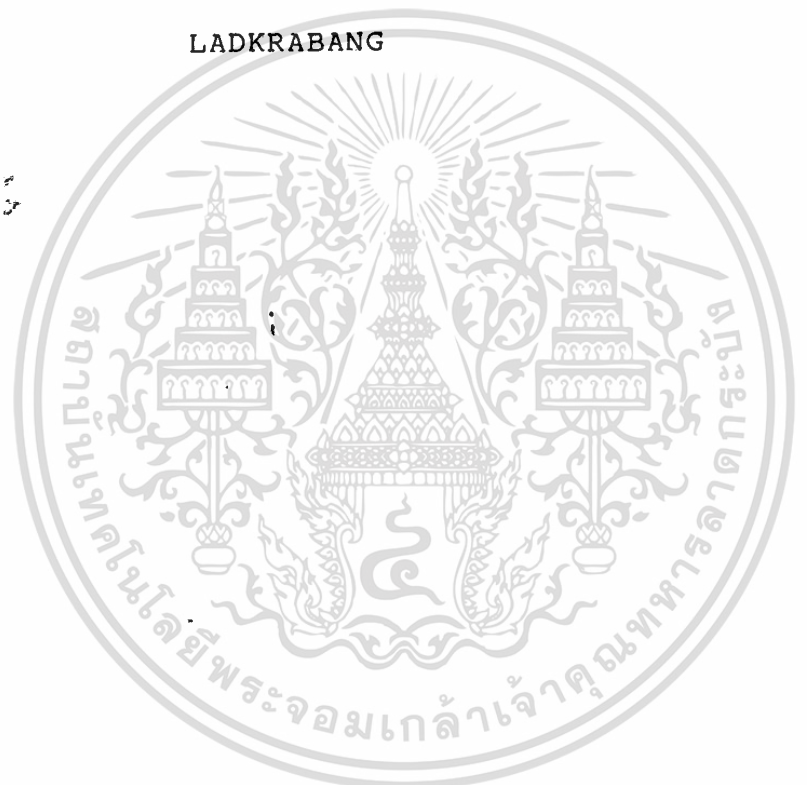
FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY

LADKRABANG



A thesis submitted to the Examining Commitee of the

Faculty of Engineering

for the partial fulfillment of the requirement for the

Degree of Bachelor of Engineering

การใช้ dBASE III+ ในการจัดเก็บข้อมูลวัสดุก่อสร้างและแสดงผล

นายศุภกรณ์ ธนฉายสวัสดิ์

อ.บวรวรรณ สาริกภูติ อาจารย์ที่ปรึกษา

2533

บทคัดย่อ

วัสดุก่อสร้างในสต๊อค จะมีการเข้าและออก เป็นประจำทุกวัน ตลอดระยะเวลาการก่อสร้าง ดังนั้น มันจึงเป็นข้อมูลที่เหมาะสม ในการให้เป็นข้อมูลที่จะถูกจัดเก็บด้วยโปรแกรมสำเร็จรูป ดีเบสทรีพลัส (dBASE III +) แต่โปรแกรมสำเร็จรูปดีเบสทรีพลัสนั้น ยังมีประสิทธิภาพ การแสดงผลในรูปแบบที่ต้องการด้อยอยู่ โครงงานนี้จึงนำเสนอ การใช้คำสั่งของโปรแกรมสำเร็จรูปดีเบสทรีพลัส มาแสดงข้อมูลวัสดก่อสร้าง ในลักษณะคล้ายกับ ที่จัดเก็บมันบนหน้ากระดาษของสมุดบัญชี

นอกจากนี้แล้ว ยังได้นำเสนอมัน ด้วยกราฟแบบต่าง ๆ ตามข้อมูลที่ผู้ใช้กำหนด โดยนำเอาโปรแกรมสำเร็จรูปเทอร์โบปาสคาล (Turbo Pascal) มาอ่านข้อมูลในไฟล์ข้อมูลดีเบส เพื่อนำไปพล็อตกราฟตามลำดับ

อีกทั้งเพื่อให้ใช้งานได้จริง จึงได้นำเอาแอสเซมเบลร์ของบริษัทไมโครซอฟต์ ชื่อ MASM มาแสดงคำอธิบายการทำงานของฟังก์ชั่นคีย์ ที่กำหนดไว้ในโปรแกรมของโครงงานนี้ด้วย ทำให้ช่วยเพิ่มความสะดวกต่อผู้ใช้งาน

ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้ dBASE III + ในการจัดเก็บข้อมูลวัสดุก่อสร้างและแสดงผล

ผู้จัดทำ

นายศุภกรณ์ ธนฉายสวัสดิ์ 301279

อ. บรรวรรณ สาริกภูติ อาจารย์ที่ปรึกษา

สารบัญ

บทที่ 1
บทนำ

ในรายงานฉบับนี้ จะนำเอาความรู้ทั้งหมด ที่ได้จากการทำโครงงานมากล่าวถึง ตั้งแต่ตัวทฤษฎีการใช้งานดีเบสทรีพลัส คำสั่งและรายละเอียดการทำงานของคำสั่งที่จะนำมาใช้ในโครงงาน จากนั้นจะอธิบายการประยุกต์ใช้งานจริง ตามโปรแกรมภาคดีเบทั้งหมด รวมถึงแสดงโครงสร้างของโปรแกรม เพื่อให้ง่ายต่อการเข้าใจด้วย

ภาคปาสดาล สำหรับการอ่านข้อมูลจากไฟล์ดีเบสมาแสดงผลนั้น จะกล่าวถึง โครงสร้างการจัดเก็บข้อมูลในไฟล์ดีเบส เพื่อโปรแกรมแยกแยะข้อมูลในส่วนที่ผู้ใช้ต้องการมาแสดงผล อีกทั้งยังได้แสดงวิธีการจัดเอาเทอร์ไบปาสดาลมาเขียนกราฟแบบต่าง ๆ

สุดท้ายก่อนถึงภาคผนวก จะแสดงวิธีการโปรแกรมภาษาแอสเซมบลี้ เพื่อเรียกใช้ในดีเบส เป็นโปรแกรมช่วยเหลือ (help) สำหรับอธิบายการทำงานของฟังก์ชั่นคีย์ในการทำงานดีเบส โดยจะกล่าวถึงอย่างละเอียดตั้งแต่ การส่งพารามิเตอร์จากดีเบส การเรียกใช้รูทีนของรวมไบออส และคอส มาทำงานตามต้องการ

ส่วนในภาคผนวกนั้น ภาคผนวก ก. จะแสดงโปรแกรมทั้งหมดของโครงงาน และภาคผนวก ข. จะแสดงไฟลร์ชาร์ทของโปรแกรมบางโปรแกรมที่สำคัญ ๆ กับผลการทำงานของโปรแกรมด้วย

บทที่ 2

ทฤษฎีการใช้ดีเบสทรีพลัส

ซอฟท์แวร์จัดการฐานข้อมูล

บนเครื่องไมโครคอมพิวเตอร์นั้น ซอฟท์แวร์ที่ได้รับการพัฒนาขึ้นเพื่อการจัดการฐานข้อมูล มีอยู่หลายโปรแกรมด้วยกัน แต่ที่นิยมกันมากในบ้านเรา ดูเหมือนว่าจะเป็น พิเอฟเอส (PFS) ดาต้าสตาร์ (Data Star) และกลุ่มซอฟท์แวร์ในตระกูลดีเบส อันได้แก่ ดีเบสทู ดีเบสทรีและดีเบสทรี-พลัส แต่ซอฟท์แวร์ตัวที่นิยมที่สุด และมีใช้ตั้งแต่เครื่อง 8 บิท, 16 บิท จนถึง 32 บิท เห็นจะได้แก่ ซอฟท์แวร์ตระกูลดีเบสนั่นเอง

ดังที่ได้กล่าวมาแล้วว่า หัวใจสำคัญของการจัดระบบข้อมูลคือการจัดเก็บข้อมูลได้ง่ายและสะดวกต่อการเรียกใช้ข้อมูบที่ได้จัดเก็บนั้นซอฟท์-แวร์แต่ละโปรแกรมที่ได้พัฒนาขึ้น ต่างก็พยายามมุ่งเน้นที่จุดนี้เป็นสำคัญ อย่างไรก็ตาม แนวความคิดที่ว่าจะพัฒนาซอฟท์แวร์ก็แตกต่างกันไปตามแต่ผู้พัฒนา ตัวอย่างเช่น PFS เป็นซอฟท์แวร์ที่มุ่งเน้นที่พัฒนาขึ้นเพื่อมุ่งเน้นให้ผู้ใช้ได้ใช้งานเครื่องคอมพิวเตอร์แทนการจัดการเอกสารด้วยมือ โดยผู้ใช้ไม่จำ-เป็นต้องมีความรู้เรื่องการฐานข้อมูลมากนัก การออกแบบโครงสร้างทำได้ง่ายและมีรูปแบบใกล้เคียงกับการบันทึกข้อมูลในแบบฟอร์มที่ใช้กัน แต่ PFS ก็มีจุดอ่อนตรงที่เป็นซอฟท์แวร์ซึ่งไม่เอื้ออำนวยให้ผู้ใช้เขียนโปรแกรม ทำให้การเรียกใช้ และประมวลผลข้อมูลมีข้อจำกัดมาก

แต่ซอฟท์แวร์ดีเบส มีความสามารถสูงกว่า โดยเฉพาะอย่างยิ่งการเรียกใช้ข้อมูล การประมวลผลข้อมูลในไฟล์ นอกจากนี้แล้วดีเบสทรีพลัสยังเอื้ออำนวยให้ผู้ใช้เขียนโปรแกรมด้วยชุดคำสั่งของดีเบส เพื่อให้เครื่องคอมพิวเตอร์ประมวลผลตามโปรแกรมนั้นอีกทีนึง และตัวโปรแกรมของดีเบส-ทรีพลัส ก็เป็นการโปรแกรมแบบโครงสร้าง ทำให้การเขียนโปรแกรมและการตรวจสอบทำได้ง่าย ดังนั้นจึงไม่น่าสงสัยว่าทำไมซอฟท์แวร์ดีเบสจึงได้รับความนิยมสูง

วิวัฒนาการของซอฟท์แวร์ในตระกูลดีเบส

ผู้พัฒนาซอฟท์แวร์ในตระกูลดีเบสขึ้นมาคือ เวนย์ แรทลิฟ (Wayne Rathlieve) วิศวกรของโครงการอวกาศนาซ่า ซึ่งเขาได้เริ่มพัฒนาดีเบสทรีพลัสมาตั้งแต่ปี พ.ศ. 2520 และจัดออกมาจำหน่ายครั้งแรกในปี พ.ศ. 2522 ภายใต้ชื่อซอฟท์แวร์ว่าวัลแคน (Valcan) แต่ไม่ได้รับความสำเร็จในยอดขายเลย จนกระทั่งเวนย์ แวทลิฟได้พบกับจอร์จ เกตแห่งบริษัทแอสตัน เทต ซึ่งเป็นบริษัทประเภทซอฟท์แวร์เฮาส์และเป็นตัวแทนจำหน่ายซอฟท์แวร์ จอร์จ เทต ได้ตกลงทำสัญญาเป็นตัวแรกแทนจำหน่าย ซอฟท์แวร์ของเวนย์ แรทลิฟ และเปลี่ยนชื่อซอฟท์แวร์นั้นเสียใหม่เป็น ดีเบสทู (dBASE II)

ตัวดีเบสทูเอง ได้เริ่มถูกวางจำหน่ายครั้งแรกในตอนต้นปี พ.ศ. 2524 และประสบความสำเร็จอย่างงดงาม จนกลายเป็นซอฟแวร์ที่มียอดขายสูงสุด ในบรรดาซอฟแวร์จัดการฐานข้อมูล นับตั้งแต่นั้นมาดีเบสทูก็ได้พัฒนาต่อเนื่องกันมาตามลำดับ จนกลายเป็นดีเบสทรีสำหรับเครื่อง 16 บิท และเมื่อต้นปี พ.ศ. 2529 ก็ได้พัฒนาเป็นดีเบสทรีพลัส (dBASE III+) ซึ่งมีความสามารถและความเร็วสูงมาก

โครงสร้างของดีเบสทรีพลัส

ดีเบสทรีพลัส เป็นซอฟแวร์ที่จัดเก็บข้อมูลในลักษณะเป็นตาราง คล้ายคลึงกับการจัดเก็บข้อมูลในแฟ้มเอกสาร เราเรียกลักษณะของระบบการจัดเก็บข้อมูลแบบนี้ว่า แบบรีเลชั่นนัลดาต้าเบส (Relational Database) ซึ่งเป็นการจัดเก็บข้อมูลที่ง่ายต่อการเรียกใช้และการทำความเข้าใจของผู้ใช้อีกด้วย

ลักษณะของตารางข้อมูลจะแบ่งเป็นคอลัมภ์ (Column) และแถว (Row) โดยแต่ละคอลัมภ์จะแทนดาต้าฟิลด์แต่ละฟิลด์ส่วนแต่ละแถวจะแทนข้อมูลแต่ละเรคคอร์ด

เริ่มต้นใช้ดีเบสทรีพลัส

เมื่อจะเริ่มต้นใช้ดีเบสทรีพลัส จะต้องทำความเข้าใจกับเครื่องคอมพิวเตอร์ที่จะใช้เสียก่อน ซอฟต์แวร์ดีเบสทรีพลัส จะต้องทำความเข้าใจกับเครื่องคอมพิวเตอร์ที่จะใช้เสียก่อน ซอฟต์แวร์ดีเบสทรีพลัสจะถูกเก็บในแผ่นดิสเก็ตต์ในลักษณะของไฟล์เช่นเดียวกัน แต่เป็นโปรแกรมไฟล์ ดังนั้นการที่จะใช้ดีเบสได้นั้นจะต้องมีคอมพิวเตอร์ที่ประกอบด้วย

1. ดิสต์ไดร์อย่างน้อย 2 ตัว
2. แผ่นดิสต์เก็ตที่มีโปรแกรมดีเบสทรีพลัสอยู่
3. เครื่องคอมพิวเตอร์ที่มีซีพียู สามารถรับได้ทั้งโปรแกรมระบบจัดการ (operation system) ทั้ง MS-DOS หรือ PC-DOS ก็ได้

4. มีหน่วยความจำ RAM อย่างน้อย 256 KB.

5. จอภาพที่ใช้จะใช้จอโมโนโครม หรือจอสีก็ได้ซึ่งตัวดีเบสก็มีคำสั่งสำหรับเลือกสีของสกรีนด้วย

ไฟล์ที่มีในแผ่นดีเบสทรีพลัส

พิจารณารายการต่าง ๆ บนจอภาพ เมื่อคุณใช้คำสั่ง DIR จะพบไฟล์ต่าง ๆ บนแผ่นดิสเก็ตต์ ดังนี้

- COMMAND.COM	โปรแกรมระบบจัดการ
- dBASE.EXE	โปรแกรมหลักของดีเบสทรีพลัส โดยจะเป็นชื่อที่สั่งจากพร้อมท์เมื่อต้องการเข้าใช้ดีเบสทรีพลัส
- dBASE.MSG	เป็นไฟล์ประเภท Message หรือ เท็กซ์ไฟล์ซึ่งเก็บเมนูและคำสั่งบางคำสั่งของ - ดีเบสทรีพลัสต้องการทำงานตามคำสั่งใดที่ผู้ใช้กำหนดก็จะโหลดไฟล์นี้ลงหน่วยความจำ
- HELP.DBS	เป็นไฟล์ที่เก็บรายละเอียดของคำสั่งที่ผู้ใช้สามารถเรียกดูได้เมื่อต้องการขอความช่วยเหลือในระหว่างการใช้ดีเบสทรีพลัส
- ASSEST.HLP	เป็นไฟล์ที่เก็บเมนูในโหมดของแอสซิสต์ (ASSIST)

การทำงานในโหมดแอสชิสต์

เมื่อเข้าสู่ดีเบสทรีพลัส เราจะเห็นเมนูบนจอภาพ ซึ่งเมนูนี้อยู่ในโหมดการทำงานแบบแอสชิสต์ เมนูของดีเบสทรีพลัสมีลักษณะเป็นเมนู 2 มิติ หรือพูลดาวน์เมนู (Pull Down Menu) ทำให้สามารถได้ใน 2 ทิศทางตามการกดลูกศรเลื่อน คำสั่งหลักในเมนูแสดงในบรรทัดบนสุดซึ่งมีอยู่ 8 คำสั่งคือ

- Set - Up ใช้หลักไฟล์จอหน่วยความจำ
- Create ใช้เพื่อสร้างไฟล์ต่าง ๆ
- Update ใช้เพื่อปรับปรุง เปลี่ยนแปลง เพิ่มลด และแก้ไขข้อมูล
- Position ใช้เพื่อค้นหาข้อมูล
- Retrieve ใช้เพื่อเรียกดูข้อมูล
- Organize ใช้เพื่อเรียงลำดับข้อมูลและก๊อปปี้ไฟล์
- Modify ใช้เพื่อแก้ไขโครงสร้างของไฟล์
- Tools ใช้เพื่อจัดการเกี่ยวกับดิสต์ไดร์ เช่น การเปลี่ยนไฟล์, ก๊อปปี้ไฟล์ เป็นต้น

แต่การใช้งานดีเบสทรีพลัสในโหมดนี้ ทำงานได้ไม่ต่อเนื่องตามลำดับกล่าวคือ เมื่อผู้ใช้สั่งงานดีเบสทรีพลัสทางเมนูหลักแล้ว ดีเบสทรีพลัสจะดำเนินงานตามนั้น จนเสร็จ แล้วก็กลับเมนูหลักนั้นอีก ผู้ใช้ก็ต้องสั่งดีเบสจากเมนูหลักนั้นอีกจนเสร็จงาน จะเห็นว่าไม่ค่อยสะดวกนัก อีกทั้งสำหรับผู้เริ่มต้นหรือไม่ได้ใช้งานกับดีเบสทรีพลัสอยู่เป็นประจำแล้ว จะสร้างความยุ่งยากให้กับผู้ใช้เป็นอันมาก ทั้งนี้เพราะการที่จะสั่งจากเมนูหลักนั้น ดีเบสทรีพลัสได้กำหนดเงื่อนไขและขอบเขต (Scope) มากมายจะสลับซับซ้อนและยุ่งยากเกินไป

ดังนั้น ผู้จัดทำโครงงานจึงเสนอโครงงานที่นำเอาคำสั่งต่าง ๆ ของดีเบสมาเขียนเป็นโปรแกรม เพื่อทดลองควบคุมจัดเก็บข้อมูลเกี่ยวกับวัสดุก่อสร้างแทน ด้วยเหตุนี้การทำงานในโหมดแอสซิสต์จึงขอกล่าวเพียงสังเขปไว้เท่านี้

บรรทัดแสดงสถานะ

การออกจากแอสซิสเมนท์ทำได้ด้วยการกดคีย์ ESC เมนูหลักจะถูกลบออกจากจอภาพมาอยู่ที่บรรทัดแสดงสถานะบรรทัดชี้นำและบรรทัดอธิบายการทำงาน เมื่อต้องกลับสู่แอสซิสเมนูใหญ่ ก็ทำได้ด้วยการกดคีย์ F2 จะเข้าสู่สภาพการทำงานแบบมีเมนูของแอสซิสต์อีกครั้ง

สำหรับบรรทัดแสดงสถานะที่อยู่ด้านล่างของจอภาพนั้น ถ้าผู้ใช้ไม่ต้องการให้แสดงเพื่อให้มีพื้นที่ว่างของจอภาพมากขึ้นเราสามารถสั่งให้ดีเบส-ทรีพลัสเลิกแสดงบรรทัดนี้ได้โดยคีย์คำสั่ง

```
SET  STATUS  OFF
```

และเมื่อต้องการให้แสดงบรรทัดนี้ใหม่ ก็ให้คีย์คำสั่ง

```
SET  STATUS  ON
```

การสั่งงานดีเบสทรีพลัส จากบรรทัดแสดงสถานะนี้ มีประโยชน์มาก เพราะเวลาคุณเขียนโปรแกรมแล้วติดขัด ไม่แน่ใจว่าคำสั่งที่ใช้นั้น มันถูกต้องไหม ก็ทดลองสั่งมันจากบรรทัดแสดงสถานะนี้ได้

การสร้างไฟล์ข้อมูล

เมื่อเราต้องการสร้างไฟล์ข้อมูลในดีเบสทรีพลัส จะใช้คำสั่ง CREATE ในบรรทัดแสดงสถานะดังนี้

.CREATE

แล้วดีเบสจะถามชื่อไฟล์ที่ต้องการให้สร้าง

.ENTER FILENAME :

โดยดีเบสทรีพลัสก็มีข้อกำหนดชื่อไฟล์เหมือนกัน ดังนี้

- ชื่อไฟล์ประกอบด้วยตัวอักษร,ตัวเลขและเครื่องหมาย '_' แต่อักษรตัวแรกของชื่อไฟล์ต้องเป็นตัวอักษรเสมอ และถ้ามีความยาวเกิน 8 ตัวอักขระ ตัวที่เกินจะถูกตัดออก ส่วนนามสกุลดีเบสจะกำหนดให้เป็น dbf(Database File) เสมอ

- สามารถระบุดิสต์ไดร์ที่จะให้ดีเบสทรีพลัสไปสร้างไฟล์ได้ เช่น ในโครงงานของเรา ต้องการให้สร้างไฟล์ Listing.dbf บนไดร์ B ถึงระบุเป็น

ENTER FILENAME : B:Listing

และหลังจากนั้น ดีเบสทรีพลัสจะให้กำหนดโครงสร้างของไฟล์ อันประกอบด้วย

- Field Name หมายถึง ชื่อของฟิลด์ที่จะตั้งให้แต่ละฟิลด์
- Type หมายถึง ประกอบข้อมูลที่เก็บในฟิลด์นั้น
- Width หมายถึง จำนวนตัวอักษรสูงสุดที่จะให้เก็บในฟิลด์นั้น

- Dec หมายถึง จำนวนตำแหน่งหลังจุดทศนิยม (Decimal Placce) สำหรับข้อมูลชนิดตัวเลข (Numeric) ของฟิลด์นั้น เช่น ในโครงงานมีไฟล์ Listing.dbf มีฟิลด์ Per_Unit สำหรับเก็บราคาขายต่อหน่วยของวัสดุก่อสร้างเป็นตัวเลข ได้กำหนด Width มีขนาด 9 ตัวอักขระโดยแบ่งเป็น 6 อักขระหน่วยบาท (ดังนั้น จะเก็บข้อมูลได้สูงสุด 100,000 บาท) จุดทศนิยม 1 ตัว และ 2 อักขระสำหรับสตางค์

นอกจากนี้แล้วโครงสร้างข้อมูลของดีเบสทรีพลัสมีข้อกำหนดเพิ่มเติมอีก ดังนี้

ข้อจำกัดของโครงสร้างข้อมูล

1. จำนวนฟิลด์สูงสุดของแต่ละเรคคอร์ดคือ 128 ฟิลด์
2. ชื่อของฟิลด์ (Field Name) ตั้งได้สูงสุด 10 ตัวอักษร จะเป็นอักษรตัวใหญ่หรือเล็กก็ได้
3. ประเภทของข้อมูลที่จะเก็บสามารถแบ่งออกได้เป็น 5 ประเภทคือ

- ตัวอักษร (Character) เป็นข้อมูลที่เก็บตัวอักษรตัวเลขสัญลักษณ์หรือเครื่องหมายต่าง ๆ มีความกว้างสูงสุดไม่เกิน 254 ตัวอักษร

- ตัวเลข (Numeric) เป็นข้อมูลซึ่งเก็บตัวเลขสามารถนำมาคำนวณทางคณิตศาสตร์ได้มีความกว้างสูงสุดไม่เกิน 19 ตัวเลข

- ตรรก (Logical) เป็นข้อมูลที่มีค่าทางตรรก มีความกว้างเพียง 1 ตัวอักษรโดยถ้าจริง (Time) จะเป็น .T.,Y.แต่ถ้าเป็นเท็จ จะเก็บข้อมูลเป็น.F.,.N.

- วันที่ (Date) เป็นข้อมูลชนิดวันที่ ซึ่งดีเบสทรีพลัสจะเก็บไว้ในลักษณะจูเลียนเดท (ตัวเลขจำนวนวันนับจากวันที่ 1 มกราคม ค.ศ.1990) ความกว้างของฟิลด์เป็น 8 ตัวอักษร เป็นเดือน/วันที่/ปี ค.ศ. แบบ MM/DD/YY

- บันทึกช่วยจำ (Memo) เป็นข้อมูลชนิดตัวอักษรใช้บันทึกข้อความเฉพาะบางเรคคอร์ มีความกว้างต่ำสุด 10 ตัวอักษร และสูงสุด 5,000 ตัวอักษร

โดยในการเลือกแบบฟิลด์นั้น จะเลือกตามอักษรตัวแรกของชื่อแบบฟิลด์ข้างต้น เช่น ชื่อวัสดุก่อสร้างของเรา จะเป็นตัวอักษรดังนั้นจึงเลือกแบบเป็น C แล้วเลือกตามขนาดอีกทีนึง ตอนนี้ก็จะได้ไฟล์ข้อมูล พร้อมจะเก็บข้อมูลต่อไปแล้ว

แล้วหลังจากที่คุณได้ไฟล์ดีเบส และทำการเก็บข้อมูลไปแล้ว ถ้าต้องการเปลี่ยนแปลงโครงสร้างของฟิลด์เพิ่มเติม เช่น ในโครงงานเดิมที่ไม่คิดจะให้มีการโอนย้ายข้อมูลกัน ระหว่างไฟล์ Listing.dbf กับไฟล์ Store.dbf แต่หลังจากเขียนโปรแกรมไปสักพักก็คิดจะทำเพิ่มในส่วนนั้น จึงต้องมาเพิ่มฟิลด์ Change สำหรับกำหนดว่าเป็นวัสดุก่อสร้างที่เพิ่งเข้า,ออกใหม่ จึงต้องแก้ไขโครงสร้างของไฟล์ทั้งสองใหม่ ด้วยคำสั่ง

Modify Structure

หลังจากสั่ง

Use B: Listing.dbf

แล้ว ซึ่งคำสั่งที่สองที่จะกล่าวถึงอีกครั้งนึง ส่วนการแก้ไขไฟล์ Store.dbf ก็ทำได้คล้ายกัน

การป้อนข้อมูลลงไฟล์

เมื่อถูกกำหนดโครงสร้างของไฟล์เรียบร้อยแล้ว ดีเบสจะถามว่าต้องการป้อนข้อมูลเลยไหม

"INPUT DATA NOW?"

ถ้าตอบตกลง ก็จะให้คุณป้อนข้อมูล ตามโครงสร้างของไฟล์ที่คุณกำหนดนั้น

แต่ถ้าคุณมีไฟล์ข้อมูลอยู่แล้ว และต้องการจะเพิ่มเติมข้อมูลเป็นเรคคอร์ดใหม่ หรือจะแก้ไขข้อมูลกับเรคคอร์ดเดิมนั้น ก็สามารถทำได้ ดังนี้ เริ่มต้นคุณต้องกำหนดไฟล์ข้อมูล ที่คุณจะใช้เพื่อทำงานเหล่านั้นเสียก่อน

USE B: Listing

กรณีคุณสั่งดีเบสให้โหลดข้อมูลของไฟล์ Listing.dbf ในไดร์ B ลงหน่วยความจำ (ถ้าไม่กำหนดดิสต์ไดร์ ดีเบสจะให้ลดไฟล์จากไดร์ปัจจุบันเสมอ คุณจะได้ข้อมูลที่แต่ละเรคคอร์ดเรียงลำดับกันตามหมายเลขเรคคอร์ด มากน้อยไปมาก แต่ถ้าคุณต้องการให้ดีเบสให้ลดไฟล์นี้แต่ให้เรียงเรคคอร์ดกันตามอินเด็กซ์ไฟล์ ที่คุณได้สร้างนั้น คุณต้องใช้คำสั่ง

USE B: Listing INDEX Deslndex

ชื่งไฟล์ Deslndex.ndx ได้ใช้ชื่อวัสดุก่อสร้างเป็นหลักในการเรียงลำดับเรคคอร์ด ดังนั้น เมื่อโหลดข้อมูลลงหน่วยความจำแล้ว คุณจะพบว่า ที่บรรทัดแสดงสถานะ หมายเลขเรคคอร์ดที่ดีเบสแสดงไว้ จะไม่ใช่หมายเลข 1 แล้ว แต่จะเป็นหมายเลขเรคคอร์ดที่อักษรตัวแรกสุดของชื่อวัสดุก่อสร้างในเรคคอร์ดนั้นเรียงอยู่ในอักษรตัวแรก ๆ จาก "A"-"Z"

นอกจากนี้แล้ว ดีเบสยังกำหนดให้คุณสามารถเปิดไฟล์เพื่อใช้งานพร้อมกันได้ทีละไฟล์ ด้วยการจองพื้นที่ใช้งานบนหน่วยความจำ ที่ไฟล์เหล่านั้นถูกโหลดลงหน่วยความจำ ด้วยคำสั่ง

```
SELECT 1
USE D: Listing INDEX Dexlnder ALIAS List
```

จากคำสั่งนี้คุณกำหนดพื้นที่ใช้งานที่ 1 ให้เก็บข้อมูลของไฟล์ข้อมูล Listing ตามที่อินเด็กซ์ไฟล์ Deslndex บอกลำดับการเรียงเรคคอร์ดโดยให้มีชื่อเรียก ALIAS ดังนั้นคราวต่อไปเมื่อคุณต้องการเรียกใช้พื้นที่ใช้งานนี้ คุณก็ใช้คำสั่ง

```
SELECT LIST
```

แทนคำสั่ง USE ได้เลย แต่จากการใช้งาน จะพบว่ามีปัญหามาก จากการใช้คำสั่งนี้ กล่าวคือ เมื่อโปรแกรมมีขนาดใหญ่ขึ้น การใช้คำสั่งนี้จะถูกคำสั่งอื่นที่จะให้ดีเบสทำงาน กับมัน จึงพบปัญหาว่าดีเบสจะฟ้องว่าไม่รู้จักชื่อ "List" นี้ และเบรคการทำงาน ดังนั้นจึงเลือกใช้คำสั่ง USE ชื่อไฟล์เป็นส่วนใหญ่ และถ้าจำเป็นต้องใช้คำสั่ง Select นี้จริง ๆ ก็ได้โปรแกรม ERR.Prg ให้แก้ไขข้อบกพร่องให้ ดังได้แสดงโปรแกรมไว้ในภาคผนวกแล้ว

มาถึงตรงนี้ ก่อนที่จะพูดถึงการเพิ่มเติม หรือแก้ไขข้อมูล มาดูการใช้ไฟล์ได้ด้วยคำสั่งพื้นฐานกันก่อน

- List คำสั่งนี้จะเหมือนกับคำสั่ง Display ในแง่ที่ว่า มันสามารถแสดงผลตามขอบเขตและเงื่อนไขที่กำหนดได้ แต่ละไม่หยุดรอการกดคีย์ ถ้ามีเรคคอร์ดที่จะแสดงผลได้ในหน้าจอเดียว หรือเกิน 20 เรคคอร์ด เหมือนคำสั่ง Display แต่ถ้าคุณต้องการเบรคก็กดคีย์ "Ctrl-S" หรือ "Break" ได้ ดังนั้น คำสั่งนี้จึงเหมาะสำหรับแสดงผลออกทางเครื่องพิมพ์เท่านั้น เช่น ถ้าคุณต้องการพิมพ์ไฟล์ Listing.dbf เพื่อดูว่าไฟล์นี้มีเรคคอร์ดได้บ้าง ที่มีชื่อวัสดุก่อสร้างเป็น "Hollow block" คุณก็ใช้คำสั่ง

```
.Set Print On
.List For Descript = "HOLLOW BLOCK"
```

มันก็จะทำงานให้จนเสร็จ แล้วคุณก็สั่งให้ดีเบสหยุดส่งข้อมูลไปทางเครื่องพิมพ์ด้วยคำสั่ง

```
.Set Print Off
```

จากข้างต้น คุณจะพบว่า ทั้งสองคำสั่งนี้ด้อยประสิทธิภาพมาก ดังนั้นเราจึงได้โปรแกรม Listing.Prg ขึ้น เพื่อให้สามารถแสดงผลข้อมูลในไฟล์ Listing.dbf ได้คล้ายหน้ากระดาษ กล่าวคือ

- สามารถเลือกจะดูข้อมูลของวัสดุก่อสร้างใด ก็ได้ด้วยการป้อนชื่อ
- สามารถเลื่อนหน้ากระดาษด้วยฟังก์ชั่นคีย์ PgUp สำหรับเลื่อนหน้าขึ้น และ PgDn สำหรับเลื่อนหน้าลง
- สามารถเพิ่มเติมเรคคอร์ดได้
- จบการทำงานด้วยการกดฟังก์ชั่นกับ Cntr-End ได้ซึ่ง

โปรแกรม Listing.Prg นี้ คุณจะดูได้จากภาคผนวกท้ายเล่ม

นอกจากจะดูรายละเอียดของวัสดุก่อสร้างแต่ละชนิดแล้ว คุณสามารถดูข้อมูลเกี่ยวกับผู้ขายได้ โดยโปรแกรม Sellman.Prg จะบอกคุณได้ว่า ผู้ขายคนใดขายวัสดุก่อสร้างให้เราบ้าง และราคาเท่าไร การใช้งานก็ทำได้เหมือนโปรแกรม Listing.Prg ทุกอย่าง

การเลื่อนเรคคอร์ดขึ้นหรือลงจากเรคคอร์ดปัจจุบัน

ดีเบสจะมีคำสั่งสำหรับเลื่อนเรคคอร์ด ดังนี้

SKIP ± n

เมื่อ n เป็นจำนวนเรคคอร์ดที่จะเลื่อนข้ามไป ถ้าค่าเป็นบวกหมายถึง เลื่อนเรคคอร์ดจากปัจจุบันไปยังเรคคอร์ดที่อยู่ถัดไป เช่นจากเรคคอร์ดที่ 1 ไปยังเรคคอร์ดที่ 2 และการทำงานจะมีอินเด็กซ์ไฟล์คอยกำกับเสมอ ถ้าเรียกใช้ไฟล์ข้อมูลนั้น กับอินเด็กซ์ไฟล์ของมัน

ส่วนคำลบ ถึงจะทำตรงกันข้ามกัน กล่าวคือ จะเลื่อนเรคคอร์ดปัจจุบันไปยังเรคคอร์ดก่อนหน้านั้น ตามจำนวนที่กำหนดโดยมีการชี้ด้วยอิเด็กซ์เช่นกัน และทั้งสองรูปแบบนี้ จะหยุดเสมอ ถ้าเลื่อนจนสุดไฟล์ (End of File) หรือต้นไฟล์ (Begin of File) แล้ว

การเลื่อนเรคคอร์ดข้างต้น จะสังเกตุเห็นว่าเป็นการเลื่อนเรคคอร์ดด้วยค่าจัด ระหว่างเรคคอร์ดปลายทางกับเรคคอร์ดปัจจุบันนอกจากวิธีนี้แล้ว ดีเบสยังมีคำสั่งเลื่อนเรคคอร์ดอีกคำสั่งนึง คือ

GOTO

มีรายละเอียดของการใช้งานดังนี้

.GOTO n

n จะเป็นหมายเลขของเรคคอร์ดปลายทาง ซึ่งอาจอยู่ก่อนหรือถัดไป เมื่อเทียบกับเรคคอร์ดปัจจุบันก็ได้

.GOTO Top

จะเลื่อนเรคคอร์ดไปยังเรคคอร์ดบนสุดของไฟล์ ตามที่อินเด็กซ์ไฟล์บอก แต่ถ้าไฟล์ข้อมูลนั้นไม่ใช้อินเด็กซ์ไฟล์ จะเลื่อนเรคคอร์ดมาอยู่ที่เรคคอร์ดหมายเลข 1 เสมอ ดังนั้นเมื่อมีคำสั่งเลื่อนขึ้นบนสุด ดีเบสก็มีคำสั่งเลื่อนเรคคอร์ดลงล่างสุดด้วย

.GOTO Bottom

จะเลื่อนเรคคอร์ดมาท้ายไฟล์ ที่เรคคอร์ดสุดท้าย ตามที่อินเด็กซ์ไฟล์ชี้บอก แต่ถ้าไม่ใช้อินเด็กซ์ไฟล์แล้ว จะเลื่อนไปเรคคอร์ดสุดท้ายของไฟล์เสมอ (Recnoc) Reccount()) แต่ถ้าต้องการเพิ่มเรคคอร์ดใหม่ ด้วยคำสั่ง APPEND รายละเอียดของคำสั่งนี้จะกล่าวถึงในหัวข้อถัดไป ก็จะต้องเลื่อนด้วยคำสั่ง SKID อีกทีนึง (วิธีนี้จะใช้สำหรับการเพิ่มเรคคอร์ดใหม่ ในโครงงานนี้ด้วย)

การหาเรคคอร์ดที่ต้องการ

จากคำสั่งทั้งสองคำสั่งข้างต้น จะพบว่ามัน บางครั้งมันยากที่จะบอกว่าข้อมูลที่เราต้องการนั้นอยู่ในเรคคอร์ดไหน ดังนั้นดีเบสจิงมีคำสั่งสำหรับหาเรคคอร์ดที่เราต้องการ ดังนี้

- Locate และ Continue คำสั่งสองคำสั่งนี้ ดีเบสกำหนดให้ใช้ร่วมกัน หรือจะใช้เนียงคำสั่ง Locate อย่างเดียวก็ได้ แต่คำสั่ง Continue นั้น ต้องใช้คำสั่ง Locate ก่อนเสมอ คำสั่งทั้งสองนี้มีไว้สำหรับค้นหาข้อมูลในไฟล์ตามเงื่อนไขที่ต้องการ เช่น

.LOCATE FOR DESCRIPT = "HOLLOW BLOCK"

ดีเบสก็จะค้นหาเรคคอร์ดในไฟล์ ที่มีชื่อในฟิลด์ชื่อ Descript ที่มีชื่อเป็น "HOLLOW BLOCK" และจะหยุดเรคคอร์ดแรกที่เจอ แล้วจะค้นเรคคอร์ดที่เหลือต่อจากนั้น ด้วยคำสั่ง

.Continue

และระหว่างสองคำสั่งนี้ อาจใช้คำสั่งอื่นสั่งงานดีเบสได้ เช่น

.DISPLAY

ดีเบสก็จะแสดงข้อมูลในเรคคอร์ดนั้นให้ ตัวคำสั่งทั้งสองนี้ มีรายละเอียดการทำงานที่อยากจะกล่าวถึง คือ หนึ่งมันสามารถใช้อินเด็กซ์ไฟล์เป็นตัวชี้เรคคอร์ดได้ ถ้ามี และสองคำสั่ง Continue จะทำให้มีการกันเรคคอร์ดที่เหลือ ถัดจากเรคคอร์ดปัจจุบันเสมอ ดังนั้น ถ้าคุณมีการเลื่อนเรคคอร์ด ด้วยคำสั่งอื่น มันก็จะค้นหาข้อมูลในเรคคอร์ดปัจจุบันขณะนั้น ไม่ใช่จากเรคคอร์ดที่มันหยุด และสุดท้ายถ้าคุณสั่งค้นข้อมูล ด้วยคำสั่ง Locate เรคคอร์ดแรกที่มันจะดูข้อมูลเพื่อเปรียบเทียบกับเงื่อนไข หรือขอบเขตที่คุณกำหนด คือเรคคอร์ดปัจจุบันนั้น แต่ถ้าคุณสั่งต่อ ด้วยคำสั่ง Continue มันจะกันที่เรคคอร์ดถัดไปแทน อันนี้ คุณลองสั่ง Locate ซ้ำๆกันคุณจะพบว่ามันหยุดที่เรคคอร์ดเดิมเสมอ

- FIND เป็นคำสั่งสำหรับค้นหาข้อมูล เฉพาะไฟล์ที่ได้ทำอินเด็กซ์ และกำลังเรียกใช้อินเด็กซ์ไฟล์นั้นเท่านั้น และข้อมูลที่ต้องการให้ค้นหา ต้องอยู่ในฟิลด์ซึ่งเป็นหลักที่ใช้ในการทำอินเด็กซ์ เช่น ไฟล์ข้อมูล Listing.dbf มีอินเด็กซ์ไฟล์ Deslndex.ndx ซึ่งทำอินเด็กซ์โดยใช้ฟิลด์ Descript หรือชื่อวัสดุก่อสร้างเป็นหลัก ดังนั้น จะค้นหาข้อมูลได้เฉพาะชื่อวัสดุก่อสร้าง ด้วยคำสั่งนี้คุณจะพบว่า แม้ว่าข้อจำกัดมันมีมาก แต่ความเร็วมันก็สูงตาม กล่าวคือ ไม่ว่าเรคคอร์ดของคุณอยู่ตำแหน่งใดของไฟล์ก็ตาม มันจะ

ใช้เวลาในการค้นหาเพียง 2 วินาทีเท่านั้น และการค้นหาข้อมูล จะค้นจากต้นไฟล์ตามที่อินเด็กซ์บอก จนพบเรคคอร์ดที่ต้องการจึงหยุดไม่เช่นนั้นแล้วจะค้นจนจบไฟล์ (End of file)

- SEEK เป็นคำสั่งที่ทำงานเหมือนกับคำสั่ง FIND ยกเว้นสามารถค้นหาข้อมูลในไฟล์ ไม่เฉพาะแต่ฟิลด์ที่ได้ทำการอินเด็กซ์ ไว้เท่านั้น แต่คำสั่ง SEEK มีข้อบังคับอยู่ว่า ถ้าข้อมูลที่ต้องการค้นหาเป็นตัวอักษร (หมายถึง ฟิลด์นั้นคุณกำหนดให้เป็นแบบ Character) คุณจะต้องเขียนข้อมูลที่ต้องการค้นไว้ในเครื่องหมายคำพูดเสมอ เช่น

```
.SEEK "KITTIPAN"
```

ชื่อ KITTIPAN เป็นชื่อผู้ขายซึ่งไม่ใช่ฟิลด์ที่ใช้อินเด็กซ์ ส่วนตัวเลข (ฟิลด์แบบ Numeric) จะต้องไม่อยู่ในเครื่องหมายคำพูด และถ้าคุณไปสั่งในเครื่องหมายคำพูด เหมือนฟิลด์แบบตัวอักษร ดีเบสจะหาไม่พบ

แต่คำสั่ง SEEK นี้ มีข้อด้อยกว่าคำสั่ง FIND อยู่นิดนึงตรงที่ ข้อมูลที่ต้องการให้ดีเบสนำไปค้นนั้น ต้องเหมือนกับที่เก็บไว้ในไฟล์ข้อมูลทุกอย่าง เช่น ชื่อสินค้า HOLLOW BLOCK เมื่อใช้กับคำสั่ง SEEK จะเป็น

```
.SEEK "HOLLOW BLOCK"
```

แต่สำหรับคำสั่ง FIND อาจใช้แต่

```
.FIND  "HOL"
```

ก็ได้ เพราะเรามีชื่อที่อักษร 3 ตัวแรก ของชื่อวัสดุก่อสร้างเป็น "HOL" เพียง BLOCK ตัวเดียว ดังนั้นโปรแกรมของโครงงานนี้จึงใช้เพียงคำสั่ง Locate , Continue กับ Find เท่านั้น

อนึ่ง การค้นด้วยคำสั่ง FIND นี้ข้อมูลที่กำหนดให้ค้นหานั้น จะอยู่ในเครื่องหมายคำพูดหรือไม่ก็ได้ ดีเบสทำงานได้ถูกต้องเหมือนกัน

การขอแก้ไขข้อมูลเก่าและการเพิ่มเติมเรคคอร์ดใหม่

- EDIT เป็นคำสั่งสำหรับแก้ไขข้อมูลในเรคคอร์ดเดิม ที่มีอยู่แล้ว โดยจะเลือกความหมายเลขเรคคอร์ด ก็ได้

```
.EDIT 5
```

หรือกำหนดขอบเขตและเงื่อนไข สำหรับเรคคอร์ดบางเรคคอร์ดก็ได้

```
.EDIT FOR DESCRIPT = "HOLLOW BLOCK"
```

จะเป็นการขอแก้ไขเรคคอร์ดที่มีฟิลด์ Descript เก็บชื่อ "HOLLOW BLOCK" อยู่ และถ้าต้องการเลื่อนไปยังเรคคอร์ดอื่น ดีเบสก็กำหนดให้ใช้คีย์ PgUp ในการเลื่อนไปยังเรคคอร์ดก่อนหน้าเรคคอร์ดนั้น 1 เรคคอร์ด และคีย์ PgDn จะใช้เลื่อนเรคคอร์ดไปยังเรคคอร์ดถัดไปตามที่อินเด็กซ์กำกับและถ้าขอแก้ไขจนสุดไฟล์แล้วยังทำการ Edit ต่อ ก็จะทำการเพิ่มเรคคอร์ดใหม่ให้ด้วย จนเสร็จผู้ใช้ก็กด Ctrl-End ก็เสร็จ

- BROWSE จะเป็นการขอแก้ไขเรคคอร์ดคราวละหลายๆ เรคอร์ด โดยกำหนดขอบเขตได้มากมาย เช่น

```
.BROWSE                                  ขอแก้ไขข้อมูลทุกฟิลด์
.BROWSE FIELD DESCRIPT                   ขอแก้ไขฟิลด์ชื่อวัสดุก่อสร้างเท่านั้น
.BROWSE  FIELDS DESCRIPT,DELIVERY        ขอแก้ไขเฉพาะฟิลด์ชื่อวัสดุ
                                         ก่อสร้าง กับจำนวนของเข้าเท่านั้น
```

.BROWSE LOCK 2 เป็นการล็อคฟิลด์ 2 ฟิลด์แรกของเรคคอร์ด บนจอภาพ ไม่ให้เลื่อนออกนอกจอถ้ามีการเลื่อนฟิลด์ไปทางซ้ายมือด้วยคีย์ลูกศรเลื่อนขวา

.BROWSE FREEZE DELIVERY ขอแก้ไขเฉพาะฟิลด์จำนวนของเข้าเท่านั้น กรณีนี้ จะเห็นฟิลด์จำนวนของเข้าแสดงผลด้วยรีเวิร์สวิดีโอ ส่วนฟิลด์อื่นเป็นตัวอักษรแบบปกติ และเคอร์เซอร์จะเลื่อนในฟิลด์นี้ตลอดการทำงาน

คำสั่ง BROWSE นี้ สามารถใช้คีย์ PgUp และ PgDn เพื่อเลื่อนหน้าขึ้นและลงได้ตามลำดับ แล้วเมื่อเสร็จงาน ต้องการเซฟข้อมูลที่ได้ทำการแก้ไขก็ให้กด Ctrl-End แต่ถ้าต้องการยกเลิกที่แก้ไข ไม่ต้องเซฟลงดิสต์แทนข้อมูลเดิม ก็ให้ใช้ ESC คีย์แทน

- APPEND เป็นการขอเพิ่มเติมเรคคอร์ดใหม่ ต่อท้ายเรคคอร์ดเดิมที่มีอยู่ในไฟล์ที่ใช้งานปัจจุบัน โดยมีรูปแบบการใช้งานตามโครงงานนี้ดังนี้

.APPEND BLANK เป็นการขอเพิ่มเรคคอร์ดว่าง 1 เรคคอร์ด

.APPEND FROM STORE FOR CHANGE คำสั่งนี้จะเขียนอยู่ในโปรแกรม Make.Prg สำหรับย้ายข้อมูลที่มีการแก้ไขเพิ่มเติม หรือสร้างใหม่จากไฟล์ Store.dbf มาเก็บเป็นประวัติไว้ในไฟล์ Listing.dbf โดยมีฟิลด์ CHANGE คอยกำกับ ว่าเรคคอร์ดใดมีการแก้ไขเพิ่มเติม

หรือสร้างขึ้นใหม่ เรคคอร์ดนั้น จะเซ็ทให้ฟิลด์ CHANGE มีค่าเป็นจริงหรือ .T.)

- INSERT เป็นคำสั่งสำหรับแทรกเรคคอร์ดใหม่ลงระหว่างเรคคอร์ดเก่าที่มีอยู่ ตัวคำสั่งมี 3 รูปแบบ คือ

.INSERT BLANK BEFORE เป็นการขอแทรกเรคคอร์ดว่าง (Blank) ก่อนหน้าเรคคอร์ดปัจจุบัน

.INSERT BEFORE เป็นการขอแทรกข้อมูลลงก่อนหน้าเรคคอร์ดปัจจุบัน กรณีนี้ดีเบสจะให้เพิ่มเติมข้อมูลคล้ายกับคำสั่ง Edit หลังทำคำสั่งนี้

.INSERT ขอแทรกเรคคอร์ดลงหลังเรคคอร์ดปัจจุบัน จะสังเกตุเห็นว่าคำสั่ง Insert นี้ จะมีประโยชน์มากกับไฟล์ที่ไม่ได้เรียงอินเด็กซ์ไว้ แต่โครงงานนี้ จะใช้ไฟล์ที่ทำอินเด็กซ์ทุกไฟล์ ดังนั้นจึงไม่นำคำสั่งนี้มาใช้

ส่วนคำสั่ง Edit กับ Brows นั้น ก็ไม่มีศักยภาพเพียงพอที่จะนำมาใช้ในการเขียนโปรแกรม เพราะดีเบสมีจุดมุ่งหมายให้ใช้สองคำสั่งนี้เป็นเอกเทศยากต่อการทำงานอื่น ดังนั้นในการแก้ไขข้อมูลพร้อมกับแสดงผล นอกจากคำสั่ง APPEND แล้ว คำสั่งที่เหลือทั้ง 3 คำสั่งของหัวข้อนี้ ไม่ได้ถูกนำมาใช้เลย

แต่จะใช้คำสั่งสำหรับการแก้ไข ร่วมกันหลายคำสั่ง ดังนี้

- @ Row,Colume SAY (ข้อความ) GET ฟิลด์ READ โดยที่ Row,Colume เป็นตัวเลข บอกตำแหน่งที่จะให้แสดงผลบนจอภาพเป็นบรรทัดที่และคอลัมภ์ที่ตามลำดับ เราจะนำเอาข้อมูลเก่าไปแสดงผลบนจอภาพก่อน แล้วรับการแก้ไขข้อมูลในฟิลด์ที่แสดงผลด้วยคำสั่งคู่ Get กับ Read โดยการรับอาจใช้ฟิลด์ที่ต้องการรับข้อมูลโดยตรงก็ได้ คีเบสจะจัดข้อมูลใหม่ที่ได้แก้ไขไปแล้ว หรือข้อมูลเก่าถ้าไม่ได้แก้ไขเพิ่มเติม (ทำได้ด้วยการกดริเทวิน) มาจัดเก็บลงในเรคคอร์ดเดินนั้น แต่ถ้าเราใช้ตัวแปรอื่นใดในการรับข้อมูลใหม่แทน หลังจากคีเบสรับค่าที่ต้องการด้วยตัวแปรนั้นแล้ว เราต้องใช้คำสั่ง

- REPLACE Delivery WITH A

กรณีนี้ ได้ใช้ตัวแปร A รับข้อมูล แล้วจึงนำข้อมูลที่ได้มาแทนลงในฟิลด์จำนวนของเข้า (Delivery) ดังนั้น การใช้สองคำสั่งนี้ร่วมกันจึงมีประโยชน์มาก กล่าวคือ เรากำหนดตำแหน่งต่าง ๆ ที่จะให้แสดงผลและรับข้อมูล บนจอภาพได้อย่างอิสระ โดยไม่ต้องมีข้อจำกัดเหมือนคำสั่ง Edit และสามารถรับข้อมูลบางฟิลด์เฉพาะที่ต้องการให้ผู้ใช้ป้อนได้ เช่น โปรแกรม Make.Prg ได้กำหนดให้ผู้ใช้ป้อนข้อมูลใหม่เฉพาะฟิลด์ที่,จำนวนของเข้า,ของออก และชื่อผู้ขายกับราคาเท่านั้น (โดยจะแสดงข้อมูลเดิม หรือ อ่านวันที่จากระบบมาแสดง ถ้าผู้ใช้พอใจก็กดรีเทิร์นผ่านได้เลย) แล้วนำเอาจำนวนของเข้ารวมกับจำนวนของเข้ารวมทั้งหมด จัดเก็บในฟิลด์จำนวนของเข้าทั้งหมดให้เองด้วยคำสั่ง Replace ส่วนจำนวนของออกรวมและจำนวนที่เหลืออยู่ในสต๊อกของวัสดุก่อสร้างก็ทำคล้ายกัน

และสุดท้าย ยังทำงานบางอย่างได้ขณะที่แสดงผลไปแล้ว กล่าวคือ คำสั่ง

@ Row,Colume SAY (ข้อความ)

จะใช้แสดงข้อมูลบางฟิลด์ของเรคคอร์ด แล้วจึงรับการกดฟังก์ชั่นคีย์ต่าง ๆ ระหว่างที่อยู่ในโปรแกรม ไม่ว่าจะเป็น Make.Prg , Listing.Prg และ Sellman.Prg ก็ตาม ดังได้แสดงรายละเอียดการทำงานตามฟังก์ชั่นไว้ในบทต่อไปแล้ว

<u>คำสั่งเกี่ยวกับการลบเรคคอร์ด</u>

ดีเบสมีคำสั่งเกี่ยวกับการลบเรคคอร์ด ดังนี้

- DELETE เป็นคำสั่งเตรียมลบเรคคอร์ด แต่ยังไม่ลบจริง เรายังสามารถใช้งานเรคคอร์ดนั้นได้ตามปรกติ โดยการทำงานนั้น ดีเบสจะมาร์คเรคคอร์ดถูกสั่งให้เตรียมลบ ด้วยเครื่องหมายดอกจัน ("*") หลังหมายเลขเรคคอร์ด โดยสามารถใช้ฟังก์ชั่น DELETE() ในการตรวจสอบว่าเป็นเรคคอร์ดที่ถูกสั่งเตรียมลบไว้แล้วหรือไม่ กล่าวคือ คำสั่งจะให้ค่าจริง (".T.") ถ้าเป็นเรคคอร์ดที่ถูกเตรียมลบ และค่าเป็นเท็จ (".F.") ถ้ายังไม่ได้ถูกสั่งเตรียมลบมาก่อน

นอกจากนี้แล้ว คำสั่งนี้ สามารถตั้งเงื่อนไขและขอบเขตสำหรับกำหนดเรคคอร์ดหรือกลุ่มของเรคคอร์ดที่จะลบได้ เช่น

.DELETE ALL FOR DEALER = "KITTIPAN"

จะเป็นการสั่งให้ดีเบสเตรียมลบเรคคอร์ด ที่ผู้ขายมีชื่อ "KITTPAN" ทุกเรคคอร์ด แต่ถ้าต้องการเรียกเรคคอร์ดคืนจากสภาพเตรียมลบ ก็สามารถทำได้โดยใช้คำสั่ง Recall

- RECALL คำสั่งนี้จะทำงานตรงข้ามกับ คำสั่งของ Delete ข้างต้น กล่าวคือ มันสามารถเรียกคืนเรคคอร์ดจากสภาพถูกเตรียมลบไว้ และสามารถกำหนดขอบเขตและเงื่อนไขให้คำสั่งทำตามได้เช่นเดียวกัน เช่น

.RECALL ALL DEALER = "KITTIPAN"

จะเรียกเรคคอร์ดคืนจากสภาพการเตรียมลบ ด้วยคำสั่ง Delete ข้างบน ตอนนี้ถ้าใช้ฟังก์ชั่น DELETE() ตรวจสอบสถานะภาพกับเรคคอร์ดเหล่านั้น จะได้ค่าเป็นเท็จ (".F.")

- PACK คำสั่งสำหรับลบจริงกับเรคคอร์ดที่ได้ถูกสั่งเตรียมลบไว้ ซึ่งจะทำการเรียกคืนเรคคอร์ดเหล่านั้นไม่ได้แล้ว และคำสั่งนี้ก็ตั้งขอบเขต หรือเงื่อนไขใด ๆ เพิ่มเติมก็ไม่ได้ เพราะมันมองที่สถานะภาพเตรียมลบของเรคคอร์ดเท่านั้น

คุณจะเห็นแล้วว่า นี่คือข้อบกพร่องของดีเบส กล่าวคือ ถ้ามีบางเรคคอร์ดที่ถูกสั่งเตรียมไว้ แต่คุณอาจยังต้องการใช้งานอยู่คุณก็ยังใช้งานได้ แต่บางเรคคอร์ดในจำนวนนั้น คุณอาจไม่ต้องการจะใช้งานมันอีกแล้ว และต้องการจะลบมันออกเสีย คุณเกิดไปสั่ง PACK เลย โดยไม่ได้เรียกคืนเรคคอร์ดอื่นที่ต้องการใช้งานต่อผลก็คือ เรคคอร์ดเหล่านั้น จะถูกลบจริงไปด้วย ดังนั้นผู้จัดทำโครงการจึงโปรแกรม MK-7-22.Prg ขึ้นเพื่อลดข้อผิดพลาดนี้ โดยการทำงานของโปรแกรมจะให้จำเรคคอร์ดที่ได้ถูกเตรียมลบไว้แล้วให้เลือกด้วยการกดคีย์ลูกศรเลื่อนขึ้นลง ถ้าไม่ต้องการให้เรคคอร์ดใดถูกลบก็จะเก็บชื่อเรคคอร์ด นั้นไว้ในตัวแปรหน่วยความจำ และเรียกคืนมันด้วยคำสั่ง Recall ก่อน จนผู้ใช้ต้องการให้ลบเรคคอร์ดที่เหลือ (จากที่สั่งเตรียมลบไว้ แล้วไม่ได้เรียกคืน) ก็จะลบจริงให้ด้วยคำสั่ง Pack จนเสร็จ

ดีเบสจะเรียงอินเด็กซ์ให้เองโดยอัตโนมัติ จึงโปรแกรมให้ไปเตรียมลบกับเรคคอร์ดที่ได้เรียกคืนเหล่านั้นใหม่ เพื่อให้สถานะเตรียมลบมันเหมือนเดิม และบนจอภาพจะมีการเขียนข้อมูลลงไปใหม่ ตามอินเด็กซ์ไฟล์ที่บอกลำดับของเรคคอร์ดที่ยังเหลืออยู่ จากเรคคอร์ดที่ได้ลบไปแล้ว จึงเสร็จงาน

การเรียงอินเด็กซ์

เนื่องจากเราต้องเขียนเรคคอร์ดเลียนแบบหน้ากระดาษ ตามลำดับตัวอักษรของชื่อวัสดุก่อสร้าง เพื่อความสะดวกต่อการใช้งานในอันที่จะเลือกเรคคอร์ด โดยพิจารณาจากชื่อในฟิลด์ชื่อวัสดุก่อสร้างของเรคคอร์ดนั้น จึงต้องใช้คำสั่ง

- Index เป็นคำสั่งสำหรับเรียงอินเด็กซ์ ของเรคคอร์ดในไฟล์พร้อมกับสร้างไฟล์อินเด็กซ์นามสกุล .ndx ให้ และในการเรียงอินเด็กซ์นี้สามารถกำหนดรูปแบบของการเรียงได้ เช่น ในโปรแกรมของโครงงานจะมีการเรียงอินเด็กซ์หลัก ๆ ดังนี้.

.INDEX ON Descript TO Deslndex

จะเป็นการสั่งให้ดีเบสเรียงอินเด็กซ์ โดยใช้ฟิลด์ Descript เป็นหลักในการจัดเรียงอินเด็กซ์ตามลำดับตัวอักษร จาก "A".."Z" และเก็บตัวชี้ที่ได้ลงในไฟล์ Deslndex.ndx

.INDEX ON Descript + Dealer TO Double-L

จะใช้ฟิลด์ Descript เป็นหลักในการเรียงอินเด็กซ์ แต่ถ้าเรคคอร์ดใดที่มีชื่อผู้ขายเหมือนกันด้วย ก็ให้เรียงอินเด็กซ์เรคคอร์ดเหล่านั้นตามลำดับชื่อผู้ขายด้วย แล้วเก็บตัวชี้ที่ได้ลงในไฟล์ Double_L.ndx เหตุผลและรายละเอียดในการใช้งานอินเด็กซ์วิธีนี้ จะอ่านได้จากบทต่อไป ในหัวข้อโปรแกรม Sellman.prg

และการทำงานด้วยคำสั่งใดๆ ของดีเบส ที่จะมีผลทำให้เกิดการเพิ่มหรือลดเรคคอร์ดในไฟล์แล้ว ดีเบสจะเรียงอินเด็กซ์เรคคอร์ดที่เหลืออยู่ในไฟล์นั้น ให้ใหม่โดยอัตโนมัติเสมอ แต่ถ้าผู้ใช้ต้องการสั่งให้เรียงอินเด็กซ์ให้ใหม่ก็ทำได้ ด้วยการใช้คำสั่ง Reindex และนอกจากการเรียงอินเด็กซ์ด้วยสองคำสั่งนี้แล้ว ดีเบสยังมีคำสั่ง

- SORT จะเป็นการสั่งให้ดีเบส สร้างไฟล์ใหม่ขึ้นอีกไฟล์ โดยนำข้อมูลเดิมจากไฟล์ที่กำหนดให้ไปสร้างเป็นเรคคอร์ดใหม่ตามลำดับที่กำหนดในคำสั่ง Sort ดังนั้น คุณจะเห็นว่าเกิดปัญหาขึ้น 2 ข้อ คือ

หนึ่ง จะต้องใช้เนื้อที่บนดิสค์เพิ่มขึ้นอีก เท่ากับไฟล์เดิม ในการใช้สร้างไฟล์ใหม่ที่ได้

และสอง การเรียงด้วยคำสั่ง Sort นี้จะเรียงลำดับเรคคอรดตามลำดับที่กำหนดก็จริง แต่จะเป็นแบบซีเควนเชียลเหมือนไฟล์ข้อมูลทั่วไป ดังนั้นการเข้าถึงข้อมูลจริงเสียเวลามากกว่าการใช้วิธีเรียงอินเด็กซ์ด้วยคำสั่ง Index ข้างต้น

ดังนั้น ในโปรแกรมของโครงการจึงใช้คำสั่ง Index ในการเรียงอินเด็กซ์เรคคอร์ด ยกเว้นก่อนที่จะนำเอาข้อมูลในไฟล์ไปใช้ในการเขียนกราฟด้วยภาษาปาสคาลเท่านั้น

การเซทสวิทช์ต่าง ๆ ของดีเบส

สำหรับให้ผู้ใช้เลือกเซทลักษณะการทำงานของดีเบส ตามต้องการ ซึ่งมีประมาณ 40 ชนิด แต่ที่ถูกเซ็ทใหม่ในโปรแกรมของเราคือ

- SET COLER TO สำหรับเลือกสีตัวอักษร, สีพื้น, สีกรอบ ที่จะให้แสดงผลบนจอ
- SET DECIMALS TO สำหรับการกำหนดเลขหลังจุดทศนิยม ว่าผู้ใช้ต้องการกี่ตำแหน่ง ซึ่งโดยปรกติจะเป็น 2 ตำแหน่ง แต่อาจเพิ่มได้ถึง 16 ตำแหน่ง
- SET INDEX TO เลือกใช้อินเด็กซ์ไฟล์ในการค้นข้อมูล
- SET PRINT ON ส่งข้อมูลบนจอภาพออกพิมพ์ ทางเครื่องพิมพ์ Cntr-P และยกเลิกไม่ต้องส่งข้อมูลถ้ามีการเพิ่มเติมแก้ไขบนจอภาพ อีกด้วยคำสั่ง Set Print OFF
- SET TALK OFF สั่งให้ดีเบสไม่ต้องแสดงคำอธิบายหรือรายละเอียดที่เกิดจากการทำคำสั่ง เช่นคำสั่ง SKIP เดิมทีดีเบสจะบอก หมายเลขเรคคอร์ดให้ด้วย

- SET ECHO OFF	บอกให้ดีเบสไม่ต้องบอกรายละเอียดหรือคำอธิบายใดๆ เพิ่มเติมเพื่อไม่ให้มันเขียนทับข้อมูลที่กำลังแสดงผลอยู่บนจอภาพ
- SET DEFAULT TO	เลือกดิสต์ไดร์
- SET SAFETY OFF	ไม่ต้องเตือนถ้ามีการสร้างไฟล์ใหม่
- SET SCOREBOARD OFF	ไม่ต้องเตือนผู้ใช้ ถ้ามีการใช้คำสั่ง Get
- SET STATUS OFF	ไม่ต้องแสดงบนบรรทัดสถานะ เพื่อให้เรามีพื้นที่ใช้งานบนจอภาพมากขึ้น

อนึ่ง นอกจากคำสั่งเซ็ทสวิทช์ต่าง ๆ ข้างต้นนี้ ดีเบสได้เซทสวิทช์ทั้ง 40 สวิทช์ไว้อย่างเหมาะสมแล้ว เช่น "Set Bell ON" สำหรับทำเสียงบี๊บได้ ถ้าต้องการด้วยคำสั่ง "??CHR(07)" แต่การเซ็ทสวิทช์ต่างๆ ข้างต้นใหม่เพื่อให้เหมาะสมกับงานที่เราจะทำเท่านั้นเอง

สรุปฟังกชั่นต่างๆที่นำมาใช้งาน

ฟังก์ชั่นของดีเบสมีหลายฟังก์ชั่นด้วยแต่มีทั้งเหมาะที่จะนำมาเขียนเป็นโปรแกรม กับที่ไม่เหมาะสม ดังนั้นโครงงานนี้จึงใช้เพียงบางฟังก์ชั่นเท่านั้น ดังต่อไปนี้

- RECNO()	เป็นการให้ดีเบสบอกหมายเลขเรคคอร์ดปัจจุบันให้ค่าเป็นเลขจำนวนเต็ม
- DELETE()	สำหรับแสดงการเตรียมลบเรคคอร์ดหรือไม่โดยจะให้ค่าเป็นตรรก

- EOF() สำหรับแสดงการจบไฟล์แล้ว ให้ค่าเป็นจริง".T."
- BOF() สำหรับบอกขอบเขตของไฟล์ที่จุดเริ่มต้นจะให้ค่าเป็นจริง ".T."
- UPPER(STRING) สำหรับเปลี่ยนสตริงข้อความจากอักษรพิมพ์เล็กเป็นอักษรพิมพ์ใหญ่
- SPACE(ตัวเลข) เป็นฟังก์ชั่นสร้าง CHR(20) หรือ" " เป็นจำนวนตามที่ระบุเป็นตัวเลข แก่ตัวแปรที่รับ
- RIGHT(STRING,จำนวน) เป็นฟังก์ชั่นคัดจริงจากตัวอักษรทางขวามือตามจำนวนที่กำหนดไปทางซ้ายมือ และมีฟังก์ชั่น LEFT (STRING,จำนวน)ทำงานตรงข้ามกัน
- SUBSTR (STRING,STRING CHARACTER,NUMBER OF CHARACTERS) เป็นฟังก์ชั่นแยกข้อความในสตริง (STRING) จากตัวอักษรที่ STRING CHARACTER เป็นจำนวน NUMBER OF CHARACTERS
- CHR (ตัวเลขฐานสิบ) เปลี่ยนรหัสแอสกี้ ตามเลขฐานสิบที่กำหนด
- LEN (STRING) หาความยาวของสตริง
- ASC (ตัวอักษร) เปลี่ยนตัวอักษรเป็นตัวเลขฐานสิบตามรหัสของแอสกี้ของมัน
- STR (NUMBER,LENGTH) เปลี่ยนตัวเลข NUMBER ให้เป็นสตริงความยาวเพียง LENGTH แรก
- TRIM(STRING) ตัดช่องว่างข้างหลังสตริงออก

- INT (ตัวเลข)	เปลี่ยนตัวเลขจากจำนวนจริง เป็นเลขจำนวนเต็ม โดยการตัดจุดทศนิยมออก
- VAL(สตริง)	ฟังก์ชั่นเปลี่ยนสตริงตัวเลข เป็นเลขจำนวนจริง ตรงข้ามฟังก์ชั่น STR
- DATE()	ฟังก์ชั่นอ่านเวลาจากระบบของคอส
- CMONTH	ฟังก์ชั่นให้ค่าเดือน ("January",..., "December") เป็นชื่อของเดือนนั้น
- YEAR()	บอกปีของวันที่ที่กำหนดให้
- ABS(นิพจน์ตัวเลข)	หาค่าสมบูรณ์ของตัวเลข
- RECCOUNT()	หาจำนวนเรคคอร์ดของไฟล์ข้อมูล
-ERROR()	ฟังก์ชั่นการตรวจสอบข้อผิดพลาด ดูจากโปรแกรม Err.prg ในภาคผนวก
- FOUND()	แสดงผลลัพธ์จากการค้นหาเรคคอร์ดด้วยคำสั่ง FIND,LOCATE และ SEEK
- INKEY(),READKEY()	ให้ค่าเลขจำนวนเต็ม 0-255 ตามรหัสแกนคีย์ที่กด ส่วนคำสั่ง Readkey() จะเก็บค่าการกดคีย์สุดท้ายจากการป้อนข้อมูลขณะทำงานแบบฟลูสกรีน เพื่อใช้ในการนำไปตรวจสอบต่อไป
- LTRIM(นิพจน์ตัวอักษร)	เป็นฟังก์ชั่นการตัดอักษรว่างทางซ้ายมือของนิพจน์อักษรออก และมีฟังก์ชั่น RTRIM (นิพจน์ตัวอักษรซึ่งทำงานตรงข้ามกัน และถ้าต้องการตัดอักษรต่อออกทั้งซ้ายและขวาของนิพจน์สตริงก็ใช้คำสั่ง TRIM ข้างต้น

- MOD (นิพจน์ตัวเลขที่1,นิพจน์ตัวเลขที่2) หาเศษที่ได้จากการตั้งนิพจน์ตัวเลขที่ 1 แล้วหารด้วยนิพจน์ตัวเลขที่ 2

- ROW(),COL() ตามบรรทัดและคอลัมภ์ปัจจุบันที่เคอร์เซอร์แสดงอยู่

รูปโครงของโปรแกรม

โปรแกรมโครงสร้างเป็นหลักการทางคอมพิวเตอร์ เพื่อให้ผู้เขียนโปรแกรมเขียนโปรแกรมงานรูปแบบต่าง ๆ ได้ง่ายขึ้น เมื่อเขียนเสร็จแล้วก็สามารถตรวจสอบ ทำเอกสารประกอบโปรแกรมได้ง่ายขึ้น และสุดท้ายสามารถดัดแปลง,แก้ไข,เพิ่มเติมโปรแกรมในภายหลังได้ง่ายขึ้นด้วยโปรแกรมโครงสร้างจึงเป็นการรวมเอาข้อดีต่าง ๆ ของหลักการคอมพิวเตอร์ไว้

ภาษาดีเบสก็เป็นภาษาหนึ่งที่ไม่มีหมายเลขบรรทัด ไม่มีจุดอ้างอิงของโปรแกรมเหมือนภาษาเบสิก หรือฟอร์แทน ไม่มีคำเฉพาะ GOTO สำหรับกระโดดไปมาตามลำดับการทำงาน ภาษาดีเบสจึงจัดได้ว่าทันสมัยและมีประสิทธิภาพดี

โครงสร้างพื้นฐานที่ใช้ในการเขียนโปรแกรม มีสามรูปแบบคือ

1. โครงสร้างเรียงลำดับ
2. โครงสร้าง IF...ELSE...ENDIF
3. โครงสร้าง DO WHILE...ENDDO

และรูปแบบของโครงสร้างทั้งสามแบบนี้สามารถแทรกปนกัน หรือผสมกันอยู่ในโปรแกรมก็ได้ ดังมีรายละเอียดต่อไปนี้

โครงสร้างเรียงลำดับ

เป็นการจัดคำสั่งมาเรียงลำดับกัน เพื่อให้ดีเบสทำงานตามลำดับคำสั่งนั้น เช่น

```
CLEAR
@ 12,12 SAY "dBASE"
@ 12,18 SAY "is good."
RETURN
* END
```

เป็นการสั่งให้ดีเบสแสดงข้อความ "dBASE is good" โดยเริ่มจากบรรทัดที่ 12 คอลัมภ์ที่ 12 หลังจากที่เคลีย์จอภาพแล้ว และเมื่อเสร็จงานทั้ง 3 คำสั่ง จึงทำคำสั่ง Return เพื่อกลับจากโปรแกรม และคืนการควบคุมให้ผู้ใช้อีกครั้ง

โครงสร้าง IF...ELSE...ENDIF

เป็นการเลือกทำงาน ด้วยการตรวจสอบเงื่อนไข ถ้าเงื่อนไขแรกเป็นจริง จะทำคำสั่งระหว่าง IF กับ ELSE ทั้งหมด แต่ถ้าเงื่อนไขแรกเป็นเท็จ จะทำคำสั่งระหว่าง ELSE กับ ENDIF แทน

โครงสร้าง DO WHILE...ENDDO

จะต้องตั้งเงื่อนไขให้เหมาะสม ในการที่จะเริ่มทำคำสั่งทำซ้ำ กล่าวคือ เงื่อนไขของ DO WHILE จะต้องเป็นจริงจึงจะมีการเข้าลูปทำซ้ำนั้น และจะทำซ้ำจนกระทั่งเงื่อนไขเป็นเท็จ จึงออกจากลูปหรือ อาจจะออกจากลูปก่อนเงื่อนไขที่ทำให้ DO WHILE เป็นจริง เกิดขึ้นก็ได้ด้วยการใช้คำสั่ง EXIT นอกจากการทำงานตามลำดับคำสั่งระหว่าง DO WHILE กับ ENDDO แล้วดีเบสยังกำหนดคำสั่ง LOOP ขึ้นมาเพื่อสั่งให้มีจำนวนลูป จาก DO WHILE ใหม่ โดยไม่ต้องสนใจ คำสั่งที่เหลือ ซึ่งอยู่ระหว่างคำสั่ง LOOP กับ ENDDO นั้น

ส่วนการใช้โครงสร้างทั้งสองผสมกันนั้น ก็ใช้ได้เหมือนกับโปรแกรมสร้างทั่วไป ยกเว้น การใช้ IF ซ้อน IF เลยโดยไม่มีคำสั่งอื่นใดมาคั่นกลาง ไม่ได้เช่น

```
IF .T.
   IF.F.
```

เพราะเงื่อนไข IF ที่สองจะไม่ได้พิจารณาจากดีเบส ดังนั้น มันจะทำเงื่อนไขที่ 2 เสมอ ถ้าเงื่อนไขแรกเป็นจริง แม้เงื่อนไขที่สองจะเป็นเท็จก็ตาม

แต่การใช้ IF ซ้อน IF ในบางครั้งอาจดูยุ่งยากมากเกินไป ดีเบสจึงกำหนดโครงสร้าง DO CASE ขึ้นมาให้ใช้แทน เพื่อตรวจสอบเงื่อนไขของแต่ละกรณี (CASE) ถ้าเงื่อนไขใดเป็นจริงก็ทำตามเงื่อนไขนั้น โดยไม่ต้องสนใจเงื่อนไขที่เหลือ แต่ถ้าไม่มีเงื่อนไขใดเลยเป็นจริง ก็จะทำคำสั่งที่อยู่ระหว่าง OTHERWISE กับ ENDCASE แทน ดังจะแสดงได้ดังนี้

```
DO CASE
      CASE (เงื่อนไขที่1)
            คำสั่งที่จะทำถ้าเงื่อนไขที่ 1 เป็นจริง
      |          |          |
      |          |          |
CASE        <เงื่อนไขที่ n-1>
            คำสั่งที่จะทำถ้าเงื่อนไขที่ n-1 เป็นจริง
      OTHERWISE
            ถ้าเงื่อนไขใดๆข้างต้นไม่เป็นจริง จะทำคำสั่งนี้แทน
      ENDCASE
```

ซึ่งถ้าคุณจะเห็นรายละเอียดการใช้โครงสร้างในการโปรแกรม คำสั่งและฟังก์ชั่น ตลอดจนการเซ็ทสวิทช์ต่าง ๆ ของดีเบสได้จากบทถัดไป

บทที่ 3

การจัด dBASE III + ไปใช้งาน

ฐานข้อมูล หรือดาตาเบส (Data Base) เป็นที่รวบรวมข้อมูลหรือข่าวสารต่าง ๆ ที่เราเก็บรวบรวมไว้และถ้าเราเอาฐานข้อมูลนั้น มาจัดเก็บในระบบคอมพิวเตอร์ โดยการนักไว้ในหน่วยความจำ ก่อนที่จะจัดเก็บลงฟลอปปี้ดิสต์หรือดิสเกตต์ และฮาร์ดดิสต์อีกต่อหนึ่ง

ดังนั้นในแต่ละฐานข้อมูลที่ได้ จะอยู่ในรูปของไฟล์ข้อมูล (Data File) โดยแต่ละไฟล์ข้อมูลนั้นจะต้องมีชื่อไฟล์ (File Name) กำกับอยู่เพื่อความสะดวกต่อการเรียกใช้ แต่ละหน่วยของไฟล์ข้อมูลจะเป็นเรคคอร์ด (Record) อันประกอบไปด้วยข้อมูลหลาย ๆ แบบ ความที่กำหนดเป็นดาต้าฟิลด์ (Data Field) แล้วสิ่งที่กำหนดหรือจัดเก็บไว้ในแต่ละดาต้าฟิลด์ก็คือข้อมูลหรือดาต้า (Data) นั่นเอง

ลักษณะการจัดเก็บของฟิลด์ต่าง ๆ ของไฟล์นี้เราเรียกว่า โครงสร้างของไฟล์ (File Structure) อย่างเช่นในไฟล์ Store.dbf จะมีโครงสร้างดังนี้

ฟิลด์ที่	ชื่อฟิลด์	สิ่งที่เก็บ
1	Data	เก็บวันที่ที่มีการเข้าหรือออกของวัสดุก่อสร้าง
2	Desoript	รายชื่อ (Description) ของวัสดุก่อสร้าง

3	Delivery	จำนวนวัสดุก่อสร้างที่เข้าสต๊อกในแต่ละครั้งที่เข้า
4	Total_Deli[illegible]	จำนวนวัสดุก่อสร้างรวมที่เข้าสต๊อก (Total Delivery)
5	Do_out	จำนวนวัสดุก่อสร้างที่นำออกไปใช้งานแต่ละครั้ง
6	Total_Dout[illegible]	จำนวนวัสดุก่อสร้างโดยรวมที่เคยนำออกไปใช้
7	Tot_In_Sto[illegible]	จำนวนวัสดุที่ยังคงเหลืออยู่ในสต๊อกได้จากการนำเอา Total Delivery - Total Do Out
8	Change	บอกสภาวะที่มีของเข้าหรือของออกในแต่ละครั้ง
9	Dealer	บอกรายชื่อผู้ขายหรือเอเยนต์ที่นำวัสดุก่อสร้างมาส่ง
10	Per_Unit	ราคาวัสดุนั้น

จากที่กล่าวมาข้างต้น จะแสดงรูปแบบของฐานข้อมูลได้ง่าย ๆ ดังนี้

ดาต้าฟิลด์

รวมเป็น

หลายฟิลด์ ———→ เรคคอร์ด

รวมเป็น

หลายเรคคอร์ด ———→ ไฟล์

รวมเป็น

หลายไฟล์ ————————→ฐานข้อมูล

ดังนั้น หลักในการจัดเก็บฐานข้อมูลจึงต้อง

- จัดเก็บข้อมูลง่าย, ไม่ซ้ำซ้อน
- เรียกใช้ข้อมูล สะดวก, รวดเร็ว

นั่นคือ งานพื้นฐานที่ต้องทำกับฐานข้อมูล จึงเป็น

- การสร้างไฟล์ข้อมูล
- การเพิ่มเติมข้อมูล ลงในไฟล์
- การจัดเรียงข้อมูลในไฟล์ตามที่ต้องการ
- การค้นหาข้อมูลจากไฟล์
- การจัดทำรายงานจากไฟล์ที่มีอยู่
- การแก้ไขปรับปรุงข้อมูลในไฟล์
- การลบข้อมูลออกจากไฟล์
- การโอนย้ายข้อมูลจากไฟล์หนึ่งไปอีกไฟล์หนึ่ง
- การเชื่อมต่อไฟล์สองไฟล์ เพื่อนำข้อมูลมาแสดงผลร่วมกัน

บริษัท Goumagai Ci.,Ltd. เป็นบริษัทก่อสร้างของญี่ปุ่น ได้มาตั้งบริษัทสาขาในไทยควบคุมการก่อสร้างสภาสิ่งแวดล้อมแห่งชาติทีปทุมธานี มีความต้องการจะนำคอมพิวเตอร์มาจัดเก็บข้อมูลควบคุมของในสต๊อกวัสดุก่อสร้าง โดยมีรายละเอียดของแบบข้อมูล ดังแสดงไว้ในหน้า๖.

นอกจากจะต้องบอกปริมาณวัสดุที่เข้ามาแต่ละครั้ง วัสดุที่เหลืออยู่ และวัสดุที่เบิกไปใช้งานแต่ละครั้งแล้วยังต้องบอกปริมาตรของไม้ (Timber) และน้ำหนักของเหล็กข้ออ้อยแต่ละครั้งที่มีการสั่งเข้าได้เพื่อนำไปเปรียบเทียบกับข้อมูลในใบสั่งของ เช็คความถูกต้องก่อนรับ และต้องบอกรายละเอียดของผู้ขายเช่นชื่อผู้ขายที่อยู่ เบอร์โทรศัพท์ และราคาขายต่อหน่วยของวัสดุนั้นได้ด้วย อีกทั้งถ้าต้องการทราบว่าวัสดุก่อสร้างชนิดไหน ผู้ขายรายใดเป็นคนขายก็ต้องให้ข้อมูลได้ เพื่อความสะดวกในการสั่งเพิ่ม

จากงานที่กำหนดให้ สามารถวิเคราะห์และออกแบบโปรแกรมได้ดังนี้

ได้ใช้ dBASE III + สร้างไฟล์ข้อมูล 4 ไฟล์ ดังนี้

1. Listing.dbf

จะเก็บรายละเอียดของวัสดุก่อสร้างทั้งหมด ที่เคยสั่งเข้ามาเพื่อทำประวัติเป็นสำคัญ ดังมีรายละเอียดโครงสร้างของไฟล์ดังนี้

Structure tor database : B : store.dbf

Number of data records : 83

Date of last update : 18/10/90

Field	Field Name	Type	Width	Dec
1	DATE	Date	8	วันที่ของเข้าออกหรือทั้งเข้าและออก
2	DESCRIPT	Character	18	ชื่อของวัสดุก่อสร้าง
3	DELIVERY	Numeric	8	จำนวนของเข้า
4	TOTAL-DELI	Numeric	6	จำนวนของเข้ารวมทุกครั้ง
5	DO-OUT	Numeric	6	จำนวนของออกไปใช้
6	TOTAL-DOUT	Numeric	6	จำนวนของออกไปใช้รวมทั้งหมด
7	TOL-IN-STO	Numeric	6	จำนวนของที่เหลืออยู่ในสต๊อก
8	CHANGE	Logical	1	บอกสถานะของการแก้ไข
9	DEALER	Character	30	รายชื่อผู้ขาย
10	PER-UNIT	Numeric	9	ราคาขายของวัสดุก่อสร้างนั้น
** Total **			97	

4. Dealer.dbf

เก็บรายละเอียดของผู้ขาย เพื่อการติดต่อ โดยจะรับข้อมูลจากทั้งโปรแกรม Make.prg และ Listing.prg หรือ Quick ก็ได้

Structure of datobase : B : dealer.dbf

Number of data record : 77

Date of last update : 10/10/90

Field	Ficld Name	Type	Width	Dec
1	DEALER	Character	30	
2	ADDRESS	Character	50	
3	PHONE	Character	8	

** Total **

โดยในโปรแกรมจะมีการช่วยในการป้อนผู้ขาย (Deater) คนเดิมที่ขายวัสดุก่อสร้างนั้นให้กับเรา แต่ถ้าเปลี่ยนแปลง ผู้ขาย(Deater)จะจัดเอาชื่อผู้ขายมาค้นหาข้อมูลในไฟล์นี้ และถ้ามีจะแสดงที่อยู่และเบอร์โทรศัพท์ให้ แต่ถ้าไม่เจอก็จะขอที่อยู่และเบอร์โทรศัพท์เองโดยอัตโนมัติ

นอกจากนี้ database file ทั้งสามแล้วยังเตรียม Index file ของแต่ละ database file ไว้ด้วยเพื่อการเรียกใช้ในโปรแกรม ดังนี้

Datobase file	Index file	Field Neme	จุดประสงค์
Store.dbf	Makindex.ndx	Descript	เป็นดัชนีของวัสดุก่อสร้างให้มีการเรียงข้อมูลตามอักษรตัวหน้าเพื่อความสะดวกต่อการนำไปใช้ในโปรแกรม

Datobase file	Index file	Field Name	จุดประสงค์
Listing.dbf	Desindex.ndx	Descript	เป็นดัชนีของวัสดุก่อสร้างที่เก็บประวัติไว้เพื่อความสะดวกในการทำโปรแกรม Listing.prg
Randt.dbf	Rindext.ndx	Descript	เป็นดัชนีของ Rebar และ Timber เพื่อนำมาแสดงผลร่วมกันกับ Store.dbf หรือทำการ Set relation กับ Listing.dbf ในโปรแกรม Listing.prg
Dealer.dbf	Purchase.ndx	Dealer	เป็นดัชนีของผู้ขายที่จะทำประวัติไว้เพื่อความสะดวกในการติดต่อครั้งต่อไปและจะค้นหาด้วยคำสั่ง Find ได้รวดเร็วกว่าการค้นด้วย Locate กับไฟล์ที่ไม่มีดัชนี

Listing.dbf Dauble-L.ndx Description,Dealer

นำเอาชื่อวัสดุก่อสร้างและผู้ขายมาทำการค้าว่าผู้ขายคนไหนขายอะไรให้เราบ้างในโปรแกรม Sellman.prg

อธิบายการทำงานโปรแกรมในโครงงานนี้

3.1 โปรแกรม Listing.Prg

จะเป็นการนำเอาข้อมูลวัสดุก่อสร้าง จัดบันทึกในไฟล์ประวัติ Listing.dbf มาแสดงผลตามชื่อของวัสดุนั้น โดยเริ่มต้น จะมีการขอป้อนชื่อวัสดุก่อสร้างที่ต้องการทราบรายละเอียด ดังภาพ 1 จากนั้นจะนำชื่อที่ได้ไปค้นหาในไฟล์ Listing.dbf ถ้าไม่เจอ จะขอให้ป้อนชื่อวัสดุก่อสร้างใหม่ หลังจากที่บอก "No.found." แล้ว และถ้าเป็นไม้แบบ (Timber) หรือเหล็กข้ออ้อย (Rebar) จะแสดงรูปแบบของชื่อเพื่อช่วยให้ผู้ใช้งาน สามารถป้อนชื่อได้สะดวกขึ้นอีกด้วย

แต่ถ้าพบ จะเริ่มต้นแสดงผล โดยทำการตีกรอบแบบฟอร์ม ด้วยโปรแกรมย่อย Listform.Prg ตัวโปรแกรมนี้จะแบ่งออกเป็น 2 ส่วน ตามวัสดุก่อสร้าง กล่าวคือ จากการที่ได้ตรวจสอบดูแล้วว่า ถ้าชื่อที่ป้อนเป็นไม้แบบ (Timber) หรือเหล็กข้ออ้อย (Rebar) จะเซ็ทตัวแปร Fact ให้มีค่าเป็นจริง ก่อนกระโดดจากโปรแกรมหลักมาทำโปรแกรมย่อยนี้ ดังนั้น จาก

โปรแกรม Listform.Prg จึงแสดงค่าปริมาตร หรือน้ำหนักที่คำนวนได้ ("C A L. Vol_Weight") กับปริมาตรหรือน้ำหนักจริงตามที่สั่ง ("R E A L Vol_Weight") แต่ถ้าเป็นวัสดุสร้างชนิดอื่นแล้ว เราจะไม่สนใจปริมาตรกับน้ำหนักของมัน แบบฟอร์มที่ได้จึงแสดงเพียง

- DATE
- DELIRY
- TOTAL DELIRY
- DO OUT
- TOTAL DO OUT
- TOTAL INSTOC
- DEALER

เท่านั้น และจากข้างต้น จะสังเกตุเห็นว่า จำนวนคอลัมภ์ของจอภาพมีจำกัดเพียง 80 คอลัมภ์เท่านั้น "DELIRY" จึงแสดงได้ไม่พอ จาก "DELIVERY" หรือจำนวนของเข้า ดังนั้นจึงแสดงเพียง 6 ตัวอักษรของมัน ตามขนาดข้อมูลของฟิลด์ DELIVERY ส่วน TOTAL DELIRY กับ INSTOC ก็ด้วยเหตุผลเดียวกัน และสุดท้ายของโปรแกรมนี้ จะแสดงคีย์ต่าง ๆ ที่โปรแกรมไว้ใช้งาน ในตอนล่างของจอภาพ และเนื่องจากพื้นที่บริเวณนี้ ลบข้อความแสดงคีย์ต่าง ๆ ที่มีใช้ แล้วจึงเขียนข้อมูลอื่นลงแทน เช่นกรณีที่ต้องการจะทราบที่อยู่ของผู้ขายเป็นต้น ดังนั้น จึงต้องให้มีโปรแกรม Help_L.Prg สำหรับเขียนข้อความแสดงคีย์ต่างๆ ที่ถูกลบไปชั่วคราว

จากนั้น จึงทำการเขียนข้อมูลของสิ่งของที่ต้องการลงบนแบบฟอร์มที่ได้ ด้วยการวนลูป

```
DO   WHILE  ROW < 20 .AND.&C2 =  NAMEL  .AND..
NOT.EOF()
```

จะเห็นว่าเงื่อนไขของการวนลูป มีหลายกรณีที่ต้องพิจารณา ดังนี้

- บรรทัดแรกที่เขียน คือบรรทัดที่ 7 จะเขียนบรรทัดละเรคคอร์ดจากบนลงล่าง จนถึงบรรทัดสุดท้ายของแบบฟอร์ม คือบรรทัดที่ 19 จึงหยุดแม้ว่ายังมีเรคคอร์ดอื่น ของสิ่งที่เราต้องการทราบข้อมูลอีกก็ตาม เพราะการแสดงผลนี้สามารถแสดงผลได้เพียงครั้งละ 13 เรคคอร์ดต่อหน้าจอหนึ่ง เท่านั้น

- &C2 = NAMEL โดยที่
C2 ="DESCRIPT" ดังนั้นการใช้ &C2 จะหมายถึงฟิลด์ DESCRIPT ของเรคคอร์ดปัจจุบันที่กำลังแอ็คทีฟอยู่ ส่วน NAMEL จะหมายถึงชื่อของวัสดุก่อสร้างที่ต้องการทราบรายละเอียด โดยถ้าตามเงื่อนไขนี้ จะหมายถึงให้แสดงเฉพาะเรคคอร์ดที่มีชื่อวัสดุก่อสร้างในฟิลด์ DESCRIPT เป็นชื่อตามตัวแปร NAMEL ของวัสดุก่อสร้างตามต้องการ

- แต่จากสองเงื่อนไขข้างต้น จะต้องจำกัดด้วยว่า ถ้าจบไฟล์หรือ EOF() เป็นจริงแล้วให้หยุด ไม่เช่นนั้น ดีเบสจะทำการวนรอบให้ไม่หยุด

และจากการที่ฟอร์มเรามี 2 แบบหลัก คือแบบที่เป็นไม้แบบ (Timber) หรือเหล็กข้ออ้อย (Rebar) กับเป็นสิ่งอื่นที่ไม่ใช่ 2 สิ่งนี้ และแต่ละแบบยังมีฟอร์มที่แยกกันอีก ตามลักษณะของที่เข้า,ออก ว่าจะเป็นแบบ

- Flow มีทั้งของเข้าและของออก
- IN มีเฉพาะของเข้า โดยไม่สนใจของออก
- OUT สนใจเฉพาะของออก โดยไม่สนใจของเข้า

ดังนั้น เราจึงต้องกำหนดคอลัมภ์และบรรทัด ที่จะให้เขียนข้อมูล ที่ต้องการตามฟอร์มดังกล่าวข้างต้นด้วย

อนึ่ง ในการจัดทำฐานข้อมูลที่ดีนั้น การเก็บข้อมูลจะต้องไม่ซ้ำกัน กล่าวคือ จากไฟล์ Listing.dbf ควรจะมีการเก็บปริมาตรของไม้แบบ (Timber) หรือน้ำหนักของเหล็กข้ออ้อย (Rebar) ทั้งค่าที่ได้จากการคำนวนและค่าจริงของพวกมัน เพิ่มอีก 2 ฟิลด์แต่ถ้าทำเช่นนี้ จะเป็นการเพิ่มเนื้อที่บนดิสต์มากเกินไป เพราะวัสดุก่อสร้างชนิดอื่น นอกจาก 2 ชนิดนี้ ไม่จำเป็นต้องใช้ฟิลด์ 2 ฟิลด์นี้ สำหรับจัดเก็บข้อมูล ดังนั้นเพื่อเป็นการแก้ปัญหานี้ เราจึงแยกเก็บปริมาตรของไม้แบบและน้ำหนักของเหล็กข้ออ้อยทั้งค่าที่ได้จากการคำนวนและค่าจริง ในอีกไฟล์นึง คือ RandT.dbf แล้วจึงค่อยเลือกมันมาแสดงผลอีกที

เมื่อเขียนข้อมูล จบในแต่ละหน้า จะมีการตรวจสอบดูว่า ยังมีเรคคอร์ดอื่น ๆ อีกไหม ที่มีชื่อตามที่เราต้องการ ถ้ามีก็ให้แสดงลูกศรบอกที่มุมล่างขวาของแบบฟอร์ม จากนั้นจึงเซ็ทค่าตัวแปรบางตัวที่สำคัญ ๆ

แล้วจึงมาเข้าลูปการรับฟังก์ชั่นคีย์ ต่างๆเพื่อเลือกการทำงาน

```
DO WHILE.T.
```

การทำงานของลูปนี้ ก่อนอื่น เก็บค่าฟังก์ชั่นคีย์ที่ใช้งานครั้งหลังสุดก่อน ในตัวแปร Last_key เพื่อใช้เป็นพารามิเตอร์ในการส่งให้โปรแกรม DBHelp.BIN (จะกล่าวถึงในบทหลัง) เลือกอ่านไฟล์สำหรับแสดงหน้าที่การทำงานของฟังก์ชั่นคีย์นั้นส่วนการรับฟังก์ชั่นคีย์นั้น จะใช้การวนลูป

```
DO WHILE Inkey = 0
   Inkey = INKEY()
ENDDO
```

แล้วนำค่าที่ได้ในตัวแปร Inkey จากการกดฟังก์ชั่นคีย์ ไปตรวจสอบ ดังนี้

```
DO CASE
   CASE Inkey = 28
```

จะเป็นการรับคีย์ F1 เพื่อนำคีย์ก่อนหน้านี้ ไปแสดงโปรแกรมช่วยเหลือ ด้วยการตรวจสอบ do case กับตัวแปร Last-Key เช่นถ้ามีการกดฟังก์ชั่นคีย์เลื่อนหน้า

```
case Last_Key = 3. OR. Last_Key = 18
     FILENAME = "B:\HELP3L 18.TXT"
```

และทำการโหลด โปรแกรม DBHelp.BIN ลงหน่วยความจำ ตามที่ดีเบสกำหนด

```
LOAD DBHelp
```

จากนั้นจึงเรียกใช้ โปรแกรมช่วยเหลือนี้ด้วย

```
CALL DBHelp WITH FileName
```

โปรแกรมช่วยเหลือ จะเก็บข้อมูลเดิม บนจอภาพ ลงหน่วยความจำในโค้ดเซคเมนท์ (Code Segment) แล้วจึงลบจอภาพนั้น เพื่อนำข้อมูลในไฟล์อธิบายฟังก์ชั่นคีย์ (เช่นกรณีนี้ จะเป็นไฟล์ Help 3L 18.TXT) มาแสดง

ผลบนจอภาพแทนและรอให้ผู้ใช้อ่านจนจบ ก็กดคีย์ "Press any key." อะไรก็ได้ โปรแกรมช่วยเหลือ จะนำเอาข้อมูลเดิมของดีเบส ที่ได้เซฟไว้ตั้งแต่ต้น มาแสดงบนจอภาพอีกครั้ง จากนั้นจึงกลับมาผังดีเบส ดีเบสก็จะลบโปรแกรมช่วยเหลือนั้นเสีย

```
RELEASE MODULE DBHelp
```

จึงวนลูปกลับไปรับการกดคีย์ใหม่ และถ้ามีการกดคีย์ Cntr-PgDn ก็จะเข้าเงื่อนไข

```
CASE Inkey = 30.AND. Deli-Dout < >"OUT"
```

เงื่อนไขหลัง Deli-Dout < > "OUT" จะใช้สำหรับกันไม่ให้แสดงข้อมูลของผู้ขาย ถ้าเลือกฟอร์มแบบสนใจเฉพาะของออกไม่เช่นนั้น จะทำให้ผู้ใช้สับสนได้

แต่ถ้าเป็นฟอร์มอื่นทั้ง "Flow" หรือ "In" จะสนใจของเข้าด้วย จะนำเอาชื่อผู้ขายวัสดุก่อสร้าง ที่เรคคอร์ดปัจจุบันขณะนั้นมาแสดงบนด้านล่าง ของแบบฟอร์ม แล้วรอการป้อนชื่อใหม่ ถ้าไม่ใช่คนนี้ แต่ถ้าผู้ใช้ต้องการดูข้อมูลของคนนี้ ก็กดรีเทรินได้เลย จะแสดงชื่อผู้ขาย ที่อยู่ เบอร์โทรศัพท์และราคาขายต่อหน่วย จากนั้น จะถามว่า ผู้ใช้ต้องการดูข้อมูลผู้ขายคนอื่นอีกไหม ถ้าต้องการก็จะเริ่มต้นทำใหม่ เพื่อแสดงผู้ขายคนอื่นอีกตามต้องการ แต่ถ้าไม่ต้องการดูผู้ขายคนอื่นอีกแล้วจะคืนข้อความแสดงฟังก์ชั่นคีย์ตอนล่างของจอภาพ ตามเดิม แล้วจึงวนลูปรับการกดฟังก์ชั่นคีย์ใหม่

แล้วถ้าคุณต้องการเลื่อนจอภาพ (คล้ายกับจอภาพเป็นแผ่นกระดาษ) จากหน้าปัจจุบัน ไปยังหน้าถัดไป เพื่อให้โปรแกรมแสดงข้อมูล ของหน้าถัดไปนั้น

```
CASE Inkey = 3.AND.&C2 = NAMEL
```

คุณก็กดฟังก์ชั่นคีย์ PgDn ถ้าหน้าปัจจุบันที่อยู่ ไม่ใช่หน้าสุดท้าย เงื่อนไข &C2 = NAMEL จะเป็นจริง จะทำการเลื่อนหน้าให้ด้วยการลบจอภาพเก่า แล้วสร้างแบบฟอร์ม จิงสั่งให้ดีเบส กระโดดออกจากลูปรับคีย์ แล้วออกไปวนลูปเขียนข้อมูลของเรคคอร์ดใหม่ แต่ถ้าอยู่ในหน้าสุดท้ายของวัสดุก่อสร้างแล้วย่อมเลื่อนไปยังหน้าถัดไปไม่ได้แน่ จะมีสัญญานบี๊บเตือนแทน

เมื่อเลื่อนไปผังหน้าถัดไปได้ ก็ต้องเลื่อนมายังหน้าก่อนได้เช่นกัน

```
CASE Inkey = 18 AND. Page>1
```

หลักการก็จะเหมือนกับ การเลื่อนไปผังหน้าถัดไป กล่าวคือ จะกำหนดตัวหน้า

```
Page = Page-2
```

เพราะเลขหน้า จะถูกเพิ่มค่าอัตโนมัติอีกหนึ่ง ในโปรแกรม Listform.Prg ดังนั้น จิงต้องลบเลขหน้าเดิมด้วย 2 และทำการเลื่อนเรคคอร์ด กลับ (Row+6) เพื่อไปผังตำแหน่งเริ่มต้น ของเรคคอร์ดแรก บนหน้าก่อนนั้น แล้วจิงออกจากลูปรับการกดฟังก์ชั่นคีย์ ไปฝังลูปเขียนข้อมูล ลูปเขียนข้อมูล จะทำการเขียนข้อมูลลงบนจอภาพ จนจบหน้าจอก็จะกลับมาฝังลูปรับการกดฟังก์ชั่นคีย์อีกครั้ง

จากการทำงานสองฟังก์ชั่นคีย์นี้ จะเห็นว่ามันเสียเวลามาก ถ้าข้อมูลมีหลายเรคคอร์ด แล้วต้องการเลื่อน จากหน้าแรกไปยังหน้าสุดท้ายที่ละหน้า ดังนั้น เพื่อให้สามารถ กระโดดจากหน้าใดก็ได้ ไปแสดงผลยังหน้าสุดท้าย ก็จะสามารถใช้ฟังก์ชั่นคีย์

```
CASE Inkey = 6
```

สำหรับเลื่อนไปแสดงหน้าสุดท้ายได้ โดยงานนี้ จะมี 2 วิธีคือ ใช้คำสั่งโดยตรงของดีเบสเองคือ

```
COUNT FOR &C2 = NAMEL To Num
```

จะเป็นการใช้คำสั่ง COUNT นับเรคคอร์ดที่มีชื่อวัสดุก่อสร้าง ตามที่เราต้องการ เก็บในตัวแปร Num แต่วิธีนี้จะเสียเวลามาก เพราะการทำงานจะเริ่มนับเรคคอร์ดตั้งแต่ ต้นไฟล์ จนจบไฟล์ ดังนั้น เพื่อไม่ให้เสียเวลาขนาดนั้น เราจึงนับเพียงเรคคอร์ดของวัสดุก่อสร้างที่ต้องการทราบ ที่อยู่ถัดจากหน้าปัจจุบัน ด้วย

```
DO WHILE Descript = NomeL
   SKIP
   NUM = NUM + 1
ENDDO
```

แล้วนำจำนวนเรคคอร์ดที่ได้บวกกับเรคคอร์ดของวัสดุก่อสร้าเดียวกัน ที่แสดงไว้แล้วในหน้าก่อน ด้วยคำสั่ง

```
SKIP-(NUM-13 * (Total-Page-1))
```

จึงเก็บหมายเลขของเรคคอร์ดนั้นไว้ ในตัวแปร Position และเซ็ทเลขหรัดให้ถูกต้อง

```
Position = RECNO()
Page = Total_Page-1
```

จึงเคลียร์จอภาพ ตีกรอบ และออกจากลูปรับฟังก์ชั่นคีย์ เพื่อทำการเขียนข้อมูลในเรคคอร์ดที่ต้องการ ลงบนแบบฟอร์มตามต้องการ

แต่ถ้าเป็นหน้าสุดท้ายของเรคคอร์ดนั้นแล้ว จะส่งสัญญาณบี๊บบอกแทน

ส่วนในการกระโดดกลับหน้าแรก ของวัสดุก่อสร้างนั้น จะใช้หมายเลขเรคคอร์ดของเรคคอร์ดแรกสุด ของหน้าแรกที่เก็บในตัวแปร First_Rec มาเซ็ทตัวแปร Position และให้ตัวแปร Page = 0 และทำซ้ำคล้ายกับ การกระโดดไปหน้าสุดท้ายแต่ทั้งนั้น ทั้งนี้ ต้องไม่มีหน้าปัจจุบันเป็นหน้าแรกสุด ตามเงื่อนไข

CASE Inkey = 1 .AND. Page > 1

และถ้าต้องการป้อนข้อมูลเพิ่มเติม ก็สามารถทำได้ด้วยการกด ริเทรินคีย์

CASE Inkey = 13 .AND. Page = Total_Page

ถ้าหน้าปัจจุบัน ที่คุณอยู่ถือหน้าสุดท้าย เงื่อนไข Page = Total_Page จะเป็นจริง เพราะโปรแกรมจะไม่ยินยอมให้คุณป้อนข้อมูลที่หน้าอื่น เพื่อป้องกันการแก้ไขข้อมูลกับเรคคอร์ดเก่าที่บันทึกเป็นประวัติไว้ จากนั้นจะถาม

"Do you want to enter material again?"

ถ้าคุณตอบตกลง ("Y") หรือกดรีเทริน คุณก็สามารถป้อนเรคคอร์ดใหม่ ได้ด้วยการให้โปรแกรมเหล็ก Listing.Prg รันโปรแกรมย่อย Enter.Prg

ตัวโปรแกรมย่อย Enter.Prg มีรายละเอียดดังต่อไปนี้

เริ่มต้น จะมีการกระโดดมายังเรคคอร์ดท้ายสุด ด้วยคำสั่ง

GOTO RECCOUNT()

แล้วเลื่อนถัดมาอีก 1 เรคคอร์ด ด้วยคำสั่ง

SKIP

พร้อมทั้งเซ็ทบรรทัด ให้เป็นตามความเป็นจริง

ROW = ROW+1

แล้วจึงขอเพิ่มเติมแรคคอร์ดใหม่ ด้วยคำสั่ง

APPEND BLANK

จากนั้นจึงเซ็ทฟิลด์ต่าง ๆ แล้วจึงตรวจสอบดูว่า วัสดุก่อสร้างเป็นไม้แบบหรือเหล็กข้ออ้อยหรือไม่ ถ้าไม่ใช่ จะให้ป้อนข้อมูลเฉพาะวันที่,จำนวนของเข้า, จำนวนของออก และชื่อผู้ขาย พร้อมทั้ง ราคาขายต่อหน่วยด้วย แต่ถ้าไม่เช่นนั้น หลังจากที่ป้อนจำนวนของเข้าแล้ว จำมีการคำนวนหาปริมาตรของไม้แบบกับหน้าหนักของเหล็กข้ออ้อย พร้อมกับรอรับการป้อนค่าจริงด้วย ส่วนการทำ

งานของโปรแกรม Enter.Prg นี้ จะอธิบายอีกครั้งนึง ตอนโปรแกรม Quick.Prg เพราะโปรแกรมทั้งสอง ทำงานคล้ายคลึงกัน

และเมื่อป้อนวัสดุก่อสร้างนั้นเสร็จแล้ว จะมีการถามคำถามเดิมซ้ำอีก เพื่อให้แน่ใจว่าไม่มีของที่ต้องการป้อนอีกแล้ว จึงตอบ "N" ก็จะถาม

"Are you want to leave this Listing ?"

ถ้าต้องการจบการ Listing กับวัสดุก่อสร้างชนิดนี้ แต่ถ้าไม่ก็จะวนลูปกับไปรับการกดฟังก์ชั่นคีย์ใหม่

มาถึงตรงนี้ จะสังเกตุเห็นว่า การออกจากการลิสติ้งวัสดุก่อสร้างนี้ มีวิธีนี้แบบนึงแต่ต้องเป็นไปตามเงื่อนไขว่าคุณต้องอยู่ในหน้าสุดท้ายของวัสดุก่อสร้างนั้น ดังนั้น เพื่อให้สามารถออกจากลิสติ้งได้อีกทาง คือออกโดยตรง ณ. จุดใดของการทำงานก็ได้ จึงมีฟังก์ชั่นคีย์ Cntr-End

```
CASE Inkey = 23
```

เพื่อกระโดดออกจากลูปรับฟังก์ชั่นคีย์ มาอยู่ในลูปนอก และตรวจสอบเงื่อนไข

```
IF (Inkey = 3 .OR. Inkey = 18 .OR. Inkey = 1;
  .OR. Inkey = 6 .OR. Inkey = 22) .AND. (&C2 = NAMEL)
   LOOP
ELSE
   EXIT
ENDIF
ENDDO
```

เงื่อนไขจะเป็นเท็จ จึงกระโดดออกจากลูปนอก ก็จะเจอเงื่อนไข

```
IF New_Enter > 0
```

หมายถึงมีการป้อนวัสดุก่อสร้างนั้นเพิ่มเติม จะทำการย้ายข้อมูล วัสดุก่อสร้าง

เข้าหลังสุด ไปอัปเดท (Undate) ข้อมูลวัสดุก่อสร้างนี้ที่ไฟล์ Store.dbf ด้วยการใช้ตัวแปร Tem_1,...Tem_10 ช่วยเก็บข้อมูลฟิลด์ต่าง ๆ ชั่วคราว ก่อนนำไปแทนข้อมูลเก่าของไฟล์ Store.dbf ด้วยคำสั่ง

REPLACE Date WITH Tem_1

เป็นต้น อนึ่งถ้าข้อมูลของวัสดุก่อสร้างในไฟล์ Store.dbf ได้ถูกลบไปจากการทำโปรแกรม ของผู้ใช้ ในไฟล์ Store.dbf จะแสดงข้อความ

"New record, create it on UPDATE BROWSE."

ก่อนทำการย้ายข้อมูลอีกด้วย

และสุดท้ายก่อนจบการทำงาน จะถาม

"Are you want to Listing another material?"

ถ้าตอบ "Y" ก็จะให้เลือกแบบของการเข้า,ออกใหม่

"Select type IN,OUT or FLOW <I,O,F> ?"

แต่ถ้าตอบไม่ต้องการก็จบการทำงาน และกลับไปยัง Main Menu เหมือนตอนเริ่มต้นอีกครั้ง

สรุปการทำงานของฟังก์ชั่นคีย์

- F1 ขอคำอธิบายฟังก์ชั่นคีย์
- Cntr-PgDn ขอรายละเอียดผู้ขาย พร้อมราคาขายต่อหน่วย
- PgUp เลื่อนหน้าขึ้น
- PgDn เลื่อนหรับลง
- End กระโดดข้ามหน้า ไปยังหน้าสุดท้าย
- Home กระโดดข้ามหน้า กลับหน้าแรกสุด
- Enter ป้อนวัสดุก่อสร้างที่มีการเข้า,ออกเพิ่มเติม
- Cntr-End จบการทำงาน

สรุป โปรแกรมย่อยอื่นๆ ที่เรียกใช้

- ListForm.Prg สร้างแบบฟอร์ม
- Enter.Prg ป้อนวัสดุก่อสร้างเพิ่มเติม
- Exam.Prg คำนวนปริมาตรไม้แบบ (Timber)
  ข้ออ้อย (Rebar)
- Help_L.Prg แสดงข้อความฟังก์ชั่นคีย์ที่มีให้เลือกใช้
- Err.Prg แก้ปัญหาเรื่องชื่อพื้นที่ใช้งาน

สรุป โครงสร้างของโปรแกรม

```
DO  WHILE (ลูปหลักสำหรับการลิสติ้งวัสดุก่อสร้างชนิดอื่นซ้ำอีก)

    DO  WHILE
        (ลูปรับชื่อวัสดุก่อสร้าง ที่ต้องการลิสติ้ง)
    ENDDO

    DO  WHILE   (ลูปการแสดงเรคคอร์ดของวัสดุก่อสร้าง
                 ตามแบบที่เลือกและรับฟังก์ชั่นคีย์ พร้อม
                 ทำงานตามนั้น)

     DO   WHILE ROW < 20 .AND. &C2 = NAME1  .AND..
NOT. ENDOFFILE
  <ลูปเขียนข้อมูลของวัสดุก่อสร้างลงในแบบฟอร์มแต่ละหน้า>
ENDDO
DO WHILE Inkey = 23 <ลูปการทำงานตามฟังก์ชั่นคีย์>
  DO WHILE          <ลูปปรับการกดฟังก์ชั่นคีย์>
```

```
            Inkey = INKEY ()
         ENDDO
         DO CASE   <ตรวจสอบเงื่อนไขเพื่อทำงานตามฟังก์ชั่นคีย์>
            CASE Inkey = 28 < "F1" ขอคำอธิบาย (help)>

CASE Inkey = 30 .AND. Deli_Dout<>"OUT" <ขอรายละเอียดผู้ขาย>
   |  |                                  |   |
CASE Inkey = 23                         <Cntr_End จบการลิสติ้ง>
ENDCASE
IF          ELSE
ENDIF   < ตรวจสอบเงื่อนไขการวนลูป รับการกดฟังก์ชั่นคีย์>
ENDDO
IF (Inkey = 3 .OR. Inkey = 18 .OR. Inkey = 1 ;
.OR. Inkey = 6 .OR. Inkey = 22) .AND. &C2 = NAME1
<ตรวจสอบเงื่อนไขการวนลูปซ้ำ เพื่อทำการเรียนหน้าใหม่ก่อน รับการกด
ฟังก์ชั่นคีย์อีก>
     ENDIF
ENDDO
IF New_Enter > 0
     <จำเป็นต้องมีการ Update ข้อมูลใน Store.dbf ด้วย>
ENDIF
     IF
     <ถาม "Are you want to Listing another material?",
     "Select type IN,OUT or FLOW <I,O,F>? >
     ENDIF
```

ENDDO <จบการทำงานของลิสติ้ง>

RETURN <กลับเมนูหลัก>

3.2 โปรแกรม Make.Prg

จุดประสงค์ของโปรแกรมนี้ เขียนขึ้นเพื่อ ให้สามารถแสดงผลแบบ Browse ของดีเบสได้ ซึ่งคำสั่ง Browse ของดีเบสนั้นยังมีศักยภาพต่ำอยู่ และเพื่อเพิ่มประสิทธิภาพการทำงานของโปรแกรม จึงให้มันมีการทำงานเลียนแบบหน้ากระดาษจริง กล่าวคือ

- ต้องมีการเลื่อนขึ้น,ลงได้ในแต่ละหน้า
- ต้องมีการเลื่อนบรรทัด ที่จะทำงานได้
- สามารถเลื่อนข้ามหน้าได้
- สามารถช่วยป้อนรายละเอียดบางส่วนแทนผู้ใช้ได้
- สามารถค้นหารายละเอียดบางอย่างที่ต้องการ ที่อยู่ผู้ขาย, ประมาตรของไม้แบบ (Timber) หรือน้ำหนักเหล็กข้ออ้อย (Rebar)
- สามารถเรียงหน้าได้โดยอัตโนมัติ ตามลำดับอักษรของชื่อวัสดุก่อสร้าง
- และสามารถลบเรคคอร์ดได้ตามต้องการ

ดังนั้น โปรแกรม Make.Prg จึงมีรายละเอียด ดังต่อไปนี้

เริ่มต้นต้องทำการกำหนดตัวแปรต่าง ๆ ไว้ใช้งานก่อน ที่จะเข้าเงื่อนไข

```
DO WHILE Total_Page >= Page
```

หมายความว่า ถ้าผิดจากเงื่อนไขนี้ หน้ารวม (Total_Page) น้อยกว่าเลขหน้าปัจจุบัน (Page) แสดงว่าโปรแกรมทำงานผิดพลาด ก็ให้จบการทำงาน ต่อมาคือเงื่อนไข

```
IF  Key = 3
```

จะเป็นการเพิ่มเลขหน้าขึ้นอีก 1 ถ้ามีการกดคีย์ PgDn ก่อนที่จะมีการสร้างแบบฟอร์ม ด้วยการทำโปรแกรมย่อย Form.Prg ตามคำสั่ง

```
DO Form
```

จากนั้นจึงทำการเขียนข้อมูล ลงบนแบบฟอร์ม ตามแบบของการเข้า,ออก ที่เลือกจากเมนูหลักก่อนแล้ว ด้วยลูป

```
DO WHILE Row < = 20 .AND..NOT. EOF<>
```

เช่น ถ้าเลือกสนใจเฉพาะของเข้า ("IN") ก็จะแสดงจำนวนของเข้า,ของเข้ารวม และจำนวนที่เหลืออยู่ในสต๊อกเท่านั้น ส่วนจำนวนของออกและจำนวนของออกรวม จะใช้ "-" แทน เป็นต้น

และในการเขียนข้อมูลของวัสดุก่อสร้างลงบนแบบฟอร์ม จะมีการเลื่อนลงบรรทัดต่อไป พร้อมกับเลื่อนเรคคอร์ดถัดไป ด้วยคำสั่ง

```
Row = Row+1
SKIP
```

จากนั้น จำทำการคำนวนหน้าที่ทั้งหมดอีกครั้งนึง ด้วยคำสั่ง

```
IF MOD(RECCOUNT<>,14) = 0.00
   Total_Page = RECCOUNT<>/14
ELSE
   Total_Page = INT(RECCOUNT<>/14)+1
ENDIF
```

ส่วนเงื่อนไขต่าง ๆ ต่อจากนี้ จะกล่าวถึงอีกครั้งในหัวข้อการลบเรคคอร์ด ด้วยการกด Del คีย์

จากนั้นจะทำการเขียนเรคคอร์ดที่กำลังแอคทีฟ ด้วยรีเวิร์ส (Reverse) เพื่อทำเป็นแถบสว่าง (highliaht) บนบรรทัดที่กำหนดในตัวแปร Row แล้วจึงเข้าลูปรับการกดฟังก์ชั่นคีย์

```
DO WHILE Key <> 23
```

โดยการทำงานของการเลื่อนแถบสว่างนั้น ถ้ากดคีย์เลื่อนขึ้น (Key = 5) หรือเลื่อนลง (Key = 24) จะต้องมีการคืนการแสดงผลบนจอภาพของเรคคอร์ดเก่า ให้เป็นแบบปรกติ (Normal display) เสียก่อน ตามเงื่อนไข

```
IF Key = 5 .OR. Key = 24
```

จากนั้น จึงค่าฟังก์ชั่นคีย์ที่ได้ไปตรวจสอบ ดังนี้

```
DO CASE
CASE Key = 22
```

จะเป็นการจัดเอาคีย์ก่อนหน้านี้ ที่เก็บในตัวแปร Last_Key ไปเปรียบเทียบ เพื่อส่งพารามิเตอร์เป็นชื่อไฟล์ ให้โปรแกรมช่วยเหลือ (help) อ่านมาแสดงผล คล้ายกับโปรแกรม Listing.Prg

และถ้าต้องการทราบที่อยู่ หรือรายละเอียดของผู้ขาย ก็สามารถทำได้ ด้วยการกดคีย์ Cntr กับเลื่อนขวา

```
CASE Key = 2
```

จะมีการตรวจสอบว่า ถ้าชื่อผู้ขายเป็น "GOUMAGAI CO.,LTD."จะแสดงข้อความ

"Use in GOUMAGAI CO.,LTD."

แต่ถ้าผู้ขายเป็นคนอื่น ไม่ใช่ใช้เองในบริษัท (จากการที่เลือกฟอร์มเป็น "OUT" จะไม่สนใจผู้ขาย ดังนั้นจึงแทนฟิลด์ผู้ขาย (Dealer) ของเรคคอร์ดนั้นด้วย "GOUMAGAI CO.,LTD.") ก็จะทำโปรแกรม MK_2_30.Prg ซึ่งมีรายละเอียดดังต่อไปนี้

จะนำเอาชื่อผู้ขายในฟิลด์ Dealer ของเรคคอร์ดที่กำลังแอ็คทีฟไปค้นในไฟล์ Dealer.dbf แล้วจึงนำที่อยู่และเบอร์ศัพท์ที่ได้มาแสดงผลพร้อมทั้งบอกราคาต่อหน่วย ซึ่งเก็บในฟิลด์ Per_Unit ในไฟล์ Store.dbf มาแสดงผลร่วมด้วย แต่จะจบการทำงานเพียงเท่านี้ จะไม่ขอถามว่าผู้ใช้ต้องการป้อนผู้ขายคนอื่นเพื่อดูรายละเอียดอีกไหม เพราะการทำงานจะสนใจแต่เพียงเรคคอร์ดที่กำลังอยู่บนแถบสว่างเท่านั้น และจะรอจนกว่าผู้ใช้กด Cntr-เลื่อนซ้าย ซึ่งเป็นฟังก์ชั่นคีย์ ซึ่งตรงกันข้ามกันกับ ก็จะกลับไปทำงานในโปรแกรม Make.Prg ตามเดิม

```
CASE  Key = 24
```

สำหรับให้ผู้ใช้เลื่อนแถบสว่างลงเพื่อเลื่อนไปแอคทีฟยังเรคคอร์ดถัดไป แต่การเลื่อนลูกศรลงนี้ ก็มีเงื่อนไขให้ตรวจสอบอีกดังนี้

```
DO  CASE
     CASE (ROW = 21) .AND..NOT.(EOF<>)
          ROW =  7
          SKIP = 14
```

สำหรับตรวจดูว่า การเลื่อนแถบสว่างลงล่าง ถึงบรรทัดล่างสุดของแบบฟอร์มแล้วหรือยัง ถ้าถึงแล้วเงื่อนไขแรกเป็นจริง และถ้าหน้าปัจจุบันที่แสดงไม่ใช่หน้าสุดท้าย จะเลื่อนกลับไปแสดงแถบสว่างกับบรรทัดบนสุดของหน้านั้น

แต่ถ้าเป็นการเลื่อนแถบสว่าง ไปบนบรรทัดสุดท้าย ของหน้าสุดท้าย เงื่อนไขแรกของ

CASE EOF<> .AND. Deli_Dout<>"OUT"

จะเป็นจริง และถ้าฟอร์มที่เลือกไม่ใช่ "OUT" หมายถึง ของที่เข้าเป็นวัสดุก่อสร้างใหม่แน่ จึงทำโปรแกรมย่อย AddNew.Prg ซึ่งมีรายละเอียด คล้ายกับโปรแกรม Enter.Prg กล่าวคือ จะรับวันที่,ชื่อวัสดุก่อสร้าง,จำนวนเข้าและออก รวมทั้งรายละเอียดของผู้ขาย และราคาขาย และถ้าเป็นไม้แบบหรือเหล็กข้ออ้อย จะคำนวนปริมาตรและน้ำหนักให้ด้วย แต่ยังมีกรณีต้องห้ามอีก 2 กรณีคือ

- ชื่อวัสดุก่อสร้างต้องถูกป้อน ไม่ให้ปล่อยว่างไว้ ไม่เช่นนั้นจะเตือนด้วยข้อความ

"How are you?"

เพราะไม่เช่นนั้นจะผิดความจริงไปมาก

- และการป้อนของเข้าที่บรรทัดล่างสุดวิธีนี้ จะอนุญาติเฉพาะวัสดุก่อสร้างชนิดใหม่เท่านั้น ถ้าเป็นของเก่าจะไม่ยอมเพราะจะซ้ำซ้อนกัน และเมื่อมีการป้อนจบ จะค้นหาชื่อวัสดุก่อสร้างนี้ ในไฟล์ Store.dbf ถ้าเจอแสดงว่าป้อนผิด ก็จะแสดงข้อความ

"It 's old"

และเก็บข้อมูลของเรคคอร์ดเก่าไว้ แล้วจึงไปเพิ่มค่ากับเรคคอร์ดใหม่ เพื่อให้ได้ค่าที่ถูกต้อง จึงลบเรคคอร์ดเก่าเสีย ก็จะได้เรคคอร์ดของวัสดุก่อสร้างที่มีข้อมูลถูกต้องตามต้องการ

อนึ่ง สำหรับคีย์เลื่อนลงนี้ จะไม่อนุญาตให้ป้อนวัสดุก่อสร้าง ที่จะเข้าใหม่ ถ้าเป็นฟอร์แบบ "OUT" นั่นคือ จะแสดงข้อความ

"Material OUT,NOT permit."

แทน เพราะกรณีผิดความจริงไปมาก นั่นคือเงื่อนไขสำหรับการตรวจสอบจะเป็น

CASE EOF<> .AND. Deli_Dout = "OUT"

ส่วนการเลื่อนขึ้น จะเลื่อนทั้งแถบสว่างและเรคคอร์ดที่จะแอคทีฟ เมื่อกดฟังก์ชั่นคีย์ลูกศรเลื่อนขึ้น

CASE Key = 5

จะมีการตรวจสอบเงื่อนไข 3 กรณีคือ

- กรณีหมุนจากเรคคอร์ดบนสุดของหน้าสุดท้ายไปยังเรคคอร์ดสุดท้ายถ้าเป็นจริง จะทำคำสั่ง

```
Row = (Row+1)+(RECCOUNTC)-((Page-1)*14+1))
GOTO BOTTOM
```

- กรณีเลื่อนเรคคอร์ดและแถบสว่างขึ้นบนตามปรกติ

```
SKIP-1
```

- กรณีหมุน (pull down) จากบรรทัดบนสุด ลงบรรทัดล่างสุด ของหน้าอื่นซึ่งไม่ใช่หน้าสุดท้าย

```
Row = 20
SKIP = 13
```

เมื่อได้บรรทัดและเรคคอร์ดที่ถูกต้องแล้ว จึงแสดงผลมันด้วยรีเวิร์สดิดีโอ (Reverse Video) ก็จะได้การเลื่อนแถบสว่างขึ้นตามต้องการ

เมื่อเลื่อนแถบสว่างในแต่ละหน้าได้ ก็ต้องเลื่อนหน้าที่จะแสดงผลได้ ด้วยการกดฟังก์ชั่นคีย์ PgDn

CASE Key = 3

ถ้าเลื่อนหน้ามาจนหน้าสุดท้าย ก็หมดหน้าที่จะเลื่อนจะเตือนด้วยสัญญาณบีบ แต่ถ้ายังไม่ถึงหน้าสุดท้าย จะเลื่อนเรคคอร์ดให้สัมพันธ์กับบรรทัดปัจจุบันของแถบสว่าง ดังนี้

```
SKIP +(21-Row)
```

ส่วนการเลื่อนหน้าขึ้นนั้น ทำได้ด้วยการกดฟังก์ชั่นคีย์ PgUp และกรณีนี้

```
CASE  Key = 18 .AND. Page > 1
```

คล้ายกับกรณีเลื่อนหน้าลง จะต่างกันตรงการเลื่อนเรคคอร์ดแรก ของหน้าก่อน จะทำคำสั่ง

```
SKIP - (Row+7)
```

แทน

และถ้าต้องการกระโดดข้ามหน้า จากหน้าอื่นไปยังหน้าสุดท้าย เพื่อความรวดเร็ว สามารถกดฟังก์ชั่นคีย์ End เพื่อให้ทำตามเงื่อนไขนี้

```
CASE Key = 6 .AND. Page < Total_Page
```

จะกระโดดไปยังเรคคอร์ดสุดท้ายก่อน

```
GOTO  BOTTOM
```

แล้วจึงเลื่อนกลับ ด้วยคำสั่ง

```
SKIP -(RECCOUNT()-14*(Total_Page-1))
```

ด้วยเซ็ทหน้าให้ถูกต้องด้วยคำสั่ง

```
Page = Total_Page
```

ส่วนการกระโดดข้ามหน้าจากหน้าอื่น กลับหน้าแรกนั้น สามารถทำได้ตามกรณี

```
CASE Key = 1 .AND. Page <> 1
```

จะเลื่อนเรคคอร์ดไปยังเรคคอร์ดแรกสุด พร้อมกับเซ็ทเลขหน้าให้เป็นหนึ่ง ด้วยคำสั่ง

```
GOTO  TOP
Page = 1
```

และจากการที่คอลัมภ์แต่ละบรรทัด ถูกจำกัดไว้เพียง 80 ตัวอักษร จึงให้แสดงข้อมูลปริมาตรของไม้หรือน้ำหนักของเหล็กข้ออ้อยด้วยไม่ได้เหมือนการลิสติ้ง เพราะแต่ละบรรทัดได้มีการแสดงชื่อวัสดุก่อสร้างด้วย ดังนั้นจึงต้องเพิ่มฟังก์ชั่นคีย์ Cntr_PgDn

```
CASE  (Key =  30).AND.(UPPER(SUBSTR(&C2,1,5))
="REBAR" .OR.; UPPER(SUBSTR(&C2,1,6))="TIMBER")
```

เพื่อให้ผู้ใช้กดคีย์ หลังจากที่เลื่อนแถบสว่าง มาตรงกับไม้แบบหรือน้ำหนักของเหล็กข้ออ้อย ตามที่ต้องการทราบข้อมูลแล้ว จะมีทำโปรแกรมย่อย Mk_2_30.Prg ซึ่งตัวโปรแกรมย่อยนี้ มีรายละเอียดการทำงานดังต่อไปนี้

เริ่มต้น จะจำเรคคอร์ดนั้นเบอร์ของเรคคอร์ดปัจจุบัน ของไฟล์ Store.dbf ไว้ก่อนในตัวแปร Address จากนั้นจึงนำเอาชื่อวัสดุก่อสร้างที่ต้องการทราบข้อมูล เก็บในตัวแปร REandTIM ดังนี้

```
Address = RECNO()
STORE &C2 TO REandTIM
```

จากนั้นจึงไปเปิดไฟล์ Randt พร้อมกับใช้อินเด็กซ์จากไฟล์ RindexT กำกับ แล้วจึงโปรแกรมชื่อวัสดุก่อสร้างในตัวแปร REandTIM ให้อยู่ในเครื่องหมายคำพูด

```
REandTIM =""+RTRIM(REandTIM)+""
```

จะต้องใช้"" เท่านั้น เพราะขนาดของไม้บอกเป็นนิ้ว ("") กำกับอยู่ จากนั้นจึงใช้คำสั่งค้นหา

```
FIND  & REandTIM
```

เมื่อค้นเจอ จะเป็นเรคคอร์ดแรกสุดของไม้หรือเหล็กข้ออ้อย ที่มีการเข้า,ออกครั้งหลังสุด จะนำจำนวนด้วยลูป

```
DO  WHILE  Descript = REandTIM
SKIP
ABC = ABC+1
ENDDO
SKIP-1
```

การทำเช่นนี้ จะทำให้มีการเลื่อนเรคคอร์ดไปยังเรคคอร์ดของไม้หรือเหล็กข้ออ้อย ที่ได้มีการเข้าครั้งหลังสุด ทำให้เราสามารถแสดงข้อมูลจากปัจจุบันไปหาอดีตได้ตามต้องการ

สำหรับการให้แสดงผลนั้น จะมีการใช้ลูกศร↑,↕ หรือ↓ กำกับ โดย

- ↑จะหมายถึงเป็นข้อมูลล่าสุด จะต้องใช้ลูกศรเลื่อนขึ้น เพื่อดูข้อมูลที่อดีตกว่า
- ↕หมายถึง สามารถใช้ลูกศรเลื่อนขึ้นหรือลงก็ได้
- ↓ใช้ได้เฉพาะลูกศรเลื่อนลงเท่านั้น เป็นเรคคอร์ดที่มีการเข้าครั้งอดีตสุดแล้ว

ส่วนการทำโปรแกรม ทั้ง 3 กรณีจะใช้การเลื่อนขึ้นหรือลง จากเรคคอร์ดที่กำลังแอคทีฟอยู่เป็นหลัก แล้วจึงทำการเขียนข้อมูลที่ต้องการและลูกศรกำกับในตอนล่างของแบบฟอร์ม

อนึ่ง สำหรับวัสดุก่อสร้างนั้น ถ้ามีการเข้า,ออกเพียงครั้งเดียว ก็จะใช้สัญลักษณ์ " ■ " บอกแทน และเมื่อต้องการจะจบการขอดูข้อมูลปริมาตรของไม้แบบหรือน้ำหนักของเหล็กข้ออ้อย แล้วก็กด Cntr PgUp เพื่อกลับไปทำงานบนแถบสว่างตามเดิมและการจะให้กลับไปยังเรคคอร์ดเดิมของไฟล์ Store.dbf จะต้องทำคำสั่งนี้

```
USE Store INDEX MakIndex ALias MakeStore
GOTO  AddGress
```

ซึ่งเป็นการนำเอาหมายเลขเรคคอร์ด ซึ่งเซฟในตัวแปร Address มาใช้

และเมื่อใช้งานโปรแกรมไปนาน ๆ จะมีของเข้า,ออกเพิ่มเรคคอร์ดมากขึ้น ดังนั้น ถ้าต้องการจะลบเรคคอร์ดก็สามารถทำได้ด้วยการกด Del คีย์ และ Ins คีย์ ดังนี้

```
CASE  Key = 7 .OR. Key = 22
```

จะมีการเรียกใช้โปรแกรมย่อย MK_7_22.Prg โดยการทำงานของโปรแกรมนี้ จะเป็นดังนี้

- ถ้าเรคคอร์ด ไม่เคยเตรียมลบด้วยคำสั่ง Delete มาก่อน จะทำคำสั่ง Delete กับเรคคอร์ดนั้น เท่านั้น
- แต่ถ้าเรคคอร์ดนั้น เคยถูกเตรียมลบ ด้วยคำสั่ง Delete มาก่อนแล้ว จะทำการลบจริง
- และถ้าต้องการเรียกคืน จากเรคคอร์ดเตรียมลบก็สามารถทำได้ด้วยการกด Ins คีย์

เริ่มต้นนั้น โปรแกรมจะนับจำนวนเรคคอร์ดที่ได้ถูกเตรียมลบก่อน ด้วยคำสั่ง

```
COUNT  FOR  DELETE  TO  Del_No
```

และถ้ามีการเตรียมลบเรคคอร์ด มากกว่าหนึ่งเรคคอร์ด จะใช้คำสั่ง PACK ไม่ได้ เพราะคำสั่ง PACK ของดีเบสจะลบข้อมูลของทุกเรคคอร์ด ที่ได้ทำการเตรียมลบ จนหมด แต่เราต้องการให้มันลบบางเรคคอร์ดเท่านั้น ดังนั้น ก่อนอื่นต้องถามก่อน

"DO You want to pack All,Some,This record."

โดยถ้าตอบ "S" หรือบางเรคคอร์ด (Some Record) จะแสดงเรคคอร์ดทั้งหมดที่ถูกเตรียมลบให้เลือก สามารถใช้คีย์ลูกศรเลื่อนขึ้น เลื่อนลง เพื่อเลือกได้ ถ้าเรคคอร์ดไหน ไม่ต้องการให้ลบก็กด Ins คีย์ ก็จะเก็บชื่อของมันลงบนตัวแปร

```
Recall_No = Recall_No + 1
Buffer = "Buffer" + LTRIM(Recall_No))
STORE  &C2 TO  &Buffer
```

เหตุที่ต้องใช้ตัวแปร Buffer เช่นนี้ เพราะเราไม่ทราบว่าควรเผื่อด้วยตัวแปรสำหรับเก็บชื่อวัสดุก่อสร้างที่ไม่ต้องการให้ลบ กี่ชื่อ จากนั้นจึงคืนเรคคอร์ดด้วยคำสั่ง

```
RECALL
```

พร้อมกับแสดงข้อความ

"OK. That 's record no pack."

และทำเช่นนี้จนกระทั่งสั่ง Pack เรคคอร์ดที่เหลือด้วยการกดฟังก์ชั่นคีย์ Cntr-PgUp หรือไม่เหลือเรคคอร์ดเตรียมลบแล้ว

```
CASE Key = 31 .OR. Del_No = 0
```

แต่ถ้าเหลือเรคคอร์ดเตรียมลบ ก็ให้ Pack ตามเงื่อนไข

```
IF Del_No <> 0
   PACK
ENDIF
```

พร้อมทั้งแสดงข้อความบอกจำนวนเรคคอร์ดที่ถูกลบจริง

```
"OK. You pack " + STR(Del_No,2)+" records."
```

แล้วจึงคืนสถานะเตรียมลบ กับเรคคอร์ดที่กู้คืนไว้ ด้วยคำสั่ง

```
DO  WHILE Recall_No > 0
Buffer = "Buffer" + LTRIM(Recall_No))
DELETE FOR &C2 = &Buffer
Recall_No = Recall_No - 1
ENDDO
```

ส่วนถ้าเลือกการลบแบบ "T" หรือเฉพาะเรคคอร์ดนี้เท่านั้น ก็จะทำการคืนเรคคอร์ดอื่น ลบจริงด้วยคำสั่ง PACK แล้วคืนสภาวะเตรียมลบกับเรคคอร์ดเหล่านั้น ด้วยวิธีคล้ายกับข้างต้นจะต่างกันก็ตรงที่เลือกเรคคอร์ดไม่ได้เท่านั้น แต่ถ้าต้องการให้ลบทุกเรคคอร์ดที่ได้ทำเตรียมลบแบบ "A" ก็จะทำการลบจริงเลย

จากนั้น จึงคำนวนจำนวนหน้าทั้งหมด,บรรทัดที่จะแสดงแถบสว่างใหม่ โดยมีหลัก ดังนี้

- ถ้ากลับหน้าเดิมได้ ก็กลับหน้าเดิม
- ถ้ากลับบรรทัดเดิมได้ ก็กลับบรรทัดเดิม

แต่ถ้าไม่ได้ ก็ให้เลื่อนไปยังหน้าก่อน,หรือบรรทัดสุดท้ายของเรคคอร์ดที่เหลือ

อนึ่ง ถ้ามีเรคคอร์ดเตรียมลบเพียงเรคคอร์ดเดียว ก็ทำการลบจริงเลย แล้วค่อยคำนวนการกลับหน้า และบรรทัดของแถบสว่างใหม่ ส่วนการกู้สภาวะเดิมจากเตรียมลบนั้น จะรับการกดฟังก์ชั่นคีย์ Ins เพื่อให้ทำงานตามเงื่อนไขกรณี นี้

```
CASE Key = 22 .AND. DELETE()
```

จะทำคำสั่ง Recall ซึ่งคำสั่งนี้ใช้ได้กับเรคคอร์ดที่กำลังแอคทีฟอยู่เท่านั้น จึงไม่ต้องโปรแกรมเพิ่มเติม แต่จะแสดงข้อความ

```
"OK. RECALL"
```

แล้วเมื่อกลับมายังโปรแกรมหลัก จะมีการเขียนข้อมูลใหม่ เพื่อทับข้อมูลเดิมของเรคคอร์ดที่ถูกลบไปให้ด้วย และเพื่อให้ผู้ใช้สามารถป้อนของเข้า,ออกเพิ่มเติมได้ จึงกำหนดให้ฟังก์ชั่นคีย์ Enter สำหรับเรียกโปรแกรมย่อย Enter_P.Prg มาทำงาน การทำงานก็จะคล้ายกับ Enter_Prg ของลิสติ้ง แต่จะทำงานกับเรคคอร์ดปัจจุบัน ที่แสดงไว้บนจอภาพด้วยแถบสว่างเท่านั้น

และฟังก์ชั่นคีย์สุดท้าย Cntr_End สำหรับการจบการทำงาน โดยจะเป็นทางออกจากการทำ Browse อีกทีนึง คือถามตอนเลื่อนลูกศรลงสำหรับการเพิ่มวัสดุก่อสร้างใหม่ โดยก่อนจบการทำงานจะมีการถาม

```
"DO you want to BROWSE another TYPE ?"
```

ถ้าไม่ต้องการ ก็จบการทำงานเลย แต่ถ้าต้องการหรือตอบ "Y" จะให้เลือกแบบของเข้า,ออก

```
"Select TYPE<F,I,O>?"
```

เพื่อนำแบบที่เลือก ไปเซ็ทตัวแปร Deli_Dout ก่อนที่จะวนลูปการทำงานซ้ำใหม่ตั้งแต่ต้น

แต่ก่อนที่จะกลับไปยังเมนูหลักนั้น จะมีการตรวจสอบเงื่อนไข

IF New_Enter > 0

ถ้ามีการแก้ไขข้อมูลบนเรคคอร์ดเก่า หรือสร้างเรคคอร์ดใหม่ ตามวัสดุก่อสร้างที่เข้าใหม่ จะมีการโอนข้อมูลไปบันทึกเป็นประวัติในไฟล์ Listing.dbf ด้วย แล้วจึงกลับไปทำงานบนเมนูหลัก เพื่อให้ผู้ใช้เลือกใช้งานอื่นอีกต่อไป

สรุป การทำงานของฟังก์ชั่นคีย์

- F1 ขอคำอธิบายจากโปรแกรมช่วยเหลือ
- Cntr-PgDn ขอดูปริมาตรไม้แบบหรือน้ำหนักเหล็กข้ออ้อย ที่ได้บันทึกเป็นประวัติไว้ในไฟล์ Rand.dbf
- Up Arrow Key เลื่อนแถบสว่างขึ้น เพื่อทำงานกับเรคคอร์ดบน
- Dn Arrow Key เลื่อนแถบสว่างลง เพื่อทำงานกับเรคคอร์ดล่าง
- PgUp เลื่อนหน้าไปยังหน้าก่อน
- PgDn เลื่อนหน้าไปยังหน้าถัดไป
- Home ข้ามหน้ากลับไปยังหน้าแรกสุด
- End ข้ามหน้าไปยังหน้าท้ายสุด
- Del ลบเรคคอร์ด
- Ins กู้เรคคอร์ดคืน จากที่ทำเตรียมลบเรคคอร์ดไว้
- Cntr_Right Arrow ขอดูรายละเอียดผู้ขาย,ราคาขาย

- Enter	ขอป้อนวัสดุก่อสร้างเรคคอร์ดปัจจุบันที่กำลังแอคทีฟอยู่
- Cntr-End	จบการทำงาน

สรุป โปรแกรมย่อยต่างๆ ที่เรียกใช้

- Form.Prg	สร้างแบบฟอร์ม
- Mk_7_22.Prg	เตรียมลบ,ลบจริงหรือกู้คืน (Delete,Pack or Recall)
- MK_2_30.Prg	แสดงรายละเอียดของผู้ขาย กับ ปริมาตรของไม้แบบ (Timber) หรือน้ำหนักของเหล็กข้ออ้อย (Rebar)
- Enter_P.Prg	แก้ไขข้อมูลในเรคคอร์ดเก่า
- AddNew.Prg	สร้างเรคคอร์ดใหม่ สำหรับเก็บวัสดุก่อสร้างชนิดใหม่
- Help_S.Prg	เขียนฟังก์ชั่นคีย์ที่มีใช้
- Exam.Prg	คำนวนปริมาตรของไม้แบบ (Timber) หรือน้ำหนักของเหล็กข้ออ้อย (Rebar) ตามขนาดและจำนวนที่มีการเข้าของมัน
- Err.Prg	ช่วยแก้ไขปัญหาการกำหนดชื่อพื้นที่ใช้งาน

สรุป โครงสร้างโปรแกรม Make.Prg

```
DO    WHILE  Total_Page >= Page
        <ลูปนอกสุด สำหรับการทำ Browse แบบต่างๆ ซ้ำ>
DO  WHILE Row <=20.AND..NOT.EOF
        <ลูปเขียนข้อมูลของเรคคอร์ด ลงบนแบบฟอร์ม>
      ENDDO
        <เงื่อนไขสำหรับการแสดงผล หลังจากที่ลบเรคคอร์ดจริง>

DO WHILE Key <>23 <ลูป สำหรับการทำงานตามฟังก์ชั่นคีย์ที่เลือกใช้>
        IF Key = 5.OR. Key = 24
        <เงื่อนไขการแสดงผล เรคคอร์ดแบบปกติ ก่อนเลื่อนแถบสว่าง>
        ENDIF
        DO CASE
          CASE kEY = 22 <"F1"  ขอคำอธิบายจากโปรแกรมช่วยเหลือ>
          CASE Key = 24 <"Arrow Down"  เลื่อนแถบสว่างลง
                        ไปยังเรคคอร์ดถัดไป>
        ENDCASE
        IF DELETE ()
        ELSE <แสดงสภาวะการเตรียมผลลบเรคคอร์ด>
        ENDIF
      ENDDO
      IF
        <ถาม "Do you want to Browse another TYPE?",
              "Select type <F,I,O> ? เพื่อเลือกการทำงานต่อ>
```

```
      ENDIF
ENDDO
      IF  New_Enter > 0
      <ตรวจเงื่อนไขสำหรับย้ายข้อมูลที่มีการเปลี่ยนแปลง      ไปเก็บเป็น
       ประวัติในไฟล์ Listing.dbf>
      ENDIF
RETURN <กลับเมนูหลัก>
```

3.3 โปรแกรม Quick.Prg

จากโปรแกรม Listing.Prg และ Make.Prg ข้างต้นจะเห็นว่า ถ้าต้องการป้อนวัสดุก่อสร้างเพิ่มเติมจะเสียเวลามากกล่าวคือ

- Listing.Prg จะต้องเลื่อนหน้าไปอยู่หน้าสุดท้ายก่อนจึงจะเริ่มป้อนวัสดุก่อสร้างนั้นเพิ่มเติมได้ แต่ถ้าเป็นการป้อนวัสดุก่อสร้างชนิดอื่น จะต้องออกจากการลิสต์ตั้งวัสดุก่อสร้างนั้น แล้วจึงเลือกวัสดุก่อสร้างที่ต้องการ ตั้งแต่ต้น

- Make.Prg จะต้องเลื่อนแถบสว่างให้ตรงกับวัสดุก่อสร้างที่ต้องการ จึงจะแก้ไขเพิ่มเติมวัสดุก่อสร้างชนิดนั้นได้แต่ถ้าเป็นวัสดุก่อสร้างเข้าใหม่ ซึ่งไม่เคยถูกจัดเก็บในฐานข้อมูลนี้มาก่อน จะต้องเลื่อนแถบสว่างลงบรรทัดสุดท้าย จึงจะเพิ่มเติมเป็นเรคคอร์ดใหม่ในไฟล์ Store.dbf ซึ่งก็จะมีการโอนย้ายข้อมูลไปในไฟล์ Listing.dbf ให้ด้วย เราจะเห็นแล้วว่าการทำงานมันจะเสียเวลามาก ดังนั้นจึงโปรแกรมให้ Quick.Prg ทำหน้าที่นี้เพื่อความสะดวกรวดเร็ว

และจากจุดประสงค์หลักอันนี้เอง จึงทำให้โปรแกรม Quick.Prg ไม่มีฟังก์ชั่นคีย์ให้เรียกใช้เลย และการทำงานของมันจะเป็นดังนี้

เริ่มต้นจะต้องเลื่อนเรคคอร์ดไปยังท้ายไฟล์ (End of file) ของไฟล์ Listing.dbf ที่ใช้ ดังนี้

```
GOTO  BOTTOM
SKIP
APPEND BLANK
```

และจำหมายเลขของเรคคอร์ด (Record number) ไว้ เพื่อความสะดวกในการโอนย้ายข้อมูล

```
Position = RECNO()
```

จากนั้นจึงทำการเริ่มต้น รับข้อมูลวันที่ (Date) และชื่อวัสดุก่อสร้าง (Description) เพื่อนำชื่อวัสดุก่อสร้างไปค้นในไฟล์ Listing.Prg ด้วยการเก็บชื่อวัสดุก่อสร้างลงตัวแปร B แล้วเติมเครื่องหมายคำพูด แล้วจึงใช้คำสั่งค้นหาดังนี้

```
STORE Descript TO B
B = "" + B + ""
FIND &B
```

ถ้าเจอแสดงว่าเป็นวัสดุก่อสร้างที่เคยมีการจัดเก็บมาก่อนหน้านี้แล้ว จะต้องเลื่อนไปยังเรคคอร์ดที่เก็บข้อมูลการเข้า,ออกวัสดุก่อสร้างนั้นครั้งล่าสุด ดังนี้

```
SKIP
DO  WHILE  &B = Descript
    SKIP
ENDDO
SKIP-1
```

เพื่อนำข้อมูลของเรคคอร์ดนั้น ไปเก็บลงบนตัวแปร C...J และเก็บชื่อผู้ขาย สงบนตัวแปร SellMan สำหรับใช้เป็นข้อมูลเก่าเพื่อรวมกับข้อมูลที่เข้าใหม่ เป็นจำนวนของเข้าออกที่ถูกต้อง

[illegible] ไม่พบ ก็จะเซ็ทตัวแปรต่างๆ ให้เป็นศูนย์ และเซ็ทตัวแปร Sellman ให้เป็นชื่อว่าง เพราะหมายความว่าของเข้านี้ ไม่เคยมีการเข้า, ออกมาก่อน แล้วจึงสั่งกลับเรคคอร์ดเดิม

GOTO Position

จากนั้นจึงทำการเขียนข้อความตามแบบของการเข้า, ออกของวัสดุก่อสร้าง เช่น ถ้าฟอร์มเป็น "IN" จะแสดงข้อความให้ป้อนจำนวนของเข้า - "(Delivery)" จำนวนของเข้ารวม "Total Delivery" จำนวนของที่เหลืออยู่ในสต๊อก ("Total in stock:") โดยจะมีแถบสว่างแสดงข้อมูลเฉพาะที่ให้ป้อน ตัวแสงสว่างเองก็แสดงข้อมูลเก่าจากที่ค้นมาข้างต้น ส่วนข้อมูลที่ไม่ต้องการให้ป้อน จะแสดงด้วยอักษรปกติแทน และเมื่อผู้ใช้ป้อนข้อมูลจนครบหมดแล้ว จึงขอรับข้อมูลชื่อผู้ขาย ("Dealer:") ซึ่งก็เหมือนกับข้างต้น กล่าวคือ ถ้าผู้ขายเคยขายสินค้านั้นให้จะแสดงชื่อผู้ขายในแถบสว่าง แต่ถ้าเป็นวัสดุก่อสร้างตัวใหม่ จะเห็นเป็นเพียงแถบสว่าง

จากนั้นจะเริ่มทำการตรวจสอบว่าผู้ใช้ป้อนชื่อวัสดุก่อสร้างแล้วยัง เพราะวัสดุก่อสร้างต้องมีชื่อที่ถูกต้องจะทำการแก้ไขภายหลังไม่ได้ โปรแกรมไม่อนุญาต จะแสดงข้อความ

"How are you"

แล้วเริ่มกลับไปทำงานใหม่ตั้งแต่ต้น และต่อมาจะตรวจว่าต้องคำนวณหาปริมาตรหรือน้ำหนักไหม โดยเก็บชื่อวัสดุก่อสร้างลงตัวแปร Randt แล้ว

ทดสอบว่าเป็นไม้แบบ (Timber) หรือเหล็กข้ออ้อย(Reber) และแบบการเข้าออกที่เลือกต้องไม่ใช่ "OUT" รวมทั้งของเข้า (Delivery) ต้องไม่เป็นศูนย์ ตามเงื่อนไขนี้

```
IF (SUBSTR(Randt,1,6) = "REBAR" .OR. SUBSTR
(Randt,1,6) = TIMBER");
    .AND..NOT.(Deli-Dout = "OUT") .AND.
    (Delivery <>0)
```

จะคำนวณหาปริมาตรไม้แบบ หรือน้ำหนักของเหล็กข้ออ้อย ด้วยการเก็บชื่อของไม้แบบหรือเหล็กข้อมูลนั้นลงตัวแปร RandT อีกครั้ง (จริง ๆ แล้วขั้นตอนนี้ไม่จำเป็น) ซึ่งจากการที่เราประกาศให้ตัวแปร RandT เป็นตัวแปรแบบสาธารณะ (Public) ดังนั้นมันจะถูกเรียกใช้ได้ในโปรแกรมคำนวณหาปริมาตรไม้แบบและน้ำหนักของเหล็กข้ออ้อย Exam.Prg (ส่วนรายละเอียดของโปรแกรมนี้ จะอธิบายอีกครั้งภายหลัง) ดังนี้

```
STORE Descript TO RandT
DO Exam
```

แล้วนำค่าที่คำนวณได้มาแสดงผลบนจอ พร้อมทั้งรับค่าจริงจากผู้ใช้ เช่นถ้าเป็นไม้แบบ (Timber ขนาดหนา 1 นิ้ว,กว้าง 6 นิ้วและยาว 4 เมตร)

```
"Volume:Timber 1"*6"*4M. is xxxxx.xx m.^3
Enter:"
```

และจะมีการนำทั้งค่าจริงและค่าที่ได้จากการคำนวณด้วยโปรแกรม Exam.Prg นั้น ไปเก็บลงไฟล์ RandT.dbf เพื่อบันทึกเป็นประวัติด้วย แล้วจึงกลับเรคคอร์ดเดิม เพื่อเขียนข้อมูลในฟิลด์ Total-Deli,Total-Dout และ Tol-in-Sto ให้เรียบร้อย

แล้วจึงมาพิจารณาถึงผู้ขาย ถ้าผู้ขายได้ขายของให้เรา แสดงว่าฟอร์มการเข้า,ออกที่เลือกใช้ จะต้องไม่เป็น "OUT"

IF Deli-Dout<> "OUT"

แล้วจะแสดงชื่อผู้ขายคนเดิม ผู้ซึ่งขายวัสดุก่อสร้างนั้นให้กับเรามาก่อน (นำมาจากผู้ขายวัสดุก่อสร้างนั้น ที่เรคคอร์ดล่าสุดของที่เคยเข้า) เพื่อให้ผู้ใช้ทำการแก้ไข ซึ่งกรณีนี้ผู้ใช้อาจ

- ไม่แก้ไขเพิ่มเติม แต่กดรีเทร์นยอมรับผู้ขายนั้นเลย

IF Sellman = Dealer

จะเก็บชื่อผู้ขายที่แสดงอยู่ ลงในเรคคอร์ดนั้น ที่ฟิลด์ Dealer เลย แล้วจึงแสดงชื่อผู้ขายพร้อมทั้งรายละเอียด

- แก้ไขเพิ่มเติม จะนำชื่อผู้ขายไปค้นในไฟล์ผู้ขาย Dealer.dbf ดูว่าเป็นผู้ขายที่เคยจัดเก็บประวัติมาก่อนไหม

Sellman =""+UPPER(Sellman)+""

LOCP

แล้วถ้าเจอจะแสดงข้อมูลของผู้ขาย แต่ถ้าไม่เจอจะขอรายละเอียดของผู้ขายคนนั้น เพื่อนำไปจัดเก็บเป็นประวัติแทน

- แต่ถ้าไม่เจอตั้งแต่ต้น แสดงว่าเป็นผู้ขายคนใหม่ จะขอรายละเอียดทั้งชื่อผู้ขายและที่อยู่พร้อมเบอร์โทรศัพท์ โดยแยกเก็บเฉพาะชื่อผู้ขายลงในฟิลด์ Dealer ของไฟล์ Listing.dbf และนำเอารายละเอียดของผู้ขายไปจัดเก็บในไฟล์ Dealer.dbf จึงเสร็จขั้นตอน

แล้วจึงถามผู้ใช้ว่า

"Are you want to enter it again?"

ถ้าต้องการจะให้เลือกแบบของการเข้า,ออกใหม่

"Material is Delivery<D>,Do out<O> or All<A>?"

แล้วจึงเซ็ทตัวแปร Deli-Dout ตามแบบที่เลือก จึงเริ่มต้นทำใหม่ตั้งแต่ต้นอีกครั้งตามแบบที่เลือกการเข้า,ออกที่เลือกนั้น แต่ถ้าไม่ต้องการจะโอนย้ายข้อมูลจากไฟล์ Store.dbf ให้คล้ายกับโปรแกรม Listing.Drg แล้วจึงกลับเมนูหลัก

สรุปโครงสร้างของโปรแกรม Quick.Prg

DO WHILE <ลูปการรับชื่อวัสดุก่อสร้าง และข้อมูลอื่นๆของมัน เป็นลูปนอกสำหรับทำซ้ำ>

<รับชื่อวัสดุก่อนสร้าง>

Do CASE

<แสดงข้อมูลเก่า ของวัสดุก่อสร้างนั้น ถ้ามี>

ENDCASE

IF Deli-Dout <> "out"

<ขอข้อมูลผู้ขาย จากผู้ใช้>

ENDIF

IF Descrip = " "

<ผู้ใช้ไม่มีการป้อนชื่อวัสดุก่อสร้าง จะเตือน แล้วเริ่มต้นใหม่>

ENDIF

IF (SUBSTR(RandT,1,5) = "REBAR" .OR.(SUBSTR (RandT,1,6)="TIMBER");.AND..NOT. (Deli-Dout ="OUT") . AND. (Delivery<>o)

<คำนวณปริมาตรไม้แบบ(Timber)หรือน้ำหนักเหล็กข้ออ้อย (Rebar) พร้อมกับแสดงผล และรับค่าจริงของมัน แล้วนำข้อมูลไปจัดเก็บใน

ไฟล์RandT.dbf>

ENDIF

<เขียนข้อมูลต่างๆ ในฟิลด์จำนวนรวมของของเข้า,ออก และของที่เหลืออยู่ ในเรคคอร์ดนั้น>

IF Deli-Dout < > "OUT"

<ขอชื่อและข้อมูลของผู้ขาย>

ENDIF

<ถามการทำเพิ่มเติมข้อมูลวัสดุก่อสร้างอื่นอีก พร้อมให้เลือกฟอร์มการเข้า ออกใหม่>

ENDDO

<ทำการโอนย้ายข้อมูลที่เพิ่มเติมมานั้น จากไฟล์Listing.dbf ไปอัปเดทข้อมูลเดิม หรือสร้างเป็นเรคคอร์ดใหม่ในไฟล์ Store.dbf>

RETURN <กลับเมนูหลัก>

3.4 <u>โปรแกรม Exam.prg</u>

จุดประสงค์ของโปรแกรมนี้ มีไว้เพื่อหาปริมาตรของไม้แบบ (Timber) หรือน้ำหนักของเหล็กข้ออ้อย(Rebar) ไว้เปรียบเทียบกับค่าจริงของมันที่สั่งเข้า เพื่อตรวจเช็คความถูกต้อง โดยมีหลักการทำงานดังนี้

<u>คำนวณน้ำหนักของเหล็กข้ออ้อย(Rebar)</u>

การคำนวณหาน้ำหนักของเหล็กข้ออ้อยนั้น เริ่มต้นจะเก็บชื่อของมันที่ถูกเก็บในตัวแปร Namel ลงในตัวแปร RandT ดังนี้

STORE Namel TO RandT

แล้วจึงเก็บข้อมูลเปรียบเทียบลงในตัวแปร Rebar

Rebar = "01070605.....25103850"

ตัวแปรนี้ได้มาจาก การเก็บขนาดของเหล็กข้ออ้อยให้ได้ 4 ตัวแรก แล้วเก็บน้ำหนักต่อเส้นของมันให้ได้เป็นเลข 4 ตัวหลัง เช่น

"Rebar 10mm.x 7m." มีน้ำหนักต่อเส้น 6.5 kg. ก็เก็บ 01070605

และ "Rebar 25mm.x 10m." มีน้ำหนักต่อเส้น 38.5 kg. ก็เก็บ 25103850

เหตุผลก็เพราะ

แล้วนำเอาข้อมูลที่เก็บในตัวแปร RandT จะตัดออกทีละตัวจนครบ 25 ครั้ง(เหตุที่ต้องตัด 25 ครั้ง ทั้งที่เรากำหนดฟิลด์ชื่อวัสดุก่อสร้าง (Descript) เพียง 18 ตัวอักษร ก็เพื่อความสะดวกในการเพิ่มเติมข้อมูลหรือขยายฟิลด์ในอนาคต โดยไม่ต้องมาเสียเวลาแก้โปรแกรม) เพื่อคัดเอาเฉพาะข้อมูลที่เป็นตัวเลขเท่านั้นเก็บลงตัวแปร Data

```
Do While No < = 25
   L-Shift 1 = SUBSTR(RandT,No,1)
   L-Shift 2 = "'" + L-Shift 1+"'"
   IF ASC( L-Shift2 > = 48 .AND. ASC(
L-Shift2) < = 57
        Data = Data + L-Shift1
ENDIF
NO = NO + 1
ENDDO
```

แล้วจึงคัดข้อมูลที่ได้ในตัวแปร Data ให้เหลือเพียง 4 ตัว ด้วยคำสั่ง

คำนวณหาปริมาตรของไม้แบบ (Timber)

การคำนวณหาปริมาตรของไม้แบบ จะต่างจากการคำนวณหาน้ำหนักของเหล็กเส้น เพราะปริมาตรของไม้แบบสามารถคำนวณได้จากขนาดกว้างคูณยาวคูณหนาได้เลย ดังมีรายละเอียดการทำงานดังต่อไปนี้

ขนาดของไม้แบบ มี

"TIMBER 1.5' * 3' * 3.5m."

"TIMBER 1' * 6' * 4m. เป็นต้น

ดังนั้น การตัดไม้แบบออกมาทีละตัวอักษร จะต้องคำนึงถึง

- ตัวเลข รหัสเอสกี (0-9) เท่ากับ 48-57
- จุดทศนิยมของตัวเลข รหัสแอสกี้ (.) เท่ากับ 46
- เครื่องหมายคูณ รหัสแอสกี้ (*) เท่ากับ 42
- เครื่องหมายคำพูดแทนนิ้ว

จะเริ่มต้น อธิบายการทำงานตามโปรแกรมดังนี้

จองตัวแปร และเตรียมตัวแปรต่าง ๆ ดังนี้

```
Buffer = SPACE (5) สำหรับใช้ชี้ตัวแปร Data1,..
,Data3
Data1 = '0'
Data2 = '0'
Data3 = '0' เก็บขนาดของแต่ละด้าน
N     =  1  ตัวชี้ตัวแปร Data1,..Data3
Dot   =  0  ตัวบอกจุดทศนิยม
NO    =  1  บอกลำดับของชื่อวัสดุก่อสร้าง
Randt = '" '+ Randt '" ' เตรียมตัวแปร  Rand
```

ซึ่งเก็บชื่อวัสดุก่อสร้างไว้สำหรับตัดแล้วจึงเข้าลูปตัด 25 ตัวอักษร ไปเก็บในตัวแปร L_Shift1

```
DO WHILE NO < = 25
L_Shift1 = SUBSTR (& RandT, NO,1)
```

ส่วนเงื่อนไขนี้ อันที่จริงแล้วไม่จำเป็น

```
IF L_Shift1 = " ' "
   L_Shift2 = ' " ' + L_Shift1 + ' " '
ELSE
   L_Shift2 = " ' " + L_Shift1 + " ' "
ENDIF
```

เพราะอาจตั้งเงื่อนไขที่ดีกว่าได้เป็น

```
IF L_Shift = " ' " .OR, L_Shift1 = ' " '
   No = No + 1
   Loop
ENDIF
```

แต่แสดงไว้ให้เห็นถึง การแก้ปัญหาการใช้เครื่องหมายคำพูดที่เหมาะสม เพื่อให้ใช้กับคำสั่ง CHR ของดีเบสได้

จากนั้น จึงเทียบว่าค่าที่ได้จากการตัดเป็นตัวเลข หรือจุดทศนิยมที่ต้องการไหม

```
IF ASC(&L_Shift2)>48. AND. ASC(&L_Shift2)
<=57 . OR. ASC(& L_Shift2) = 46
   Dat = Dot
   IF ASC (& L_Shift2) = 46
      Dot = Dot + 1
```

```
            ENDIF
            IF Dot <=1
               Buffer = 'Data' + STR (N,1)
            ENDIF
         ENDIF
```

จะใช้ตัวแปร Dot ในการนับจุดทศนิยม โดยจะสังเกตุเห็นว่าด้านแต่ละด้านรวมทั้งหน่วยบอกขนาด (เช่น ม.) ของชื่อไม้แบบ จะมีทศนิยมได้ไม่เกิน 2 ตัว แต่ขนาดของด้านใด ๆ เป็นตัวเลข ต้องมีจุดทศนิยมตัวเดียวเสมอ ดังนั้น ถ้าตัวแปร Dot น้อยกว่าหรือเท่ากับหนึ่ง (ค่าเริ่มต้นของมันคือศูนย์) จะเก็บค่าที่ได้จากการตัดลงในตัวแปร Data1,...,Data3 ตามที่ตัวแปร N บอก โดยจะใช้ตัวแปร Buffer ชี้ตัวแปรเหล่านั้นอีกครั้งหนึ่ง

และจากชื่อของไม้แบบ จะเห็นว่าด้านแต่ละด้านจะมีเครื่องหมายคูณ (*) หรือจุลภาค(,)กั้นอยู่ ดังนั้นเราจึงใช้มันเป็นตัวบอกว่า ถึงเวลาแล้ว ที่ต้องเพิ่มค่า N เพื่อใช้เป็นตัวแปร Data2 หรือ Data3 เพื่อเก็บขนาดด้านอื่นต่อไป ดังนี้

```
    IF ASC( & L_Shift2) = 42. OR. ASC (& L_Shift2) = 44
       N = N+1
       Dot = 0
    ENDIF
```

พร้อมทั้งเซทตัวแปร Dot ให้เป็นศูนย์ ตรงนี้คงเห็นประโยชน์ของตัวแปร Dot แล้วว่า ถ้า "Timber 1.5' * 4' * 3.5 ม.
มันจะไม่เก็บจุดทศนิยมของหน่วยเมตร เข้าไปในตัวแปร Data3 ก็จะได้ขนาดความยาวของไม้แบบที่ถูกต้องตามต้องการ

เมื่อได้ขนาดแต่ละด้านของไม้แบบแล้ว ก็นำมาคำนวนหาปริมาตร ด้วยคำสั่ง

ปริมาตร (หน่วยลูกบาศก์เมตร) = ความหนา (หน่วยนิ้ว) x 0.0254
x ความกว้าง (หน่วยนิ้ว) x 0.0254
x ความยาว (หน่วยเมตร)

1 นิ้วเท่ากับ 0.0254 เมตร หรือ

ปริมาตร (ลูกบาศก์เมตร) = 0.0228 x ความหนา (นิ้ว) x
ความกว้าง(นิ้ว) x ความยาว(เมตร)

นั่นคือ

```
Volume = &Data1 * &Data2 * &Data3 * 0.0228
```

เพราะ Data1,..Data3 จะเก็บขนาดนิ้วไว้ในเครื่องหมายคำพูด หรือเป็นตัวอักษร ดังนั้น ถ้าจะใช้เป็นข้อมูลแบบตัวเลข ก็ต้องอ้างทางอ้อมโดยใช้เครื่องหมาย " & " แทน ก็จะได้ปริมาตรของเหล็กข้ออ้อย (Rebar) หน่วยลูกบาศก์เมตร ตามต้องการ

5. โปรแกรม Sellman.Prg

จากโปรแกรม Listing'Prg จะเห็นว่า ผู้ใช้สามารถทราบได้ว่า วัสดุก่อสร้างตัวนี้ ซื้อมาจากผู้ขายคนใดบ้าง แต่สำหรับการจะตอบคำถามที่ว่า ผู้ขายคนนี้ขายอะไรให้แก่เราบ้างนั้น มันใช้เพียงโปรแกรม Listing. Prg อย่างเดียว คงยังไม่มีประสิทธิภาพเพียงพอ ดังนั้น จึงโปรแกรม Sellman.Prg ให้สามารถบอกเราได้ว่า ผู้ขายสินค้าคนนี้ขายอะไรให้เราบ้าง โดยจะบอกราคาขายครั้งหลังสุด พร้อมที่อยู่ของผู้ขาย เพื่อความสะดวกในการติดต่อ สิ่งวัสดุก่อสร้างครั้งต่อไปให้ด้วย

การทำงานของโปรแกรม จะใช้ไฟล์ข้อมูล Listing.dbf ส่วนอินเด็กซ์ไฟล์จะใช้ Deslndex.ndx เหมือนโปรแกรม Listing.Prg ไม่ได้ เพราะอินเด็กซ์ไฟล์ Deslndex.ndx ทำอินเด็กซ์โดยใช้เพียงชื่อสินค้าเป็นตัวเรียงอินเด็กซ์เท่านั้น จึงต้องสร้างอินเด็กซ์ไฟล์ใหม่ชื่อ Double_L.ndx ซึ่งเรียงอินเด็กซ์โดยใช้ชื่อสินค้าเป็นหลัก แต่ถ้าวัตถุก่อสร้างนั้น มีผู้ขายให้หลายคนก็จะเรียงอินเด็กซ์ให้ตามลำดับชื่อผู้ขายด้วย นั่นคือ สร้างไฟล์ Double_L.ndx

```
INDEX ON Descript + Dealer TO Double_L
```

หลังจากที่ได้ไฟล์ Purchase.ndx ตามต้องการแล้ว เมื่อจะใช้ไฟล์ Listing.dbf ด้วยอินเด็กซ์ไฟล์ Double_L.ndx ในโปรแกรม Sellman.Prg นี้ ก็ต้องสั่งให้เรียงอินเด็กซ์ใหม่ ด้วยคำสั่ง

```
USE Listing Index Double_L
REINDEX
```

แต่ก่อนนี้จะสั่งให้เรียงอินเด็กซ์ใหม่นี้ จะต้องถามผู้ใช้ให้แน่ใจก่อนว่า ผู้ขายคนที่ต้องการทราบรายละเอียดนั้นเป็นใคร โดยให้ผู้ใช้ป้อนชื่อผู้ขาย แล้วนำชื่อผู้ขายคนนั้นไปค้นในไฟล์ Dealer.dbf เพื่อให้แน่ใจว่า ผู้ใช้ป้อนถูก

จากนั้น จึงทำการเรียงข้อมูลตามที่กล่าว และหาว่าผู้ขายคนนั้นขายวัสดุก่อสร้างให้เรามาแล้วกี่ครั้ง ด้วยคำสั่ง

```
Same_No = 0
COUNT FOR Dealer = & Deal_Name TO Same_No
```

ดังนั้น ตัวแปร Same_No จะบอกผู้ขายคนนั้น เคยขายสินค้าให้กี่ครั้งแล้ว ต่อมาจึงเซ็ทมันกับตัวแปร Same_Ref อีกทีนึง แล้วจึงสั่งกลับไปเริ่มต้นใหม่ที่เรคคอร์ดแรกสุด ตามที่อินเด็กซ์ไฟล์ Double_L.ndx ชี้

```
GOTO TOP
```

และเขียนแบบฟอร์มลงบนจอภาพ โดยมีตัวแปร Page_Ref บอกเลขหน้า จึงเข้าลูปการแสดงผลและรับการกดฟังก์ชั่นคีย์

```
DO WHILE Key < >23 .AND..NOT.EOF( )
```

ลูปของการแสดงผลจะอยู่ที่

```
DO  WHILE ROW <20.AND..NOT.EOF ( )  .AND.
Same_No>0
```

และแต่ละบรรทัด จะมีลูปหาเรคคอร์ดวัสดุก่อสร้างใด ๆ ที่ผู้ขายคนนั้นได้ขายให้ ดังนี้

```
DO WHILE Descript = Material .AND. Dealer
= Deal_Name ;
        .AND. Same_No >0
        Same_No = Same_No-1
        SKIP
ENDDO
```

ตรงนี้ จะสังเกตุเห็นว่า ทำไมเราต้องใช้อินเด็กซ์ทั้งวื่อวัสดุก่อสร้าง (Descript = Material) และชื่อผู้ขาย (Dealer = Deal_Name) เพราะไม่เช่นนั้นแล้ว การเลื่อน (SKIP) เรคคอร์ดเพื่อจุดประสงค์ดังกล่าวจะทำไม่ได้ และตรงนี้จะเห็นประโยชน์อีกอย่างหนึ่งของ Samw_No กล่าวคือ ถ้า Same_No = 0 แล้ว ก็ไม่ต้องเลื่อนเรคครอร์ดอีกแล้ว (หรือไม่ต้องรอให้ลบไฟล์ก่อนจึงหยุด) เพราะข้อมูลที่เหลือ จากเรคคอรีดที่เจอและทำให้ Same_No ลดลงเป็นศูนย์นั้น ไม่ได้ซื้อมาจากผู้ขายที่ต้องการทราบรายละเอียดนั่นเอง

แล้วจากที่เรามีโปรแกรมให้ฟังก์ชั่นคีย์ Pg Dn, Pg Up ในการเลื่อนหน้าได้นั้น จำเป็นต้องจำ

-Same_No หรือจำนวนวัสดุก่อสร้าง ที่ผู้ขายคนนั้นขายให้ ที่ยังเหลืออยู่จากบรรทัดแรกของหน้านั้น

-RECNO () หรือหมายเลขเรคคอร์ด ที่ผู้ขายคนนั้นขายสินค้าชนิดใด ที่เขียนในบรรทัดแรกของหน้านั้น

ดังนั้น จึงโปรแกรมเป็น

```
IF ROW = 13
     Same_Ref = 'Same_Ref' + STR (Page_Ref, 1)
     STORE Origin TO &Same_Ref
     Return_At = 'Return_At' + STR (Page_Ref,1)
     STORE RECNO () & Return_At
ENDIF
```

Same_Refn แทน เหตุที่ต้องกำหนด Same_Ref 1,...,n เก็บ Same_No และ Return_AGtT1,...,Return_Atn เก็บหมายเลขเรคคอร์ด ดังนี้ ก็เพราะว่าไม่สามารถคำนวนหน้าทั้งหมด จาก Same_No ได้นั่นเอง และถึงแม้ว่าจะคำนวนได้ ก็ไม่สามารถกำหนดตัวแปรเพื่อไว้ใช้พอตามจำนวนหน้านั้นได้ แล้วถ้าจะเผื่อเกิน ๆ ไว้ ก็จะกินหน่วยความจำมากเกินไป ดังนั้น จึงต้องใช้วิธีนี้แทน

และถ้าดูในโปรแกรมนี้ต่อไป จะพบคำสั่งที่น่าสนใจคือ

```
IF Same_No > 0 .AND..Not. EOF ()
   CONTINUE
   IF FOUND () .AND. Row = 20.AND. Same_No>0
```

คำสั่ง Continue นี้ เป็นคำสั่งควบคู่กันของ

```
LOCATE FOR Dealer = Deal_Name WHILE Same_No >0
```

หมายถึง เป็นการให้ดีเบสค้นหาเรคคอร์ด ที่มีชื่อผู้ขายเป็นคน ๆ เดียวกันกับที่ผู้ใช้ต้องการทราบรายละเอียด โดยที่จะค้นได้ก็ต่อเมื่อ จำนวนเรคคอร์ด

หรือครั้ง ที่ผู้ขายคนนั้นขายสินค้าให้เรายังมีอยู่เหลืออีกในไฟล์ นั่นคือ ตัวแปร Same_No มากกว่าศูนย์อยู่

หมายเหตุ คำสั่งนี้ จะเริ่มทำการค้นจากเรคคอร์ดปัจจุบัน ไปยังท้ายไฟล์ ตามที่อินเด็กซ์ไฟล์บอก และตามเงื่อนไขที่กำหนด ส่วนคำสั่ง FIND จะค้นตั้งแต่ต้นไฟลีเลย แต่ถ้าเจอเรคคอร์ดตามเงื่อนไขที่กำหนดแล้ว ทั้งสองคำสั่งนี้ก็จะหยุดเหมือนกัน ที่เรคคอรีดนั้น แต่คำสั่ง Locate สามารถใช้คำสั่ง Locate เอง หรือ Continue ค้นต่อจากเรคคอร์ดนั้นได้ แต่คำสั่ง FIND ทำไม่ได้

ส่วนเงื่อนไขหลังจากคำสั่ง Continue นั้น มีไว้สำหรับแสดงลูกศรขวากระพริบ บอกว่ามีหน้าถัดไปอีก

มาถึงการรับฟังก์ชั่นคีย์บ้าง

- ถ้าผู้ใช้กด PgUp จะเอา Same_No และหมายเลขเรคคอร์ดของหน้าก่อน ซึ่งเป็นข้อมูลที่เขียนบนบรรทัดแรกของหน้านั้นมาใช้ใหม่ ดังนั้น

```
Page_Ref = Page_Ref-1
                เซ็ทเลขหน้ากลับหน้าเดิม
Same_Ref = 'Same_Ref' + STR (Page_Ref,1)
                เซ็ท Same_Refให้ชี้ Same_Ref ของหน้าเดิม
STORE &Same_Ref TO Same_No
                นำค่า Same_No ของหน้าเดิม ที่ Same_Ref ชี้บอก
                มาเซ็ท Same_No
Return_At = 'Return_At' + ATR (Page_Ref,1)
                เซ็ทตัวแปรซึ่งจำหมายเลขเรคคอร์ด ให้ชี้กลับหน้าเดิม
```

GOTO & Return_At

เลื่อนเรคคอรีดไปที่เรคคอร์ดของบรรทัดบนสุดของหน้าเดิม

Page_Ref = Page_Ref -1

เลขหน้าทีจะมีการเพิ่มค่าให้อีก 1 เมื่อมีการเขียนแบบฟอร์ม ดังนั้น จึงต้องลดค่าเลขหน้าที่ตัวแปร Page_Ref ลดอีก 1

Loop กลับวนลูปใหม่

- ถ้าผู้ใช้กดคีย์ PgDn เลื่อนหน้าลง ก็วนลูปต่อไปได้เลย เพราะการโปรแกรมของเขา จะเอื้ออำนวยหรือเขียนโดยฝึกการเลื่อนลงเป็นหลัก เพราะโอกาศผู้ใช้จะเลือกคีย์นี้ย่อมมีมากกว่าคีย์ PgUp เนื่องดูข้าหน้าเดิม จริงไหม

- ถ้าผู้ใช้ได้ข้อมูลตามต้องการแล้วจะกด Cntr-End

อนึ่ง ถ้าค้นหาวัสดุก่อสร้างที่ผู้ขายที่ต้องการทราบรายละเอียดไม่พบ จะแสดงข้อความ

" NOW, not deal."

แต่กรณี จะไม่เกิดขึ้นเด็ดขาด ถ้าผู้ใช้ทำงานเฉพาะกับโปรแกรมของโครงงานนี้ แต่จะเกิดขึ้น ถ้าผู้ใช้ออกนอกโปรแกรมของโครงงานแล้วลบเรคคอร์ดด้วยคำสั่งของดีเบสเอง หลังจากที่แสดงข้อความนั้นแล้ว จะขอให้ผู้ใช้ป้อนชื่อผู้ขายใหม่ โดยการกลับไปวนลูปตั้งแต่ต้นนั่นเอง

สรุปโครงสร้างโปรแกรม Sellman.Prg

```
DO WHILE <ลูปนอก สำหรับทำซ้ำผู้ขายกันใหม่ที่ต้องการทราบรายละเอียด>
IF. NOT.FOUND ()
   <เงื่อนไขที่จะวนลูปซ้ำ ถ้าผู้ขายที่ป้อนไม่มีประวัติ หรือไฟล์ข้อมูลผู้ขาย
    beater.dbf>
ELSE
   <นับจำนวนเรคคอร์ด หรือครั้ง ที่ผู้ขายคนนั้น ได้เคยขายสินค้าให้พร้อม
    ทั้งเตรียมตัวแปรต่าง ๆ ไว้เรียกใช้งาน และตีแบบฟอร์ม>
   DO WHILE Key <> 23.AND..NOT.EOF ()

   IF Same_No >0
      DO WHILE Row <20.AND..NOT.EOF () Same_No>0
      <ลูปการเขียนวันที่,ชื่อสินค้า,ราคาขาย ที่ผู้ขายนั้นขายสินค้าให้
      ครั้งหลังสุด>
      ENDDO
      <เก็บจำนวนเรคคอร์ด ที่ผู้ขายสินค้าคนนั้น เคยขายสินค้าให้ เป็น
      จำนวนที่เหลืออยู่ในไฟล์ นับจากที่เขียนลงในบรรทัดแรกของแบบ
      ฟอร์มและหมายเลขเรคคอร์ดด้วย>
      <แสดงลูกศรเลื่อนหน้าได้ ถ้ายังไม่มีหน้าคัดจากหน้านั้นอีก>
      DO WHILE (Key = 0) .AND. (Same_No>=0)
         <รับการกดฟังก์ชั่นคีย์>
      ENDDO
      Do CASE
         <ทำงานตามฟังก์ชั่นคีย์ ที่กดโดยผู้ใช้>
```

```
        ENDCASE

      ELSE
        <ลบประวัติของซื้อจากผู้ขายคนนั้น ในไฟล์ Listing.dbf แล้วจะ
        แสดงข้อความ "NOW, not deal.">
      ENDIF
      <ถ้าลบประวัติแล้ว จะให้ป้อนชื่อผู้ขายใหม่ ด้วยการวนลูปซ้ำ>
       ENDDO
ENDIF
ENDDO
```

โปรแกรม Horizon.Prg.

โปรแกรมนี้ ใช้สำหรับแสดงเมนูหลัก หลักการทำงานสามารถสรุปได้ดังนี้

- เริ่มต้นต้องกำหนดข้อความที่ต้องการแสดงไว้ในเมนูหลัก โดยแยกเป็น
  - ส่วนหัวของเมนู ที่บรรทัดบนสุด จะแสดงหัวข้อหลัก ๆ ที่สามารถเลือกทำได้ ในตัวแปร Header1, .., Header6
  - ส่วนตัวเลือกของแต่ละหัวข้อหลัก ๆ ในตัวแปร Choice1,..,Choice6
- นำหัวข้อหลักไปเขียนบนจอภาพก่อน เป็นตัวอักษรปกติ
- การทำพูลดาวน์เมนู(Pull down menu)จะต้องจำทั้งบรรทัดและคอลัมภ์ ในตัวแปร Rw และ Cl ตามลำดับ
- และการแสดงหัวข้อกับตัวเลือก จะต้องแสดงจากซ้ายมือสุดก่อน

- แล้วถ้ากดคีย์เลื่อนขวาหรือซ้าย จะทำการเลื่อนหัวข้อและตัวเลือกไปตามนั้น โดยคำนวนคอลัมภ์ที่จะแสดงบนจอจากตัวแปร C1 แล้วเซ็ทบรรทัด Rw ให้เริ่มต้นที่บรรทัดของตัวเลือกด้วย แต่ทั้งนี้ทั้งนั้น ต้องลบของเดิมก่อน ถ้าหัวข้อเดิมนั้นมีตัวเลือกด้วยคำสั่ง

```
@ 6,C1 CLEAR TO 12,C1 + 20
RW = 8
```

และถ้าเลื่อนจนสุดแล้ว ก็ต้องเซ็ทให้วนกลับ

- แต่ถ้าเลื่อนขึ้นบนหรือลงล่าง ก็ทำคล้ายกัน

- และเมื่อผู้ใช้เลือกกดรีเทริน ค่าที่ได้จากฟังก์ชั่น ReadKey () เท่ากับ 15 ก็จะออกจากลูปนี้ เพื่อเซ็ทแบบที่เลือกในตัวแปร Deli_Dout ตามตัวแปร Choice ของโปรแกรม I ซึ่งตัดจากหัวข้อหรือ Header มาเรียกใช้โปรแกรมด้วยคำสั่ง Do ชื่อโปรแกรมนั้น ส่วนการออกจากโปรแกรมนี้ ก็เลือก Exit เพื่อทำคำสั่ง Exit และถ้าต้องการจบการทำดีเบสก็จะใช้คำสั่ง Quit ซึ่งคุณดูได้จากโปรแกรม

<u>หมายเหตุ</u> IIF เป็นฟังก์ชั่นตัวนิงของดีเบส มีการใช้งานดังนี้

```
C1 = IIF (C1>71, 6, C1)
```

หมายความว่า

- ตัวแปร C1 หรือคอลัมภ์ จะเท่ากับ 6 ถ้า C1 ขณะนั้นมีค่ามากกว่า 71

- ไม่เช่นนั้นแล้ว ก็ให้คงค่าตัวแปร C1 ไว้ตามเดิม

ซึ่งถ้าเขียนด้วย เงื่อนไข IF จะเป็น

```
IF C1 > 71
C1 = 6
```

```
            ELSE
               C1 = C1
            ENDIF
```

หรือเขียนเป็นคำสั่งง่าย ๆ ได้เป็น

```
            IF  C1 >71
               C1 = 6
            ENDIF
```

บทที่ 4

การนำข้อมูลดีเบสทรีพลัสมาแสดงผลด้วยกราฟ

ส่วนการสร้างกราฟ

โปรแกรมนี้จะทำการอ่านข้อมูลจากไฟล์ดีเบส เพื่อนำมาแสดงกราฟ โดยเริ่มต้น จะให้เลือกแบบของกราฟ ซึ่งมีอยู่ 5 แบบ ดังนี้

- กราฟเส้น สำหรับแสดงข้อมูลในฟิลด์ใด ๆ 2ฟิลด์ เพื่อเปรียบเทียบกัน เช่น เราอาจต้องการเปรียบเทียบดูจำนวนของเข้า (Deliery) กับจำนวนของออก (Do out) หรือจำนวนของเข้ารวม (Total delivery) กับจำนวนของออกรวม (Total Do out) ของวัสดุก่อสร้างชนิดใดชนิดหนึ่ง เพราะแต่บางครั้งเราอาจต้องการแสดงกราฟเส้น ของการเข้าหรือออกอย่างใดอย่างหนึ่ง ของวัสดุก่อสร้าง เราก็อาจเลือกเอาฟิลด์ใดฟิลด์หนึ่ง กราฟเส้นที่ได้ จะเห็นการที่ออกของวัสดุก่อสร้างนั้น เพียงชนิดเดียว

- กราฟแท่ง จากกราฟเส้น เราจะเห็นว่า ประโยชน์ของมัน จะอยู่ตรงที่มันแสดงให้เห็นถึง การเปลี่ยนแปลงปริมาณวัสดุก่อสร้างชนิดใดชนิดหนึ่ง ณ เวลาใด ๆ ของการสั่งเข้าหรือนำไปใช้ ดังนั้น กราฟแท่งจะถูกโปรแกรมขึ้นมา ให้มันแสดง การเปลี่ยนเทียบการเข้าหรือออกอย่างใดอย่างหนึ่ง ของวัสดุก่อสร้างหลายชนิด (10 ชนิด เนื่องจากข้อจำกัดทางด้านจอภาพ) ทำให้ประโยชน์ของมัน จะอยู่ตรงที่ การนำไปแสดงปริมาณวัสดุก่อสร้างที่เหลืออยู่ในสต๊อก

- กราฟวงรี ซึ่งเป็นกราฟแบบสุดท้ายของโครงงานนี้ จะอาศัยคุณลักษณะของกราฟวงรี ที่มันสามารถแสดงการเปรียบเทียบชนิดของข้อมูลให้เห็นได้ชัดเจน ดังนั้น เราจึงให้มันแสดงชนิดของวัสดุก่อสร้างให้เราเห็น โดยชนิดของวัสดุก่อสร้างนี้ จะมีอยู่ 3 ชนิด หรือ material เช่น อิฐ, หิน, ปูน, ทราย Equipment เช่น บุ้งกี๋, แปรงทาสี, เกียง ฯลฯ และ - machine เช่น เครื่องสูบน้ำ, เครื่องผสมปูน, มอเตอร์ ฯลฯ แต่เมื่อเรานำมันมาพร็อตกราฟมันดูไม่สวยงาม ดังนั้น จึงเพิ่มชนิดของวัสดุก่อสร้างเป็น 5 ชนิด จาก A ถึง E โดยสุ่มเอาตามความเหมาะสม

ดังนั้น จากกราฟทั้งสามแบบ จะเห็นว่า กราฟแต่ละแบบ จะต้องเลือกข้อมูลที่จะให้มันนำมาแสดงผล ดังนี้

- กราฟเส้น หลังจากเลือกกราฟแบบนี้แล้ว จะต้องเลือกฟิลด์ข้อมูล ที่จะอ่านมาแสดงผล จำนวน 2 ฟิลด์ แล้วโปรแกรมจะอ่านข้อมูลจากไฟล์ Make.dbf ซึ่งเป็นไฟล์แบบ update มาแสดงชื่อของวัสดุก่อสร้างทั้งหมดที่เหลืออยู่ในสสต๊อก ให้เลือกวัสดุก่อสร้างหนึ่งชนิด จากนั้น มันจะอ่านไฟล์ Listing.dbf เพื่อจัดเอาการเข้าหรือออกของวัสดุก่อสร้างที่ได้บันทึกไว้เป็นประวัติ มาแสดงผล ตามการเข้าออกของวัสดุก่อสร้างชนิดนั้น

- กราฟแท่ง จะทำคล้ายกับกราฟเส้นแต่การเลือกวัสดุก่อสร้าง 10 ชนิด เพื่อนำเอาข้อมูลจากไฟล์ Make. dbf มาแสดงผลตามฟิลด์ของวัสดุก่อสร้างที่เราเลือกไว้ เป็นกราฟ 10 แท่ง

- กราฟวงรี จากกราฟทั้งสองแบบ เราจะเห็นว่า จะมีความจำเป็นที่เราต้องกำหนดไฟล์ข้อมูลที่แน่นอน ในการอ่านข้อมูลมาแสดงผล กล่าวคือ กราฟเส้นจะอ่านข้อมูลจากไฟล์ Listing.dbf และกราฟแท่งจะอ่านข้อมูลจากไฟล์ Make.dbf มาแสดงผล ดังนั้นเพื่อให้เห็นกราฟสามารถในการเลือกไฟล์ใกล้ได้ มาแสดงผล จึงกำหนดในกราฟแบบนี้ เลือกไฟล์ข้อมูลที่จะอ่านมาแสดงผลได้ ดังนั้นหลังจากที่เลือกแบบและฟิลด์ข้อมูลแล้ว มันจะให้เลือกไฟล์ข้อมูลคือ Make.dbt และ Listing. bdf เพื่อนำไปอ่านข้อมูลมามาแสดงผลต่อไป จะสังเกตเห็นว่า กราฟแบบนี้ จะไม่มีการให้เลือกชื่อวัสดุก่อสร้าง เพราะมันต้องอ่านไฟล์ข้อมูลทั้งไฟล์ แล้วแยกข้อมูลตามชนิดของวัสดุก่อสร้างอยู่แล้ว

อนึ่ง สำหรับไฟล์ข้อมูลนั้น จะต้องมีผลข้อมูลด้วยคำสั่ง Sort ของดีเบส ก่อนนำมาใช้ ดังนั้น ในโครงงานนี้จะใช้ไฟล์ Make.dbf มาทำการเรียงข้อมตามคำดับชื่อของวัสดุก่อสร้าง เป็นไฟล์ List Grph.dbf และการเลือกกราฟ, แบบข้อมูลและไฟล์ข้อมูลตามที่กล่าวมานี้จะอยู่ในโปรซิเดียว Select Type Graph ในโปรแกรม DbGraph.PAS ที่ภาคผนวกแล้ว

สำหรับโปรซีเดียว Select Type Graph จะมีการสร้างเมนูเลือกแบบของกราฟ ในส่วนของโปรแกรมหลักของโปรซีเดียว ซึ่งสามารถเลื่อนแถบสร้างของของเมนูขึ้น-ลงได้ และจะสร้างเมนูเลือกฟิลด์ข้อมูล หลังจากเลือกแบบของกราฟแล้ว โดยเรียกใช้โปรซีเดียว

Block 2 ของมันมาใช้ ซึ่งเมนูเลือกฟิลด์ข้อมูลนี้ก็สามารถเลื่อนแถบสร้างขึ้น-ลงได้เช่นเดียวกันกับเมนูเลือกแบบของกราฟ จากนั้นเมื่อเลือกแบบฟิลด์ข้อมูลแล้วโปรแกรมจะพิจารณาว่ากราฟที่เลือกเป็นกราฟวงรีหรือไม่ ถ้าใช่ มันจะแสดงเมนูเลือกไฟล์ข้อมูลที่จะอ่านข้อมูล เพื่อนำมาแสดงผล โดยเรียกใช้โปรซิเดียว Block 3 หลังจากนั้น จะแสดงกราฟ. ฟิด์ข้อมูลและไฟล์ข้อมูล ที่เราได้เลือกไว้ ก่อนจะเริ่มทำการพล็อตกราฟต่อไป

การอ่านข้อมูลดีเบส

ก่อนที่เราจะโปรแกรมบาสกาล ให้อ่านข้อมูลจากไฟล์ด้วยสมาใช้งานนั้น จะต้องทำความเข้าใจเกี่ยวกับการจัดเก็บข้อมูลของดีเบส เราจะพบข้อมูลที่สำคัญ ๆ ดังนี้

1. วันที่ที่มีการแก้ไขข้อมูลครั้งสุดท้ายจะเก็บ คศ. ตามด้วยเลขลำดับที่ของเดือนและวันที่ ตัวอย่างเช่น วันที่ 11/09/90 จะเก็บเป็น 5AOB09h

2. จำนวนเรคคอร์ดทั้งหมด ที่ถูกเก็บในไฟล์ข้อมูล

3. ขนาดของเรคคอร์ด

4. ชื่อของฟิด์. ชนิดของฟิลด์ และขนาดของฟิลด์ที่กำหนดในเรคคอร์ดนั้น มีขนาด 32 ไบท์ ดังนั้น ถ้าเรคคอร์ด มีหลายฟิลด์ ก็จะเก็บชื่อของฟิด์. ชนิดของฟิลด์และขนาดของฟิลด์ ในลำดับทัดไปอีก 32 ไบท์ และชื่อของฟิลด์นั้น ทั้งดีเบสจะกำหนดไว้ให้มีขนาดไม่เกิน 10 ไบท์ เช่น ในโปรแกรม Makc.prg จะกำหนดฟิลด์จำนวนของคงเหลือ (Total in stock) ไว้เป็น Tol_lni_Sto เป็นต้น และชื่อฟิลด์นี้ จะมีขนาด 10 ไบท์แน่นอน โดยไม่สนใจว่า เราจะกำหนดชื่อฟิลด์ไว้ครบทั้ง 10 ไบท์ หรือไม่ เช่น ชนิดของวัสดุก่อสร้าง แม้ว่าเราจะกำหนดชื่อฟิลด์ไว้เป็น Type ก็ตาม

ดังนั้น ชนิดของฟิลด์จะถูกเก็บเป็นอักษรตัวเดียวในไบท์ที่ 11. เช่น ฟิลด์แบบ Neimeric จะเก็บ N ไว้เป็นต้น ส่วนไบท์ที่ 16 จะเก็บขนาดหรือความกว้างของฟิลด์ และถ้าฟิลด์นั้นเป็นแบบเลขจำนวนจริง จะเก็บจำนวนจุดทศนิยม ไว้ในไบท์ถัดไปด้วย และในโครงงานนี้ จะมีโปรซิเดียว Read Structure สำหรับอ่านโครงสร้างข้อมูลของไฟล์

มิติที่สองจะใช้อ้างในการเก็บชื่อฟิลด์ทีละตัวอักษรตามกำลังข้างบน

สุดท้ายจะเก็บจุดทศนิยมของฟิลด์แบบ Numeric ดังนี้

```
If Type Of Field[No] = 'N' Then
   FDec[No] : = Head [32 * No+17];
```

การเก็บโคครงสร้างของฟิลด์นี้จะใช้การวนรอบเท่ากับจำนวนฟิลด์ทั้งหมด (Count Of Field)

การแสดงวัสดุก่อสร้างคงเหลือในสต๊อค

สำหรับกราฟเส้นและกราฟแท่งนั้น จะมีการรับวัสดุก่อสร้างคงเหลือในสต๊อคก่อนที่จะนำไปพล็อตกราฟ โดยวัสดุก่อสร้างนี้จะอ่านมาจากไฟล์ Make.dbf แล้วนำมาสร้างเป็นลิงก์ลิสต์สองทางแบบมีเซลเตอร์โหนด (Double Linked List with Header Node) เพื่อสามารถเลื่อนแถบสร้างไปมา เพื่อเลือกวัสดุก่อสร้างที่ต้องการให้แสดงผลได้

โปรซีเดียว Display Database จะแยกเอาฟิลด์ชื่อวัสดุก่อสร้างหรือ Description ไปแสดงบนจอภาพและสร้างโหนดใหม่บนลิงค์ลิสต์สองทาง โดยข้อมูลในฟิลด์ใด ๆ จะใช้การวนลูป

```
FOR S : = x TO (x+Width Of Field [No]-1) Do
     Tem_Str : = Tem_Str+Data Dbase [s];
```

เพื่อเลื่อนข้อมูลของไฟล์ดีเบส ในอาเวย์ Data Dbase มาเก็บในตัวแปร Tem_Str หลังจากนั้นจะพิจารณาว่าเป็นข้อมูลของฟิลด์ชื่อวัสดุก่อสร้างหรือไม่

```
if Name Of Field [No] = "DESCRIPT" THEN
```

ถ้าใช่ก็ให้แสดงผลบนจอภาพ

```
GotoXY(Where Y+1);
Write (Tem_Str);
```

จากนั้น จะนำข้อมูลที่ได้ไปเก็บในเรคคอร์ด Now_Rec ซึ่งเป็นเรคคอร์ดแบบเดียวกับเรคคอร์ดของดีเบส และจะทำการวนลูปเช่นนี้ จนครบทุกฟิลด์ของเรคคอร์ดนั้นแล้ว จึงทำการเชื่อมโหนด

ตัวแปร Temp_Rec จะเป็นพวยค่าเตอร์ชั่วคราว สำหรับใช้ช่วยเชื่อมโหนด โดยให้โหนดสุดท้ายของลิงก์ลิสต์ที่ชี้โดยพวยค่าเตอร์ At_Rec นำพวยค่าเตอร์เชื่อมโยง (Linkage Pointer) Next มาเซ็ท Temp_Rec ดังนี้

```
Temp_Rec : = At_Rec Next;
```

หลังจากนั้น ให้พวยค่าเตอร์เชื่อมโพง Previous ของ Tem_Rec ชี้ไปยังโหนดใหม่

```
Temp_Rec ^. Previous := Now_Rec;
```

และเซ็ท Next ของโหนดใหม่ ให้ชี้มายัง Temp_Rec

```
Now_Rec ^. Next : = Temp_Rec;
```

ตอนนี้ก็สามารถเซ็ท Previous ของ Now_Rec ให้ชี้ไปยังโหนดสุดท้ายของลิงก์ลิสต์เดิม นั้นคือ At_Rec

```
Now_Rec ^. Previous: = At_Rec;
```

พร้อมกับเซ็ทโหนดสุดท้ายเดิม At_Rec ให้ชี้มายังโหนดใหม่ด้วย

```
At_Rec ^. Next : = Now_Rec;
```

ก็จะจบการเชื่อมโหนดใหม่ Now_Rec เข้ากับโหนดเดิม At_Rec และสำหรับพวยค่าเตอร์เชื่อมโยง Previous และ Next จะชี้ไปยังโหนดก่อนและหลังโหนดนั้นเสมอ แต่โหนดสุดท้ายจะต้องปิดท้ายด้วย NIL เสมอ ดังนั้น จากกำลังข้างต้น จะเห็นว่าเราจะใช้โหนดชั่วคราวมาเซ็ทเป็น NIL ด้วยพวยค่าเตอร์ Next ของโหนดสุดท้ายเดิม At_Rec เพื่อนำมันไปเซ็ท Now_Recของโหนดใหม่ Now_Rec ให้ปิดท้าย Next ด้วย NIL ในคำสั่งที่สาม

จากการแสดงวัสดุก่อสร้างข้างต้น ด้วยโปรซิเดียว Display Data Base นี้ จะถูกเรียกใช้โดยโปรซิเดียว Read Dbase ซึ่งโปรซีเดียวนี้จะทำการอ่านโครงสร้างของไฟล์ Make.dbf ด้วยการเรียกใช้โปรซีเดียว Read Structure และอ่านข้อมูลด้วยโปรซิเยว Read Dbase จากนั้นจึงตีกรอบด้วยโปรซีเดียว Draw Window ในยูนิต Utility เพื่อเขียข้อมูลลงไป ด้วยการเรียกใช้โปรซีเดียว Display Data Base เท่ากับจำนวนเรคคอร์ดของไฟล์ดีเบสนั้ ด้วยการวนลูป

```
REPEAT
    Last Rec : = Head Rec ^ .  Previous;
    Display Database  (Last Rec;i);
    Inc (i);
UNTIL i = Total_Thing+1;
```

การวนลูปนี้ จะมีการเซ็ท Previous ของ Header ให้ชี้ไปยังโหนดใหม่หรือ Last Rec เสมอ เพื่อเชื่อมโยงลิงค์ลิสต์สองทางแบบมีเซตเดอร์โหนดให้เป็น circular Double Linked List

การเลือกวัสดุก่อสร้างที่ต้องการ

การเลือกวัสดุที่ต้องการจะทำโดยการเลื่อนแถบสว่างไปยังวัสดุก่อสร้างนั้น ที่กำลังแสดงอยู่บนจอภาพ แล้วจึงกดคีย์รีเทิร์นโดยงานนี้ เราจะใช้โปรซีเดียว Display Choice ซึ่งประกาศไว้ตรงส่วนหัวเป็น

```
Procedure Display(Choicec No_Next : Byte; Shift_Next:Boolean);
```

พารามิเตอร์ No_Next จะเป็จำนวนเรคคอร์ดหรือโหนดในลิงค์ลิสต์ ที่จะต้องเลื่อนไปตามการเลื่อนแถบสว่าง และพารามิเตอร์ Shift_Next จะกำหนดทิศทางที่ต้องการเลื่อนโดยที่

- Shift_Next เป็นจริง จะใช้พวยต์เตอร์เชื่อมดยง Next ของโหนดปัจจุบันน เลื่อนไปยังโหนดถัดไป ตามจำนวนที่กำหนดในพารามิเตอร์ No_Next
- Shift_Next เป็นจริง จะทำตรงข้าม กล่าวคือจะใช้พวยต์เตอร์เชื่อมโยง Previous เลื่อโหนดไปยังโหนดก่อนโหนดนั้นเท่ากับจำนวนโหนดที่กำหนดในพารามิเตอร์ No_Next แทน

หลังจากั้น จะนำเอาชื่อวัสดุก่อสร้างมาแสดงผลบนจอภาพ ด้วยกำลัง

```
Write(Pointer ^ .  Des(ript);
```

สำหรับการกำหนดจำนวนเรคคอร์ดและทิศทางในพารามิเตอร์ ของโปรซีเดียว Display Choice นั้น จะถูกกำหนดจากการเรียกใช้มันในฟังก์ชั่น Get Choice ซึ่งกำหนดในส่วนหัวไว้เป็น

FUNCTION Get Choice (VAR Pt : Rec Ptr) : String ;

ฟังก์ชั่นนี้ จะส่งสตริงของชื่อวัสดุก่อสร้างที่เลือก และคอยควบคุมการเลื่อนแถบสว่าง ตามคีย์ที่กด โดยมีรายละเอียด ดังนี้

มันจะวนลูป

```
REPEAT
    :     :
    :     :
UNTIL Select;
```

จนกว่าจะมีการเลือกวัสดุก่อสร้าง โดยการวนลูป จะทำการอ่านคีย์ที่กดจากผู้ใช้

```
Read Func Key (Key, Move_Key);
```

ถ้าผู้ใช้กดคีย์เลื่อนแถบสว่าง โปรซีเดียว Read FuncKey ของยูนิต Utility จะเซ็ทพารามิเตอร์ Move_Key ให้เป็นจริง เพื่อทำการเลื่อนแถบสว่างให้ โดยการคืนแถบสว่างนี้ จะเซ็ทแอททริบิวท์ของชื่อวัสดุก่อสร้าง ที่แถบบนแถบสว่างเดิม ให้เป็นแบบปกติ

```
SetAttr (Normal Display);
GotoXY (Whore X-18, Where Y):
Write (Pointer ^ . Descript);
Set Attr (Reverse Display);
```

จากนั้น จึงเขียนมันซ้ำ แล้วเซ็ทแอททริบิวท์ให้เป็นริเวิร์สวีดีโอ เพื่อเขียนชื่อวัสดุก่อสร้างที่จะเลื่อนแถบสว่างไป จากนั้นจะตรวจสอบคีย์ที่กด ดังนี้

- Up_Key เลื่อนแถบสว่างขึ้น ด้วยการเรียกใช้โปรซีเดียว Mave Up
- Down_Key เลื่อนแถบสว่างลง ด้วยการเรียกใช้โปรซีเดียว Move Down
- Left_Key เลื่อนแถบสว่างไปทางซ้าย ด้วยการเรียกใช้โปรซีเดียว Move Left
- Right_Key เลื่อนแถบสว่างไปทางขวา ด้วยการเรียกใช้โปรซีเดียว Move Right

การวาดกราฟแท่ง 3 มิติ

เทอร์โบปาสกาลมีรูทีน สำหรับสร้างกราฟแท่ง 3 มิติ ดังนี้

```
Bar 3 D(x1, y1, x2, y2, Depth, Top);
```

อากิวเมนท์ (x2, y1) เป็นพิกัดของจุดล่วงซ้ายของกราฟแท่ง ขณะที่พิกัด (x2, y2) เป็นพิกัดของจุดบนขวาของกราฟแท่งนั้น ดังนั้นความสูงของกราฟแท่งจะเท่ากับ y1-y2 (จุดออริจินของกราฟแท่งจะอยู่มุมบนซ้าย แต่จอภาพจะอยู่มุมบนซ้ายของจอภาพแทน) ความกว้างของกราฟแท่ง จะเท่ากับ x2-x1 และความหนาของกราฟแท่งสามมิติจะกำหนดด้วยอากิวเมนท์ Depth โดยความหนานี้จะเป็นจำนวนฟิดเซล เช่นเดียวกับความกว่า 3 และสูงบ้าง ต้น สุดท้ายอากิวเมนท์ Top จะเป็นแบบบูลีน กำหนดการปิดขอบ 3 มิติด้านบนถ้าให้ทำเป็น True ถ้าไม่เช่นนั้น (False) จะไม่มีการปิดขอบบน เพื่อว่าผู้ใช้ต้องการจะพล็อตกราฟแท่ง ซ้อนหรือต่อกราฟแท่งออกไปได้

สำหรับในโครงงาน จะนำรูทีนนี้มาใช้ ในการวาดกราฟแท่งทั้ง 10 แท่ง ตามข้อมูลของวัสดุก่อสร้างทั้ง 10 ชนิดนั้น ดังนี้

- กำหนดความหนาของกราฟแท่ง ไว้ 10 ฟิดเซล

```
Depth : = 10;
```

ก่อนที่จะวนลูปวาดกราฟแท่งทั้ง 10 แท่ง ดังนี้

- กำหนดแนว (บรรทัด) ที่จะแสดงชื่อวัสดุก่อสร้าง ให้เยื้องกันบน. ล่าง

```
If (i MOD2) = 0 Then
WhereY : = Get Max Y-25
Else
Where Y : = Get Max Y-14;
```

เพื่อไม่ให้มันซ้อทับกัน

- นำสตริงชื่อวัสดุก่อสร้างที่เลือก มาเขียนตรงแนวบรรทัดนั้น

```
Material_Name : = V_Name Of Matr [i];
Out Text XY (Where x, Where Y, Shiftchr (Material_
Name));
```

สำหรับฟังก์ชั่น Shift Chr จะใช้ในการตัด Space ของชื่อวัสดุก่อสร้างออก ให้เหลือเพียงตัวอักษร ทำให้การเขียนชื่อวัสดุก่อสร้างอยู่กึ่งกลางระหว่างสองด้านของกราฟแท่ง

- นำสตริงของขนาดวัสดุก่อสร้าง มาเปลี่ยนเป็นเลขจำนวนจริง เพื่อคำนวณหาความสูงของกราฟแท่ง ในหน่วยฟิดเซล

```
Val(V_Value Of Matr [i,i], Valuc Of Y)/ RangeY);
```

- จากนั้นจึงเริ่มทำการพล็อตกราฟแท่งตามข้อมูลเหล่านั้น

```
Bar 3D (WhereX-15, GetMaxY-30, Where X+15,
        (GetMax Y-30)-HeightOfY, Depth, True);
```

เพื่อให้การอ่านกราฟแท่งที่ได้ง่ายขึ้น

หลังจากนั้นจะวนลูปเพื่อแสดงกราฟแท่งของข้อมูลในวัสดุก่อสร้างถัดไป จนครบทั้ง 10 แท่ง จึงทำการเขียนส่วนหัวกำกับกราฟแท่ง ดังนี้

- แปลงจำนวนวัสดุก่อสร้างทั้งหมด ให้เป็นสตริง แล้วเขียนมันด้านบนของจอภาพ

```
Str(Total-Choose, Number_Str);
Out Text XY(GetMaxX Div 2,5, Num Ber_Sir+ 'th Material');
```

ก็จะได้กราฟแท่งตามต้องการ หลังจากนั้นจะรอการกดคีย์ เพื่อจบการแสดงผล

กราฟวงรี

กราฟวงรีในโครงงานนี้ จะเป็นกราฟวงรี 3 มิติ หรือ Pie Slice SD สำหรับการแสดงอัตราส่วนของข้อมูลในฟิลด์ของไฟล์ดีเบสที่เลือกโดยกราฟนี้ จะทำให้เราทราบได้ว่าสต๊อคมีวัสดุก่อสร้างชนิดต่าง ๆ คงเหลืออยู่ในสต๊อคเป็นจำนวนเท่าไร ถ้าเราเลือกให้มันแสดงข้อมูลตามฟิลด์ Tol_In_Sto (จำนวนวัสดุคงเหลือในสต๊อก Total In Stock) หรือถ้าเราเลือกให้มันแสดงกราฟวงรีตามฟิลด์ Delivery หรือจำนวนวัสดุที่มีการเข้าสต๊อค) มันก็สามารถบอกได้ว่า เราได้สั่งวัสดุก่อสร้างชนิดใดมาเท่าไร

จากการที่กราฟแบบนี้ จะนำข้อมูลจากวัสดุก่อสร้างทั้งหมดในไฟล์ที่เลือก มาแยกตาม

ชนิด เพื่อหาอัตราส่วนบนวงรี ดังนั้น การทำงานของโปรซีเดียว Pie Slice 3D จึงมีการทำงานดังต่อไปนี้

- อ่านโครงสร้างและข้อมูลในไฟล์ดีเบสที่ผู้ใช้เลือก ด้วยโปรซีเดียว Read Structure และ Read Data Base ตามลำดับ
- วนลูปเพื่อแยกข้อมูลตามฟิลด์ ของเรคคอร์ดทั้งหมดที่อยู่ในไฟล์ เพื่อที่จะ
- แยกข้อมูลในฟิลด์ที่เลือกไปเก็บรวมในอาเรย์ V_Arr_Type Matr ตามชนิดของวัสดุก่อสร้างนั้น พร้อมกับคำนวณจำนวนวัสดุก่อสร้างโดยรวมของฟิลด์นั้นด้วย
- เริ่มวาดกราฟวงรีสามมิติ โดยเริ่มจากวาดวงรี 2 วงซ้อนกัน พร้อมกับดีเส้นความหนา
- คำนวณหาอัตราส่วนของพื้นที่หรือมุมของส่วนของวงรี ตามข้อมูลที่ได้
- วาดส่วนของวงรีตามมุมที่ได้ พร้อมกับแรเงาลงไป
- วาดส่วนของวงรีตามวัสดุก่อสร้างชนิดอื่น จนครบทั้ง 5 ชนิด
- เขียนส่วนหัวของกราฟวงรีสามมิติ
- แสดงคำบรรยาย การแทนชนิดของวัสดุก่อสร้างด้วยการแรเงา
- รีบการกดคีย์ใด ๆ เพื่อจบการทำงาน

และต่อไปนี้ จะได้กล่าวคือส่วนต่าง ๆ นี้โดยละเอียด

<u>การแยกข้อมูลในฟิลด์ตามชนิดของวัสดุก่อสร้าง</u>

การวนลูปในส่วนนี้ของโปรแกรม จะคล้ายกับส่วนอื่น ๆ ที่แยกข้อมูล เพื่อใช้สร้างกราฟเส้นและกราฟแท่ง แต่ต่างกันตรงที่หลังจากที่ได้ข้อมูลในฟิลด์ของเรคคอร์ดมาแล้ว จะตรวจสอบดูว่าเป็นฟิลด์ที่ต้องการหรือไม่

```
if Name Of Field [No_F] = Field 2 then
begin
    Tem_Value : = 0.0;
    Val (Tem_Str, Tem_Value, Result);
end;
```

ถ้าใช่ก็เก็บมันไว้ในตัวแปร Temp_Value ก่อนที่จะตรวจสอบข้อมูลในเรคคอร์ดนั้นต่อไป ว่าเป็นฟิลด์ข้อมูลของวัสดุก่อสร้างชนิดใดด้วยการเลื่อนข้อมูลตามฟิลด์ จนกว่าจะพบฟิลด์ชื่อ Type หรือชนิดของวัสดุก่อนสร้าง ซึ่งชนิดจะถูกเก็บเป็นตัวอักษร "A" ถึง "E" ตามที่กล่าวไว้ในตอนต้น

```
if Name ofField [No_F] = 'TYPE' then
Begin
Temp_Num : = 0.0
if Temp_Str = 'A' then
begin
Temp_Num : = V_Arr_Type Matr [1];
V_Arr_Type Matr [1] : = Temp_Num = Temp_Value;
end;
if Temp_Str = 'B' then
        :    :    :    :
        :    :    :    :
        :    :    :    :
        :    :    :    :
if Temp_Str = 'E' then
        begin
        Temp_Num : = V_Arr_Typematr [5];
        V_Arr_Type Matr [5] ; = Temp_Num + Temp_Value;
end;
Sum_Matr : = Sum_Matr + Temp_Value;
End;
```

เมื่อพบว่าเป็นข้อมูลของวัสดุก่อสร้างชนิดใดแล้ว ก็ให้รวมข้อมูลเพิ่มเข้าไปกับข้อมูลเก่าในอิลิเมนท์นั้นของอาเรย์ เช่น ถ้าเป็นข้อมูลของวัสดุก่อสร้างชนิด A ก็เก็บมันไว้ในอิลิเมนท์แรกของอาเรย์

V_Awi_Type Matr เป็นต้น พร้อมกับรวมวัสดุก่อสร้างเข้าไว้ในตัวแปร Sum_Matr ซึ่งเป็นตัวแปรที่ใช้สำหรับเก็บวัสดุก่อสร้างทุกชนิดด้วย จึงวนลูปเพื่อพิจารณาเรคคอร์ดถัดไป จนหมดไฟล์

การสร้างกราฟวงรี

ในโครงงานจะใช้วิธีการสร้างวงรีสามมิติ ดังนี้ จะวาดวงรี 2 วงซ้อนกัน โดยวงรีบน จะวาดเพียงครึ่งวง แล้วจึงตีขอบทั้ง 2 ด้าน จากนั้นจึงแบ่งวงรีเป็นส่วน ๆ ตามมุม ที่คำนวนจากข้อมูลชนิดวัสดุก่อสร้าง พร้อมกับแรเอาจนครบทั้ง 5 ชนิด จะได้วงรีเต็มวงพอดี แต่การที่จะวาดวงรีให้เป็น 3 มิติแบบ Pie Slice 3D นั้น จำเป็นต้องตีขอบความหนาตามส่วนที่ถูกตัด ในด้านที่เห็นด้วย ดังนั้นการวาดกราฟวงรีจะมีรายละเอียด ดังนี้

- วาดวงรีด้านบนเต็มวง จากมุม 0-360 องศา และวงรีด้านล่างครึ่งวง จากมุม 197-340 องศา ด้วยฟังก์ชัน Ellipse ของเทอร์ดบปาสคาล

```
Ellipse((GetMaxX Div 23+60, (Get MaxY Div 2)-40, 0360, 100, 65);
Ellipse((GetMaxX Div 2)+60, GetMaxY Div 2), 197, 340, 160, 80);
```

- ตีขอบด้านข้างของวงรีที่ได้ทั้งสองด้าน ที่พิกัดตามที่หาได้จากฟังก์ชัน เริ่มจากขอบด้านซ้ายก่อน

```
Get Arc Coords (Arc Coords);
WITH Arc Coords DO
    begin
        Line((Get MaxX Div 2)+60-150, (GetMaxY Div 2)-40,
            (Get MaxX Div 2)+60-150, (Get MaxY Div 2)-40+44);
        Line((Get MaxX Div 2)+60+150, (Get MaxY Div 2)-40, XEnd, YEnd);
    end;
```

สำหรับ XEnd และ YEnd นั้น จะเป็นจุดปลายสุดของวงรีด้านล่าง ซึ่งไม่เต็มวง และเป็นฟิลด์ของเรคคอร์ด Arc Coords ดังนั้น ในตอนนี้เราจะด้วงรี 3 มิติ ที่พร้อมจะให้เรา

แบ่งมันเป็น Pie Slice 3D แล้ว

- วนลูปการแบ่งวงรี เริ่มจากคำนวณหามุมตามอัตราส่วนข้อมูลชนิดนั้น

```
Tem_Value : = V_Arr_Type Matr [i];
Portion : = Round(360.0 * (Tem_Value/Sum_Matr));
```

พร้อมกับคำนวณหาอัตราส่วนเป็นเปอร์เซ็นต์ สำหรับเขียนในคำบรรยายด้วย

```
Tem_Value : = 100.0 * (Tem_Value/Sum_Matr);
Str (Tem_Value :2 : 2, Percent [i]);
```

แต่มุมที่จะใช้ในการแบ่งวงรี จะเป็นมุมเริ่มต้น จนถึงมุมสุดท้าย ดังนั้น จึงคำนวณมุมสุดท้าย จาก Portion หรือมุมระหว่างที่หาได้

```
If i = Total_Type Then
  End AngLe : = 360
Else
  End Angle : = Start Angle & Portion;
```

เหตุที่ต้องกำหนดเงื่อนไขไว้เช่นนี้เพราะ การปัดเศษมุม Portion ด้วยฟังก์ชั่น Round จะทำให้วงรีถูกตัดไม่เต็มหรือเกินวงได้ ดังนั้นในครั้งสุดท้ายของการแบ่งวงรี จึงต้องกำหนดมุมสุดท้ายให้เท่ากับ 360 องศา

- แบ่งส่วนของวงรีตามมุมที่ได้ สำหรับโครงงานที่จะใช้โปรซิเดียว Sector ของเทอร์โบปาสคาล วาดส่วนของวงรีพร้อมกับแรเงาลงไป โดยสีของแรเงาจะกำหนดโดยคำสั่ง

```
Set FillStylec 5+i, Get Max Color);
Sector((GetMaxX Div 2)+60, (Get Max Y Div 2)-40,
  Stait Angle, End Angle, 150, 65;
```

หลังจากนั้นจะทำเส้นตามส่วนของวงรีที่ได้ เริ่มจากคำนวณจุดกึ่งกลางของวงรี

```
InX : = (GetMazX Div 2) +60;
InY : = (GetMaxY Div 2)-40;
```

พร้อมกับรับพิกัด (x, y) ของจุดเริ่มต้นและจุดปลายของวงรีที่ได้

```
Get Arc Coords (Arc Coords);
```

จากนั้นจึงทำเส้นตรงตามส่วนของวงรีที่ได้

```
Line (InX, InY, XStart, YStart);
Line(InX, InY, XEnd, YEnd);
```

จากนั้นจึงตีเส้นความหนา โดยถ้าจะพิจารณาการตีเส้นแบ่งรีตามความหนานี้ จะตีเพียงเส้นถัดขวาของส่วนของวงรี เพียงเส้นเดียวก็พอ ยกเว้นเส้นนั้น เกิดจากมุม 360 องศา ไม่ต้องตีขอบข้าง ใช้เส้นเดิมของขอบข้างตอนต้นแทน ดังนั้น จึงต้องตรวจสอบมุมก่อนตีเส้นขอบข้าง

```
if EndAngle <> 360 then
   Line (XEnd, YEnd, XEnd, YEnd+37);
```

จากนั้นจะวนลูป เพื่อแบ่งวงรีตามข้อมูลชนิดอื่นต่อไป จนครบทั้ง 5 ชนิด ก็จะได้กราฟวงรีตามต้องการ

- การเขียนส่วนหัวของกราฟวงรี

```
Out TextXY(40, 60, "Goumagai Co., Ltd.");
```

พร้อมกับขีดเส้นใต้ 2 เส้น

```
Line(30, 60+(TextHeight ('G') MOD 2)+15+2*i,
Text Width ("_Goumagai Co., Ltd._")+15, 60+(Text Height
("G") MOD 2)+15+2*i);
```

- เขียนคำบรรยาย การแทนชนิดของวัสดุก่อสร้าง ด้วยการแรเงา ซึ่งจะคล้ายกับคำบรรยายของกราฟทั่วไป กล่าวคือ จะมีการตีกรอบสี่เหลี่ยมเล็ก ๆ และแรเงาลงไป พร้อมกับบอกชนิดที่มันแทน และเปอร์เซ็นต์ของวัสดุก่อสร้างนั้น ด้วยการวนลูป เซ็ทสีของแรเงา และสร้างกรอบสี่เหลี่ยมเล็ก ๆ แล้วจึงแรเงา ดังนั้น

```
Set Fill Style (5+i, GetMaxColor);
Rectangle (35, (Get MazY-35-((25+5)*(Total_Type-i));
   60, (Get MazY-35)-((25+5)*(Total_Type_i))+1,
    Get Max Colo);
Flood Fill (36, (GetMaxY-35)-((25_5)*(Total_Type-i))+1, GetMax
```

จากนั้น จึงเขียนชนิดของวัสดุก่อสร้างกำกับ

```
Out TextXY670, (GetMaxY-35)-((25_5)*(Total_Type-i))+12, Name_Type[i]);
```

พร้อมด้วยเปอร์เซ็นทต์ของวัสดุก่อสร้างชนิดนั้น

```
Out TextXY(70+Text Width ("AMaterial")+5,
        (GetMazY-35)-((25+5)*Total_Type-i))+12;
        Percent [i]+"%");
```

และเมื่อเขียนคำบรรยาย ด้วยการวนลูปข้างต้น ครบทั้ง 5 ชนิดแล้ว จึงแสดงฟิลด์ข้อมูล ซึ่งเป็นที่มาของข้อมูลพล็อตกราฟวงรี บริเวณด้านล่างของวงรี

```
Out TextXY(Get MaxX-30, Get MaxY-40, "Type Material
["+Field2+']");
```

พร้อมกับรอรับการกดคีย์ เพื่อจบการแสดงผล

คำสั่งในการวาดกราฟวงรี

คำสั่ง Sector มีรูปแบบเป็น

```
Sector(X, Y, Start Angle, End Angle, XRadius, YRadius);
```

เป็นคำสั่งสำหรับวาดส่วนของวงรี ณ จุดศูนย์กลาง (x, y) เป็นมุม Start Angle ถึง End Angle และมีรัศมีแนวนอน XRadius กับรัศมีแนวตั้ง YRadius แต่คำสั่งนี้จะไม่มีการมีเส้นล้อมรอบส่วนของวงรีให้ด้วย คำสั่งจะวาดวงรีด้วยการแรเงาจะไปในส่วนของวงรีที่กำหนดด้วยอากิวเมนท์เหล่านั้น เท่านั้น

เราจึงต้องวาดเส้นล้อมรองวงรีเอาจากคำสั่ง Line ตามพิกัดที่ได้จาก

```
Get Arc Coords (Arc Coords);
```

เรคคอร์ด ArcCoords เป็นแบบที่กำหนดไว้ในยูนิต Graph ของเทอร์โบปาสคาลเป็น

```
type ArcCoords Type = record
                X, Y          : Integer;
                XStarf, YStart : Integer;
                XEnd, YEnd     : Integer;
                end;
```

ฟิลด์ XStart, YStart และ XEnd, YEnd จะเป็นพิกัดฟิดเชคของจุดตั้งต้นและจุดปลายของส่วนเส้นตรง, เส้นโค้งหรือวงรี ที่เพิ่งวาดครั้งหลังสุด ก่อนที่จะหาโคออร์ดิเนต ด้วยคำสั่ง Get Arc Coords ข้างบน

ดังนั้น หลังจากที่วาดส่วนของวงรีด้วยการแรเงา ตามคำสั่ง Sector แล้ว เราจึงตีเส้นตรงรอบส่วนของวงรีที่ได้ด้วยคำสั่ง Line จากจุดศูนย์กลางของวงรี (InX, inY) จนถึงจุดตั้งต้นของส่วนโค้งวงรี

```
Line (InX, InY, XStart, YStart);
```

โดยที่จุดศูนย์กลางของวงรี อยู่ ณ ตำแหน่ง

```
InX : = (GetMaxX div 2)+60;
InY : = (Get MaxY Div 2)-40;
```

และตีเส้นตรงงอีกเส้น ปิดส่วนของวงรีตามพิกัดจุดปลาย

```
Line(InX, InY, XEnd, YEnd):
```

พร้อมกับตีเส้นขอบความหนา ของวงรีแบบสามมิติอีกหนึ่งเส้น

```
Line (XEnd, Yend, XEnd, YEnd+37);
```

ตามพิกัดจุดปลายของส่วนวงรี และเส้นนี้จะทำเพียงเส้นจุดปลายเท่านั้น เพราะจุดปลายนี้จะเป็นจุดเริ่มต้นของส่วนของวงรี ส่วนถัดไปจึงไม่ต้องตีขอบความหากของจุดตั้งด้านนั้นเอง

อนึ่ง สำหรับเทอร์โบปาสคาลนั้น ยังมีคำสั่ง Arc กับ Flood Fill สำหรับวาดส่วนของวงรีด้วยการตีเส้นกรอบ และแรเงาลงไปตามลำดับ แต่คำสั่ง Flood Fill มีข้อจำกัดในการใช้ไว้ว่า ถ้าแรเงาในส่วนปิดของเส้นรอบรูป จะต้องกำหนดอาร์กิวเมนท์ของจุดที่จะให้สั่ง อยู่ในเส้นรอบรูปนั้น ไม่เช่นนั้นแล้ว จะแรเงาพื้นที่นอกเส้นรอบรูปทั้งหมด ยกเว้นพื้นที่ในเส้นรอบรูปและสำหรับส่วนของวงรี ก็ยากจะกำหนดพิกัดในของเส้นรอบส่วนของวงรี จึงวาดวงรีด้วยคำสั่ง Sector และตีเส้นรอบรูปด้วยคำสั่ง Line แทนแต่สำหรับเส้นโค้งส่วนของวงรีนั้น ไม่ต้องตีอีก เพราะเราตีมันไว้แล้ว ด้วยคำสั่ง Ellipse ตอนก่อนที่จะแบ่งส่วนของวงรีนั่นเอง

โปรแกรมภายนอก

ในโครงงานนอกจากจะมีโปรแกรมตามที่ได้กล่าวไป ในการเรียกใช้เพื่อสร้างกราฟแล้ว ยังได้แยกโปรแกรมที่จำเป็นสำหรับช่วยเหลือเพิ่มเติมไว้ในยูนิต Utility และโปแกรม Shiftchr ดังมีรายละเอียดต่อไปนี้

ยูนิต Utility

ยูนิต Utility ได้รวมเอาโปรแกรมยูลิทิตี้

- Set Attr สำหรับเปลี่ยนแอททริวท์ ตัวอักษรที่จะแสดงบนจอภาพ ในการ
- Set Cursor On สำหรับใช้เปิดเคอร์เซอร์อีกครั้ง
- Set Cursor Off สำหรับใช้ปิดเคอร์เซอร์
- Draw Window สำหรับสร้างกรอบของหน้าต่าง 3 แบบ
- Read Funckey อ่านฟังก์ชั่นคีย์ที่กด เพื่อเลื่อนแถบสว่าง

และแต่ละโปรซีเดียว้ จะมีรายละเอียด ดังต่อไปนี้

โปรซิเดียว Set Attr ใช้ในการเซ็ทตัวแปร Text Attr ของเทอร์โบปาสคาล ซึ่งกหนดไว้เป็น

```
var textAttr : byte;
```

เพื่อกำหนดแอททริบิวท์ของตัวอักษรที่จะให้แสดงผล ในคำสั่ง Write ของการเลือกวัสดุก่อสร้าง เช่น เซ็ทมันเป็น

$70 หรือ 112 มันจะแสดงอักษรแบบรีเวิร์สวีดีโอ

$0F หรือ 15 มันจะแสดงอักษรเป็นตัวเข้ม (Intense)

หรือ $07 หรือ 7 จะแสดงเป็นตัวอักษรปรกติ เป็นต้น

ดังนั้น โปรซิเดียวนี้ จึงเป็น

```
Procedure Set Attr (Attrib : Byte);
Begin
  Text Attr : = Attrib;
end;
```

โปรซีเดียว Set Cursor On ใช้ในการเปิดเคอร์เซอร์ผ่าน ไอซี 6845 ("CRT Controller") ซึ่งทำหน้าที่ควบการแสดงผลบนจอภาพ ด้วยการเปลี่ยนค่ารีลิสเตอร์ของมันบางตัว ตามค่าที่ได้จากการดำเนินงานบางอย่าง กับพื้นที่เก็บข้อมูลของรวมไบออส (ROM BIOS Data Area) ดังนี้

ในหน่วยความจำ จะมีพื้นที่ส่วนหนึ่งใช้ในการเก็บข้อมูล ที่ได้จากการดำเนินงานของรวมไบออส เช่น แอดเดรสของพอร์ทอนุกรมพอร์ทขนาด. คีย์บอร์ดบัฟเฟอร์ รวมไปถึง ค่าเริ่มต้น. สุดท้ายของสแกนไลน์ (scan line) เกอร์เซอร์บันจอกาฟด้วย ซึ่งสองค่าหลังนี้จะกำหนดไว้ที่แอดเดรส 0040:0060h และ 0040:0061h ตามลำดับ

นอกจากนี้แล้ว ที่แอดเดรส 0040:0064h ของพื้นที่เก็บข้อมูลของรวมไบออส ยังเก็บหมายเลขพอร์ทอะแคปเตอร์จอภาพ ซึ่งเป็นพอร์ทเดียวกันกับที่ ไอซี 6845 ควบคุมอยู่ด้วย ซึ่งไอซี 6845 ก็มีริจัสเตอร์หมายเลข 11 และ 12 สำหรับเก็บค่าเริ่มต้น. สุดท้ายของสแกนไลน์

ดังนั้น เราจึงเรียกริจัสเตอร์ของไอซี 6845 จากพอร์ทที่กำหนดในรวมไบออส มาเปิดเกอร์เซอร์ ดังนี้

- กำหนดหมายเลขพอร์ทตามแอดเดรส ที่เก็บในรวมไบออส

```
VPort : Word Absolute $0040 : $0063;
```

- กำหนดริจัสเตอร์ที่ต้องการจากพอร์ทที่ชี้โดยตัวแปร

```
Port [VPort] : = 10;
```

- เขียนค่าในริจัสเตอร์นั้น ตามที่กำหนด ม่านพอร์ทถัดไปที่ชี้โดยพอร์ท VPort

```
Port [VPort+1 : = HI(Cursor Mode) And $DF;
```

เช่น ถ้าเป็นจอโมโนโครม พอร์ทที่ถูกชี้โดยตัวแปร VPort คือ พอร์ทหมายเลข 3B4h ซึ่งเป็นพอร์ทกำหนดริจัสเตอร์ที่จะใช้งานตามหมายเลขของมัน และพอร์ทหมายเลข 3b5h จะใช้ในการเซ็ทค่าริจัสเตอร์ที่เลือก แต่สำหรับจอสี หมายเลขของพอร์ทจะเปลี่ยนไปเป็น 3D4h และ 3D5h ตามลำดับ แต่เราใช้หมายเลขของพอร์ทเหล่านี้จากพื้นที่ของรวมไบออส ดังนั้น เราจึงไม่ต้องยุ่งยากในเรื่องเหล่านั้น

และสำหรับการเปิดเคอร์เซอร์ โดยไม่แตะต้อง ขนาดหรือค่าสแกนไลน์เดิมนั้น ก็ทำได้โดยการรีเซ็ทบิทที่5 ของค่าเริ่มต้นสแกนไลน์ให้เป็น 0 นั่นคือ แอนด์มันด้วย DFh หรือ 1101 1111b นั้นเอง ซึ่งค่าเริ่มต้นของสแกนไลน์ ก็ได้มาจากพื้นที่ไงานของรวมไบออส ใน

ในตัวแปร Cursor Mode ซึ่งกำหนดไว้เป็น

```
Cursor Mode : Word Absolute $0040:$0060;
```

เพียงเท่านี้ ก็สามารถเปิดเคอร์เซอร์ได้แล้ว แต่โครงงานนี้ จะแสดงการติดต่อกับริจัสเตอร์หมายเลข 11 อีกตัวหนึ่ง ดังนี้

- เรียกริจัสเตอร์หมายเลข 11 ซึ่งใช้เก็บค่าสุดท้ายของการสแกนไลน์

```
Port [VPort] : = 11;
```

- เซ็กมันด้วยค่าเดิม

```
Port [VPort+1]:=Lo(Cursor Mode);
```

อนึ่งสำหรับการปิดเคอร์เซอร์ด้วยโปรซีเดียว Setcursor Off นั้น ก็มีการโปรแกรมคล้ายกัน จะต่างกันก็เพียงการเซ็ทบิทที่ 5 ของค่าเริ่มต้นของสแกนไลน์ให้เป็น 1 ก่อนที่จะเขียนมั่นคงริจิสเตอร์หมายเลข 10 ผ่านพอร์ทที่กำหนดในตัวแปร VPort นั่นคือ

```
Port [VPort+1] : = Hi (Cursor Mode) OR $20;
```

<u>การวาดกรอบหน้าต่าง</u> โปรซีเดียว Drarr Window ได้ถูกกำหนดไว้เป็น

```
Procedure Draw Window (X1, Y1, Y2, Style : Byte);
```

สำหรับวาดกรอบหน้าต่างตามพิกัดที่กำหนด แบบ Style โดยแบบนี้จะกำหนดไว้ในอาเรย์ 2 มิติ Style Win สำหรับเก็บชีกขระแอสกี้ แบบละ 8 ตัวอักษร เริ่มจากเส้นบน เส้นขอบซ้าย ขอบขวา เส้นล่าง มุมบนซ้าย มุมบนขวา มุมล่างซ้าย และมุมล่างขวา พร้อมเก็บแทนมันด้วยตัวคงที่จาก 1 ถึง 8 ตามอินเดทซ์ของอาเรย์ข้างต้น ในโปรซีเดียว Draw Window

ส่วนการทำงานของโปรซีเดียว จะใช้คำสั่ง GotoXY ในการเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ตั้งการ ก่อนที่จะเขียนแอสกี้ของกรอบที่ตำแหน่งนั้น ด้วยคำสั่ง write โดยการเขียนนี้ จะกำหนดแน่นอนไว้ในโปรแกรมว่า ถ้าเป็น

- มุมบนขวา, ซ้าย หรือมุมล่างขวา, ซ้าย จะแสดงอักขระเพียงตัวเดียว
- เส้นบน, ล่าง จะเขียนจากมุมซ้ายจนถึงมุมขวาของกรอบนั้น
- เส้นข้างซ้ายและขวา จะเขียนเส้นขอบซ้าย แล้วเลื่อนเคอร์เซอร์ด้วยคำสั่ง GotoXY ไปเขียนเส้นขอบขวา แต่สำหรับโปรซีเดียว Draw Window นี้ จะทำ

เพียงตีกรอบหน้าต่างเท่านั้น จะไม่เซ็ทหน้าต่าง ด้วยคำสั่ง GotoXY ของเทอร์โบปาสคาล เพื่อไม่ต้องยุ่งยากในการกำหนด Window อื่นอีก ในการตีกรอบหน้าต่างถัดไป

โปรซีเดียว Read Funckey จะทำหน้าที่อ่านฟังก์ชั่นคีย์ที่กด เพื่อเลื่อนแถบสร้าง ด้วยการใช้คำสั่ง Read Key จองเทอร์โบปาสคาลมาตรวจสอบว่าเป็นคีย์ที่ต้องการหรือไม่ ถ้าใช่ก็ส่งคีย์ที่ได้กลับผู้เรียกใช้ แต่ถ้าไม่ใช่จะวนลูปเพื่ออ่านคีย์ต่อไป

แต่สำหรับฟังก์ชั่น ReadKey จะกำหนดการทำงานไว้เป็น

```
Key : = ReadKey;
if Key = #0 then
Key : = ReadKey;
```

ในการแฟล็ทช์ที่ได้จากการกดคีย์ ออกจากคีย์บอร์ดบัฟเฟอร์นั้น ถ้าเป็นคีย์จำพวก printing Keys จำพวก "A","1" ฯลฯ จะแฟล็ทช์ เพียงครั้งเดียว ก็ได้รหัสแอสกี้ของตัวอักษรตามต้องการ แต่ถ้ากดคีย์พวก Non-Printing Keys จะต้องแฟล็ทช์ข้อมูลขึ้นมาอีกครั้งหนึ่งจึงจะได้ค่าสแกนโค้ดของคีย์ตามต้องการ

ที่เป็นเช่นนี้ เพราะฟังก์ชั่นในอินเทอร์รีพท์หมายเลข 9h จะแปลงสแกนโค้ดที่ได้เป็นรหัสแอสกี้ แล้วเก็บรหัสแอสกี้และสแกนโค้ดของคีย์ลงไปในคีย์บอร์ดบัฟเฟอร์ แต่ถ้าเป็นคีย์ Non-Printing Keys เช่น F1 หรือ Ctrl_C จะไม่มีรหัสแอสกี้ดังนั้นไบออสจะเก็บ null หรือ #0 จะไม่มีรหัสแอสกี้ ดังนั้นไบออสจะเก็บ null หรือ #0 ลงในไบท์แรก และตามด้วยรหัสสแกนโค้ดของคีย์นั้น ในไบท์ถัดไป ในคีย์บอร์ดบัฟเฟอร์ ทำให้เราต้องแฟล็ทช์มันออกมา ด้วย ReadKey ตามที่แสดงไว้

ฟังก์ชั่น ShiftChr

เป็นรูทีนภาษาแอสเซมบลี้ ซึ่งทดลองเขียนขึ้น เพื่อใช้แทนรูทีนของเทอร์โบปาสคาล สำหรับตัด space ในสตริงชื่อวัสดุก่อสร้างก่อนที่จะเขียนมันในกราฟแท่ง อันจะทำให้ชื่อวัสดุก่อสร้างแบ่งซ้าย วางอยู่ตรงกึ่งกลางแกน x ตามมาตร

โดยที่เราจะส่งสตริงชื่อวัสดุก่อสร้าง มาเป็นพารามิเตอร์ที่เก็บในตัวแปรโกลบอล มาให้ฟังก์ชั่น ฟังก์ชั่นจะนำเอาแอดเดรสของตัวแปรโกลบอล มาเก็บในพอยต์เตอร์ Src เพื่อเลื่อนอักขระของสตริงมาพิจารณาคราวละตัว จนกว่าจะเจอ space จึงส่งค่าหรือสตริงที่ถูกเลื่อนแล้วกลับโปรแกรมเทอร์โบปาสคาล เป็นแอดเดรสในเอ็กซ์ตร้าเซ็ดเมนท์ ที่เก็บสตริงนั้น ดังนี้

```
Src EQU DWORD PTR [BP+4]
Dest EQU DWORD PTR [BP+8]
```

พารามิเตอร์ Dest จะเป็นพอยต์เตอร์ที่ชี้หน่วยความจำ ณ ตำแหน่งที่ระบุใน BP+8

สำหรับการนำหน่วยความจำที่ชี้โดยตัวแปรทั้งสองมาใช้นั้น จะใช้รีจิสเตอร์คู่ DS:Si และ ES:D1 มาชี้ตำแหน่งแทน ดังนี้

```
Lds Sl,Src
Les Dl,Dest
```

แต่ก่อนที่จะเลื่อนสตริงจะถือเอาว่าสตริงนั้นไม่มี space ดังนั้น จึงเซ็ทสตริงด้านและปลายให้มีขนาดเท่ากัน

```
Mov CL, Ds :[Sl]
Mov ES:[Dl],CL
```

เหตุที่ต้องทำเช่นนี้ เพราะไบท์แรกของสตริงนั้น จึงต้องเซ็ทไบท์แรกของสตริงปลายทางไว้ก่อนนั่นเอง แต่จะเซ็ทมันใหม่ หลังจากที่เลื่อนจอ space แล้วตามตัวนับหรือริลิสเตอร์ Ch ในโปรแกรมแล้ว

แต่จากการที่ชื่อวัสดุก่อสร้าง มักจะมี spaceที่เกิดจากการเว้นวรรค เช่น Hollow Block ดังนั้น การเลื่อนอักขระมาพิจารณา จะต้องไม่ตัดสตริงหลังวรรคนี้ จึงต้องใช้รีจิสเตอร์ BL มานับ spaceเพื่อตัดส่วนของสตริงหลังจากเจอ space ติดกัน 2Space

และสำหรับการเลื่อนสตริงมาพิจารณานั้น จะโหลดอักขระของสตริงมาเก็บในรีจิสเตอร์ AL ก่อน ด้วยคำสั่ง

```
Lodsb
```

คำสั่งนี้ จะโหลดสตริงที่ชี้โดยรีจิสเตอร์คู่ DS:Sl มาเก็บในรีจิสเตอร์ AL แล้วเทียบค่าที่ได้กับ

```
Cmp AL,' '
```

ถ้าไม่ใช่ space ก็เพื่อตัวนับตัวอักษรซึ่งไม่ใช่ space ในริจัสเตอร์ CH ขั้นหนึ่ง พร้อมกับนำมันไปเก็บในหน่วยความจำที่ชี้โดยริจัสเตอร์คู่ ES; DI ดัง

```
Inc CH
Stosb
```

แต่ถ้าอักษรตัวนั้นเป็น space จะเพิ่มตัวนับ space ในริจัสเตอร์ BL เพื่อตรวจดูว่ามันนับถือ 2 space แล้วหรือยัง

```
Inc BL
Cmp BL, 2
```

ถ้ายังนับไม่ถือสอง จะวนลูปเลื่อนตัวอักษรถัดไป มาพิจารณาต่อ แต่ถ้านับได้ครบแล้ว จะทำการเซ็ทสตริงปลายทาง ที่ได้จากการตัด space เสียใหม่ตามขนาดจริงของมัน ดังนี้

- ลดขนาดของสตริงลงหนึ่ง เพื่อตัดspace แรกที่ตรวจพบออก

```
Dec CH
```

- นำขนาดของสตริง มาเก็บในริจัสเตอร์ CX

```
Mov CL, CH
Xor CH, CH
```

- เซ็ตริจิสเตอร์ DI เสียใหม่ให้มันเลื่อนไป ชี้ที่ไบท์เก็บขนาดของสตริง

```
Add CX, 2
Sub DI, CX
```

เหตุที่ต้องเลื่อนริจิสเตอร์ DI เพิ่มอีก 2 ไบท์จากขนาดสตริงริงของสตริงนั้น ก็เพราะขณะนี้มันชี้ไบท์ถัดจากไบท์ที่เก็บ 1 ไบท์ และไบท์เก็บขนาดสตริง จะอยู่ที่ไบท์ก่อนหน้าไบท์เก็บข้อมูล 1 ไบท์ด้วย ดังนั้นการที่จะเลื่อน DI จากไบท์ถัดจากไบท์แรกที่เก็บ space ไปยังไบท์เก็บขนาดของสตริง เราจึงต้องเลื่อนกลับเท่ากับ ขนาดสตริงเพิ่มอีก 2 ไบท์ นั่นเอง

- เซ็ทขนาดจริงของสตริง

```
Sub CX, 2
Mov ES : [DI], CL
```

แต่ก่อนที่จะกลับโปรแกรมหลักปาสคาลนั้น จะต้องสั่ง

RET 4

เพื่อให้คอมไพเลอร์ของเทอร์โบปาสกาล ทำการยกเลิกการใช้งานหน่วยความจำสแต็ก 4 ไบท์ เพราะที่เก็บแอดเดรสของโปรแกรมจุดที่กระโดดมาตามรีจิสเตอร์ CS, IP กับพารามิเตอร์ Src เท่านั้น โดยคำสั่งนี้จะเป็นการกำหนดให้รีจิสเตอร์ SP เลื่อนมาชี้ที่ตำแหน่งเก็บแอดเดรสของสตริงที่ได้ ตามที่ชี้มันด้วย Dest เพื่อส่งแอดเดรสนี้กลับผู้เรียกใช้ฟังก์ชั่นนี้ในโปรแกรมต่อไป

บทที่ 5

โปรแกรมช่วยเหลือ

โปรแกรมช่วยเหลือหรือ help จะใช้ในการแสดงคำอธิบายการทำงานของฟังก์ชั่นคีย์ที่กดครั้งหลังสุด ให้ผู้ใช้ได้เข้าใจการทำงานโดยโปรแกรมดีเบส จะตรวจสอบคีย์ F1 เพื่อเรียกไปยังโปรแกรม Db Helb Ein ที่ถูกเขียนและคอมไพล์ให้เป็นภาษาแอสเซมบลี้พร้อมกับส่งชื่อไฟล์ที่เก็บคำบรรยาย ของคีย์ที่ถูกกดครั้งหลังสุด ก่อนกดคีย์ F1 ให้โปรแกรม Db Help. Bin ทำการเซพจอภาพเดิมไว้ก่อนที่จะอ่านไฟล์ตามพารามิเตอร์ที่ส่งมาจากดีเบส เพื่อแสดงผลมันลงบนจอภาพ จากนั้นจะรอการกดคีย์ เมื่อผู้ใช้อ่านจบ จึงคืนจอภาพเดิมที่มันเซพไว้ก่อนที่จะกระโดดกลับไปยังโปรแกรมดีเบสเพื่อทำงานต่อไป

แต่ก่อนจะกล่าวถึงรายละเอียดของโปรแกรม Db Help. Bin จะขออธิบายถึงรายละเอียดและข้อกำหดในการเขียนภาษาแอสเซมบลี้เพื่อเชื่อมต่อกับดีเบสก่อนดังนี้

1. จะต้องกำหนด ORG ของโปรแกรมไว้ที่ 0 เสมอ
2. การส่งค่าผ่านตัวแปรในหน่วยความจำ จะใช้รีจิสเตอร์ DS:BX
3. หากมีการย้ายค่าเซทเมนต์รีจิสเตอร์ไปใช้งานอย่างอื่น ก่อนกลับโปรแกรมดีเบสนั้น จะต้องคืนค่าเดิมทุกครั้ง
4. กระบวนความหลักที่กำหนดให้กระโดดไปจากดีเบส จะต้องกำหนดให้เป็น PROC แบบ FAR อีกทั้งการกระโดดกลับจากโปรแกรมแอสเซมบลี้ จะกระโดดกลับด้วย RET แบบ FAR
5. ดีเบสจะกันหน่วยความจำสำหรับโหลดโปรแกรมแอสเซมบลี้มเก็บ มีขนาดไม่เกิน 32 กิโลไบท์ ดังนั้นโปรแกรมแต่ละโปรแกรมจะต้องมีขนาดไม่เกินนี้ แต่ถ้ามีหลายโปรแกรมย่อยที่จะโหลดลงหน่วยความจำ ก็สามารถทำได้ โดยเมื่อมีการโหลดโปรแกรมใหม่เข้ามาดีเบสจะโหลดมันทับโปรแกรมเดิม ในหน่วยความจำส่วนนั้นเสมอ

คำสั่งของดีเบส

การติดต่อกับโปรแกรมภายนอก ในไฟล์นามสกุล Bin นั้น ดีเบสทรีพลีสมีคำส่งใช้งาน 3 คำสั่ง คือ

LOAD (File Name)

เป็นคำสั่งที่สั่งให้ดีเบสพรีพลีส โหลดไฟล์ที่กำหนดในคำสั่ง ลงมาเก็บในหน่วยความจำ และถึงแม้ว่าดีเบสจะยอมให้โหลดโปรแกรมได้ถึง 5 โปรแกรมก็ตาม แต่มันจะโหลดโปรแกรมใหม่ทับโปรแกรมเดิมเสมอ ทั้งนี้เพราะดีเบสจะทำงานตามโปรแกรมได้ทีละโปรแกรมเท่านั้นและถ้าเราต้องการดูว่ามีไฟล์อะไรถูกโหลดเข้ามาบ้าง เราก็สามารถใช้คำสั่ง

DISPLAY STATUS

ตรวจสอบได้ และถ้ามีการโหลดไฟล์เดิมซ้ำอีก การโหลดจะนำเอาชื่อไฟล์ทับชื่อเดิมเสมอ

การโหลดดังที่กล่าวไว้ข้างต้น จะทำเพียงนำโปรแกรมในไฟล์ที่กำหนด มาเก็บไว้ในหน่วยความจำเท่านั้น และเมื่อต้องการวันโปรแกรมดังกล่าวก็ให้ใช้คำสั่ง

CALL (File Name) WITH (Parameter)

ดีเบสจะโอนการควบคุมให้กับโปรแกรมในไฟล์ที่กำหนดใน FileName ให้มันรันโปรแกรมตามพารามิเตอร์ที่กำหนด โดยการทำงานในส่วนของดีเบสนั้น มันจะเซ็ทรีจิสเตอร์ CS ให้ชี้ไปยังแอดเดรสของโปรแกรม พร้อมกับเซ็ทรีจิสเตอร์คู่ DS:BX ให้ชี้ไปยังแอดเดรสของพารามิเตอร์ โดยพารามิเตอร์นี้จะถูกดีเบสเก็บไว้ในรูป ASCIIZ หรือสตริงซึ่งปิดท้ายด้วยศูนย์

และเมื่อหมดความจำเป็นต้องใช้โปรแกรมที่โหลดมาแล้ว เราก็สั่งให้ดีเบสยกเลิกการใช้หน่วยความจำ ในส่วนที่โปรแกรมถูกโหลดมาเก็บไว้ด้วย

RELEASE (File Name)

และต่อไปนี้จะได้กล่าวถึงรายละเอียดของโปรแกรม DbHelp.ASM ของโครงงานนี้

ส่วนหลักของโปรแกรม

ในโปรแกรม DbHelp.Asm จะกำหนดโปรซีเดียว DbHelp ไว้เป็นส่วนหลักของโปรแกรม ซึ่งพิจารณาได้จาก

ก) มันเป็นเพียงโปรซีเดียวเดียวที่ได้กำหนดไว้เป็นแบบ FAR

ข) มันถูกกำหนดให้สามารถเรียกใช้ได้จากโปรแกรมภายนอก ตามที่ประเทศเป็น

ค) ถูกกำหนดให้เป็นโปรแกรมหลักใน END Db Help คำสั่งสุดท้าย

ง) มันเรียกใช้โปรซิเดียว่อยทั้งหมดของโปรแกรม

ดังนั้น เราจะมาว่ากันในรายละเอียดของโปรซีเดียวนี้กันก่อน ที่จะกั้นในโปรซีเดียว่อยที่มันเรียกใช้

- เปิดไฟล์ (ลาเบล Open_File) ตามพารามิเตอร์ ที่ส่งมาจากดีเบสด้วย ฟังก์ชั่น 21h ของอินเทอร์รีพท์หมายเลข 2 ซึ่งจะต้องใช้วีจิสเตอร์เก็บค่าดังนี้

AL = 000 ให้เปิดไฟล์เพื่ออ่านอย่างเดียว

= 001 เปิดไฟล์เพื่อเขียนอย่างเดียว

= 010 เปิดไฟลเพื่ออ่านและเขียน

DS: DX เซกเมนท์และอ็อฟเซ็ทแอดเดรสที่เก็บ ASCIIZ

และผลของการเปิดไฟล์ จะใช้แฟล็กตัวกดลอกผลการดำเนินการ ถือถ้ามันเป็นศูนย์ หมายถึงเปิดไฟล์ได้สำเร็จ ส่วนถ้าเป็นค่าอื่น ๆนั้นจะกล่าวถึงอีกครั้งหนึ่ง

แต่จากการที่พารามิเตอร์ที่เก็บชื่อไฟล์ เป็นแอดเดรสนั้นจะถูกดีเบสส่งมาด้วยริจิสเตอร์คู่ DS: BX ดังนั้น โปรแกรมส่วนของการเปิดไฟล์เพื่ออ่านจะเป็น

```
MOV   DX, BX
MOV   AX, 3Dooh  ; open file
INT   11h
```

จากนั้น จะตรวจสอบผลของการเปิดไฟล์ ด้วยการทดสอบแฟล็กตัวทด (Carry Flag)

```
JNC ReadF
```

ถ้ามันเป็นศูนย์ จะเซพ ASCIIZ ของชื่อไฟล์ (handle) ในริจิสเตอร์ AX ไว้ในตัวแปร Handle แต่ถ้าไม่เช่นนั้น จะกระโดดไปแสดงข้อความบอกความผิดพลาดจากการเปิดไฟล์

```
JMP   Error_Level_10
```

- การเคลียร์หน่วยความจำ(ลาเบล Clear_Buffer) ก่อนนำไฟล์ลงไป

เซพ ณ แอดเดรสของ Old_Buffer เป็นจำนวนไบท์ที่กำหนดในส่วนของดาต้าเซดเมนท์ โดยการทำงาน เริ่มต้นจะเก็บจำนวนไบท์ในริจิสเตอร์ CX พร้อมกับเซ็ทริจัสเตอร์ ES:DI ให้ชี้ไปยังแอดเดรสของ Old_Buffer

```
MOV  CX, 4000
MOV  AC, CS
MOV  ES, AX
LEA  DI, Old_Buffer
```

เพื่อเคลียร์มันด้วย Space

```
MOV  AX, 2020h
CLD
REP  STOSW
```

คำสั่ง STOSW จะนำแอสกี้ของ SPACE ในริจิสเตอร์ AX ไปเก็บ ณ แอดเดรสที่กำหนดด้วยริจิสเตอร์คู่ ES:DI เป็นจำนวนครั้งของการทำซ้ำ 4000 ครั้ง ตรงนี้จะสังเกตเห็นว่า Old_Buffer จะกำหนดได้เป็น

```
Old_Buffer DB 4000 DUP (?)
```

หน่วยความจำสำหรับเก็บข้อมูลอะไรก็ได้ในภายหลัง ขนาด 4000 ไบท์ แต่เราเคลียร์มัน 4000 ครั้งด้วยเรีร์ด AX มันก็ทำงานได้ไม่มีปัญหาอะไรและสำหรับ Cld จะใช้ในการเคลียร์แฟล็กทิศทาง (Clear direction flag) ให้คำสั่ง Stosw เก็บข้อมูลจากไบท์ต่ำไปสูง

- การอ่านไฟล์(ลาเบล Read_File) จะอ่านไฟล์ข้อมูลที่เปิดไว้มาเซพลงในหน่วยความจำของ Help_Buffer ด้วยฟังก์ชั่น 3Fh ของอินเทอร์รีพท์หมายเลข 21h โดยจะใช้ริจิสเตอร์

AH = 3Fh (Lead file)

BX = แฮนเดิลของไฟล์

CX = จำนวนไบท์ข้อมูลของไฟล์ที่จะอ่านเท่ากับจำนวนไบท์ของบัฟเฟอร์หน่วยความจำ

DS: DX = เซตเมนท์และอ๊อฟเซ็ทแอดเดรสของบัฟเฟอร์หน่วยความจำ

และผลของการดำเนินงาน จะเซ็ตแฟลิกตัวทดเป็นศูนย์ ถ้าการอ่านไฟล์สำเร็จ และริจิสเตอร์

AX = 0 ถ้าอ่านจนหมดไฟล์

= จำนวนไบท์ที่อ่านเข้ามา

= 5,6 ตามความผิดพลาดที่เกิดจากการอ่านไฟล์ไม่สำเร็จ

ดังนั้น คำสั่งในส่วนนี้จะเป็น

```
MOV  BX, Handle
LEA  DX, Help_Buffer
MOV  CX, Max_Size
MOV  AH, 3Fh
INT  21h
```

ตัวคงที่ Max_Size จะเก็บจำนวนไบท์สูงสุดเท่ากับจำนวนไบท์ของบัฟเฟอร์ 4000 ไบท์แต่ในไฟล์ข้อมูลที่อ่านมาจริง ๆ ไม่ใหญ่ขนาดนั้น และจะใช้แฟล็กตัวทด ทดสอบผลของการอ่านไฟล์ที่เปิด

```
JNC  CloseF
```

เพื่อกระโดดไปทำการปิดไฟล์นั้นต่อไป

และสำหรับกรณีที่เกิดข้อผิดพลาดเนื่องจากการอ่านไฟล์ จะทดสอบริจิสเตอร์ AL

เหมือนการเปิดไฟล์

```
MOV  Error_Code, AL
JMP  Error_Level_11
```

แต่กระโดดไปลาเบลต่างกัน

- การปิดไฟล์ (ลาเบล Close F) การปิดไฟล์นี้จะใช้ฟังก์ชั่น 3Eh ของ

อินเทอร์รีพท์หมายเลข 21h โดยใช้ริจิสเตอร์เพียง

AH = 3Eh (Close file)

BX = ไฟล์แฮนเดิล

คอสจะเคลียร์แฟล็กตัวทด ถ้าปิดไฟล์ได้เรียบร้อย ไม่เช่นนั้นแฟล็กตัวทดจะถูกเซ็ท และริจิสเตอร์ AX จะเก็บรหัสบอกข้อผิดพลาดซึ่งรหัสบอกข้อผิดพลาดจะมีเพียงกรณีเดียว คือกำหนดแชนเดิลของไฟล์ไว้ไม่ถูกต้อง และริจิสเตอร์ AX จะเท่ากับ 6 ดังนั้น โปรแกรมในส่วนนี้จะเป็น

```
MOV   BX, Handle
MOV   AH, 3Eh
INT   21h
CMP   AL, 6
JNE   DISPLAY
```

ถ้าปิดไฟล์ได้สำเร็ จะทำการแสดงข้อมูลในไฟล์ ไม่เช่นนั้น จะบอกข้อผิดพลาดของการกำหนดไฟล์แชนเดิลไม่ถูกต้องแทน

- การเซพสกรีนเดิมก่อนนำไฟล์ข้อมูลมาแสดงผล(ลาเบล Display) ส่วนนี้ของโปรแกรมจะเรียกใช้โปรซิเดียว่อยมาเตรียมแสดงผล เริ่มจาก

```
CALL   Vedio_Status
```

ทดสอบโหมดการแสดงผลของจอภาพ เพื่อจะ

```
CALL   Save_Old
```

เซพสกรีนข้อมูลดีเบส ที่กำลังแสดงผลปัจจุบันบนจอภาพ ตามแอดเดรสของส่วนเก็บข้อมูลของจอภาพ ("Screen Segment") ก่อนที่จะเคลียร์จอภาพนั้น เพื่อแสดงข้อมูลในบัฟเฟอร์ที่โหลดข้อมูลลงมาเก็บโดยมีริจิสเตอร์ S1

```
CALL Cwin
```

คอยชี้ไบท์ข้อมูลในบัฟเฟอร์ Help_Buffer นั้น

- การแสดงข้อมูลในบัฟเฟอร์ ในโครงงานนี้จะใช้ฟังก์ชั่นหมายเลขของอินเทอร์รัพท์หมายเลข 9h ซึ่งฟังก์ชั่นนี้ต้องใช้ริจิสเตอร์ 10h

```
AH =  9h (Write Character and attribute at cursor)
AL =  รหัสแอสกี้ของตัวอักขระที่จะให้แสดงผล
```

BH = หมายเลขประจำตัว (เพล)

BL = แอททริบิวท์ที่จะแสดงตัวอักษร

CX = จำนวนตัวอักษรที่จะเขียน

แต่ฟังก์ชั่นจะไม่มีการเลื่อนเดอร์เซอร์ให้หลังจากเขียนตัวอักษรเป็นแบบแอททริบิวท์ที่กำหนด ณ ตำแหน่งเดอร์เซอร์ ปัจจุบันแล้ว และสำหรับแอททริบิวท์ของข้อมูลบางค่า มีขนาดเกินกว่าจะเก็บได้ในไบท์เดียว ของเท็กซ์ไฟล์ เช่น แอททริบิวท์ ให้แสดงตัวอักษรแบบรีเวิร์สวีดีโอมีค่า ดังนั้น ถ้าเราสร้างเท็กซ์เก็บข้อมความอธิบายการทำงานของฟังก์ชั่นคีย์ และแอททริบิวท์ในลักษณะนี้ จึงไม่อาจทำได้ ทำให้เราต้องกำหนดแอททริบิวท์ในเท็กซ์ไฟล์ เป็นตัวอักษรซึ่งมีขนาดแน่นอน 1 ไบท์เช่นเดียวกับตัวอักษร และโครงงานนี้จะกำหนดตัวอักษรแทนแอททริบิวท์ไว้ดังนี้

I = Fh แทนแอทบริบิวตัวอักษรสีเข้ม (Interse)
M = 9h แทนแอททริบิวท์ตัวอักษรสีเข้มแซะปีดเส้นใต้ สำหรับเน้น (Mark)

ข้อความ

R = 70h แทนแอททริบิวท์ตัวอักษรแบบริเวิร์สวีดีโอ
U = 1h แทนแอททริบิวท์ตัวอักษรปีดเส้นใต้
N = 7h แทนแอททริบิวท์ตัวอักษรแบบปรกติ

แล้วให้โปรแกรมตรวจสอบอักษรแทนแอททริบิวท์นี้ พร้อมกับเซ็ทแอททริบิวท์ในวิจจิสเตอร์ ก่อนที่จะนำมันไปแสดงบนจอภาพ ตามฟังก์ชั่นที่กล่าวไว้บ้างต้น

การแสดงตัวอักษรนั้น จะทำโดยการเลื่อนเดอร์เซอร์ไปยังตำแหน่งที่ต้องการก่อน ที่จะเขียนตัวอักษรนั้นด้วยการใช้วิจิสเตอร์ DX เก็บตำแหน่งที่ต้องการให้เลื่อนแล้วเรียกใช้โปรซีเดียว Move_Cursor (จะกล่าวถึงอีกครั้งหนึ่ง) แต่ก่อนที่จะเขียนตัวอักษรนั้นจะตรวจสอบ

ก) เป็นรหัสแอสกี้ของเครื่องหมายเท่ากับ ("=") หรือไม่

```
MOV   AL, ocdh
CMP   BUTE PTR [S1], Double_Line
```

ถ้าใช่จะกระโดดไปเขียนเส้นคู่ ตามรหัสแอสกี้ OCDh ที่เก็บในริจิสเตอร์ AL สำหรับเหตุผลที่ต้องการโปรแกรมไว้ลักษณะนี้ก็เพราะ หนึ่งอักษรตัวแรกที่จะเขียนบนจอภาพคือเส้นคู่ และสองตัวอักษรกราฟฟิคของเส้นคู่ ไม่สามารถป้อนได้จากคีย์บอร์ด เพราะเป็นรหัสแอสกี้ส่วนขยาย Exfended ASC11) และสุดท้ายมันเป็นเพียงอักษรตัวเดียวที่ต้องแทนด้วยอักษาตัวอื่นในบรรดาตัวอักษรทั้งหมด ที่ให้แสดงข้อความอธิบายการทำงานของฟังกีชั่นคีย์ ดังนั้นจึงสะดวก

ที่จะกำหนดมันโดยตรงลงริจิสเตอร์ AL ก่อนที่จะตรวจสอบมัน

ข) เป็นรหัสแอสกี้ที่13ของ carritage หรือไม่

```
CMP BYTE PTR [SI], Carritage_Return
```

ถ้าใช่ให้กระโดดข้าม Carritage Return และ linefeed ด้วยการเพิ่มค่าในริจิสเตอร์ ที่ชี้ไบท์ข้อมูลในบัฟเฟอร์ขึ้นสองไบท์พร้อมกับตรวจสอบมันว่าเป็นคอลัมน์สุดท้าย (62) ของข้อความช่วยเหลือที่บรรทัดนั้น หรือไม่

```
ADD  S1, 2
CMP  DL, 62
```

ถ้าใช่ก็จะเลื่อนบรรทัดด้วยการย้ายเดอร์เซอร์มายังคอลัมน์แรกที่จะให้แสดงอักษรตัวถัดไป ในบรรทัดใหม่

```
INC  DH
MOV  DL, 17
CALL Move_Cursor
Mov  AL, [S1]
```

หรือไม่เช่นนั้นก็กระโดดไปแสดงตัวอักษรถัดไป โดยไม่ต้องเลื่อนบรรทัด และสำหรับเหตุผลที่ต้องทำเช่นนี้เพราะการที่เรากำหนดแอททริบิวท์ของตัวอักษรลงไปในเท็กซ์ไฟล์ ทำให้ขนาดหรือจำนวนตัวอักษรในแต่ละบรรทัดของเท็กซ์ไฟล์ เป็นสองเท่าของอักษรในบรรทัดที่นำมาแสดงผลบนจอภาพ ดังนั้นการเลื่อนบรรทัดของข้อความบนจอภาพ จึงพิจารณาตาม carritage return ของเท็กซ์ไฟล์ไม่ได้ จึงต้องพิจารณาจากตำแหน่งจริงบนจอภาพแทน

ค) ถ้าไม่ใช่ทั้ง ก. และ ข. มันเป็น Eof หรือไม่

```
CMP BYTE PTR [S1], End_Of_File
```

ถ้าใช่ก็จะจบการเขียนตัวอักษรและแอททริบิวท์จะกล่าวถึงอีกครั้งหนึ่ง

แต่ถ้าไม่ใช่ จะเก็บรหัสแอสกี้ของอักษาตัวนั้นใช้ในริจิสเตอร์ AL พร้อมกับเลื่อนให้ไปชี้ไบท์แอททริบิวท์ เพื่อนำมันมาเก็บในริจิสเตอร์ BL

```
MOV  AL, [S1]
INC  S1
MOV  BL,[S1]
```

แล้วจึงเทียบตัวอักษรแทนแอททริบิวท์ เพื่อเปลี่ยนให้มันเป็นค่าแอททริบิวท์ที่ถูกต้อง เช่น ทดสอบดูว่าต้องการแอททริบิวท์ตัวเข้มหรือไม่

```
CMP  BL,'I'
JNZ  Not_Intense
MOV  BL, 00001111b
```

ถ้าไม่ใช่ ก็กระโดดไปลาเบลถัดไป แต่ถ้าใช่จะเซ็ทแอททริบิวท์ที่ถูกต้องตามอักษรนั้น ลงริจิสเตอร์ BL ก่อนที่จะกระโดดไปเขียนตัวอักษรนั้น ด้วยการเรียกอินเทอร์รีพท์หมายเลข 10h จึงเลื่อน S1 ไปชี้อักษรตัวถัดไปที่จะนำมาแสดงผล ณ ตำแหน่งถัดไป ตามเคอร์เซอร์ที่ถูกเลื่อนตามริจิสเตอร์ DL จนกระทั่งจบไฟล์

ก็จะคอยการกดคีย์ CALL Wait_Key

เพื่อเคลียร์จอภาพ

```
CALL CWin
```

ก่อนที่จะนำข้อมูลดีเบสของสกรีนเดิม มาแสดงผลเพื่อทำงานตามฟังก์ชั่นในโปรแกรมดีเบส ที่เรียกมาอีกครั้ง

- การแสดงข้อผิดพลาด (ลาเบล Error_Level_10, Error_Level_11)

การแสดงข้อผิดพลาดนั้น จะนำเอารหัสบอกความผิดพลาดจากการเปิดไฟล์, อ่านไฟล์หรือบิลไฟล์ จากริจิสเตอร์ AL มาเป็นตัวกำหนดสตริงข้อความบอกข้อผิดพลาดที่เหมาะสมที่จะจัดมันมาแสดงผล ตัวอย่างเช่น ถ้าเกิดความผิดพลาดจากการปิดไฟล์ ริจิสเตอร์ AL จะมีค่าเป็น 6 ซึ่งถูกเก็บในตัวแปร Error_Code จะนำมันมาเซ็ทริจิสเตอร์ AL อีกครั้ง

```
MOV AL, Error_Cade
```

ก่อนที่จะทดสอบมันเพื่อกระโดดไปยังลาเบลที่เหมาะสม

```
CMP AL, 6
JZ  CODE_6
```

เพื่อโลหดแอดเดรสของสตริงบอกข้อผิดพลาดของรหัส 6

```
LEA  S1, Message_6
```

แล้วนำมันมาแสดงผลบริเวณด้านล่างของจอภาพ ด้วยฟังก์ชั่นหมายเลข 9h ของอินเทอร์รีพท์หมายเลข 21h ซึ่งเป็นฟังก์ชั่นสำหรับนำสตริงที่ปิดท้ายด้วยเครื่องหมายดอลล่าร์ (Dollar sign)"$" ที่ชี้โดยริจิสเตอร์ DX มาแสดงผลบนจอภาพ ณ ตำแหน่งของเคอร์เซอร์ปัจจุบัน ดังนั้น คำสั่งจึงเป็น

```
MOV  DX, S1
MOV  AH,9h     ; output character string
INT  21h
```

พร้อมกับแสดงสัญญาณเสียงบี๊บเตือน ด้วยการพยายามจะเขียนแอสกี้รหัสที่ 07h กับฟังก์ชั่นหมายเลข 2 ของอินเทอร์รัพท์ที่ 21h ดังนี้

```
MOV  DL, Bell
MOV  AH, 2     ; character output
INT  21h
```

แล้วจึงกระโดดกลับโปรแกรมดีเบส ณ ลาเบล ณ บาเบล EXIT ซึ่งเป็นจุดเดียวกับการกลับหลังจากแสดงข้อมูลในไฟล์ข้อมูลช่วยเหลือ, รอการกดคีย์, เคลียร์จอภาพ และคืนจอภาพ กรณีมีไฟล์ข้อมูลช่วยเหลือ นั่นเอง

- <u>การเลื่อนเคอร์เซอร์</u> (โปรซีเดียว Move_Cursor)

จากการที่ฟังก์ชั่น 9h ของอินเทอร์รัพท์หมายเลข 10h จะดำเนินการเพียงเขียนตัวอักษรตามแอททริบิวท์ที่กำหนด ณ ตำแหน่งเคอร์เซอร์ปัจจุบันเท่านั้น ไม่มีการเลื่อนตำแหน่งเคอร์เซอร์ไปยังคอลัมน์ถัดไปให้หลังจากที่แสดงตัวอักษรนั้นแล้ว ดังนั้นจึงจำเป็นต้องเลื่อนเคอร์เซอร์ไปยังคอลัมน์ถัดไป หรือคอลัมน์เริ่มต้นของบรรทัดใหม่ อีกทั้งตำแหน่งใด ๆ ตามต้องการ ด้วยฟังชก์ชั่นหมายเลข 2 ของอินเทอร์รีพท์หมายเลข 10h โดยฟังก์ชั่นนี้ จะกำหนดการใช้ริจิส

เตอร์ได้เป็น

AH = 02h (Set cursor position)

BH = หมายเลขประจำหน้า (เพล)

PH = บรรทัด

PL = คอลัมภ์

ดังนั้น ในการเลื่อนเคอร์เซอรไปยังตำแหน่งใหม่ จึงมีการกำหนดคำสั่งไว้เป็น

```
PUSH    AX
MOV     AH, 2
INT     10h
POP     AX
```

ค่าพิกัดคอลัมภ์และบรรทัด ในริจิสเตอร์ DL และ DH จะกำหนดไว้ก่อนที่จะเรียกใช้ฟังก์ชั่นนี้ และสำหรับริจิสเตอร์ AX จะเก็บหมายเลขฟังก์ชั่น ของการอินเทอร์รีพท์ตัวอื่น ดังนั้นเราจึงต้องเซพมันไว้ ก่อนที่จะให้มันเก็บหมายเลขของฟังก์ชั่นนี้ เพื่อได้นำมาเซ็ทหมายเลขฟังก์ชั่นใหม่นั่นเอง

- การเซพจอภาพ (โปรซีเดียว Save_Old)

ก่อนที่จะนำข้อความในไฟล์ช่วยเหลือมาแสดงผลบนจอภาพนั้น จำเป็นต้องเซพสกรีนเดิม ของข้อมูลดีเบสไว้ก่อน เพื่อที่ว่าหลังจากแสดงข้อความช่วยเหลือเรียบร้อยแล้ว จะได้นำเอาข้อมูลดีเบสที่เซพไว้นั้น มาแสดงบนจอภาพกลับอีกครั้งหนึ่ง เพื่อทำงานดีเบสต่อไปและสำหรับการเซพสกรีนเดิมนั้นจะนำเอาข้อมุลในแอดเดรส B000:0 หรือ B800:0 ตามการ์ดอะแดปเตอร์ของจอภาพโมโนโครมหรือจอภาพสีมาเก็บในบัฟเฟอร์ Old_Butter ขนาด 4000 ไบท์ตามจำนวนคอลัมภ์คูณบรรทัด ของจุดที่จะแสดงอักษรบนจอภาพทั้งหมด รวมถึงไบท์แอททริบิวท์ของพวกมันด้วย

สำหรับการโปรแกรมนั้น จะเซ็ทริจิสเตอร์คู่ DS:S1 ให้ชี้ไปยังส่วนเซดเมนท์ของจอภาพ เช่น จอโมโนโครมจะชี้ไปที่แอดเดรส B000:0 ซึ่งเซดเมนท์แอดเดรสจะเซ็ทในตัวแปร SCR_Seg (จะกล่าวถึงอีกที) และใช้ริจิสเตร์คู่ ES:D1 ชี้ไปยังบัฟเฟอร์ Old_Buffer

ดังนี้

```
MOV AX, SCR_Seg
MOV DS, AX
XOR S1,S1
LEA D1, Old_Buffer
```

จากนั้น จึงเซ็ทจำนวนไบท์ข้อมูลลงในริจิสเตอร์ CX พร้อมกับเคลียร์แฟล็คทิศทาง

```
MOR CX, Max_Size
CLD
```

ก่อนที่จะจะวนลูป เพื่อนำข้อมูลในเซคเมนท์จอภาพ มาเก็บในบัฟเฟอร์ Old_Buffer ตามจำนวนไบท์ที่กำหนดในริจิสเตอร์ CX ด้วยคำสั่ง

```
MOVSB
```

คำสั่งนี้ จะนำสตริงที่ชี้โดยริจิสเตอร์คู่ CS:S1 ไปเก็บในหน่วยความจำตามที่ริจิสเตอร์คู่ ES:D1 ชี้อกตำแหน่ง ในทิศทางเพิ่มจากแอดเดรสต่ำไปสูง(โดยการเพิ่มอีฟเซ็ทแอดเดรส ในริจิสเตอร์ S1, D1 ให้โดยอัตโนมัติ) คราวละไบท์ และจะมีการวนลูปซ้ำ ตามจำนวนไบท์ที่กำหนดในริจิสเตอร์ CX ผลของการทำงาน ก็เป็นการก๊อปปี้ข้อมูลจากสกรีนเดิม มาเก็บในบัพเฟอร์ที่กำหนด นั่นเอง

การคืนสกรีนเดิมของจอภาพ (โปรซีเดียว Dump)

ก็ทำตนคล้ายกัน แต่กลับทิศทางริจิสเตอร์คู่ที่ชี้หน่วยความจำ ทั้งสองพื้นที่ ดังนี้

```
MOR   AX, SCR_Seg
MOV   ES, AX
XOR   D1, D1
XOR   D1, D1
LEA   S1Old_Buffer
```

และจากโปรซีเดียวทั้งสองนี้ จะเห็นว่าตัวแปร SCR_Seg ชี้ไปยังเซทเมนท์ของจอภาพ ซึ่งหามาจาก

การกำหนดเซตเมนท์ของจอภาพ (โปรซิเดียว Video_Status)

ในรวมไบออสจะมีฟังก์ชั่นหมายเลข 15 ของอินเทอร์รีพท์หมายเลข 10h สำหรับทดสอบโหมดของการแสดงผลปัจจุบันโดยฟังก์ชั่นนี้ จะใช้รีจิสเตอร์

AH = 15 (Git current display made)

โดยมีค่าที่ส่งกลับมาในรีจิสเตอร์

AH = จำนวนตัวอักษรในหนึ่งบรรทัด (40/80)

AL = โหมดของการแสดงผล

BL = หมายเลขของเพลปัจจุบัน

และโหมดที่สำคัญ ๆ จะเป็น

AL = ooh สำหรับ 40×25 ขาดำ เท็กซ์ ของอะแคปเตอร์สี

AL = 01h สำหรับ 40×25 สี เท็กซ์ ของอะแคปเตอร์สี

AL = 02h สำหรับ 80×25 ขาวดำ เท็กซ์ ของอะแคปเตอร์สี

AL = 03h สำหรับ 80×25 สี เท็กซ์ ของอะแคปเตอร์สี

AL = 07h สำหรับ 80×25 ขาวดำ เท็กซ์ ของโมโนโครมอะแคปเตอร์

ส่วนโหมดอื่น ๆ จะเป็นกราฟฟิคโปรด ซึ่งไม่ใช่โหมดที่เราสนใจ

ดังนั้น จากค่าที่ส่งกลับมา เราจึงตรวจสอบมันโดยการตรวจสอบโหมดว่า มีค่าน้อยกว่า 7 หรือไม่ ถ้าใช่ก็เป็นโหมดของอะแคปเตอร์สี ถ้าไม่ใช่ก็เป็นโมโนโครมอะแคปเตอร์ โดยเริ่มต้น

- สมมุติว่าเป็นโหมดของอะแคปเตอร์สีก่อน จึงกำหนดเซทเมนท์ของจอภาพ ในตัวแปร SCR_Seg ชี้ไปที่

```
MOV SCR_Seg, Color_Seg
```

ตามตัวคงที่ Color_Seg

- ก่อนที่จะทดสอบนำโหมดที่แท้จริง

```
MOV   AH, 15
INT   10h
```

- แล้วจึงเก็บค่าที่ส่งกลับมาจากรวมไบออส ในริจิสเตอร์ที่กำหนด ไว้ในตัวแปร

```
MOV CRT_Mode, AL
MOV Current_Fage, BH
```

- ก่อนที่จะทดสอบมัน

```
SUB   AL, 6
```

ถ้าเท่ากับ 7 หรือเท็กซ์โหมดของโมโนโครมอะแคปเตอร์ ก็เซ็ทเซดเมนท์ของจอภาพเสียใหม่ ให้ชี้ไปยัง Boooh ตามตัวคงที่ Monochrome_Seg

```
MOV SCR_Seg, Monochrome_Seg
```

จากนั้น หรือถ้าเป็นโหมดของอะแคปเตอร์สีตามที่สมมุติไว้

- เก็บตำแหน่งเดิมของเคอร์เซอร์ไว้ เพื่อที่จะนำไปเซ็ทเคอร์เซอร์ใหม่ หลังจากที่คืนจอภาพข้อมูลของคีเบสแล้วโดยการนำฟังก์ชั่นหมายเลข 3 ของอินเทอร์รีพท์หมายเลข 10h ซึ่งฟังก์ชั่นนี้ จะส่งค่ากลับมาในริจิสเตอร์

DH = บรรทัด

DL = คอลัมน์ของเคอร์เซอร์ปัจจุบัน

เราจะเอาค่าเหล่านี้ มาเซพในตัวแปร Row และ Colume ตามลำดับ ดังนี้

```
MOV   AH, 3
INT   10h
MOV   ROW, DH
MOV   Colume, DL
```

<u>การเคลียร์จอภาพ</u> (โปรซีเดียว CWin)

ก่อนที่จะแสดงข้อความอธิบายโปรแกรม หรือก่อนที่จะนำเอาข้อมูลดีเบสมาแสดงมันกลับคืนบนจอภาพนั้น จะลบหรือเคลียร์ข้อมูลเดิมของจอภาพก่อน ด้วยฟังก์ชั่นหมายเลข 6 ของการ

อินเทอร์รีพท์หมายเลข 10h โดยฟังก์ชั่นนี้จะใช้รีจิสเตอร์

AH = 6 (Initialize window or scroll window contents up)

AL = จำนวนบรรทัดที่จะเลื่อน หรือ ถ้าต้องการเลื่อนหมดทั้งหน้าจอ ด้วยการเคลียร์มันออกไปเสียจากจอภาพก็กำหนดให้มันเป็นศูนย์

AL = แอททริบิวท์

CH, CL พิกัดบรรทัด, คอลัมน์มุมบนซ้าย

DH, DL พิกัดบรรทัด, คอลัมน์ของมุมล่างขวา ของพื้นที่ต้องการเลื่อนหน้าต่างขึ้น

อนึ่งสำหรับฟังก์ชั่นนี้ ถ้าต้องการเพียงกำหนดขอบเขตของหน้าต่าง จะต้องกำหนดรีจิสเตอร์ไว้เท่ากับ 1 และแอททริบิวท์จะกำหนดเหมือนแอททริบิวท์ของตัวอักษร

และสำหรับโปรซีเดียวนี้ ต้องการจะเคลียร์จอภาพทั้งจอภาพ ดังนั้น

```
XOR   AL, AL
MOV   AH, 6
MOV   CX, 0
MOV   DX, 1950h
MOV   BH, 7
INT   10h
```

มุมบนซ้ายจึงเป็น (0,0) และมุมล่างขวาจะกำหนดพิกัดเป็น (24,79) แต่พิกัดมุมล่างขวากำหนดไว้เป็น 1950h ในรีจิสเตอร์ DX ก็ไม่มีปัญหาอะไร

การรอการกดคีย์ (โปรดซีเดียว Wait_Key)

หลังจากที่แสดงผล ข้อความอธิบายการทำงานของฟังก์ชั่นคีย์ หรือบอกข้อผิดพลาดจากการเปิด, อ่านหรือปิดไฟล์จะรอให้ผู้ใช้ได้อ่านข้อความนั้นให้จบ ด้วยการเรียกใช้โปรซีเดียวนี้มาอ่านคีย์ใด ๆ ที่กด ก่อนที่จะเริ่มทำงานอื่นต่อไป

โปรซีเดียวนี้ จะใช้ฟังก์ชั่นหมายเลข 8 ของการอินเทอร์รีพท์หมายเลข 21h เพื่ออ่านคีย์ที่กด โดยไม่ต้องแสดงมันทางจอภาพ

ซึ่งโปรแกรมไว้เป็น

```
MOV AH, 8
INT 21h
```

สำหรับโปรแกรม Db Help. Asm นี้ได้ถูกรวมไว้ในภาคผนวกแล้ว

บทที่ 6

บทสรุป

จากข้อมูลวัสดุก่อสร้าง ที่ได้จากบริษัท กูมาไก จำกัด นั้น ได้นำมาเขียนโปรแกรมดีเบส เพื่อให้สามารถใช้งานได้จริง แต่เพิ่มความสะดวกต่อการใช้ ด้วยฟังก์ชั่นคีย์ต่าง ๆ จนเป็นที่น่าพอใจ อีกทั้งการที่มีคำอธิบายการทำงานของฟังก์ชั่นคีย์นั้นให้ จึงใช้ได้ง่ายขึ้น

และเมื่อต้องการพิจารณาปริมาณการเข้า-ออกของวัสดุก่อสร้างในสต็อคนั้น ก็สามารถพิจารณาได้จากกราฟแบบต่าง ๆ จึงเห็นว่า โปแกรมในโครงงานนี้สามารถประยุกต์ใช้งานกับการจัดเก็บวัสดุก่อสร้างในสต็อค ได้เหมาะสมพอสมควร

อนึ่ง สำหรับผลการทำงานของโปรแกรมในโครงงานนี้ ผู้จัดทำได้ใช้โปแกรมสำเร็จรูป ชื่อ Pizazz ของบริษัท Application Techniques มาพิมพ์ผลบนจอภาพ ออกทางเครื่องพิมพ์ เพื่อให้ผู้อ่านเห็นภาพ ได้เหมือนกับภาพบนจอภาพทุกประการ แต่สำหรับกราฟนั้น ผู้จัดทำได้ทดลองพิมพ์หลายครั้ง ก็ไม่สำเร็จ จึงต้องขออภัยมา ณ ที่นี้ด้วย

กิตติกรรมประกาศ

การทำงานใด ๆ ก็ตามล้วนแต่ต้องมีอุปสรรคในการทำงานด้วยกันทั้งสิ้น แต่ทั้งนี้หากเรามีผู้ให้คำปรึกษาแนะนำที่ดี ช่วยเหลือให้คำแนะนำต่าง ๆ ย่อยทำให้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการทำงาน สามารถถูกแก้ไขให้ลุล่วงได้ การทำวิทยานิพนธ์นี้ก็เช่นกัน คงจะไม่สำเร็จลุล่วงไปได้ด้วยดี ถ้าหากขาดซึ่งคำแนะนำและความช่วยเหลือจากบุคคลต่าง ๆ

ดังนั้นในนามของผู้จัดทำ จึงขอขอบคุณอาจารย์ทุกท่าน ที่เคยสั่งสอนวิชาความรู้ ตลอดจนให้คำปรึกษาแนะนำในการทำวิทยานิพนธ์ครั้งนี้ โดยเฉพาะอย่างยิ่ง อาจารย์บวรวรรณ สาริกภูติ ผู้ซึ่งช่วยเหลือเป็นอย่างมากในการทำวิทยานิพนธ์นี้ ทั้งด้านคำสั่ง, การใช้งานและอัลกอลิซึ่มของโปรแกรม ต้องขอขอบพระคุณเป็นอย่างสูง มา ณ ที่นี้

หนังสืออ้างอิง

1. พงษ์ระพี เดชพาหพงษ์. "แอดวานส์เอ็มเอสดอส". บริษัท ซีเอ็ดยูเคชั่น จำกัด. 300 หน้า. 2533
2. ยืน คู่วรวรรณ. "แอดวานซ์ดีเบส". บริษัท ซีเอ็ดยูเคชั่น จำกัด. 251 หน้า. 2533
3. ยืน คู่วรวรรณ. "โปรแกรมคอมพิวเตอร์ dBASE III PLUS" บริษัท ซีเอ็ดยูเคชั่น จำกัด. 354 หน้า. 2532
4. สุรศักดิ์ สงวนพงษ์. "เทคนิคการเขียนโปรแกรมชั้นสูง แอดวานซ์เทอร์โบปาสคาล เวอร์ชั่น 4.0". บริษัท ซีเอ็ดยูเคชั่น จำกัด. 200 หน้า. 2532
5. หัสรังษี ศิริวิมลวรรณ. อภิชาติ พัฒนไพรสณฑ์. อาทิตย์ จิตต์จุฬานนท์. วรีศักดิ์ วิทวัสกุล. "เทคนิคการเขียนโปรแกรมภาษาแอสเซมบลี้ สำหรับเครื่อง IBM PC", บริษัท ซีเอ็ดยูเคชั่น จำกัด. 326 หน้า. 2533
6. ยืน คู่วรวรรณ. สุรศักดิ์ สงวนพงษ์. "โปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี้ 8086/8088". บริษัทซีเอ็ดยูเคชั่น จำกัด. 202 หน้า. 2531
7. สุรชาติ พ่วงพุ่ม. "โครงสร้างข้อมูล กับ ปาสคาล". สำนักพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ". 256 หน้า. 2531
8. "ไมโครคอมพิวเตอร์". สำนักพิมพ์ซีเอ็ดยูเคชั่น จำกัด. ฉบับที่ 66,67 และ 68. 2534
9. "Turbo Pascal V. 5.0", บริษัท บอร์แลนด์ จำกัด. 844 หน้า. 2533
10. Douglas Hergert, "Mastering Turbo Pascal 5", Tech Publications, 630 p, 2533
11. Ben Ezzell, "Graphics Programming In Turbo C 2.0", บริษัท บอร์แลนด์ จำกัด. 568 หน้า. 2532