บทที่ 3

แนวคิดและทฤษฎี

จากบทที่แล้วได้ทำการศึกษาถึงเรื่องการจัดการด้านการเรียนการสอนที่ใช้ทั่วไป รวมถึงเรื่องการคำนวณค่าสถิติต่าง ๆ ที่มีการใช้ในการวัดผลและวิธีการวิเคราะห์ข้อสอบ ขั้นตอนถัดไปคือต้องทำการศึกษาวิธีการนำเอาคอมพิวเตอร์เข้ามาประยุกต์ใช้ในระบบการจัดการด้านการเรียนการสอนที่ได้ศึกษาไว้แล้ว โดยผู้วิจัยได้ทำการศึกษาแนวคิดและทฤษฎีทางคอมพิวเตอร์ในเรื่องต่าง ๆ ดังหัวข้อต่อไปนี้

3.1 ลักษณะการจัดเก็บข้อมูล

ข้อมูลที่ใช้ในระบบการจัดการด้านการเรียนการสอนสามารถจำแนกได้เป็น 2 ประเภท คือ ข้อมูลที่เป็นข้อความและข้อมูลที่เป็นรูปภาพ ซึ่งการนำข้อมูลเหล่านี้ไปจัดเก็บไว้ในภาษาคอมพิวเตอร์มีลักษณะการจัดเก็บที่แตกต่างกันตามลักษณะของข้อมูล โดยมีรายละเอียดดังนี้

3.1.1 การจัดเก็บข้อมูลแบบข้อความ

ข้อมูลที่นำมาจัดเก็บจะเป็นลักษณะที่ประกอบด้วย ตัวอักษร ตัวเลข เช่น ในระบบการเรียนการสอน ข้อมูลที่จะจัดเก็บได้แก่ รหัสผู้เรียน ชื่อ-นามสกุลผู้เรียน เป็นต้น ตัวอย่างข้อมูลนี้จะแสดงในรูปที่ 3.1

รหัสผู้เรียน	ชื่อ-นามสกุล		กลุ่มเรียน
C200000	สิริรัตน์	ทิพวงศา	วิศวกรรมศาสตร์
C200111	ดนุช	ตันเทอดทิตย์	นิติศาสตร์
C201023	นวลจันทร์	จิตตนาสวัสดิ์	อักษรศาสตร์

รูปที่ 3.1 แสดงข้อมูลที่มีลักษณะการจัดเก็บข้อมูลแบบข้อความ

โดยการเก็บข้อมูลในระบบคอมพิวเตอร์นั้นจัดเก็บลงหน่วยความจำสำรองของคอมพิวเตอร์ในรูปแบบของแฟ้มข้อมูล การบันทึกข้อมูลลงแฟ้มข้อมูลจะต้องมีวิธีการหรือเทคนิคในการจัดเก็บ เพื่อให้เกิดความสะดวกและรวดเร็วในการค้นหาข้อมูลที่ต้องการ โดยเทคนิคการจัดแฟ้มข้อมูลสามารถแบ่งออกเป็น 3 แบบ ได้แก่

3.1.1.1 แฟ้มข้อมูลแบบเรียงลำดับ (Sequential File)

มีลักษณะการจัดเก็บข้อมูลโดยเรียงลำดับระเบียนตามลำดับของเขตข้อมูลที่ใช้เป็นคีย์ การปฏิบัติหรือการเปลี่ยนแปลงข้อมูลแต่ละระเบียนในแฟ้มข้อมูลแบบนี้ ต้องทำอย่างต่อเนื่องตั้งแต่ระเบียนแรกจนถึงระเบียนสุดท้าย การจัดแฟ้มแบบนี้จะเหมาะกับงานที่มีการเปลี่ยนแปลงข้อมูลมาก และมีการเรียกใช้ข้อมูลเป็นจำนวนมาก โดยมีระยะเวลาที่แน่นอน เช่น ทุกสิ้นวัน ทุกสิ้นเดือน

3.1.1.2 แฟ้มข้อมูลแบบสุ่ม (Random File)

การจัดแฟ้มแบบสุ่มนี้เหมาะกับงานที่มีความต้องการเข้าระเบียนใด ๆ ในแฟ้มข้อมูลทันทีโดยไม่ต้องอ่านระเบียนตั้งแต่ต้นแฟ้ม ในการบันทึกข้อมูลลงแฟ้มข้อมูลแบบนี้จะต้องนำเขตข้อมูลที่เป็นคีย์ไปผ่านฟังก์ชันความสัมพันธ์ระหว่างค่าคีย์กับตำแหน่งเพื่อเปลี่ยนค่าคีย์ให้เป็น ที่อยู่ (address) ซึ่งเป็นตำแหน่งสำหรับบันทึกระเบียน และการเข้าถึงข้อมูลจะใช้วิธีคำนวณผ่านฟังก์ชันเดียวกันนี้ เพื่อให้ได้ระเบียนข้อมูลที่ต้องการ

3.1.1.3 แฟ้มข้อมูลแบบเรียงลำดับเชิงดรรชนี (Index Sequential File)

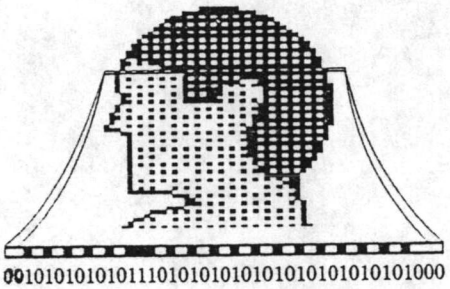
เป็นการจัดเก็บข้อมูลที่รวมเอาข้อดีของการจัดแฟ้มข้อมูลแบบเรียงลำดับและแบบสุ่มเอาไว้ด้วยกัน โดยระเบียนต่าง ๆ ของแฟ้มข้อมูลจะถูกบันทึกติดต่อกันไป ทำให้หน่วยความจำสำรองถูกใช้อย่างเต็มที่ แต่ขณะเดียวกันการค้นหาระเบียนสามารถทำได้รวดเร็ว เนื่องจากการจัดเก็บแบบนี้มีการกำหนดเขตข้อมูลหนึ่งเป็นคีย์ซึ่งจะใช้ในการอ้างอิงระเบียนเมื่อต้องการอ่านหรือบันทึก และเพื่อให้การอ่านหรือบันทึกทำได้รวดเร็วจะมีการสร้างแฟ้มอีกแฟ้มหนึ่งเพื่อเก็บดรรชนี คือเก็บ

ข้อมูลที่กำหนดให้เป็นคีย์คู่กับตัวชี้ที่ชี้ไปยังตำแหน่งของระเบียนที่มีคีย์ค่านั้นอยู่ สำหรับแฟ้มข้อมูลที่เก็บดรรชนีนั้นโปรแกรมระบบที่ทำหน้าที่เกี่ยวกับการจัดเก็บข้อมูลแบบเรียงลำดับเชิงดรรชนีจะเป็นตัวสร้างให้ โดยที่โครงสร้างข้อมูล (Data Structure) ของแฟ้มข้อมูลที่เก็บดรรชนีจะต้องเป็นโครงสร้างที่ทำให้ค้นหาคีย์ได้รวดเร็ว ซึ่งโครงสร้างข้อมูลที่นำมาใช้จัดเก็บดรรชนีของแฟ้มข้อมูลมีหลายแบบ เช่น โครงสร้างแบบต้นไม้ (Tree) โครงสร้างลิงค์ลิส (Link List) โครงสร้างแบบต้นไม้แบบทวิภาค (Binary Tree) โครงสร้างบีทรี (B-Tree) แต่ละโครงสร้างข้อมูลจะมีลักษณะและข้อดีข้อเสียที่แตกต่างกันไป ในงานวิจัยนี้ได้ทำใช้โครงสร้างข้อมูลแบบบีทรี (B-Tree) เนื่องจากโครงสร้างข้อมูลแบบนี้เป็นโครงสร้างข้อมูลที่โปรแกรมบีทรีฟ (Btrieve) ได้นำมาใช้จัดแฟ้มดรรชนี ซึ่งโปรแกรมบีทรีฟนี้เป็นโปรแกรมจัดการแฟ้มข้อมูลที่ผู้วิจัยเลือกมาใช้จัดการแฟ้มข้อมูลดรรชนีของระบบการจัดการด้านการเรียนการสอนด้วยคอมพิวเตอร์ โดยรายละเอียดของโครงสร้างข้อมูลแบบบีทรีและโปรแกรมบีทรีฟได้อธิบายรายละเอียดในหัวข้อ 3.2 และ 3.4 ตามลำดับ

3.1.2 การจัดเก็บข้อมูลแบบกราฟิก

เป็นการจัดเก็บข้อมูลที่มีลักษณะเป็นรูปภาพ รูปแบบของการจัดเก็บข้อมูลประเภทนี้แบ่งเป็น 2 ประเภทใหญ่ คือ

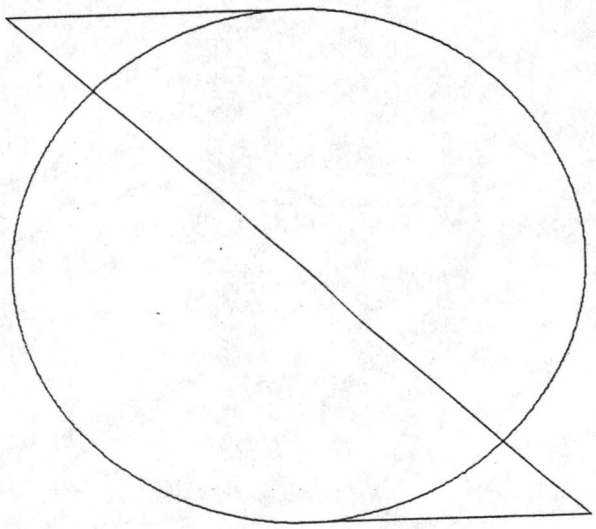
3.1.2.1 แฟ้มข้อมูลแบบบิตแมพ คือการเก็บแผนที่บิตต่าง ๆ ที่ประกอบขึ้นเป็นภาพกราฟิก หลักการของบิตแมพ คือภาพที่ได้จะประกอบด้วยจุดเล็ก ๆ เรียกว่า พิกเซล (Pixel) สำหรับภาพขาวดำในจอภาพแบบโมโนโครมจะแทนแต่ละพิกเซลในภาพด้วยข้อมูลเพียง 1 บิต แต่ละจุดของภาพมีสีขาวหรือดำก็ขึ้นอยู่กับค่าของบิต (1 หรือ 0) โดยรูปที่ 3.2 แสดงการแทนค่าข้อมูลบิตแมพใน 1 บรรทัด



รูปที่ 3.2 แสดงการแทนค่าข้อมูลแบบบิตแมพใน 1 บรรทัด

ถ้าเป็นสีเทาก็จะสามารถแสดงให้เห็นได้ด้วยการกระจายความหนาแน่นของจุดขาวดำคละกันไป แฟ้มข้อมูลบิตแมพบางชนิดสามารถจัดเก็บรูปภาพกราฟิกเป็นแบบเทาต่างระดับ (Gray Scale)ได้ โดยการจัดเก็บจะแตกต่างกับโมโนโครมตรงที่ แต่ละพิกเซลสามารถเก็บระดับความแตกต่างของสีเทาได้ถึง 256 ระดับ การจัดเก็บในลักษณะนี้ทำให้ได้รูปภาพกราฟิกที่สมบูรณ์กว่า การใช้เทคนิคนี้เรียกว่า ดิตเธอร์ (Dither : การใช้ความหนาแน่นของจุดดำและขาวแทนความแตกต่างของระดับสี) สำหรับการจัดเก็บภาพสีจะเก็บความแตกต่างของระดับสีเหมือนกับการเก็บความแตกต่างของระดับเทาในแต่ละพิกเซล แต่จำนวนสีที่แตกต่างกันได้นั้นก็จะขึ้นอยู่กับจำนวนบิตที่ใช้แทนแต่ละพิกเซล ตัวอย่างเช่นถ้าแต่ละพิกเซลถูกกำหนดด้วยข้อมูลจำนวน 4 บิต จะแทนได้ 16 สี ถ้าเป็น 8 บิตจะแทนได้ 256 สี และสำหรับ 16.7 ล้านสี จะแทนด้วยข้อมูล 24 บิตต่อ 1 พิกเซล สรุปได้ว่าจำนวนสีที่เพิ่มขึ้นนั้นขึ้นอยู่กับจำนวนบิตที่ใช้แทนข้อมูลแต่ละพิกเซลที่เพิ่มขึ้น

ดังนั้นขนาดของแฟ้มที่จัดเก็บก็จะมีขนาดเพิ่มขึ้นการจัดเก็บแฟ้มภาพสี 24 บิตขนาดเต็มจอภาพใช้เนื้อที่บนฮาร์ดดิสด์ถึง 16 เมกะไบต์ ตัวอย่างของแฟ้มข้อมูลแบบบิตแมพได้แก่ .BMP ของโปรแกรมวินโดว์บิตแมพ (Window Bitmap) .PCX ของโปรแกรมพีซีเพ้นท์บรัช (PC Paintbrush) ซึ่งรูปแบบนี้เป็นรูปแบบการจัดเก็บกราฟิกตัวแรกที่ใช้สำหรับเครื่องคอมพิวเตอร์พีซีและเป็นประเภทของแฟ้มข้อมูลกราฟิกที่ใช้แพร่หลาย ตัวอย่างภาพที่มีการจัดเก็บแบบบิตแมพที่มีรูปแบบ PCX แสดงไว้ในรูปที่ 3.3 และรูปที่ 3.4 แสดงข้อมูลรูปภาพของแฟ้มข้อมูลแบบบิตแมพที่มีรูปแบบ PCX ของรูปที่ 3.3



รูปที่ 3.3 แสดงภาพที่มีการจัดเก็บข้อมูลแบบบิตแมพที่มีรูปแบบ PCX

```
0A 05 01 01 01 00 01 00 80 02 5E 01 80 02 5E 01
00 00 00 00 00 AA 00 AA 00 00 AA AA AA 00 00 AA
00 AA AA 55 00 AA AA AA 55 55 55 55 55 FF 55 FF
55 55 FF FF FF 55 55 FF 55 FF FF FF 55 FF FF FF
00 04 50 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF FF FF FF FF FF FF FF C5 FF FF FF FF FF
FF FF FF FF FF FF C5 FF FF FF FF FF FF FF FF FF
FF FF C5 FF FF FF FF FF FF FF FF FF FF FF C5 FF
FF FF FF FF FF FF FF FF FF FF C5 FF FF FF FF FF
FF FF FF FF FF FF C5 FF FF FF FF FF FF FF FF FF
FF FF C5 FF FF FF FF FF FF FF FF FF FF FF C5 FF
FF FF FF FF FF FF FF FF FF FF C5 FF FF FF FF FF
FF FF FF FF FF FF C5 FF FF FF FF FF FF FF FF FF
FF FF C5 FF FF FF FF FF FF FF FF FF FF FF C5 FF
FF FF FF FF FF FF FF FF FF FF C5 FF C7 FF C1 E0
D0 00 FF FF C1 E0 D0 00 FF FF C1 E0 D0 00 FF FF
C1 E0 D0 00 F8 FF D2 FF C1 FC C3 00 7F C1 FF D0
00 0F F9 FF C1 FC C3 00 7F C1 FF D0 00 0F F9 FF
C1 FC C3 00 7F C1 FF D0 00 0F F9 FF C1 FC .....
```

รูปที่ 3.4 แสดงข้อมูลรูปภาพของแฟ้มข้อมูลบิตแมพที่มีรูปแบบ PCX ของรูปที่ 3.3

3.1.2.2 แฟ้มข้อมูลแบบเว็กเตอร์ จะมีการเก็บลักษณะรูปร่างของภาพแทนการเก็บเป็นจุดพิกเซลเหมือนแบบบิตแมพ โดยรูปร่างของภาพได้แก่ รูปสี่เหลี่ยม รูปวงกลม เส้นตรง หรือส่วนโค้ง เป็นต้น ซึ่งภายในแฟ้มข้อมูลจะมีการเก็บข้อมูลที่มีลักษณะเป็นข้อความคำสั่งที่เป็นภาษาอังกฤษ และตามด้วยข้อมูลต่าง ๆ ซึ่งจะบันทึกอยู่ในรูปของรหัสแอสกี้ ตัวอย่างเช่น รูปที่เป็นวงกลม มีรัศมี 100 จุดศูนย์กลางอยู่ที่ x = 2.25 y = 5 ข้อมูลที่ถูกบันทึกอยู่ในแฟ้มข้อมูลคือ circle(100,2250,5000) เป็นต้น โดยตัวอย่างการเก็บข้อมูลแบบเว็กเตอร์ซึ่งเป็นของแฟ้มข้อมูล .DXF ของโปรแกรมออโตแคด จะแสดงในรูปที่ 3.5 ซึ่งเป็นข้อมูลของรูปภาพในรูปที่ 3.3 จุดเด่นของการเก็บรูปภาพแบบเวกเตอร์คือความสามารถในการพิมพ์ที่สามารถพิมพ์ภาพออกมาด้วยขนาดเท่าใดก็ได้อย่างชัดเจนและสมบูรณ์ที่สุดเพราะเป็นการเก็บรูปร่างต่าง ๆ ซึ่งสามารถนำมาย่อหรือขยายได้ตามลักษณะของรูปร่างนั้น แฟ้มข้อมูลแบบเว็กเตอร์จะมีลักษณะการเก็บข้อมูลตัวอย่างของแฟ้มข้อมูลแบบเวกเตอร์ได้แก่ .DXF ของโปรแกรมออโตแคด เป็นต้น

```
  0
SECTION
  2
HEADER
  9
$CECOLOR
 62
     0
  0
ENDSEC
  0
SECTION
  2
TABLES
  0
TABLE
  2
LAYER
  0
LAYER
  2
0
 70
     0
 62
     7
  6
CONTINUO
US
  0
LAYER
  2
obj1
 70
     0
 62
     7
  6
CONTINUO
US
  0
ENDTAB
  0
ENDSEC
  0
SECTION
  2
ENTITIES
  0
CIRCLE
  8
obj1
 62
     7
 10
9.000000
0000
 20
7.500000
0000
 30
0.000000
0000
 40
1.750000
0000
  0
POLYLINE
  8
obj1
 62
     7
 66
     1
 70
     8
  0
VERTEX
  8
obj1
 62
     7
 10
9.000000
0000
 20
9.250000
0000
 30
0.000000
0000
 70
    32
  0
VERTEX
  8
obj1
 62
     7
 10
7.250000
0000
 20
9.250000
0000
 30
0.000000
0000
 70
    32
  0
VERTEX
  8
obj1
 62
     7
 10
10.75000
00000
 20
5.750000
0000
 30
0.000000
0000
 70
    32
  0
VERTEX
  8
obj1
 62
     7
 10
9.000000
0000
 20
5.750000
0000
 30
0.000000
0000
 70
    32
  0
SEQEND
  0
ENDSEC
  0
EOF
```

รูปที่ 3.5 แสดงข้อมูลรูปภาพจากแฟ้มข้อมูลแบบเวกเตอร์ .DXF ของออโตแคด

ข้อแตกต่างระหว่างการจัดเก็บแบบเวกเตอร์ และการจัดเก็บแบบบิตแมพ คือ

- สำหรับรูปภาพที่เหมือนกัน การเก็บข้อมูลแบบเวกเตอร์จะใช้เนื้อที่ในการเก็บรูปภาพน้อยกว่าการจัดเก็บแบบบิตแมพ เนื่องจากการเก็บข้อมูลแบบเวกเตอร์เก็บเป็นลักษณะคำสั่งในรูปของรหัสแอสกี้แต่การจัดเก็บบิตแมพจะเก็บจุดพิกเซล

- การจัดเก็บรูปภาพแบบเวกเตอร์จะใช้ได้กับรูปภาพที่มีลักษณะประกอบไปด้วยรูปร่าง สี ระดับสี ที่ง่าย ๆ ถ้าเป็นรูปภาพที่เป็นภาพลักษณ์ (Image) ที่มี

ความซับซ้อนในเรื่องของ สี ระดับสี และรูปร่าง ต้องจัดเก็บในรูปแบบของบิตแมพ

เนื่องจากในงานวิจัยนี้ต้องสร้างภาพที่มีลักษณะเป็นรูปภาพที่เป็นรูปร่างเช่น วงกลมรูปเหลี่ยมต่างๆ เส้นตรง เส้นโค้ง และรูปภาพที่มีลักษณะเป็นภาพลักษณ์ และต้องสามารถจัดเก็บรูปภาพเหล่านั้นได้ ผู้วิจัยจึงเลือกวิธีการจัดเก็บข้อมูลแบบพิกเซลของบิตแมพ ซึ่งเหมาะสมกว่าการจัดเก็บแบบเวกเตอร์และได้เลือกรูปแบบการเก็บข้อมูลบิตแมพแบบ PCX ของโปรแกรมพีซีเพ้นท์บรัช เนื่องจากรูปแบบ PCX ของโปรแกรมพีซีเพ้นท์บรัช เป็นที่นิยมใช้แพร่หลายในโปรแกรมวาดภาพที่มีอยู่ในปัจจุบัน ซึ่งจะทำให้ระบบที่พัฒนาขึ้นในงานวิจัยนี้สามารถดึงรูปภาพที่ถูกสร้างจากโปรแกรมอื่นมาใช้ได้ ทำให้ระบบที่พัฒนาขึ้นมีประสิทธิภาพมากขึ้น โดยรายละเอียดโครงสร้างการเก็บข้อมูลบิตแมพตามรูปแบบ PCX จะได้อธิบายในหัวข้อ 3.3

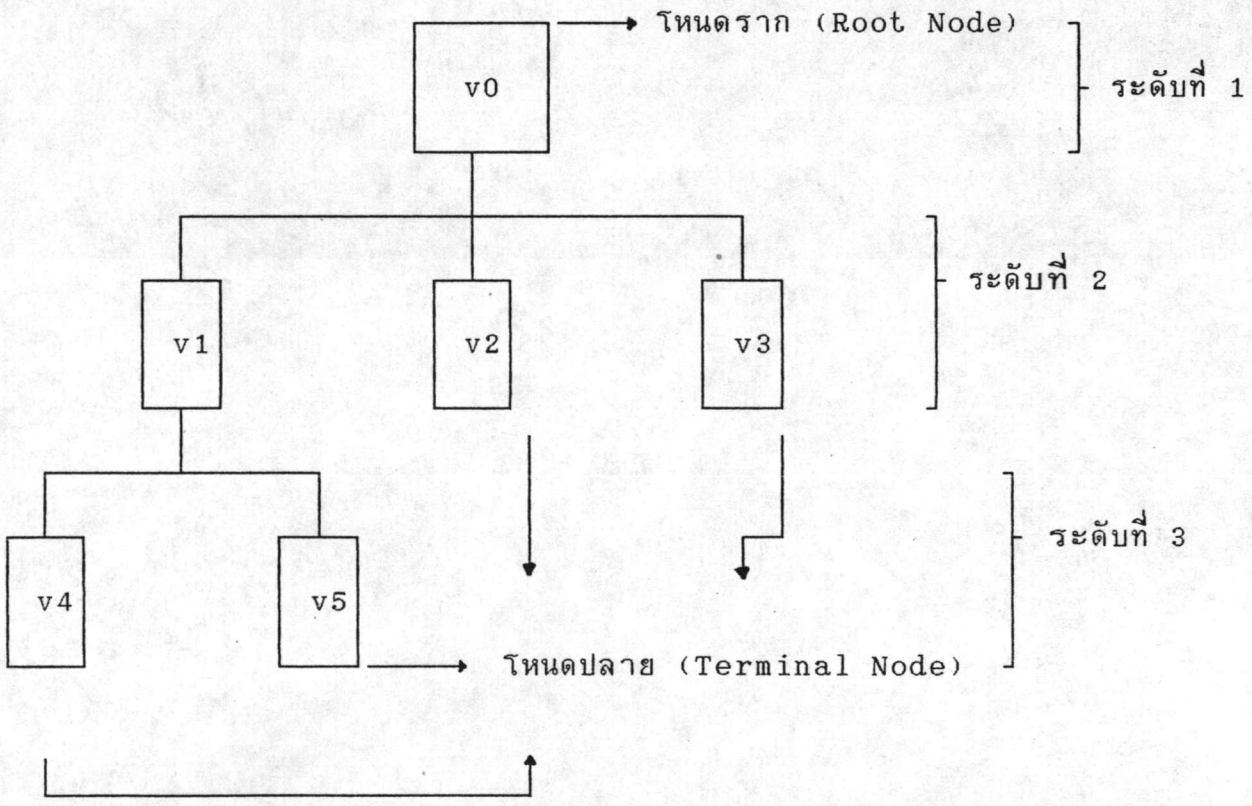
3.2 โครงสร้างข้อมูลแบบบีทรี

3.2.1 ลักษณะของโครงสร้างข้อมูลแบบต้นไม้

ลักษณะของโครงสร้างข้อมูลแบบต้นไม้ ประกอบไปด้วยเซตของโหนด (Node) ตั้งแต่ 1 โหนดขึ้นไป และจะต้องมีโหนดที่มีลักษณะพิเศษอยู่ 1 โหนด เรียกโหนดนี้เรียกว่า โหนดราก (Root Node) ซึ่งรูปที่ 3.6 จะแสดงลักษณะโครงสร้างต้นไม้ที่ประกอบไปด้วยโหนด 6 โหนด โดยจะมี v0 เป็น โหนดราก โหนดอื่น ๆ จะถูกจัดแบ่งเป็นเซลเล็ก ๆ แต่ละเซลจะเรียกว่า ต้นไม้ย่อย (Subtree) จากรูป 3.6 ต้นไม้ย่อยของ v0 คือ {v1,v4,v5},{v2},{v3} จำนวนของต้นไม้ย่อยของแต่ละโหนดจะเรียกว่าดีกรีของโหนดนั้น (Degree of Node) เช่น v0 มีต้นไม้ย่อย 3 ต้น ดังนั้น v0 จะมีดีกรีเท่ากับ 3 v1 มีต้นไม้ย่อย 2 ต้น มีดีกรีเท่ากับ 2 v2 มีต้นไม้ย่อย 0 ต้น มีดีกรีเท่ากับ 0 ซึ่งโหนดที่มีดีกรีเท่ากับ 0 นี้เรียกว่า โหนดปลายทาง (Terminal Node) ในรูปที่ 3.6 โหนดที่เป็นโหนดปลายทางคือ v2 v3 v4 v5

ระดับของโหนด (Level of Node) หมายถึงระยะที่ห่างจากรากของโหนดใด ๆ ตามแนวดิ่งโดยตัวอย่างโครงสร้างข้อมูลต้นไม้นี้จะมีระดับของโหนดดังที่ได้แสดงไว้ในรูปที่ 3.6

การเรียกชื่อโหนดจะยึดหลักของครอบครัว (Family) มาแสดงความสัมพันธ์ของโหนดต่าง ๆ โดยโหนดรากของแต่ละต้นไม้ย่อยจะเรียกว่าโหนดพ่อ (Parent) โหนดที่เกิดจากพ่อจะเรียก โหนดลูก (Child) ตัวอย่างเช่นจากรูปที่ 3.6 v1 เป็นโหนดพ่อซึ่งมีโหนดลูกคือ v4 v5 เป็นต้น



รูปที่ 3.6 แสดงส่วนประกอบของโครงสร้างข้อมูลแบบต้นไม้ (Tree Structure)

3.2.2 ลักษณะโครงสร้างข้อมูลบีทรี (B-Tree)

ลักษณะโครงสร้างข้อมูลบีทรีเป็นโครงสร้างข้อมูลต้นไม้ (Tree Structure) รูปแบบหนึ่ง โดยโครงสร้างข้อมูลบีทรีนี้จะมีลักษณะการสร้างโครงสร้างจากโหนดปลายและขยายขึ้นไปสู่โหนดด้านบน ซึ่งจะต่างจากโครงสร้างข้อมูลต้นไม้อื่นที่จะเริ่มสร้างโครงสร้างจากโหนดราก ส่วนประกอบภายในแต่ละโหนดของโครงสร้างแบบบีทรีได้แก่ชุดของคีย์ที่มีการเรียงลำดับตามค่าของข้อมูลและชุดของตัวชี้(Pointer) ที่ชี้ไปยังโหนดต่อไป ซึ่งในโครงสร้างข้อมูลบีทรีนี้จำนวนตัวชี้ในโหนดหนึ่ง ๆ มีจำนวนมากกว่าจำนวนคีย์อยู่ 1 เสมอ จำนวนตัวชี้สูงสุดที่สามารถเก็บได้ใน 1 โหนดเรียกว่าลำดับของบีทรี ตัวอย่างเช่น บีทรีลำดับ 8 หมายถึงโครงสร้างข้อมูลบีทรีที่มีแต่ละโหนดสามารถเก็บคีย์ได้มากที่สุด 7 คีย์ และเก็บตัวชี้ได้มากที่สุด 8 คีย์

ดังนั้นลักษณะสำคัญของโครงสร้างข้อมูลแบบบีทรี ที่มีลำดับ m จะสรุปได้ดังนี้

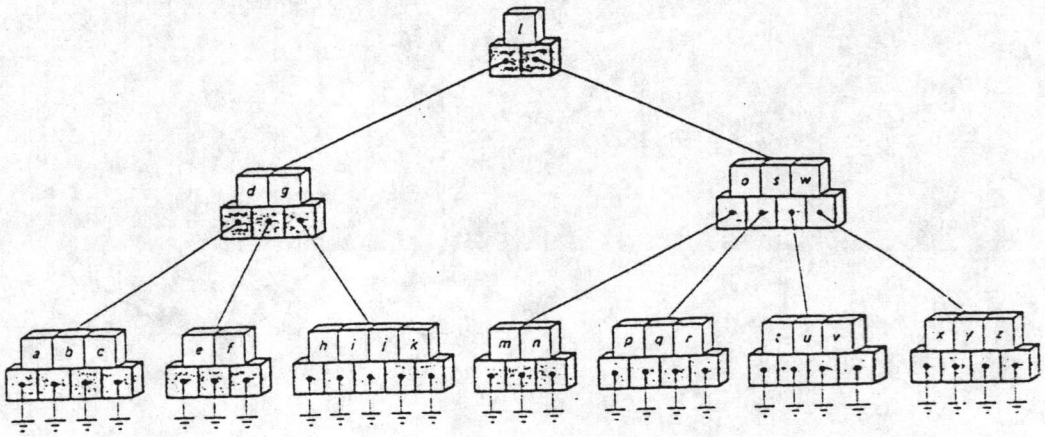
1) ภายในโครงสร้างบีทรี ทุกโหนด ยกเว้นโหนดราก มีสมาชิกได้มากที่สุดไม่เกิน m ตัว และ อย่างน้อยที่สุด m/2 ตัว

2) โหนดปลายทั้งหมดภายในบีทรี ต้องอยู่ในระดับเดียวกัน และไม่มีการชี้โยงไปยังโหนดใด ๆ อีก

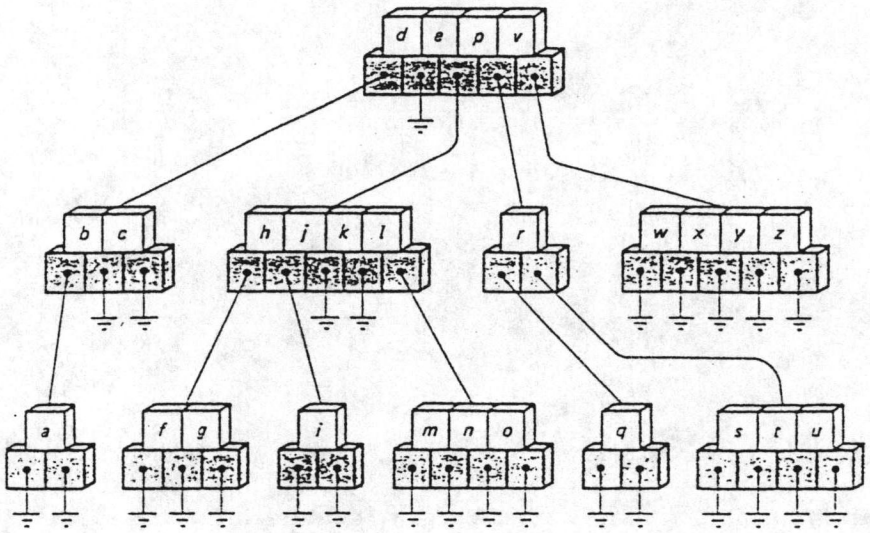
3) ในโหนดที่มีจำนวนคีย์เท่ากับ i คีย์ มีจำนวนตัวชี้ที่ชี้ไปยังโหนดที่อยู่ในระดับต่ำกว่าเท่ากับ i+1 ตัว

เนื่องจากโครงสร้างข้อมูลบีทรีนี้มีการจัดความสมดุลย์ให้กับโหนดปลายคือมีการจัดให้โหนดปลายอยู่ในระดับเดียวกัน การกระจายข้อมูลที่ถูกบันทึกภายในโครงสร้างจะมีการแผ่ออกไป ไม่มีการเอนเอียงน้ำหนักไปทางปลายข้างใดข้างหนึ่ง ทำให้การค้นหาข้อมูลในโครงสร้างทำได้เร็วมากขึ้น ซึ่งเป็นการเพิ่มประสิทธิภาพให้กับโครงสร้างข้อมูลบีทรี

โครงสร้างแบบต้นไม้ในรูป 3.7 จะแสดงโครงสร้างต้นไม้แบบบีทรีลำดับ 5 ซึ่งคีย์เป็นตัวอักษร 26 ตัว แต่โครงสร้างต้นไม้ในรูป 3.8 เป็นโครงสร้างต้นไม้ที่มีจำนวนโหนดลูกสูงสุด 5 โหนด แต่ไม่ใช่โครงสร้างต้นไม้แบบบีทรี เนื่องจากมีโหนดที่มีโหนดลูกว่างเปล่า และโหนดปลายไม่ได้อยู่ในระดับเดียวกัน



รูปที่ 3.7 แสดงโครงสร้างข้อมูลแบบบีทรีลำดับ 5



รูปที่ 3.8 แสดงโครงสร้างต้นไม้ที่ไม่ใช่โครงสร้างต้นไม้แบบบีทรี

3.3 โครงสร้างแฟ้มข้อมูลกราฟิกรูปแบบ PCX

โครงสร้างแฟ้มข้อมูลกราฟิกรูปแบบ PCX เป็นโครงสร้างแฟ้มข้อมูลกราฟิกที่พัฒนาโดยบริษัทแซดซอฟต์ (Zsoft) ซึ่งเป็นเจ้าของโปรแกรมวาดภาพพีซีเพ้นท์บรัช โครงสร้างแฟ้มข้อมูลกราฟิกแบบ PCX จะขึ้นอยู่กับฮาร์ดแวร์ที่ใช้ในการสร้างรูปภาพโดยจะแบ่งออก เป็น 3 ประเภทคือ

1. PCX สำหรับจอภาพสีเดียว (Monochrome PCX)
2. PCX สำหรับจอภาพ 16 สี (16-color PCX)
3. PCX สำหรับจอภาพ 256 สี (256-color PCX)

แฟ้มข้อมูล PCX ทุกประเภทมีส่วนหัวของแฟ้มข้อมูล (File Header) ที่ใช้เก็บข้อมูลต่าง ๆ ที่จำเป็นในการสร้างภาพ โดยแฟ้มส่วนหัวนี้มีความยาว 128 ไบต์ โครงสร้างแฟ้มข้อมูลกราฟิกแบบ PCX จะแสดงได้ดังรูปที่ 3.9

ไบต์ที่	รายละเอียด	หมายเหตุ และค่าที่เป็นไปได้
0	บอกว่าเป็น PCX หรือไม่	ค่า '0aH' = ไฟล์ PCX
1	บอกเวอร์ชันของ พีซี เพ้นท์บรัช ที่ใช้สร้างภาพ	0 = เวอร์ชัน 2.5 1 = เวอร์ชัน 2.8 3 = เวอร์ชัน 2.8 ที่ไม่มีรายละเอียดของสี(color pallete) 5 = เวอร์ชัน 3.0 ขึ้นไป
2	วิธีการเข้ารหัส	เป็น 1 เสมอ
3	จำนวนบิตต่อพิกเซล	จำนวนของบิตที่ใช้แสดง 1 พิกเซล จาก 1 ระนาบ 1 = EGA,VGA, or HERC 4 = CGA

ไบต์ที่	รายละเอียด	หมายเหตุ และค่าที่เป็นไปได้
4-11	โคออร์ดิเนต	ค่าจำนวนเต็ม 4 ค่า (ค่า 2 ไบต์) ให้ค่ามุมซ้ายและมุมล่างขวาของภาพ กำหนดให้รูปแบบ x1,y1,x2,y2
12-13	ความละเอียดของภาพตามแนวนอน	ความละเอียดของการแสดงภาพในแนวนอน 640=EGA,VGA 320=CGA 720=HERCULES
14-15	ความละเอียดของภาพตามแนวตั้ง	ความละเอียดของการแสดงภาพในแนวตั้ง 480=VGA 350=EGA 200=CGA 348=HERC
16-63	รายละเอียดของสี	สำหรับ 16 สี หรือน้อยกว่า
64	สำรอง (reserved)	มีค่าเป็น 0 เสมอ

ไบต์ที่	รายละเอียด	หมายเหตุ และค่าที่เป็นไปได้
65	จำนวนบิตของ ระนาบ	จำนวน ระนาบ ที่ใช้แสดงภาพ
66	จำนวนไบต์ต่อบรรทัด	แสดงจำนวนไบต์ที่ใช้เก็บข้อมูล ในแต่ละบรรทัด
68	ข้อมูลเกี่ยวกับสี (Pallete Information)	1 = สี หรือขาว-ดำ 2 =เกรย์สเกล (Gray Scale)
70-127	-	สำหรับ Fractal Programming

รูปที่ 3.9 แสดงรายละเอียดของแฟ้มข้อมูลส่วนหัวของการจัดเก็บแบบ PCX

การอ่านข้อมูลเก็บแฟ้มข้อมูลบิตแมพ จะทำได้โดยการอ่านภาพตามแนวของจอภาพคือต้องอ่านในแนวนอนจากซ้ายไปขวา เริ่มที่พิกเซลที่ตำแหน่งบนซ้ายไปทางขวาจนสุดภาพแล้วจึงอ่านในแถวถัดไป โดยในจอภาพชนิด EGA และ VGA ซึ่งมีหน่วยความจำหลาย ๆ ระนาบ ดังนั้นจะต้องอ่านทุก ๆ ระนาบใน 1 แถว (scan line) เริ่มที่ ระนาบ 0,1,2 และ 3 ตามลำดับจนครบ แล้วจึงอ่านแถวถัดไปจนหมดภาพที่ต้องการเก็บ การบันทึกข้อมูลในแฟ้มข้อมูล PCX ใช้วิธีการเข้ารหัสที่เรียกว่า Run Length Encoding (RLE) ซึ่งเป็นวิธีการที่ใช้ในการอัดข้อมูลรูปภาพ เพื่อลดขนาดของแฟ้มข้อมูลการเข้ารหัสแบบนี้มีรายละเอียดดังนี้ คือ ถ้าไบต์ใด ๆ มีค่าไม่เหมือนกับไบต์อื่น ๆ และถ้า 2 บิตบนไม่เท่ากับ '11' ไบต์นั้นจะถูกเก็บลงไฟล์ นอกนั้นจะมีตัวนับ (Counter) นับจำนวนไบต์ที่เก็บลง

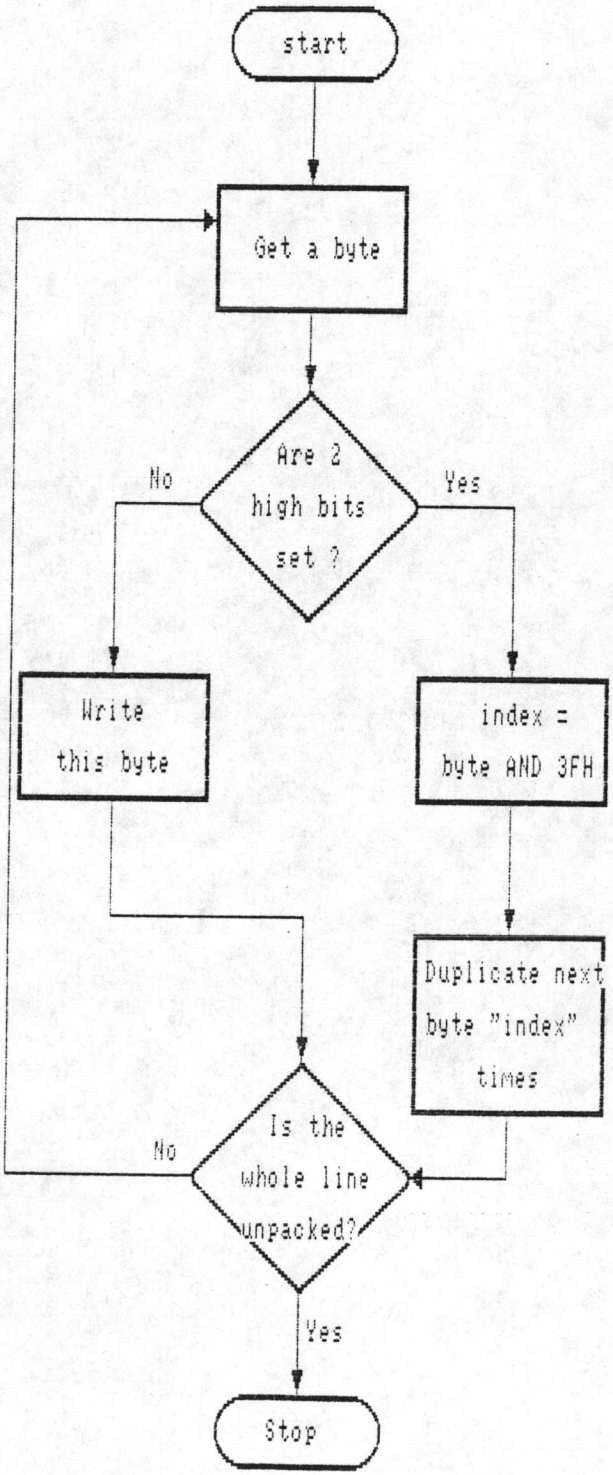
แต่ต้องไม่เกิน 63 ถ้าเกินให้นำค่าตัวนับที่ได้ OR กับ 'C0h' แล้วเก็บค่าตัวนับนั้นลงแฟ้มแล้วตามด้วยค่าของไบต์นั้น ๆ จากนั้นจึงเริ่มนับใหม่เป็น 1 ถ้ากรณีที่ไบต์เดียวมีค่าบิตบนเป็น '11' ให้เขียนตัวนับเท่ากับ 1 (OR กับ C0h = C1h) ลงไปในแฟ้มข้อมูลและตามด้วยค่าไบต์นั้น สรุปการเข้ารหัสแบบ RLE แสดงในรูปที่ 3.10

เงื่อนไข	การกระทำ
ค่าที่เหมือนไบต์อื่น ค่า < 'C0h' ไม่เหมือนไบต์อื่น ค่า > 'C0h' ไม่เหมือนไบต์อื่น Counter > 63	นับ Counter + 1 เก็บลงไฟล์เลย เก็บ 'C1h' ลงไฟล์ตามด้วยค่าไบต์นั้น Counter=1 เก็บค่า 'FFh'และค่าไบต์ลงไฟล์

รูปที่ 3.10 แสดงขั้นตอนการเข้ารหัส PCX

การถอดรหัสจะใช้วิธีการตรงข้ามกับการเข้ารหัส คือ อ่านข้อมูลมา 1 ไบต์แล้วตรวจสอบดูว่าค่า 2 บิตบนเป็น '11' หรือไม่ (หรือมากกว่า C0h) ถ้าใช่จะเป็นค่าตัวนับทันที (ซึ่งไบต์ต่อไปก็จะเป็นค่าของไบต์) แล้วทำการขยายข้อมูลออกมาตามจำนวนตัวนับนั้น ๆ แล้วนำมาเก็บไว้ในหน่วยความจำ ถ้าค่าไบต์ที่อ่าน 2 บิตบนไม่ใช่ '11' จะทำการเก็บข้อมูลไบต์นั้นตัวเดียวลงหน่วยความจำ และจะทำเช่นนี้ไปจนจบแฟ้มข้อมูล

สำหรับผังงาน (Flow Chart) แสดงการอ่านแฟ้มข้อมูลภาพ PCX 1 บรรทัดสามารถแสดงได้ดังรูปที่ 3.11



รูปที่ 3.11 แสดงการอ่านแฟ้มข้อมูลภาพ PCX

3.4 การเลือกใช้โปรแกรมเพื่อการพัฒนาระบบการจัดการด้านการเรียนการสอน

สำหรับงานวิจัยนี้ ผู้วิจัยได้เลือกภาษา ซี ช่วยในการพัฒนาระบบ เนื่องจากภาษา ซี เป็นภาษาที่สามารถทำงานบนเครื่องคอมพิวเตอร์หลาย ๆ ประเภท ทำให้ระบบที่จะพัฒนาขึ้นนี้สามารถใช้ได้กับเครื่องคอมพิวเตอร์หลาย ประเภท นอกจากนี้ภาษา ซี ยังเป็นภาษาที่มีความสามารถจัดการเกี่ยวกับหน่วย ความจำ (Memory Allocation) ซึ่งเรื่องนี้เป็นเรื่องสำคัญมากในการพัฒนา ระบบงานใหญ่ ๆ และในงานวิจัยนี้จะต้องพัฒนาระบบในโหมดกราฟิก เนื่องจาก จะต้องมีการสร้างและแสดงภาพต่าง ๆ ซึ่งในภาษาซีจะมีฟังก์ชันทางด้านกราฟิก เป็นจำนวนมากให้เรียกใช้ ทำให้เกิดความสะดวกในการพัฒนาระบบมากขึ้น แต่ สำหรับภาษา ซี นั้นจะไม่มีคำสั่งที่ใช้ช่วยในการจัดการแฟ้มข้อมูล เช่น การค้นหา ระเบียนข้อมูล การเพิ่มระเบียนข้อมูล การลบระเบียนข้อมูล ถ้าจะให้ระบบ สามารถทำงานเหล่านี้ได้ ต้องเขียนโปรแกรมเป็นฟังก์ชันที่ทำหน้าที่จัดการแฟ้มขึ้นมา เองซึ่งจะต้องใช้เวลามาก ดังนั้นเพื่อเป็นการประหยัดเวลาในการพัฒนาระบบผู้วิจัย จึงเลือกใช้โปรแกรมบีทรีฟมาช่วยในการจัดการแฟ้มข้อมูลของระบบ เนื่องจาก โปรแกรมบีทรีฟ เป็นโปรแกรมจัดการแฟ้มข้อมูลที่สามารถใช้พัฒนางานร่วมกับภาษาซี ได้และเป็นโปรแกรมที่สามารถจัดการแฟ้มข้อมูลได้อย่างมีประสิทธิภาพ นอกจากนี้ยัง เป็นโปรแกรมที่ได้ความนิยมใช้กันแพร่หลาย โดยรายละเอียดของโปรแกรมบีทรีฟ มีดังนี้

3.4.1 การใช้โปรแกรมบีทรีฟกับภาษา ซี

เมื่อต้องการใช้โปรแกรมบีทรีฟ จัดการแฟ้มข้อมูลในโปรแกรม ประยุกต์ที่เขียนขึ้นด้วยภาษาซีนั้น ในโปรแกรมประยุกต์จะต้องเรียกใช้ฟังก์ชันจำนวน เต็ม BTRV พร้อมกับส่งรหัสการทำงานและพารามิเตอร์ที่จำเป็นให้ ซึ่งรูปแบบการ เรียกใช้จะแสดงในรูปที่ 3.12 และเมื่อให้โปรแกรมประยุกต์ทำงานก็จะต้องทำการ เรียกโปรแกรม BTRIEVE.EXE ก่อน ซึ่งโปรแกรมนี้เป็นโปรแกรมเฉพาะงาน

ที่อยู่ประจำในหน่วยความจำ (Resident Program) โดยใช้เนื้อที่ในหน่วยความจำประมาณ 76.6 กิโลไบต์ เป็นโปรแกรมกระทำการหรือจัดการฟังก์ชันของโปรแกรมบีทรีฟ (Btrieve Function Executor)

```
int BTRV(OP,POS_BLK,DATA_BUF,BUF_LEN,KEY_BUF,KEY_NUM)
    int  OP                 /* operation code
    int  POS_BLK[]          /* position block
    int  DATA_BUF[]         /* data buffer
    int  *BUF_LEN           /* length of data buffer
    char  KEY_BUF[]         /* key buffer
    int   KEY_NUM           /* key number
```

รูปที่ 3.12 แสดงรูปแบบฟังก์ชัน BTRV ของโปรแกรม บีทรีฟ

รูปแบบของฟังก์ชัน BTRV มีรายละเอียดดังนี้

- รหัสกระทำการ (Operation Code) เป็นรหัสของงานที่ต้องการให้โปรแกรมทำงานเช่น 14 เป็นรหัส CREATE คือ สร้างแฟ้มข้อมูล 4 เป็นรหัส DELETE คือ ลบระเบียนในแฟ้มข้อมูล

- ตำแหน่งของบล็อก (Position Block) เป็นตัวแปรแถวลำดับ (Array) ที่ใช้เก็บข้อมูลต่าง ๆ ของแฟ้มข้อมูล เช่น เส้นทางที่ใช้เข้าถึงระเบียนหรือหมายเลขคีย์ เส้นทางที่ใช้เข้าถึงตัวชี้ดรรชนี (Index Pointer) ตัวชี้ระเบียนที่ชี้ระเบียนปัจจุบัน ตัวชี้ที่ชี้ระเบียนก่อนหน้าระเบียนปัจจุบัน ตัวชี้ที่ชี้ระเบียนถัดจากระเบียนปัจจุบัน ในภาษาซีให้ขนาดของตัวแปรนี้เท่ากับ 128 ไบต์

- บัฟเฟอร์ของข้อมูล (Data Buffer) เป็นตัวแปรที่เก็บที่อยู่ (address) ของบัฟเฟอร์

- ความยาวของบัฟเฟอร์ของข้อมูล (Data Buffer Length) จะต้องกำหนดความยาวของบัฟเฟอร์ของข้อมูลให้กับโปรแกรมบีทรีฟเพื่อใช้ในขั้นตอนการถ่ายข้อมูล ซึ่งความยาวที่กำหนดให้ควรจะเท่ากับความยาวของระเบียนที่กำหนดไว้ในขั้นตอนการสร้างโครงสร้างแฟ้มข้อมูล

- คีย์บัฟเฟอร์ (Key Buffer) เป็นตัวแปรที่เก็บที่อยู่ของบัฟเฟอร์ของคีย์

- หมายเลขคีย์ (Key Number) เนื่องจาก บีทรีฟ สามารถมีคีย์ได้ถึง 24 คีย์ ดังนั้นก่อนที่จะสั่งให้โปรแกรมบีทรีฟเข้าถึงระเบียนในแฟ้มข้อมูลต้องบอกหมายเลขของคีย์ที่จะใช้เข้าถึงข้อมูลด้วย

ฟังก์ชันของโปรแกรมบีทรีฟที่ได้ใช้ในงานวิจัยนี้ ได้แก่

- ฟังก์ชันการสร้างโครงสร้างแฟ้มข้อมูล รหัสในการสร้างโครงสร้างแฟ้มข้อมูล คือ 14

- ฟังก์ชันการเปิดแฟ้มข้อมูล รหัสในการเปิดแฟ้มข้อมูล คือ 0

- ฟังก์ชันการปิดแฟ้มข้อมูล รหัสในการปิดแฟ้มข้อมูล คือ 1

- ฟังก์ชันการเพิ่มระเบียน รหัสในการเพิ่มระเบียน คือ 2

- ฟังก์ชันการปรับปรุงข้อมูลในระเบียน รหัสในการปรับปรุงข้อมูลในระเบียน คือ 3

- ฟังก์ชันการลบระเบียน รหัสในการลบระเบียนออกจากแฟ้มข้อมูล คือ 4

- ฟังก์ชันการดึงระเบียนตามต้องการ รหัสในการดึงระเบียนตามต้องการในแฟ้มข้อมูล คือ 5

- ฟังก์ชันการดึงระเบียนถัดจากระเบียนปัจจุบัน รหัสในการดึงระเบียนถัดจากระเบียนปัจจุบัน คือ 6

- ฟังก์ชันการดึงระเบียนแรกของแฟ้มข้อมูล รหัสในการดึงระเบียนแรกของแฟ้มข้อมูล คือ 12

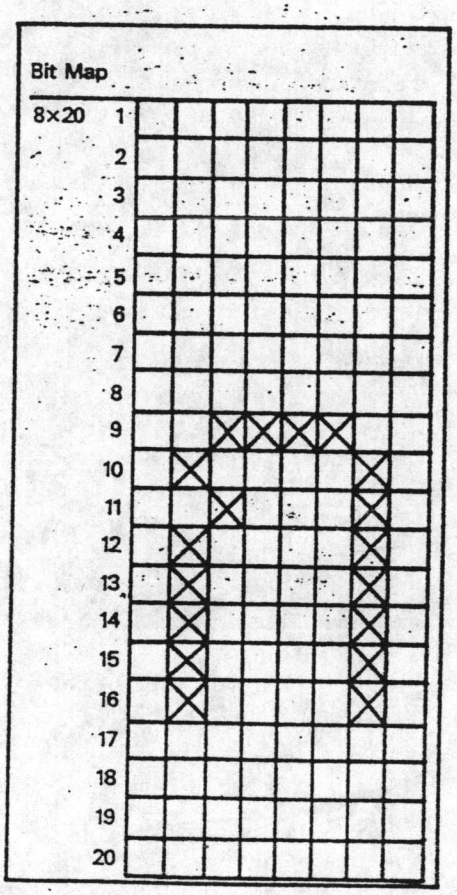
- ฟังก์ชันการดึงระเบียนสุดท้าย รหัสในการดึงข้อมูลระเบียนสุดท้ายของแฟ้มข้อมูล คือ 13

3.5 การแสดงและรับข้อมูลภาษาไทยในโหมดกราฟิก

3.5.1 การแสดงข้อความตัวอักษรในโหมดกราฟิก

การเขียนโปรแกรมให้แสดงอักษรไทยออกจอภาพในโหมดกราฟิก มีขั้นตอนดังนี้

3.5.1.1 ทำการสร้างฟอนต์ของตัวอักษรแล้วเก็บลงแฟ้มข้อมูลไว้ในการสร้างฟอนต์ก็ต้องสร้างฟอนต์เอดิเตอร์ขึ้นมาสำหรับเป็นเครื่องมือที่ใช้ออกแบบตัวอักษรแต่ละตัว สำหรับฟอนต์ที่ใช้ในงานวิจัยนี้จะเป็นฟอนต์ขนาด 20x8 และ 14x8 นั่นคือเป็นฟอนต์ที่มีจำนวนจุดตามแนวนอน 20 จุด จำนวนจุดตามแนวคอลัมน์ 8 จุดดังที่แสดงในรูปที่ 3.13 และฟอนต์ที่มีจำนวนจุดตามแนวนอน 14 จุด จำนวนจุดตามแนวคอลัมน์ 8 จุด ซึ่งในแต่ละแถวที่มี 8 จุดจะให้แทนข้อมูลด้วยเลขฐานสอง เช่น ให้จุดของกราฟิก (pixel) ที่เป็นจุดสว่างแทนด้วยเลข 1 จุดมืดแทนด้วยเลข 0 ดังนั้นถ้าบนแถวแรกมีจุดมืดอยู่ทางซ้าย 4 จุด และถัดมาเป็นจุดสว่างอีก 4 จุด จะเขียนเป็นเลขฐาน 2 จำนวน 8 บิต ได้เท่ากับ 00001111 ถ้าเป็นเลขฐาน 10 จะได้เท่ากับ 15 นั่นคือฟอนต์ตัวอักษร 1 ตัวใช้ 8 บิต จำนวน 20 แถว คือใช้เนื้อที่ 20 ไบต์ และถ้าเก็บตัวอักษร 255 ตัว ต้องใช้เนื้อที่ทั้งหมด 255x20 = 5120 ไบต์ และในกรณีเดียวกันถ้าเป็นฟอนต์ที่มีขนาด 14x8 จะต้องใช้เนื้อที่ 255x14 = 3570 ไบต์ สำหรับในงานวิจัยนี้จะมีฟอนต์แบบต่าง ๆ ทั้งหมด 6 ฟอนต์ ซึ่งเก็บอยู่ในแฟ้มข้อมูลดังนี้ NORMAL.X20 NORMAL.X14 ITALIC.X20 OFFSET.X20 COMPUTE.X20 LIGHT.X20



รูปที่ 3.13 แสดงตัวอย่างฟอนต์ที่มีขนาด 20x8

3.5.1.2 เมื่อสร้างฟอนต์ได้แล้วต่อไปเป็นขั้นตอนการเขียนฟังก์ชันจัดการอ่านฟอนต์ที่เก็บอยู่ในแฟ้มข้อมูลขึ้นมาเก็บไว้บนหน่วยความจำเพื่อเตรียมเอาไว้ให้โมดูลที่ต้องการใช้ฟอนต์เรียกใช้ ฟังก์ชันที่อ่านฟอนต์ขึ้นมาเก็บในหน่วยความจำแสดงในรูปที่ 3.14

```
int readfont() {
    for ( i=0;i<NUMFONT;i++ )  {
        fp = NULL;
        handle = 0;
        size   = 0;
        if ((fp = fopen(tempfile,"rb")) == NULL) {
            box_error("ไม่สามารถเปิดแฟ้มข้อมูลได้","");
            return(NO);
          }
        handle = fileno(fp);
        if ((handle = fileno(fp)) == 0)
            return(NO);
         FONT[i].buffont = temp = (void far *)
             farmalloc(size = filelength(handle));
        if (temp == NULL) {
            box_error("เนื้อที่หน่วยความจำไม่พอ");
            return(NO);
        }
        fread(FONT[i].buffont,1,size,fp);
        fclose(fp);
    }
    return(YES);
}
```

รูปที่ 3.14 แสดงฟังก์ชันอ่านแฟ้มข้อมูลฟอนต์

3.5.1.3 ขั้นตอนการแสดงตัวอักษรออกทางจอภาพ ทำได้โดยการนำเอารหัสตัวอักษรภาษาไทยที่ต้องการแสดงมาคำนวณหาตำแหน่งที่อยู่ของฟอนต์ในหน่วยความจำ เมื่อได้ตำแหน่งของฟอนต์แล้วก็จะทำการอ่านรูปแบบฟอนต์แล้วใช้คำสั่ง putpixel ของภาษา ซี ในการแสดงจุดสว่างบนจอภาพเมื่อรูปแบบฟอนต์ที่อ่านได้เป็นเลข 1 สำหรับฟังก์ชันแสดงตัวอักษรภาษาไทยบนจอภาพได้แสดงไว้ในรูปที่ 3.15

```
void put_str() {
     /*  คำนวณหาตำแหน่งของฟอนต์ในหน่วยความจำ */
     ptr = &FONT[font].buffont[*st * FONTHIGH];
     /* แสดงตัวอักษรตามรูปแบบฟอนต์ */
     for(y=0 ; y<FONTHIGH ; y++)
         for(x=0 ; x<FONTWIDTH ; x++)
             if((*ptr << x) & 0x80)
                putpixel(Loc_x+x,Loc_y+y,color);
       ptr++;
}
```

รูปที่ 3.15 ฟังก์ชันแสดงตัวอักษรภาษาไทย

3.5.1.4 การจัดเรียงตัวอักษรภาษาไทยเป็น 3 ระดับ

การแสดงข้อความภาษาไทยจะต้องทำการจัดตัวอักษรภาษาไทยให้แสดงออกมาเป็น 3 ระดับ เช่นคำว่า กุ้ง หรือ กั้น นั่นคือต้องแยกตัวอักษรภาษาไทยออกเป็น 3 ประเภท คือ อักษรระดับบน อักษรระดับกลาง อักษรระดับล่างและเมื่อจะแสดงข้อความภาษาไทยก็จะต้องนำรหัสภาษาไทย

นั้นไปเทียบกับค่าในตารางแสดงระดับตัวอักษร (Code_level Table) ซึ่งแสดงไว้ในรูปที่ 3.16 จะทำให้ทราบว่ารหัสตัวอักษรภาษาไทยนั้นเป็นอักษรระดับใด เพื่อจะได้แสดงอักษรนั้นได้ถูกตำแหน่ง

```
/*  1 = ระดับกลาง  2 = ระดับบน   3 = ระดับล่าง  */
/*  0 = รหัสที่ไม่ใช่ตัวอักษร   -1 = รหัส attribute */
signed char code_level[255] = {
 0, 0,-1, 0, 0,-1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
 0, 0,-1,-1,-1, 0,-1,-1, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 2, 1, 1, 2, 2, 2, 2, 3, 3, 1, 0, 0, 0, 0, 1,
 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
   };
```

รูปที่ 3.16 แสดงตารางระดับตัวอักษร (Code_Level Table)

3.5.2 การรับข้อมูลภาษาไทยในโหมดกราฟิก

เมื่อมีการรับข้อมูลจากแป้นพิมพ์ จะต้องนำรหัสที่ได้จากแป้นพิมพ์ไปเทียบกับค่าในตาราง ซึ่งตารางนี้เก็บรหัสภาษาไทยไว้ตามตำแหน่งรหัสของแป้นพิมพ์ ตัวอย่างตารางแสดงไว้ในรูปที่ 3.17 สำหรับรหัสภาษาไทยที่กล่าวถึงหมายถึงรหัสภาษาไทยที่มีใช้ในปัจจุบันซึ่งมีมากมาย เช่น รหัสสมอ. รหัสเกษตร แต่ในงานวิจัยนี้จะเลือกใช้เฉพาะรหัสของสมอ. เนื่องจากเป็นรหัสมาตรฐานของสมาคมมาตราฐานอุตสาหกรรม

```
unsigned char key_smo[127] = {
 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,160,
44, 46, 50, 51, 52, 38,167, 54, 55, 53,  57,
193,162,227,189,168,197, 47, 45,192,182,216,
214,164,181,171,199,178,170,204, 63, 49,196,
46,169,175,174,226,172,231,179,235,201,200,
63,236,207,173, 48,177,166,184,234,206, 34,
41,237, 40,186, 96,197,217, 56, 37,191,212,225,
161,211,180,224,233,195,232,210,202,183,215,185,
194,230,190,203,208,213,205,228,187,209,188,176,
126, 44, 45
};
```

รูปที่ 3.17 แสดงตารางรหัสแป้นพิมพ์