในภาษาอังกฤษ เช่น ตำแหน่ง Font ใน File ของอักษร a เราก็จะนำ Bitmap รูปตัวอักษร a ของอักษรเบรลล์เก็บไว้ ตัวอื่น ๆ ก็เช่นกัน

จุดอักษรเบรลล์นั้นมี 2 สถานะคือ Active กับ Non Active จุดที่ Active ถ้าเปรียบเทียบกับการเขียนจริง ๆ แล้ว ก็คือจุดที่นูนขึ้นมานั่นเอง การออกแบบจะแทนด้วยจุด pixel จำนวน 8 จุด วางตำแหน่งคล้ายเครื่องหมายบวกเมื่อแสดงผลออกมาทางจอภาพแล้วจะทำให้มองเห็นเป็นจุดกลมเล็ก ๆ ส่วนจุดที่เป็น Non Ative จะแทนด้วยจุด pixcel จำนวน 1 จุด ทำให้มองเห็นความแตกต่างระหว่างจุด Active กับ Non Active ได้อย่างชัดเจน

การแสดงผลอักษรเบรลล์ เหมือนกับการแสดงผลภาษาอังกฤษทั่วไป โดยเลือก font การแสดงผลจาก file braille.fon มาแสดงแทนจากธรรมดาที่ใช้ font จาก file normal.fon แต่การแสดงผลของอักษรเบรลล์แตกต่างจากการแสดงผลภาษาไทยและภาษาอังกฤษ คือ ตำแหน่งบนจอภาพในการแสดงผลจะต้องมี ช่องว่างระหว่างตัวอักษร ขนาด 1 ตัวอักษร เป็นเพราะรูปของอักษรเบรลล์แบ่งจุดออกเป็นกลุ่ม ๆ คล้ายกัน ดังนั้นถ้าแสดงติด ๆ กันจะทำให้อ่านได้ลำบาก

ดังนั้นการแสดงผลอักษรเบรลล์จึงต้องเพิ่มตำแหน่งในการแสดงผลที่จอภาพเป็น 2 เท่า ของอักษรปกติ ทั้งในแนวแกน x และแนวแกน y จึงส่งผลให้การแสดง

อักษรเบรลล์ใน 1 จอภาพมีขนาดไม่เท่ากับอักษรปกติ ดังนั้นจึงมีตัวแปรสำหรับการกำหนดกั้นหน้า กั้นหลัง และจำนวนบรรทัดในการแสดงผลขึ้นมาสำหรับอักษรเบรลล์โดยเฉพาะจำนวนตัวอักษรเบรลล์ในการแสดงผลต่อหนึ่งบรรทัดบนหน้าจอสามารถที่จะแสดงได้ 30 ตัวอักษร ถ้าข้อมูลมีมากกว่านั้น สามารถใช้ key ลูกศรเลื่อนเพื่อดูข้อมูล ตำแหน่งต่อไปได้ และจำนวนบรรทัดหนึ่งจอภาพเท่ากับ 10 บรรทัดเท่านั้น การคำนวณตำแหน่งแสดงผลบนจอภาพสำหรับอักษรเบรลล์เป็นดังนั้น

```
PosX = bx * 2 + BLmarg ;
PosY = by * 2 + Bstart ;
```

การเก็บข้อมูลแบบ BITMAP ของตัวอักษรเบรลล์

ลักษณะการเก็บจะเป็นดังนี้ จำนวน 20 ไบต์ ตัวอักษร < หรือ สระ แอ

	0	1	2	3	4	5	6	7	
0									00H
1									00H
2		■							40H
3	■	■	■				■		E2H
4	■	■	■						E0H
5		■							40H
6									00H
7									00H
8		■							40H
9	■	■	■				■		E2H
10	■	■	■						E2H
11		■							40H
12									00H
13							■		02H
14						■	■	■	07H
15		■				■	■	■	47H
16							■		02H
17									00H
18									00H
19									00H

Function สำหรับการแสดงตัวอักษรบนจอภาพ

Function ในการ Load Font ตัวอักษรมาเก็บในหน่วยความจำ

```
FILE * Fp;
FILE * Fp2;
Char * filename = "normal.fon";
Char * filename = "braill.fon";
if ((fp = fopen (filename , "rb")) ==Null)
{ printf ("Font not found");
  exit (1)'
}
fread (Alphl,sizof Alphl,1,fp2);
fclose (fp2);
fclose (fp):
}
```

Function ในการแสดงผลหน้าจอภาพ 1 ตัวอักษรโดยใช้ bitmap font ณ ที่ตำแหน่ง PosX และ PosY

```
Putchar (unsigned char asciicode)
{
register int I;
unsigned char code;
int  tx,ty;
int sclmode = XOR;                กำหนดโหมดการแสดงผลเป็นแบบ
                                  การพิมพ์ทับในตอนเริ่มต้น
Switch (asciicode) {              ตรวจชนิดของอักษรที่จะพิมพ์
Case 'In' :                       เครื่องหมาย      new line
        Increase Posy ( )         เพิ่มตำแหน่งการแสดงผล
                                  ทางจอภาพแนวแกน y ขึ้นหนึ่ง
Case '#' :      it(Type =BRAIL) {     เป็นเครื่องหมาย newline
```

```
                    IncrementPosY ( ); line          สำหรับอักษรเบรลล์
if (Mode == 1) {                    ป็นการแสดงในโหมดของอักษรภาษาอังกฤษ
                                    หรืออักษรเบรลล์
Increse PosX ( );                   เพิ่มตำแหน่งทางแนวแกน x ได้เลย
Selmode=Fill;                       ให้โหมดการแสดงผลเป็นแบบพิมพ์ทับได้เลย
 }
else {                              เป็นการแสดงผลในโหมดภาษาไทย
    if (Isalpha (asciicode))        ถ้าเป็นพยัญชนะทั่วไป
      {
       IncreasePosX( );
        selmode = FILL;
       }
      }
พิมพ์ตัวอักษรออกจอภาพด้วย function
vscreen จนกระทั่งหมดความยาวของตัวอักษร
Function IncreasePosX( );
if (PosX >= RghtMar) {              ถ้าตำแหน่งจำภาพถึง Right margin
PosX = LftMar;                      กำหนดให้กลับไปเริ่มต้นที่จุด left
Increase PosY ( );                  Margin  และเพิ่มตำแหน่งบรรทัดขึ้นหนึ่ง
}
++PosX;                             เพิ่มตำแหน่งการแสดงผลทางแนวแกน X
Function การเพิ่มตำแหน่งแสดงผลทางแนวแกน Y
        IncreasePosY ( )
        {
         if ( ++ PosY>MaxCharY)     ถ้าเพิ่มบรรทัดจนถึงบรรทัดสุดท้าย
                                    ลดตำแหน่ง PosY ลงหนึ่ง
           {
             PosY- - :
           }
```

```
            PosX = LftMar-1; ให้ตำแหน่ง      Posx  ชี้ไปที่ต้นบรรทัด
          }}
        }|
```

Function **ลดตำแหน่งการแสดงผล แนวแกน x**

```
DecreasePosX( )
{
  - - PosX;
    if (PosX<=LftMar)      ถ้าตำแหน่ง PosX ลดลงถึงตำแหน่งต้นบรรทัด
    DecreasePosY( )
     }                     ลดตำแหน่ง PosY ลง 1 ตำแหน่ง
```

Function ลดตำแหน่ง การแสดงผล แนวแกน Y

```
DecreasePosY( )
 {
 if ( - - PosY<0)   เมื่อลดตำแหน่งจนกระทั่งถึงขอบบนของจอภาพ
  {
   PosY++;          เพิ่มตำแหน่งขึ้นเท่ากับอยู่ตำแหน่งเดิม
   }
    PosX = RghtMar;  กำหนดให้ PosX อยู่มุมขวาของจอภาพ
    }
```

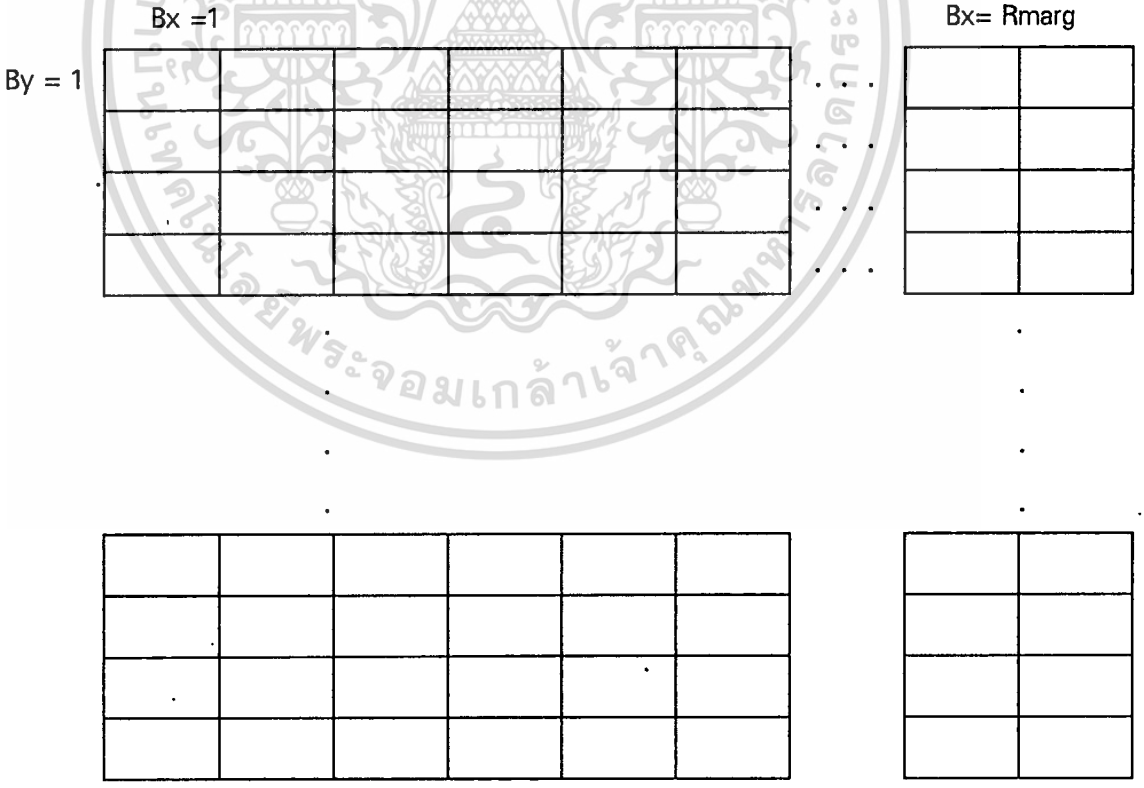
บทที่ 4

Braille Editor

Data Structure of Program

โครงสร้างข้อมูลที่ใช้ในการเขียนโปรแกรมนี้ จงใช้โครงสร้างข้อมูล แบบ Array 2 มิติ เนื่องจากตัวโปรแกรม เป็นโปรแกรมแปลงอักษรเบลล์มากกว่าที่จะเป็น Word Processsor ดังนั้น จึงใช้โครงสร้างข้อมูลที่ไม่ซับซ้อนมาก ซึ่งก็สามารถที่จะทำงานได้ดี ในระดับหนึ่ง

ลักษณะของ Data Structure ในรูปแบบของ Array 2 มิติ



BRMarg : ตำแหน่งกั้นขวา

By : เป็นตัวชี้ตำแหน่ง Line ใน Memory และตำแหน่ง Cursor ที่หน้าจอ

Bx : จะเป็นตัวชี้ตำแหน่งของ Colum ใน Memory และตำแหน่ง Cursor ที่ หน้าจอ

การแทรกและลบตัวอักษร

การแทรกตัวอักษรในแง่ของหน่วยความจำ คือ การย้ายข้อมูลไป 1 ตัวอักษร จากตำแหน่งปัจจุบัน แล้วนำตัวอักษรที่ต้องการแทรกไปไว้แทน ส่วนการลบตัวอักษรคือ การย้ายข้อมูลในตำแหน่ง ถัดไปจนกระทั่งถึงท้ายข้อมูลมาข้างหน้า

เนื่องจากการแก้ไขข้อมูลจงต้องทำใน Buffer ของข้อมูล ตามที่โปรแกรมกำหนดไว้ถ้ามีการแทรกข้อมูลจนกระทั่งเกินขนาดของ Buffer โปรแกรมย่อมทำงานไม่ได้

การอ่านค่าจาก Keyboard

การอ่านค่าจาก Keyboard จงใช้คำสั่ง geth() ในการรับค่า Keyboard โดยจะทำการตรวจสอบ Key แรกที่รับเข้ามาจะเป็นศูนย์หรือไม่ ถ้าเป็นศูนย์ก็จะทำค่า Key ตัวที่ 2 ไปตรวจสอบว่าเป็น Key อะไรแล้วก็ทำตามคำสั่งของ Key นั้น

ถ้าในการรับ Key ครั้งแรกไม่เท่ากับศูนย์ แสดงว่าเป็น Charactor ทั่วไป ก็จะทำค่าที่ได้ไปเติมใน Buffer ก่อนนำข้อมูลเก็บใน Buffer นั้นต้องตรวจสอบว่าอยู่ในโหมด Insert หรือไม่ก่อน

Algorithm การรับค่าจาก Keyboard

```
While ((Key = getch())!= 27)
        รอรับ Key จาก User จนกระทั่ง = ESC
  if (Key == 0)
        ถ้าตัวแรกเป็นศูนย์แสดงว่าเป็น Key พิเศษ
    Switch (getch())
        ตรวจสอบรหัสของ Charector ตัวที่ 2
    {
      case L_Arrow:
        DecBPosX()
        break;
      case R_Arrow:
        IncBposx()
      case D_Arrow:
        IncBposY()
      case U_Arrow:
        DecBposY();
      case PageUp:
        Inc_page();
        Show_page();
     case PageDown
        Dec_page();
        Show_page();
     case Delete:
        DeleteChar(Bx);
     case Insert:
        Insert =! Insert.
        เปลี่ยน Status ของ Mode Insert
```

```
        case Homkey:
            Scrool(); Colum แรก
        case Endkey:
            Scroll(); Colum สุดท้าย
            }
    else เป็น Charector ทั่วไป
            if(Insert == 1) ตรวจสอบว่าอยู่ใน Mode Insert หรือไม่
        InsertChar(Bx);
            ย้ายข้อมูลไปทางขวา 1 ตำแหน่ง
           นำข้อมูลจาก Keyboard ใส่ใน Buffer ที่ตำแหน่งปัจจุบัน
          Update ข้อมูลที่จอภาพ ณ บรรทัดปัจจุบัน
    else
        นำข้อมูลใส่ใน Buffer ณ ตำแหน่งที่ Bx ชี้อยู่
        แสดงข้อมูล ณ ตำแหน่งปัจจุบันออกจอภาพ
    }
}
```

Algorithm Insert_Char(Bpos);

```
for (!=Maxcolm ;i>= Bpos ;i--) {
    Buffs [i][By] = Buffs [i-1][By];
}
```

ย้ายข้อมูลจากทางขวาไป 1 ตำแหน่ง จนกระทั่งถึงตำแหน่งที่จะ Insert Charector

```
Algorithm  Delete_Char(Bpos)
   {
        กำหนด Loop จากตำแหน่งปัจจุบันถึงตำแหน่งท้าย Colum
       for (i=Bpos ;i<= 50;i++) {
       Buffs [i][By] = Buffs [i+1][By];
       ย้ายข้อมูลทางซ้าย 1 ตำแหน่งโดยทับข้อมูลเดิม
    }
```

การเลื่อนตำแหน่งของจอภาพและ Buffer

สิ่งสำคัญของ Editor อีกอย่างก็คือความสัมพันธ์ระหว่างจอภาพ และหน่วยความจำ และจากความแตกต่างระหว่างขนาดของจอภาพ กับจำนวนตัวอักษรใน Buffer ไม่เท่ากัน ดังนั้นการเลื่อนตำแหน่งของ Cursor ที่จอภาพ และตำแหน่งซ้ายหรือขวา ตามตำแหน่งข้อมูลใน Buffer โดยจะมีตัวแปร Global ที่สำคัญ ดังนี้คือ

BLmarg : ตำแหน่งกั้นซ้ายของการแสดงผล

BRmarg : ตำแหน่งกั้นขวาของการแสดงผล

Bstart : ตำแหน่งเริ่มต้นของการแสดงผล

Bend : ตำแหน่งสุดท้ายของการแสดงผล

Bx : Index สำหรับชี้ข้อมูลของ Colum ใน Buffer

BY : Index สำหรับชี้ข้อมูลของ Line ใน Buffer

Pcurx : ตำแหน่ง Cursor ก่อนหน้า 1 ตำแหน่ง

Ncurx : ตำแหน่ง Cursor ปัจจุบัน

Scroll : ตำแหน่งที่จะต้องทำการ Scroll

Page : หน้าปัจจุบันขณะที่กำลังทำงานอยู่

Posx : ตำแหน่งแกน X บนจอภาพ

Posy : ตำแหน่งแกน Y บนจอภาพ

ในการแสดงผลอักษรเบรลล์นั้นจะต้องแสดง 1 Charectorและเว้น 1 Charector เนื่องจากการออกแบบ Map ตัวอักษรนั้นไม่ได้มีช่องว่างระหว่างตัวอักษร

Function ต่าง ๆ ในการเลื่อนตำแหน่งของจอภาพและ Buffer

Function GotoBxy (int bx, int by)

จะปรับการแสดงผลจากการแสดงของตัวอักษรปกติ ที่เป็นแบบต่อเนื่องไปเป็นการแสดงแบบพิมพ์ 1 Charector และเว้น 1 Charector แล้วทำการแปลงข้อมูลจากตำแหน่งในหน่วยความจำไปเป็นตำแหน่งการแสดงข้อมูลบนจอ

```
Posx = bx*2 + BLmarg;
Posy = by*2 + BStart;
```

Function IncBposx()

จะเพิ่ม Index ในการชี้ข้อมูลใน Buffer ขึ้น 1 ตำแหน่งและเพิ่มตำแหน่ง Cursor หน้าจอภาพขึ้น 1 ตำแหน่ง

```
if(++Bx>Maxcol) {          ถ้า Bx เพิ่มถึงตำแหน่งสุดท้ายของ
   Bx = Maxcol;            Buffer = Maxcol
    }

if(Bx >= Scroll)           ถ้าเพิ่มถึงตำแหน่งขวาสุดของจอภาพต้อง
   Scroll = Bx;            ทำการ ScrollLelf
   ScrollLeft(scroll);
Posx = ( Bx* 2 ) +Blmarg;  คำนวณตำแหน่งบนจอภาพจากตำแหน่ง
                           Bx ในที่ชี้ข้อมูลใน Memory
```

Function DecBposX( )

จงลดตำแหน่งข้อมูลใน Buffer 1 ตำแหน่งในแนว Colum

```
if (--Bx<1) {
   DecBposY();
   Bx = BRmarg;
if (NcurX<=1)              ถ้าตำแหน่ง CURSUR บนจอภาพถึงด้านซ้ายของจอ
   {
   Scroll = Scroll-1;      ลดตำแหน่งที่ต้อง SCROLL ลง
```

```
ScrollLeff(Scroll);          SCROLL ข้อมูลบนจอภาพไปทางขวา 1 ตำแหน่ง
Curs();                      แสดง CURSOR ที่จอภาพ
Ncurx = Pcurx=1;
}
PosX=Bx * 2 Blmarg;          แปลงข้อมูลในหน่วยความจำไปเป็นตำแหน่ง
DecCurs();                   การแสดงผลบนจอภาพ
  }
```

Function เพิ่มตำแหน่งชี้ข้อมูลใน buffer ขึ้น 1 บรรทัด

```
IncBposy()
{
if (++By>10)                 ถ้า pointer เพิ่มขึ้นมากกว่า 10 ให้ set กลับเป็น 1
ใหม่
   By=1;                     เพื่อให้สัมพันธ์กับการแสดงผลทางจอภาพ
PosY = By*2+ Bstart;         คำนวณตำแหน่งการแสดงผลทางจอภาพ
```

Function DecBposY ( )

```
   {
      By--;
if ( By < 1 )                ลดตำแหน่ง pointer ลง ถ้า
                             น้อยกว่า 1 แล้ว
      By = 10;               ให้วนกลับไปที่ 10 ใหม่
   PosY = By * 2 + Bstart;   คำนวณตำแหน่งการ
                             แสดงผล
      CURS( );               วาด CURSOR ที่จอภาพ
      return;
      }
```

บทที่ 5
การแปลงอักษรเบรลล์

ตามหลักการที่นางสาว เจเนวีฟ คอลฟิลด์ ได้ออกแบบและสร้างกฎภาษาศาสตร์อักษรเบรลล์สำหรับภาษาไทยขึ้น พยัญชนะ วรรณยุกต์ และเครื่องหมายพิเศษอื่น ๆ ของภาษาไทย ก็จะเกิดจากการผสมกันของอักษรเบรลล์ภาษาอังกฤษนั่นเอง บางกรณีก็สามารถเปรียบเทียบได้ตัวต่อตัวเลย หมายความว่า ภาษาไทย 1 ตัว แทนด้วย ตัวอักษรภาษาอังกฤษ 1 ตัว เช่นกัน หรือ ในอักษรภาษาไทยที่ไม่ได้ใช้บ่อยนัก ก็จะแทนด้วยอักษรภาษาอังกฤษ 2 ตัว ในการเขียนอักษรเบรลล์นั้น ก็มีกฎในการเขียนของภาษาเบรลล์เองซึ่ง จะไม่เหมือนกับการอ่านหรือเขียนของคนปกติ เนื่องจากคนตาบอด มีข้อจำกัดในการอ่านและเขียน กฎต่างๆของอักษรเบรลล์จึงออกแบบให้เอื้ออำนวยความสะดวกในการอ่านและเขียนแก่คนตาบอดมากที่สุด ยกตัวอย่างเช่น การเขียนจะมีเพียงระดับเดียว ไม่มีวรรณยุกต์ หรือสระอยู่บนตัวอักษร ไม่มีสระล่าง ดังนั้น ในการแปลงอักษรของคนปกติเป็นอักษรเบรลล์จึงแบ่งทฤษฎีและ Alogorithm ในการเขียนโปรแกรมแปลง ได้ดังนี้

1. การแปลงพยัญชนะ

พยัญชนะภาษาไทย จะแทนด้วยรูปอักษรเบรลล์ตัวใดตัวหนึ่งในภาษาอังกฤษ มีลักษณะคล้าย กับการเปรียบเทียบพยัญชนะภาษาไทยกับภาษาอังกฤษของคนปกติ ตัวอย่าง เช่น อักษร "ก" ภาษาไทยเมื่อแปลงเป็นอักษรเบรลล์ก็จะตรงกับตัวอักษร "g" ในภาษาอังกฤษของอักษรเบรลล์ แต่การเปรียบเทียบ ในลักษณะเช่นนี้จะไม่เหมือนกับการเปรียบเทียบของคนปกติทุกตัวอักษรเลย เช่น "ง" จะแทนด้วย "]" ในภาษาเบรลล์ ซึ่งของคนปกติ "ง" จะแทนด้วย "ng"

ในการเขียนโปรแกรมแปลงพยัญชนะ จะใช้วิธีการ Open Table ซึ่งภายในตาราง ก็จะเก็บข้อมูลเปรียบเทียบกันระหว่างพยัญชนะภาษาไทยและอักษรเบรลล์ โดยจะใช้รหัส Ascii ของอักษรภาษาไทย ที่ส่งเข้ามาแปลงเป็น index ในการชี้ตาราง

No	รหัสภาษาไทย IdexTable	อักษรเบรลล์ 1 Brll1	อักษรเบรลล์ 2 Brll2
0	161 - ก	g	0
1	162 - ข	k	0
2	163 - ข	-	0
3	164 - ค	u	0
4	165 - ต	-	0
5	166 - ฆ	,	u
6	167 - ง	l	0
7	168 - จ	j	0
8	169 - ฉ	/	0
9	170 - ช	+	0
10	171 - ซ	!	0
11	172 - ฌ	,	+
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
69	235 - +	8	0

ตารางเก็บข้อมูลเปรียบเทียบระหว่างภาษาไทยกับอักษรเบรลล์

ตัวเลข 161-237 : เป็นรหัส Ascii ของภาษาไทย ตามมาตรฐาน

ตัวอักษรภาษาอังกฤษ : เป็นภาษาอังกฤษในอักษรเบรลล์ที่ตรงกับภาษาไทย

ตัวเลข '0' : ตารางบรรทัดนั้นภาษาไทย 1 ตัวแทนด้วยอักษรเบรลล์ 1 ตัว

Data Strueture ของตาราง

```
Typedet  Sturct  table {
        Char    brll;
        Char    brll;
                        } tables;
tables Data_table[MAXTABLE];
```

ซึ่งลักษณะของตารางจะ Initial ในโปรแกรม ดังตัวอย่าง

```
Init_Table()
{
Table(161,'g','\0');    Table(162,'k','\0');
Table(163,' ','\0');    Table(164,'u','\0');
Table(165,' ','\0');    Table(166,',','u');
Table(167,']','\0');    Table(168,'j','\0');
Table(169,'/','\0');    Table(170,'+','\0');
Table(171,'!','\0');    Table(172,',','+');
Table(173,',','y');     Table(174,',','d');
Table(175,',','/');     Table(176,'-','\\');
Table(177,',',')');     Table(178,'-',')');
Table(179,',','n');     Table(180,'d','\0');
Table(181,'\\','\0');   Table(182,'t','\0');
Table(183,')','\0');    Table(184,'0','\0');
Table(185,'n','\0');    Table(186,'v','\0');
Table(187,'&','\0');    Table(188,'p','\0');
Table(189,'x','\0');    Table(190,'?','\0');
Table(228,':','\0');    Table(229,'*','\0');
        .                       .
Table(234,'7','\0');    Table(235,'8','\0');
Table(236,'0','\0');    Table(237,' ','\0');
```

Algorithm ในการแปลงพยัญชนะ

รับค่ารหัส Ascii ของพยัญชนะที่จะแปลงเข้ามา

แปลงค่ารหัส Ascii เป็น Index เพื่อใช้ในการชี้ตารางแปลง

เปิดตารางที่ Index ตรวจสอบการแปลง

นำข้อมูลในตารางซึ่งเป็น Field ของอักษรเบรลล์ ไปใส่ใน Output Buffers หรือ Buffer ที่ใช้สำหรับเก็บผลการแปลง

ตรวจสอบว่า มีตัวอักษรที่จะต้องนำไปเก็บใน Output Buffer อีกหรือไม่

```
if (Data_Table [Index].brll2!=0)
Begin:
    เพิ่มตำแหน่ง Index ใน Output Buffer
    นำข้อมูลในตารางไปเก็บใน Output Butter
End:
```

โปรแกรม แปลงพยัญชนะ

```
Trans_consonant ( )
{
 int index;
index = Tdata [IndexT];
index= index-161;
Fill_Buffs(Data_table[index].brll1);
   if(Data_table[index].brll ! = 0
  {
      Inc_IndexB0;
      Fill_Buffs(data[index].brll2;
}}
```

2.การแปลงสระ

ตามหลักภาษาศาสตร์ของอักษรเบรลล์แล้วการประสมคำก็จะประกอบด้วยอักษรต่างๆ ผสมกัน ซึ่งแบ่งตามชนิดได้ดังนี้คือ

- พยัญชนะต้น (CONSONANT)
- สระ (VOWEL)
- ตัวสะกด (CONSONANT)
- วรรณยุกต์ (TONE)

ในการเขียนนั้นอักษรเบรลล์ทั้งพยัญชนะ สระ วรรณยุกต์ จะอยู่บนบรรทัดเดียวกันหมด สิ่งสำคัญอีกอย่างหนึ่งของการประสมคำของอักษรเบรลล์ก็คือ สระที่เกิดจากการผสมกันของสระมากกว่า 1 ตัว เช่น สระ เอา สระเอือ เมื่อแปลงเป็นอักษรเบรลล์ลักษณะของสำดับในการเรียง พยัญชนะ สระ และวรรณยุกต์จะแตกต่างจากอักษรปกติและแต่ละสระก็จะมี รหัสที่เป็นเฉพาะของตัวเอง ไม่ได้เกิดจากการรวมกันของหลายๆ สระอย่างเช่นอักษรปกติ ยกตัวอย่าง เช่น เรา ลักษณะการเขียนของอักษรเบรลล์ ร เอา

และ ลักษณะการเขียนของอักษณเบรลล์ ล แอะ
เหมาะ ลักษณะการเขียนของอักษรเบรลล์ หม เอาะ
เหมือน ลักษณะการเขียนของอักษรเบรลล์ หม เอือ น
โต๊ะ ลักษณะการเขียนของอักษรเบรลล์ ต โอะ อ๊

จากความแตกต่างตรงจุดนี้เอง ทำให้การเขียนโปรแกรม ในการแปลงเกิดความยุ่งยากและซับซ้อนขึ้น ดังนั้นเมื่อพบสระแล้วจะแบ่งการแปลงออกเป็น 2 กรณี คือ

2.1 การแปลงสระที่ไม่ใช่สระผสม เป็นสระที่สามารถแปลงได้ทันทีเลยไม่ต้องมีการทำ Forward Cheeking เช่น สระ อะ อั อา อำ อิ อี อึ อื อุ อู ดังนั้น เราจึงมองการแปลงเหมือนการแปลงพยัญชนะทั่วไปเพื่อที่จะให้สามารถใช้

Alogorithm การแปลงเหมือนกับ การแปลงพยัญชนะเราจึงใส่ข้อมูลของสระเหล่านี้ในตารางของการแปลงพยัญชนะด้วย เมื่อเราแปลงสระเหล่านี้เราก็ใช้ Table การแปลงเดียวกับการแปลงพยัญชนะได้เลย

2.2 การแปลงสระที่เป็นสระผสม เป็นสระที่ไม่สามารถรวมได้ทันทีว่าเป็นสระอะไร ต้องมีการทำ Forward Check เพื่อทำมาเข้ากฎตรวจสอบว่าเป็นสระอะไร ซึ่งสระเหล่านี้ จะขึ้นต้นด้วย สระ "เ","แ","โ" และจากรูปแบบการประสมคำของภาษาไทยที่มีมากมายหลายรูปแบบ ทำให้เกิดคำยกเว้น คำที่ไม่ตรงกับกฎที่เขียนไว้อีกมากมาย ยกตัวอย่างการประสมคำ เช่น

สระ+พยัญชนะ = เสือ
สระ+พยัญชนะ+วรรณยุกต์ = เสื่อ
สระ+พยัญชนะ+ตัวควบกล้ำ = เกลือ
สระ+พยัญชนะ+ตัวสะกด = เฉือน
สระ+พยัญชนะ+วรรณยุกต์+ตัวสะกด = เกลื่อน

จะเห็นว่าการประสมคำของสระ "เอือ" เพียงสระเดียวก็เป็นไปได้อย่างน้อยที่สุด 5 รูปแบบ อาจจะมีคำยกเว้นบางคำของสระเอือที่เขียนตามรูปแบบที่กล่าวมาแต่ความหมายไม่เป็นไปตามนั้น

ดังนั้นในการแปลงสระผสมเหล่านี้จะใช้วิธี Forward Checking และเปรียบเทียบกับ Dictionarg ภายใน Dictionary ก็จะเก็บกฎต่าง ๆ ของการประสมคำของแต่ละสระไว้ และให้ผู้ใช้สามารถเพิ่มกฎในการประสมคำของสระหรือเพิ่มคำยกเว้นเข้าไปใน Dictionary ได้

Algorithm ในการแปลงสระผสม

ตรวจสอบข้อมูล Input ตรงตำแหน่งปัจจุบัน และอีก 6 ตัวอักษรข้างหน้า ตรงกับข้อมูลใน Dic_table หรือไม่

if (เท่ากัน)

แปลงข้อมูลจาก Dictable

Return;

else

เพิ่ม Index เพื่อชี้ตาราง dictronanry

ตรวจสอบเปรียบเทียบจนหมดตาราง Dictionry

if (ครบตาราง Dic แล้วไม่พบ)

แสดงว่าไม่ใช่สระผสมหรือหาไม่พบ

แปลงข้อมูลตรงตำแหน่งนั้นโดยวิธีปกติ

ตัวอย่าง โปรแกรม

```
Trans_vowel()
 {
  int id;
  int Flag;
  for(id=0;id<=MAXDIC;id++){
   Flag=Compare_Dic(id);
   if(Flag==1){
     Trans_dic(id);
     Tdic=0;
     id=MAXDIC;
     return 1;
     }
  }
return 0;
}
```

3. การแปลงภาษาอังกฤษ

อักษรเบรลล์ในภาษาอักฤษ ไม่ค่อยมีความยุ่งยากซับซ้อนเท่าใหร่เพราะว่ามีลักษณะคล้ายๆ กัน การแปลงก็จะสามารถแปลงไปโดยตรงเลย แต่มีข้อแตกต่างที่เห็นได้ชัดคือ ระหว่างอักษรตัวเล็กกับอักษรตัวใหญ่ เนื่องจากคนตาบอดไม่

สามารถมองเห็นความแตกต่างระหว่างอักษรตัวเล็ก กับอักษรตัวใหญ่ได้ ดังนั้นในภาษาเบรลล์จะใช้เครื่องหมาย "," นำหน้า เพื่อแสดงว่าอักษรที่ต่างจากเครื่องหมาย "," เป็นอักษรภาษาอังกฤษตัวใหญ่ และถ้ามีตัวอักษรภาษาอังกฤษตัวใหญ่ติดกันหลายๆ ตัว เราต้องใช้เครื่องหมาย ",," สองตัวนำหน้า

Algorithm **ในการ แปลงภาษาอังกฤษ**

ตรวจสอบข้อมูลต่อจากตำแหน่งปัจจุบัน 1 ตำแหน่ง
ตรวจสอบข้อมูลก่อนตำแหน่งปัจจุบัน 1 ตำแหน่ง
if (ข้อมูลข้างหน้าเป็นภาษาอังกฤษตัวใหญ่)
นำข้อมูลตำแหน่งนั้นเก็บใน output Buffer ได้เลย
else if (อักษรตัวก่อนไม่ใช่ภาษาอังกฤษตัวใหญ่ && อักษรตัวต่อไปเป็นภาษาอังกฤษตัวใหญ่)
ให้นำหน้าอักษรตัวนั้นด้วยเครื่องหมาย "," 2 ตัว
else if (อักษรตัวก่อนไม่ใช่ภาษาอังกฤษตัวใหญ่ && อักษรตัวต่อไปไม่ใช่ภาษาอังกฤษตัวใหญ่)
นำหน้าอักษรตัวนั้นด้วยเครื่องหมาย "," เพียงตัวเดียว

Function **การแปลงภาษาอังกฤษ**

```
Trans_engcap()
  {
   int TType1,TType2;
   TType1=Get_token(Tdata[IndexT-1]);
   TType2=Get_token(Tdata[IndexT+1]);

  if(TType1==ENG1 )
        Fill_Buffs(Tdata[IndexT]);
  else if(TType1!=ENG1&&TType2==ENG1){
        Fill_Buffs(44);      /*   ,    */
        Inc_IndexB();
```

```
    Fill_Buffs(44);
    Inc_IndexB();
    Fill_Buffs( [ Tdata[IndexT ] );
    }

else if(TType1!=ENG1&&TType2!=ENG1){
    Fill_Buffs(44);
    Inc_IndexB();
    Fill_Buffs(Tdata[IndexT]);
  }
 }
```

4. การแปลงตัวเลข

ตัวเลขในอักษรเบรลล์ จะต้องนำด้วยเครื่องนำเลขคือ # เช่น เลข 3 เขียนเป็นอักษรเบรลล์ได้เป็น # 3 กฎเกณฑ์ในการแปลงตัวเลขในอักษรเบรลล์มีดังนี้คือ

4.1 ตัวเลขทุกตัวต้องทำด้วยเครื่องหมายนำเลขคือ #

4.2 ถ้ามีเลขตั้งแต่ สองหลักขึ้นไปจะต้องมีเครื่องหมายนำเลข เพียงตัวเดียวเท่านั้น

4.3 ตัวเลขหลักเครื่องหมายทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร ไม่ต้องมีเครื่องหมายนำเลข

Algorithm ในการแปลงตัวเลข

ตรวจสอบชนิดข้อมูลก่อนหน้าตำแหน่งปัจจุบัน 1 ตำแหน่ง

if(เป็น ตัวเลข)

แปลงได้เลยไม่ต้องมีเครื่องหมาย นำเลข

else if (เป็นเครื่องหมาย ทางคณิตศาสตร์)

แปลงโดยไม่ต้องมีเครื่องหมายนำเลข

else

ใส่เครื่องหมายนำเลขหน้าตัวเลขที่ตำแหน่งนั้น

เก็บข้อมูลลง Output Buffer

Function การแปลงตัวเลข

```
Trans_Number( )
{
    int Type;
Type=Cet_token(Tdata[IndexT-1);

if(Type==NUM
    Fill_Buffs(Tdata[Index]);
else if (Type==MATH)
    Fill_Buffe(Tdata[IndexT];
else
    Fill_Buffs(#);
    Fill_Buffs(Tdata[IndexT]);
}
```

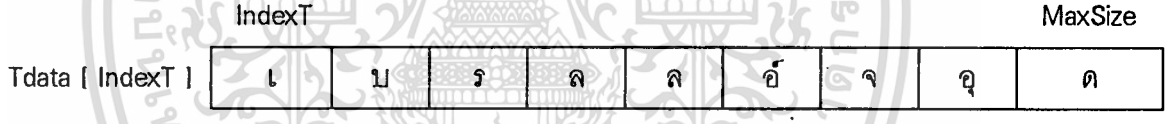
บทที่ 6
การแปลงข้อมูลใน File และใน Buffer

การแปลงข้อมูลใน File

รูปแบบของ File ข้อมูลที่โปรแกรมสามารถมาแปลงเป็นอักษรเบรล์ล เป็น Text File ที่ใช้รหัส Ascii ตามมาตรฐาน สมอ. ไม่ว่าจะพิมพ์จาก Editor ตัวใดก็ตามถ้ามีรูปแบบข้อมูลตัวที่ได้กล่าวมาแล้วสามารถแปลงเป็นอักษรเบรล์ลได้โดยไม่มีข้อจำกัดในเรื่องของขนาดของ File ซึ่งจะมีขนาดเท่าใดก็ได้

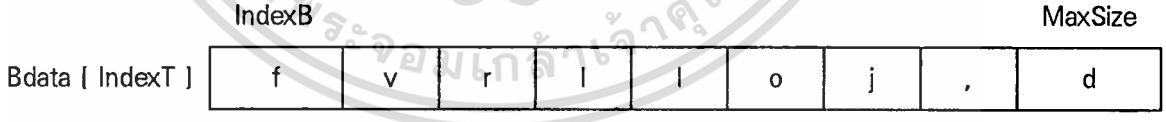
การแปลงข้อมูลจาก File นั้นจะใช้ Buffer สองตัวในการแปลงคือ แปลง Buffer สำหรับเก็บข้อมูล Input ที่จะแปลงและ Buffer สำหรับเก็บข้อมูล Output ที่ได้จากแปลงซึ่ง Data Structure ของ Buffer ทั้งสองจะมีลักษณะเป็น Array ขนาด 1 มิติ สาเหตุที่ใช้ Buffer เป็นแบบ Array ก็เพื่อความง่ายในการเขียนโปรแกรมแปลง

INPUT BUFFER



TRANSLATE TO BRAILLE

OUTPUT BUFFER



เมื่อ

Tdata [ ] : เป็น Buffer สำหรับเก็บข้อมูลภาษาไทยที่จะแปลง

Bdata [ ] : เป็น Buffer สำหรับเก็บข้อมูลที่ได้จากการแปลง

IndexT : เป็น Index สำหรับชี้ข้อมูลที่จะแปลง

IndexB : เป็น Index สำหรับชี้ตำแหน่งสำหรับเก็บข้อมูลที่ได้จากการแปลง

MaxSize : ขนาดสูงสุดของ Buffer

เมื่ออ่านข้อมูลจาก File มาเก็บใน Buffer จนเต็มแล้ว แต่ยังไม่หมด File ก็จะแปลงข้อมูลใน Buffer ก่อน จากนั้นก็จะ Save ข้อมูลที่ได้จากการแปลงใน Buffer ลงใน File แล้วทำการแปลงข้อมูลส่วนที่เหลือจนจบไฟล์

Algorithm ของฟังก์ชั่น Trans_File()

Open File พร้อมกับตรวจสอบการเปิดแฟ้ม

```
do {
    อ่านข้อมูลจาก File 1 อักษรเก็บใน Buffer ที่ Tdata[ p ] เพิ่มตำแหน่ง
    buffe = p++;
if(P>P_SIZE) ถ้าอ่านข้อมูลจนเต็ม Buffer แล้ว
{
    ใส่รหัสพิเศษที่ Tdata เพื่อบอกตำแหน่งสิ้นสุดข้อมูล
    แปลงข้อมูลใน Tdata.Buffer
    เก็บผลการแปลงใน Bdata Buffer ลง file
    set Buffer Pointer p=o;
}
}while (!feoj(fp)) ตรวจสอบดูว่าสิ้นหรือ file หรือยัง
    ใส่รหัสพิเศษที่ Tdata เพื่อบอกตำแหน่งสิ้นสุดข้อมูล
    แปลงข้อมูลใน Tdata Buffer
    เก็บผลการแปลงใน
    Bdata Buffer ลง file
    ปิดแฟ้มข้อมูล
```

การแปลงข้อมูลใน Buffer

สำหรับกรแปลงข้อมูลใน Buffer จะมี Grobal varrable ทำหน้าที่เป็น Index ในการชี้ข้อมูลใน Buffer ทั้ง Input Buffer และ Output Buffer ดังนี้คือ

IndexT : เป็น Index สำหรับชี้ข้อมูลใน buffer Tdata [ ]

Index B : เป็น Index สำหรับชี้ข้อมูลใน buffer Bdata [ ]

Algorithm function Trans_Buff()

Initial index IndexT = IndexB=o

do

แปลงข้อมูล ณ. ตำแหน่งที่ IndexT ชี้อยู่โดย function Translate()

เก็บข้อมูลใน Buffer ตรงตำแหน่งที่ IndexB ชี้

เพิ่ม IndexT ขึ้น 1 ชี้ข้อมูลตัวถัดไป

เพิ่ม IndexB ขึ้น 1 ชี้ตำแหน่งถัดไป ใน Bdata[IndexB]

while (!Check_end())

ตรวจสอบจนสิ้นสุดข้อมูลที่จะแปลงหรือยัง

ใส่เครื่องหมายแสดงจุดสิ้นสุดการแปลงใน Bdata[IndexB] return ;

Function Translate()

โดย funetionนี้จะมีหน้าที่แปลงข้อมูล 1 ตัวอักษรจาก Input Buffer ณ ตำแหน่งที่ IndexT ชี้อยู่ ผลจากการแปลงจะเก็บไว้ใน Buffer Bdata[ ] ตำแหน่งที่ IndexB ชี้อยู่

Algorithm Function Translate()

ตรวจสอบข้อมูลด้วย function Get_token ว่าเป็น Charector ชนิดใด

Type=Get_token(Tdata[IndexT])

Switch(Type){

case: ENG_CAP:

Trans_EngCap();

case: Eng_Nolm:

Trans_EngNolm();

Case: CON "TONE

Trans_Stardart();

Case: VOWEL

Dicton = Trans_VOWEL()

if (ไม่พบใน Dictronary)

Trans_char(Tdata[IndexT]);

แปลงข้อมูลแบบธรรมดา

ตัวอย่าง โปรแกรม

```
Translate()
{
int Type;
Type=Get_token(Tdata[IndexT]);
if(Type==NUM)
        Trans_num();
if(Type==ENGCAP)
        Trans_engcap();
else if(Type==ENGNOL)
        Trans_engnol();
else if(Type==CON || Type==TONE)
         Trans_std();
else if(Type==EXTRA)
         Trans_extra();
else if(Type==VOWEL){
         Type=Trans_vowel();
         if(Type==0)              /* no data in dic translate one vowel */
         Trans_Char(Tdata[IndexT]);
 }
 }
```

Function Get_token()

มีหน้าที่ตรวจสอบชนิดของอักษร ณ. ตำแหน่งที่ IndexT ชี้อยู่แล้วส่งผลการตรวจสอบไปให้ Function ที่เรียกมา

โดยใช้ Marco BETWEEN(Volue,low,high) ในการตรวจสอบการกำหนดเงื่อนไขสำหรับ Morco คือ

BETWEEN (valve,low.high) ((valve>=law) && (valve x = high))

Function Get_token จะแบ่งชนิดของอักขระ ในรหัส Ascii ตามมาตรฐาน สมอ. เป็นชนิด ต่าง ๆตามกฎของอักษรเบรลล์ดังนี้คือ

```
Get_token(>nt ch)
{
if(BETWEEN(ch,00,47))
    return(EXTRA)
if(BETWEEN(Ch,48,57))
    return(NUMBER)
if(BETWEEN(ch,65,90))
    return(ENG_CAPS)
if(BETWEEN(ch,97,122))
    return(ENG_NOLM)
if(BETWEEN(ch,161,217))
    return(con)
if(BETWEEN(ch,224,228))
    return(VOWEL)
if(BETWEEN(ch,230,236))
    return(TONE)
}
```

บทที่ 7
Dictionary

ในการแปลงภาษาไทยเป็นภาษาเบรลล์นั้น จะต้องมีการวิเคราะห์คำในบางกรณี ดังนั้นการกำหนดกฎตายตัวลงไป ในการแปลงจะเกิดความผิดพลาดได้ง่าย เนื่องจาก คำในภาษาไทยเกิดจากการประสมของสระ พยัญชนะ และวรรณยุกต์หลายรูปแบบ มีคำยกเว้นมากมาย คำบางคำ รูปสระ และตำแหน่งเหมือนกัน แต่หลังในการอ่าน และความหมายต่างกัน

ยกตัวอย่างเช่น

เหมา อ่านว่า เหมา เขียนเป็นอักษรเบรลล์ หม เอา

เวลา อ่านว่า เว ลา เขียนเป็นอักษรเบรลล์ เ ว ล า

ดังนั้น ภายใน Dictionary จะเก็บ คำ ยกเว้นเหล่านี้ไว้และก็จะเก็บผลของการแปลงเป็นอักษรเบรลล์ ไว้อีก Field หนึ่ง ซึ่งคำยกเว้นต่างๆ ผู้ใช้สามารถเพิ่มเติมหรือลบออกจาก Dictionary ได้อย่างง่ายดาย เนื่องจากการออกแบบ การเก็บ Dictionary เป็น text file ธรรมดา เพียงแต่จะมีข้อกำหนดในการเขียนบางประการที่ผู้เขียนโปรแกรมกำหนดขึ้นเพื่อให้โปรแกรมสามารถทำข้อมูลจาก file ไปเก็บใน memory ได้อย่างถูกต้อง

ข้อกำหนดในการเขียน file Dictionary

1. แต่ละ Field จะต้องกั้นด้วยเครื่องหมาย #
2. แต่ละ Field จะห่างกันเท่าใดก็ได้
3. ถ้าต้องการแทนด้วยพยัญชนะใดๆ ให้ใช้เครื่องหมาย "?" ใส่ที่ตำแหน่งนั้น
4. เมื่อ save file ต้องใช้ชื่อ file name เป็นDic.Dat.

ยกตัวอย่าง file Dic.Dat,

ตัวอย่าง FILE DIC.DAT

เ?าะ	#	oa	#
เกราะ	#	groa	#
เหมาะ	#	hmoa	#
เ?า	#	6	#
เ?่า	#	69	#
เ?้า	#	64	#
เ?๊า	#	67	#
เ?๋า	#	68	#
เวลา	#	wl6	#
เหมา	#	hm6	#
เ?ะ	#	fa	#
เ?อะ	#	%a	#
เ?อ	#	%	#
เ?ียะ	#	(a	#
เ?ีย	#	(	#
เ?ี้ย	#	(4	#
เ?ือ	#	q	#
เ?ื่อ	#	q9	#
แ?ะ	#	<a	#
แ?ละ	#	l<a	#
แสวง	#	sw<]	#
โ?ะ	#	la	#
เศร้า	#	,s64	#

Data Structure ของ Dictionary

เมื่อจะแปลงข้อมูล จะต้องเปิด file Dic.dat เพื่อนำข้อมูลภายใน Dictionary มาเก็บใน memory ใช้สำหรับเปรียบเทียบแปลงในกรณีคำที่เป็นสระผสมหรือคำยกเว้นอื่นๆ รูปแบบโครงสร้างข้อมูลในการเก็บข้อมูล Dictionary ใน โปรแกรมเป็นดังนี้คือ

```
typedet Struuct Stable{
        int ch[6];
        int cnt;
        int bcnt;
        int brll [4];
    }Dic_Table;

Dic_Table  Dic[MAX DIC]
```

เมื่อ

ch : เก็บข้อมูลสำหรับเปรียบเทียบกับข้อมูล Input

cnt : จำนวนอักษร ใน ch

brll : ผลของการแปลงคำจาก ch เป็นอักษรเบรลล์

bcnt : จำนวนอักษรที่แปลงใน brll

Dic[MAXDIC] : จำนวน Record ใน Dictionary

จากรูปแบบ Data structure สามารถเขียนอยู่ในรูปของตาราง เพื่อให้สามารถมองเปรียบเทียบได้ง่ายขึ้น

Dic [I]	ch [6]	Brll [4]	cnt	bcnt
0	เ ? า ะ	oa	4	2
1	เ ก ร า ะ	groa	5	4
2	เ ห ม า ะ	hmoa	5	4
3	เ ? า	56	3	2
4	เ ว ล า	bwl*	4	4
5	เ ห ม า	hmb	4	3
6	เ ? อ	%	3	1
7	เ ? อ ะ	%a	4	2
8	เ ? อี ย	c	5	1
9	แ ? ะ	<a	3	2
	เ ? ะ	fa	3	2

Algohthm การนำข้อมูลจาก Dictronary file เก็บใน Memory จาก Data structure ของ Dictronary จะเห็นว่าแบ่งได้เป็น 2 Field หลักใหญ่ ๆ คือ THAI และ BRAILL ส่วนที่เพิ่มเติมของโปรแกรมคือ Field END ดังนั้น เมื่อเราอ่านข้อมูลพบเครื่องหมายกั้นระหว่าง Field "#" เราก็จะเปลี่ยน status จนไปเรื่อยๆ เช่น THAI เปลี่ยนเป็น BRAILL วนไปเป็น END เป็น THAI โดย แต่ละ Field เราจะกำหนดมี status เป็นดังนี้คือ

THAI # BRAILL # END

0 # 1 # 2

ดังนั้นการเปลี่ยน status จะวนจากศูนย์ หนึ่ง สอง แล้วกลับมาศูนย์ใหม่

วิธีเขียนโปรแกรมให้ status วนอยู่เช่นนี้ ก็ใช้คำสั่ง mod

status = status mod 3

Algorithm ฟังก์ชั่น Load_table()

เปิด file Dic.Dat

Initial status เป็น THAI

อ่านข้อมูลจาก file มา 1 ตัวอักษร

ตรวจสอบชนิด ตัวอักษร

if (ch == ' # ') เป็นเครื่องหมายกั้น field

เปลี่ยน status

if(ch == ช่องว่างหรือเครื่องหมายอื่นๆ)

ข้ามไปอ่านอักษรตัวถัดไป

else if(ถ้าเป็นพยัญชนะข้อมูล)

switch(status) ตรวจสอบ status ขณะนั้น

case THAI:

เก็บข้อมูลใน Dictionary table Field THAI

เพิ่ม cnt ตัวแปร count จำนวนอักษรขึ้นหนึ่ง

case BRAILL:

เก็บข้อมูลใน Dictionary table Field BRAILL

เพิ่มค่าตัวแปรที่ count จำนวนอักษรเบรลล์ขึ้นหนึ่ง

case END:

เก็บจำนวน อักษรภาษาไทยและเบรลล์ลง Dic table

clear ตัวแปรต่างๆ

ตรวจสอบว่าจบ กระทั่งจบแฟ้มข้อมูล

โปรแกรม ฟังก์ชั่น Load_Dic_table()

```
        Load_Dic_Table(char fname[10])
{
#define THAI 0
#define BRAILL 1
#define END 2
FILE *fp;
int ch;
int stat=0,i=0,j=0,i1=0;
if((fp=fopen(fname,"rb"))==NULL){
printf("cannot open file");getch();
return 0;
}
do{
ch=getc(fp);
  if(ch == 35){
                    stat++;
                    stat=stat%3;
            if(stat == END){
                              Dic[j].cnt=i;
                              Dic[j].bcnt=i1;
                              i=i1=stat=0;
                              j++;

 else if(ch==32||ch==13||ch==10)
                    ch=ch;
  else{
          switch(stat){
                    case THAI:
```

```
                    Dic[j].ch[i]=ch;
                    i++;
                    break;
                case BRAILL:
                    Dic[j].brll[i1]=ch;
                    i1++;
                    break;
                }
                }
        }while(!feof(fp));
        fclose(fp);
}
```

Algorithm การเปรียบเทียบข้อมูลกับ Dictionary

ในการเปรียบเทียบนั้นจะนำข้อมูล Input ที่จะเปรียบเทียบมาเทียบกับข้อมูลใน Dic table ที่ละตัวอักษร จนกระทั่งพบว่าทุกตัวอักษรที่นำมาเปรียบเทียบตรงกับข้อมูลใน Dictable ณ.ที่ Record นั้น

ข้อดีอีกอย่างของ Dictionary คือการใช้เครื่องหมาย " ? " แทนพยัญชนะใดๆ ก็ได้ 1 ตัวทำให้ลดจำนวนข้อมูลใน Dictionary ลงไปมาก เนื่องจากคำในภาษาไทยจะมีรูปสระที่เหมือนกัน แต่ต่างกันที่อักษรนำเท่านั้น ดังนั้น ถ้าสระใดที่เข้ากฎเช่นนี้ได้ก็ให้แทนพยัญชนะตรงตำแหน่งนั้นด้วยเครื่องหมาย " ? " ได้ ยกตัวอย่างเช่น สระเอา เขียนใน Dictionary เป็น เ ? า เมื่อเปรียบเทียบกับคำเหล่านี้โปรแกรมจะถือว่าเท่ากัน เรา เอา เมา เขา หลักในการเขียนโปรแกรมเปรียบเทียบก็คือ จะเปรียบเทียบข้อมูล Input ที่เสริมกับ Dictionary ในบรรทัดนั้นทีละตัว ถ้าเปรียบเทียบแล้วแต่ละตัวเท่ากัน ก็จะเปรียบเทียบไปเรื่อยๆ จนครบจำนวนอักษรใน Dictionary แต่ถ้าเปรียบเทียบแล้วมีตัวใดตัวหนึ่งไม่เท่ากัน ก็จะล้มเลิกการเปรียบเทียบในบรรทัดนั้นแล้ว เปลี่ยนเป็นบรรทัดถัดไปของ Dictionary ทันที

Algorithm **ฟังก์ชั่น** Compare_Dic()

ตรวจสอบข้อมูลได้ dictionary ว่าเป็น เครื่องหมาย " ? "

```
if(Dic[ i ] == " ? " ) {
```

นำข้อมูลตำแหน่งนั้นเก็บใน Bufferก่อน

ข้ามการเปรียบเทียบตำแหน่ง i โดยถือว่าเท่ากัน

```
if (ch[i]==Tdata[ IndexT+i ]
        Flag =1;
```

เพิ่มตำแหน่ง iเพื่อเปรียบเทียบอักษรถัดไป

```
else
        Flag=0;
```

เพิ่มบรรทัดใน Dictionary table ขึ้น

ตรวจสอบจนจบ Dic table ทั้งหมด

รูปแสดงการเปรียบเทียบข้อมูล Input Buffer กับ Dic table

```
for ( i  = 1 ; i <= cnt ; i++  )
```

IndexT + i

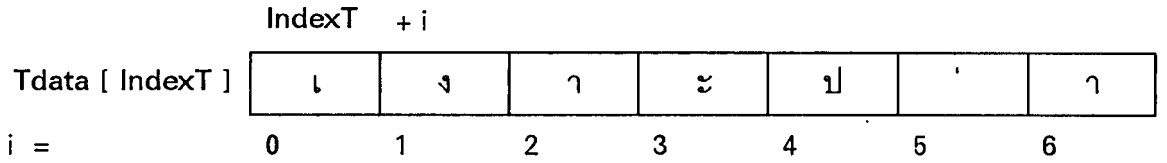
Tdata [IndexT]	เ	ก	ล	า	ม	า	ก
i =	0	1	2	3	4	5	6

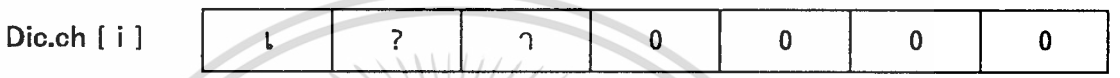
Dic.ch [i]	เ	?	ล	า	0	0	0

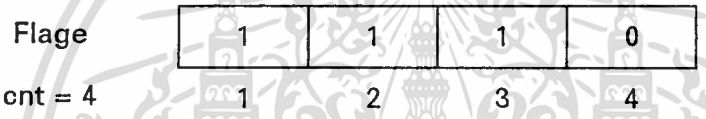
Flage	1	1	1	1
cnt = 4	1	2	3	4

ผลการเปรียบเทียบเท่ากัน

for (i = 1 ; i <= cnt ; i++)







ผลการเปรียบเทียบ = ไม่เท่ากัน

ตัวอย่าง โปรแกรม

```
Compare_Dic(int Index)
{
int flag,i,Tmp;
for(i=0;i<=Dic[Index].cnt-1;i++){
Tmp=Dic[Index].ch[i];
 if(Tmp==63){
        Tdic=Tdata[IndexT+i];
        Tmp=Tdata[IndexT+i];
                        }
   if(Tdata[IndexT+i]==Tmp){
              flag=1;
              }
```

```
        else{
                flag=0;
                Tdic=0;
                return flag;
        }
        }
  return flag;
}
```

Algorithm **แปลงใน** Dictionary

เมื่อเราเปรียบเทียบข้อมูล Input กับข้อมูลใน Dictionary จนกระทั่งผลการเปรียบเทียบเท่ากันแล้ว ในการแปลงข้อมูลนั้นสิ่งที่ต้องคำนึงถึงในการแปลงคือ

1. ขนาดความยาวของคำ หรือจำนวนอักษรที่แปลง เมื่อแปลงแล้วต้อง set Index ที่ชี้ข้อมูลใน Input Buffer ให้ถูกต้อง
2. ขนาดความยาวหรือจำนวนอักษรในการแปลงก็เช่นกัน เมื่อแปลงแล้วต้อง set Index ที่ชี้ข้อมูลใน Output Buffer

Algorithm **ฟังก์ชั่น** Trans_dic()

ตรวจสอบว่าบรรทัดนั้นมีการเก็บพยัญชนะที่ตรงกับเครื่องหมาย "?" ใน Dictionary ไว้หรือไม่

```
if(Tdic!=0){
  Trans_char(Tdic)
  แปลงพยัญชนะที่เก็บไว้
  }
```

นำข้อมูลใน Dictionary table ไปเก็บไว้ใน Output Buffer

ทำจนครบจำนวนอักษรใน Field Dic[Index].bcnt-.

เพิ่ม Index ชี้ข้อมูลใน Input Buffer ไปเป็นจำนวนเท่ากับจำนวนอักษรใน field dic[Index].cnt

ตัวอย่าง โปรแกรม

```
Trans_dic(int index)
{
int I;
if(Tdic!=0){
  Trans_char(Tdic);
  Inc_IndexB();
  Tdic=0;
  }
for(i=0;i<=Dic[index].bcnt-1;i++){
        Fill_Buffs(Dic[index].brll[i]);
        Inc_IndexB();
        }
        Dec_IndexB();
        IncN_IndexT(Dic[index].cnt-1);  /* -1 because sub trans will +1 */
}
```

บทที่ 8
การพิมพ์เอกสารอักษรเบรลล์

ส่วนการจัดพิมพ์ เอกสารเป็นอีกส่วนหนึ่งที่สำคัญของโปรแกรม เนื่องจากหากเอกสารที่สร้างขึ้นหรือผ่านการแปลงแล้ว ไม่สามารถพิมพ์ได้ ก็ไม่มีประโยชน์แต่อย่างใด ข้อกำหนดในการพิมพ์นั้น เครื่องพิมพ์ สำหรับพิมพ์อักษรเบรลล์นั้นใช้รหัส Ascii มาตรฐานทั่วไป ดังนั้น การพิมพ์ก็เพียงแต่ส่งตัวอักษรภาษาอังกฤษที่ผ่านการแปลงไปให้เครื่องพิมพ์เท่านั้น

ขั้นตอนการพิมพ์เอกสาร

1. ทำข้อมูลที่จะพิมพ์เข้ามาอยู่ ใน Buffer สำหรับการพิมพ์ โดย Data structure ของ Buffer เป็นแบบ Array 2 มิติ

2. กำหนดค่าเริ่มต้นสำหรับการพิมพ์ ได้แก่ จำนวนหน้าที่จะพิมพ์ ตำแหน่งกั้นขวาในการพิมพ์ จำนวนบรรทัดต่อหน้า

3. ตรวจสอบความพร้อมของเครื่องพิมพ์ซึ่งใช้ฟังก์ชั่น biosprint() ในการตรวจสอบ ซึ่งฟังก์ชั่นนี้จะส่งค่ากลับทุกครั้งหลังจากใช้คำสั่งให้ส่งข้อมูลไปยังเครื่องพิมพ์แล้วค่าที่ส่งคืนจะมีความหมายในแต่ละบิตดังนี้

บิตที่ 0 Derice time out

บิตที่ 3 I/O error

บิตที่ 4 Selected

บิตที่ 5 Out of paper

บิตที่ 6 Acknowlege

บิตที่ 7 Not busy

สำหรับการตรวจสอบข้อผิดพลาดของเครื่องพิมพ์อักษรเบรลล์อย่างง่ายๆ ก็คือตรวจสอบว่ามี I/O error หรือไม่ ถ้ามีก็แจ้งให้ผู้ใช้ทราบ สาเหตุที่ไม่ใช้ค่าที่ส่งกลับทั้งหมด เพราะว่า สำหรับเหตุการณ์เดียวกัน จะมีค่าส่งกลับมาหลายค่า

4. พิมพ์ข้อมูลจนถึงตำแหน่งที่ผู้ใช้กำหนด หรือจนกระทั่งหมดข้อมูลใน Buffer

Alogorithm ในการพิมพ์เอกสาร

สำหรับในการพิมพ์เอกสารจะพิมพ์ทีละ 1 หน้า ตามจำนวนที่ผู้ใช้กำหนดได้แก่จำนวนบรรทัดต่อหน้า หน้าเริ่มต้น หน้าสุดท้าย ตำแหน่งกั้นขวา ถ้าไม่มีการกำหนดค่าจากผู้ใช้จะใช้ค่า มาตราฐานที่ได้กำหนดไว้แล้ว

จำนวนตำแหน่งหน้าเริ่มต้นและหน้าสุดท้ายในการพิมพ์

Page = StartPage -1

Endpage = EndPage -1

do{

คำนวนตำแหน่งบรรทัดที่จะพิมพ์ใน Buffer

บรรทัดแรกที่จะพิมพ์ = หน้าที่จะพิมพ์ * จำนวนบรรทัดต่อหน้า

บรรทัดสุดท้ายที่จะพิมพ์ = บรรทัดแรก + จำนวนบรรทัดต่อหน้า

พิมพ์ข้อมูลในหน้านั้น เริ่มพิมพ์ข้อมูลจากบรรทัดแรกถึงบรรทัดสุดท้ายจากที่

คำนวนได้โดยส่งบรรทัดแรกและบรรทัดสุดท้ายไปให้ function พิมพ์ทีละหน้า

if (ข้อมูลที่จะพิมพ์สิ้นสุด)

return

else

Page ++ พิมพ์หน้าถัดไป

} พิมพ์จนครบจำนวนหน้าที่กำหนด

Algorithm **ในการพิมพ์ข้อมูล ตามจำนวนบรรทัดที่กำหนด**

กำหนดให้ pointer ของบรรทัด ชี้ไปที่บรรทัดแรกที่จะพิมพ์

กำหนดให้ colum ที่จะพิมพ์ชี้ไปที่ colum แรก

if (ตรวจสอบว่า เป็นจุดสิ้นสุดข้อมูลหรือไม่)

Return o;

else

พิมพ์ข้อมูล ณ.บรรทัดที่ pointer ชี้จนหมดบรรทัด

เพิ่มตำแหน่ง pointer ของบรรทัดที่จะพิมพ์ขั้นหนึ่ง

if(ถึงบรรทัด สุดท้ายที่จะพิมพ์หรือไม่)

return;

else

กลับไปพิมพ์บรรทัดที่ pointer ชี้

ในการนำข้อมูลเข้ามาเก็บใน Buffer จะต้องมีการกำหนดตำแหน่งสิ้นสุดของข้อมูลไว้เสมอ เมื่อมีการอ่านข้อมูลใน Buffer สำหรับการประมวลผลอย่างใดอย่างหนึ่ง เช่น การแปลงข้อมูล ใน buffer การพิมพ์ข้อมูล หรือการเก็บข้อมูล จะต้องมีการตรวจสอบรหัสพิเศษนี้ สำหรับรหัสพิเศษที่กล่าวถึงเป็นตัวอักษรภาษาอังกฤษคือ !BRL สาเหตุที่ต้องใช้ถึง 4 ตัว ก็เพื่อป้องกัน กรณีที่ข้อมูลใน Buffer มาตรงกับรหัสพิเศษเหล่านี้พอดี ทำให้การตรวจสอบของโปรแกรมผิดพลาด

Algorithm ในการใส่เครื่องหมายสิ้นสุดข้อมูลใน Buffer

กำหนด เครื่องหมายพิเศษที่จะใช้ = !BRL
ทำเครื่องหมายพิเศษใส่ใน Buffer ณ. ที่ Index ชี้อยู่
เพิ่มตำแหน่ง Index ขั้นหนึ่ง
if(ครบตามจำนวน เครื่องหมายพิเศษที่จะใส่)
Return 1;
else
กลับไปทำเครื่องหมายพิเศษใส่ใน Buffer

Algorithm สำหรับการตรวจหาเครื่องหมายพิเศษ

กำหนดเครื่องหมายพิเศษที่จะหา
กำหนดค่า flag แสดงการตรวจสอบ
if(ข้อมูลใน Buffer == เครื่องหมายพิเศษ)
flag=1
เพิ่มตำแหน่ง pointer ชี้ข้อมูล
เพิ่มตำแหน่ง pointer เครื่องหมายพิเศษ
else
สิ้นสุดการตรวจสอบ
Return o;
ทำจนกระทั่งครบจำนวนอักขระที่จะตรวจสอบ

บทที่ 9

การออกแบบ Menu และส่วนติดต่อกับผู้ใช้

ลักษณะ Menu ของโปรแกรมที่จะเป็น Menu แบบกราฟฟิก เนื่องจากการแสดงผล อักษรเบรลล์นั้นต้องแสดงใน Mode กราฟฟิกอยู่แล้ว

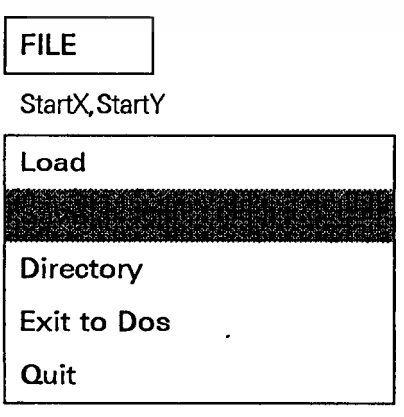
การออกแบบ Menu นั้น มีความยืดหยุ่น ในการใช้งานเพิ่มเติม แก้ไขได้ง่ายเพื่อความสะดวกในการพัฒนาในโอกาสต่อไป ยกตัวอย่างเช่น การเพิ่มหรือลดขนาดของกรอบเมนูการเปลี่ยนหัวข้อ เมนู การเพิ่มหรือลดจำนวนเมนูย่อย หรือเมนูหลัก

Menu จะเป็นแบบ window มี Menu หลักๆและแต่ละ Menu ก็จะประกอบด้วยตัวเลือกย่อย ๆ ภายใน Menu นั้น

Data Structure ของ Mainwindow

```
Struct window_frame{
    int Startx,starty , endx,endy;
    Char *header;
    void far*;
    int Maonstate;
    int Substate;
    int Maxsubrwn;
}frame[MAXFRAME];
```

แต่ละ Field ตัวแปรใน data structure ของ Mainwindow จะหมายถึงตำแหน่งต่างๆ บน Window ที่จอภาพคือ



Dat Structure ของ Sub Window

```
Struct  Subwin_frame{
        int Startx,starty;
        Char  *header;
        int  active;
} subwin[25];
```

ตัวแปรแต่ละตัวใน subwin_tarme หมายถึงตำแหน่งต่างๆบน จอภาพของ Subwindow นั้นๆ คือ

StartX,StartY

Exit to Dos

จาก Data Structure คำที่ได้กล่าวมาแล้ว เมื่อเราต้องการสร้าง window สำหรับติดต่อกับผู้ใช้ไม่ว่าจะเป็นรูปแบบใด ตำแหน่งใดบนจอภาพชื่อ window อะไรก็ตาม สามารถสร้างได้อย่างง่าย ๆโดยการกำหนดค่า ต่างๆ ให้กับแต่ละตัวแปร ของ Mainwindow นั้นออกหน้าจอภาพโปรแกรมก็จะแสดงตามตำแหน่งและขนาดตาม ที่เก็บไว้ใน Data structture นั่นเอง และถ้าต้องการลบ window ออกจาก จอภาพ ก็ไปลบที่ตำแหน่ง การแสดงผลของ window นั้น

การสร้าง Main window จะใช้ function Make Main Window ในการ สร้างโดยมีรูปแบบดังนี้คือ

Function การสร้าง Subwindow

```
MakesSubwindows(num,header,startx,starty)
int num;
char *header;
int startx,starty;
{
subwin[num].startx=startx;
subwin[num].starty=starty;
```

```
subwin[num].header=header;
subwin[num].active=0;
return 1;
}
```

Function การสร้าง Mainwindow

```
MakeMainWindows(num,header,maxsubwin,startx,starty,endx,endy)
int num;
char *header;
int maxsubwin;
int startx,starty,endx,endy;
{
frame[num].startx=startx;frame[num].endx=endx;
frame[num].starty=starty;frame[num].endy=endy;
frame[num].header=header;
frame[num].mainstate=0;
frame[num].maxsubwin=maxsubwin;
return 1;
}
```

การ Initial MainWindows และ Sub Windows

```
initwindows()
{
makemainwindows(0,"FILE",5,4X,6Y,29X,30Y);
makemainwindows(1,"EDIT",2,14X,6Y,39X,22Y);
makemainwindows(2,"TRANS",4,24X,6Y,48X,30Y);
makemainwindows(3,"PRINT",5,34X,6Y,58X,30Y);
makemainwindows(4,"OPTION",2,44X,6Y,68X,18Y);
```

การสร้างเมนูย่อยของ FILE

```
makesubwindows(1,"Load          ",5X,12Y);
makesubwindows(2,"Save          ",5X,15Y);
makesubwindows(3,"Directory     ",5X,18Y);
makesubwindows(4,"Exit to Dos   ",5X,21Y);
makesubwindows(5,"Quit          ",5X,24Y);
```

การสร้างเมนูย่อยของ EDIT

```
makesubwindows(6,"New File      F3",15X,12Y);
makesubwindows(7,"Filename      F2",15X,15Y);
```

การสร้างเมนูย่อยของ TRANS

```
makesubwindows(11,"Translate...  ",25X,12Y);
makesubwindows(12,"Check Data...  ",25X,15Y);
makesubwindows(13,"Simulate...   ",25X,18Y);
makesubwindows(14,"Set Right Margin ",25X,21Y);
```

การสร้างเมนูย่อยของ PRINT

```
makesubwindows(16,"Print ....    ",35X,12Y);
makesubwindows(17,"Set Line/Page ...",35X,15Y);
makesubwindows(18,"Set Begin Page...",35X,18Y);
makesubwindows(19,"Set End  Page...",35X,21Y);
makesubwindows(20,"Set Right Marg...",35X,24Y);
```

การสร้างเมนูย่อยของ OPTION

```
makesubwindows(21,"Set Menu Color...",45X,12Y);
makesubwindows(22,"Set Right Marg...",45X,15Y);
```

จาก Data Structure ทำให้ Main menu แต่ละตัวสามารถมี Sub meun ได้ 5 ตัวเลือก และแต่ละตัวเลือกของ Sub menu จะมีค่า address ประจำ Sub Window แน่นอน ดังนี้

Main Menu 1 : File

Sub 1 : Load	หมายเลขประจำ Menu คือ 1
Sub 2 : Save	หมายเลขประจำ Menu คือ 2
.	
Sub 5 : Quit	หมายเลขประจำ Menu คือ 5

Main Menu 2 : Edit

Sub 1 : New file	หมายเลขประจำ Menu คือ 6
Sub 2 : Open	หมายเลขประจำ Menu คือ 7

Main Menu 3 : Translate

Sub 1 : Translate	หมายเลขประจำ Menu คือ 11
Sub 2 : Check data	หมายเลขประจำ Menu คือ 12
Sub 3 : Simulate	หมายเลขประจำ Menu คือ 13
Sub 4 : Set Righ Margin	หมายเลขประจำ Menu คือ 14

Main Menu 5 : Option

Sub 1 : Set Menu color	หมายเลขประจำ Menu คือ 21
Sub 2 : Set Margin	หมายเลขประจำ Menu คือ 22

ดังนั้นเมื่อเราทราบว่า Main menu ใด Active อยู่ (รู้ได้จากตัวแปร mactive) แล้ว Menu ย่อยตัวใด Active อยู่ก็สามารถที่จะคำนวณหาหมายเลขประจำ Menu จริง ๆ เพื่อนำไปเป็น Index ในการ execute คำสั่งได้ ลักษณะการคำนวณเป็นดังนี้คือ

```
addr = mactive*5 + SubActive
```

นำค่า addr ที่ได้ไป switch เพื่อทำตามคำสั่งตามหมายเลขประจำเมนูที่ได้กล่าวมาแล้ว โดย function ExecuteCommand ( );

Algarithm ในการรับค่า Keyboard จากการเลือก Manu ของ User

รอรับ Key จนกระทั่งผู้ใช้กด Key ESC
if (key==0) หมายความว่ามีการกด control key
Switch(key)
case : ปุ่ม ซ้ายหรือขวา
clear Mainwindow ปัจจุบัน
Show Mainwindow ถัดไป
set ค่า Main Actire เป็น window ที่จอภาพ
Case: ปุ่มบน ล่าง หรือ SPace bare
Clear sub window ปัจจุบันให้อยู่ในสภาวะ NON ACTIVE
Show subwindow ตำแหน่งใหม่ เป็นสภาวะ Active
Set Active Menu
Case: Enter
ทำตาม Menu ที่ ตัวแปร Active_Menu ชี้อยู่
else{
กรณีที่ผู้ใช้กด key ธรรมดาทั่วไป

ส่งสัญญาณเตือน beep 1 ครั้ง ออกทางลำโพง
}

ตัวอย่าง Function Select เพื่อรับค่าตัวเลือกจากผู้ใช้

select( ) ตัวอย่าง Funetion Seclect ซึ่งจะรับ Key board จาก User แล้วนำไปตรวจสอบและปฏิบัติตามการทำงาน พร้อมทั้งแสดงและลบ เมนูด้วย

```
{
char key;
showmain(oldmain);
แสดง Menu เก่าล่าสุดที่แสดงครั้งสุดท้าย
while((key=getch())!=27)
รอการลด Key จากผู้ใช้จนกระทั่งพบ Key ESC
{
        if(key==0)
        ถ้า KEY ตัวแรกเป็นศูนย์แสดงว่าเป็น Key พิเศษ
        switch(getch())
                ตรวจสอบ KEY ตัวต่อไปว่าเป็นรหัส KEY อะไร
                case R_ARROW:       กรณีเป็นลูกศรขวา เป็นการเลื่อน Mainmenu
ไปทางขวา
                                    mactive=uppos(mactive,MAXMENU);
                                    คำนวณหา หมายเลข Mainmenu ใหม่ที่จะ Active
                                    activemenu( );
                                    Active Menu
                                    oldmain=mactive;
                                    นำค่าหมายเลข window ที่ active ไปเก็บเป็นตัวแปร old
                                    break;
            case L_ARROW:   กรณีเป็น KEY ลูกศรซ้ายเป็นการเลื่อน Main menu
ไปทางซ้าย
                                    mactive=downpos(mactive,MAXMENU);
                                    คำนวณตำแหน่งใหม่ของ Mainmenu ที่ active
```

```
                activemenu();
                แสดง Menu ใหม่ทางจอภาพ
                oldmain=mactive;
                เก็บค่าหมายเลข Menu ที่ active เป็นค่า old
                break;
case D_ARROW:   กรณีเป็น KEY ลูกศรลงเป็นการเลื่อน Submenu ของ Main
                 Menu ที่ Active อยู่ขณะนั้น
                oldsub=frame[mactive].substate;
                เก็บค่า Status ของ Subwindows เดิมก่อนจงเปลี่ยนค่า active
               sactive=subuppos(frame[mactive].substate,
               frame[mactive].maxsubwin);
               คำนวณตำแหน่งหมายเลข Subwindows ใหม่โดยคำนวณจาก Main
               menu   ที่ active อยู่ขณะนั้น
                        frame[mactive].substate=sactive;
                        เก็บค่า Windows ที่ active ใน active ใน Mainwindows
                        activesubmenu();
                        แสดง Submenu ออกทางจอภาพ
    break;
case U_ARROW:  กรณีเป็น Key ลูกศรนั้นเป็นการเลื่อน Submenu ขึ้น 1ตำแหน่ง
oldsub=frame[mactive].substate;   เก็บ ค่า Subactive เดิมไว้ในตัวแปร
        active=subdownpos(frame[mactive].substate,
        frame[mactive].maxsubwin);
        คำนวณตำแหน่งของ Subactive ใหม่จาก Mainmenu ขณะนั้น
            frame[mactive].substate=sactive;
            เก็บค่าที่คำนวณได้ใน Mainwindows
            activesubmenu();
            แสดง Submenu ทางจอภาพด้วย Status active
```

```
        case F3:  กรณีเป็น KEY F3 จงเปลี่ยนโหมดการแสดงผล
                  จากภาษาไทยอังกฤษเป็นอักษรเบรลล์
            Mode=!Mode;
            PrintMode();
            break;
    }
    }
    else if(key=='\r')        ถ้าเป็น ENTER
    {
                restorescreen(mactive);
                เก็บภาพหน้าจอขณะนั้นไว้ในหน่วยความจำ
                ExecuteCommand();
                ทำตามคำสั่งโดยคำนวณจากหมายเลข Subactive ที่ active ขณะนั้น
                showmain(mactive);
            }       กลับมาแสดง Mainwindows หลังจากทำคำสั่งแล้ว
    else  if(key==SPACBAR){          ถ้าเป็น KEY Spacbar จะเลื่อนการแสดง Submenu
     oldsub=frame[mactive].substate;        การทำงานเหมือน Key ลูกศรลง
                sactive=subdownpos(frame[mactive].substate,
               frame[mactive].maxsubwin);
                    frame[mactive].substate=sactive;
                    activesubmenu();
    else    ถ้าเป็นการกดตัวอักษรทั่วไป
    printf('\007');
    }       ส่งเสียงเตือนบอกความผิดพลาด
    }
```

Function execute คำสั่ง ตามหมายเลข menu

```
ExecuteCommand()
int menu;
menu=subaddress(mactive,sactive);
คำนวณหมายเลข Menu จาก Mainactive และ Subactive
switch(menu)
{       Execute คำสั่งตามหมายเลข Menu
        case 1:
                ExcuteLoadFile();
                break;
        case 6:
                LoadFile1(fname);
                edt();
                break;
        case 11:
        TranslateFile();
          break;
        case 16:
          ExecutePrintFile();
          break;
        case 17:
         E17SetLinePage();
         break;
        case 18:
         E18SetStartPage();
         break;
```

```
            case 19:
             E19SetEndPage();
             break;
            case 20:
             E20SetRightMarg();
             break;
    default:
             break;
    }
    }
```

บทที่ 10
สรุป

จากการทำงานที่ผ่านมา ความสัมพันธ์ระหว่างอักษรทั่วไปกับอักษรเบรลล์นั้นมีความคล้ายคลึงกันในด้านการอ่านและการเขียนมาก ดังนั้นจึงทำให้การเขียนโปรแกรมแปลงไม่ค่อยยุ่งยากซับซ้อนมากนัก ดังนั้นผลการแปลงที่ได้จึงเป็นที่น่าพอใจ แต่จะมีบางกรณีที่การเขียนโปรแกรมและการออกแบบจะมีความยุ่งยากและซับซ้อน ก็คือคำที่เกิดจากสระผสม เนื่องจากมีความแตกต่างกับอักษรทั่วไปอย่างสิ้นเชิง ดังนั้นจึงแก้ปัญหาโดยเก็บกฎการแปลงสระเหล่านี้ไว้ใน dictionary ทำให้ผู้ใช้สามารถเพิ่มเติมและเปลี่ยนแปลงได้โดยง่าย โดยสรุปและผลการแปลงที่ได้ เป็นที่น่าพอใจ

สำหรับในส่วนอื่นๆ ก็เป็นส่วนประกอบซึ่งถือว่าเป็น tool ของโปรแกรม ได้แก่ editor สามารถพิมพ์ข้อมูลจัดเก็บข้อมูล นำข้อมูลที่ได้จากการแปลงมาตรวจสอบ แก้ไข เพิ่มเติมได้ในระดับหนึ่ง ซึ่งการใช้งานก็ยังไม่ดีเท่าที่ควร เนื่องจากสามารถใช้งานได้ในระดับพื้นฐาน เท่านั้น เช่นการลบตัวอักษรและการแทรกตัวอักษร ส่วนการค้นหา การกั้นบล๊อก ยังทำไม่ได้ แต่มีข้อดีคือสามารถแสดงผลได้ทั้งภาษาไทย ภาษาอังกฤษ และรูปอักษรเบรลล์ ทำให้ผู้ใช้มองเห็นรูปลักษณะของอักษรเบรลล์ได้ดีขึ้น

การพิมพ์อักษรเบรลล์ออกเครื่องพิมพ์ ก็เช่นกัน พิมพ์ได้ดีในระดับหนึ่งสามารถกำหนดหน้าที่เริ่มต้นที่จะพิมพ์ หน้าสุดท้าย จำนวนบรรทัดต่อหน้า สำหรับข้อที่ควรปรับเพิ่มเติมได้แก่ การเลือกชนิด เครื่องพิมพ์ การเลือก port สำหรับต่อเครื่องพิมพ์เป็นต้น

การแสดงผลอักษรเบรลล์ภาษาไทย ภาษาอังกฤษ สามารถใช้งานได้ในระดับที่ดี เนื่องจากใช้วิธีการแสดงผลแบบ direct video ram ทำให้มีความเร็วในการแสดงผล

แนวทางในการพัฒนา

สำหรับโปรแกรมในส่วนของการแปลงข้อมูลถือว่ามีความสมบูรณ์เพียงพอแล้ว ส่วนที่เห็นควรมีการพัฒนาต่อไปได้แก่

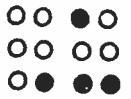
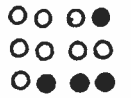
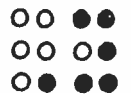
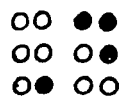
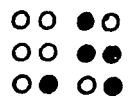
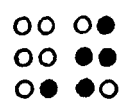
Editor : เพิ่มความสามารถ copy การ move การค้นหา การกั้นบล๊อก สำหรับข้อมูล

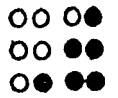
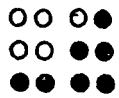
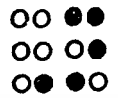
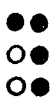
Print : พัฒนาให้สามารถพิมพ์จุดอักษรเบรลล์ที่ปรากฏบนจอภาพ ออกทางเครื่องพิมพ์ธรรมดาทั่วไปได้ ซึ่งเป็นการพิมพ์ในโหมดกราฟฟิก

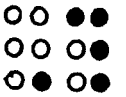
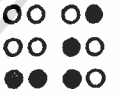
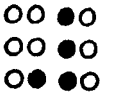
TransLate : ให้สามารถเปรียบเทียบระหว่างข้อมูลที่แปลงกับผลการแปลงทีละบรรทัดได้เพื่อตรวจสอบความถูกต้องของการแปลงให้ถูกต้องมากยิ่งขึ้น

ภาคผนวก

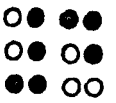
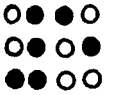
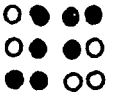
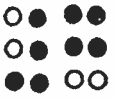
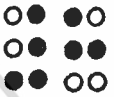
ตารางเปรียบเทียบ ภาษาไทย ภาษาอังกฤษ อักษร Braille

พยัญ ชนะภาษาไทย	พยัญ ชนะภาษาอังกฤษ	รูปอักษรเบรล์ล
ก	g	
ข	k	
ค	u	
ฆ	,u	
ง	]	
จ	j	
ฉ	/	
ช	+	
ซ	I	
ฌ	,+	
ญ	,y	
ฎ	,d	
ฏ	,/	
ฐ	-\	

พยัญ ชนะภาษาไทย	พยัญ ชนะภาษาอังกฤษ	รูปอักษรเบรล์ล
ฑ	,)	
ฒ	-)	
ณ	,m	
ด	d	
ต	\	
ถ	t	
ท	)	
ธ	0)	
น	n	
บ	v	
ป	&	
ผ	p	
ฝ	x	
พ	?	

พยัญชนะภาษาไทย	พยัญชนะภาษาอังกฤษ	รูปอักษรเบรล์ล
ฟ	$	
ภ	,?	
ม	m	
ย	y	
ร	r	
ฤ	/1	
ล	l	
ว	w	
ศ	,s	
ษ	-s	
ส	s	
ห	h	
พ	,l	
อ	o	

พยัญชนะภาษาไทย	พยัญชนะภาษาอังกฤษ	รูปอักษรเบรล์ล
ฮ	-	●● ○○ ●● ○○ ●● ○○
ะ	a	●○ ○○ ○○
า	*	●○ ○○ ○●
ำ	z	●○ ○● ●●
เ	f	●● ●○ ○○
แ	<	●○ ●○ ○●
โ	i	○● ●○ ○○
ใ	:1	●○ ○○ ○● ●○ ○● ○○
ไ	:	●○ ○● ○●
ๆ	1 -	○○ ●○ ○○
ฯ	;2	○○ ○○ ○● ●○ ○● ●○
์	c	●● ○○ ○○
๋	3	○○ ●● ○○
็	6	●○ ●○ ○○

พยัญ ชนะภาษาไทย	พยัญ ชนะภาษาอังกฤษ	รูปอักษรเบรล์ล
4	#d	
5	#e	
6	#f	
7	#g	
8	#h	
9	#i	

กิตติกรรมประกาศ

ทางผู้จัดทำโครงงานโปรแกรม แปลงอักษรเบรลล์ ขอขอบคุณอาจารย์ที่ปรึกษา รศ.ดร. ครรชิต ไมตรี ที่ให้คำแนะนำปรึกษา และสนับสนุนโครงงาน ขอขอบคุณอาจารย์ ทัศนีย์ กาญจนิษฐิติ ที่เอื้อเฟื้อข้อมูลและให้คำปรึกษาอย่างดีเกี่ยวกับอักษรเบรลล์

ขอขอบคุณบิดามารดาของผู้จัดทำที่ให้กำลังใจในการทำโครงงานนี้ตลอดมา และขอขอบคุณบุคคลอื่นๆ ที่มีส่วนช่วยให้โครงงานนี้สำเร็จลงด้วยดี

บรรณานุกรม

1. ทัศนีย์ กาญจนิษฐิติ , "อักษรสำหรับคนตาบอด" ,โรงเรียนสอนคนตาบอด กรุงเทพฯ, 123 หน้า , 2529
2. ศิริวัฒน์ ศิวะบวร , พรชัย จักรธำรงค์ , จิรศักดิ์ ชัยวิริยะกุล , "การประยุกต์ใช้งานภาษาซี" ,608 หน้า , 2527
3. มนตรี พจนารถลาวัณย์ ,"การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี" , บริษัท-ซีเอ็ดยูเคชั่น , 346 หน้า , 2521
4. พ.อ เจนวิทย์ เหลืออร่าม , "การใช้ Turbo C ++ เขียนโปรแกรมภาษา C" , สถาบัน-บัณฑิตพัฒนบริหารศาสตร์ , สำนักพิมพ์สุขภาพใจ , 416 หน้า , 2521
5. พันธุ์ปิติ เปี่ยมสง่า , "มาสร้าง สกรีนเอดิเตอร์กันเถอะ" , วารสาร
6. Jack Purdum , "C Programmer's Toolkit" , Programming Series , 350,1989
7. Peter Norton , "PC Programmer's bible", Penguim Books , 615 , 1973
8. Mare J. Rochkind , "Advanced C Programming for Displays" , prentice Hall solfware Series , 331 , 1970
9. Herbert Schildt , "Turbo C The complete Reference" United State of America : McGraw-Hill , 771 , 1988