บทที่ 5

การออกแบบและพัฒนาโปรแกรม

<u>คุณสมบัติของโปรแกรม</u>

1. สามารถอ่านและเขียนแฟ้มข้อมูลโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 (PFB) เฉพาะโปรแกรมแบบอักษรปรกติได้อย่างถูกต้อง ไม่สนับสนุนโปรแกรมแบบอักษรแบบ hybrid หรือ synthetic

2. สามารถอ่านและเขียนแฟ้มข้อมูลความกว้างตัวอักษรและช่องไฟ (PFM) เพื่อให้สามารถนำแฟ้มข้อมูลโปรแกรมแบบอักษรโพสต์สคริปต์ที่ได้ไปใช้กับซอฟต์แวร์เอทีเอ็มบนไมโครซอฟต์วินโดวส์ได้

3. โปรแกรมที่ออกแบบและพัฒนาจะสามารถทำงานได้กับ

3.1 เครื่องไมโครคอมพิวเตอร์ไอบีเอ็มหรือแบบเดียวกับไอบีเอ็ม ที่มีไมโครโปรเซสเซอร์เบอร์ 80286 ขึ้นไป และมีหน่วยความจำตั้งแต่ 2 เมกะไบต์

3.2 ซอฟต์แวร์ไมโครซอฟต์วินโดวส์รุ่น3.xโดยจะต้องทำงานในภาวะสแตนดาร์ดหรือเอ็นฮานซ์เท่านั้น

3.3 ซอฟต์แวร์เอทีเอ็มรุ่น 1.1 ขึ้นไป

3.4 จอภาพและวงจรแสดงผลแบบวีจีเอ ที่สามารถแสดงสีได้ 16 สีขึ้นไป

3.5 ต้องมีเมาส์และแผงแป้นอักขระ

4. สนับสนุนการใช้ความสามารถพิเศษของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ดังนี้

4.1 การกำหนดและเรียกใช้โปรแกรมย่อยแบบ Subrs

4.2 การกำหนดสเตมและอไลเมนท์โซน

4.3 การกำหนดและเรียกใช้ตัวอักษรแบบแอ็คเซนต์

5. ไม่สนับสนุนการใช้คำสั่งและความสามารถพิเศษดังนี้

5.1 การเปลี่ยนสเตมภายในตัวอักษร

5.2 การกำหนดและเรียกใช้เฟลกซ์

5.3 การกำหนดและเรียกใช้คำสั่ง dotsection

5.4 การกำหนดสเตม 3 ระดับ(คำสั่ง hstem3 และ vstem3)

โดยทั้งนี้แฟ้มข้อมูลที่กำหนดและเรียกใช้คำสั่งเหล่านี้จะถูกแปลงโดยโปรแกรมในขณะที่อ่านแฟ้มข้อมูลให้กลายเป็นคำสั่งพื้นฐานอื่นๆหรือยกเลิกคำสั่งนี้

6. เครื่องมือที่โปรแกรมจะมีให้ใช้ ประกอบด้วย
   6.1 การลากเส้นตรงและเส้นโค้ง
   6.2 การเปลี่ยนรูปร่างของเส้น
   6.3 การเลือกส่วนของตัวอักษรเพื่อเปลี่ยนแปลง
   6.4 การย่อ-ขยาย หมุน โย้และการย้าย
   6.5 ไม้บรรทัดสำหรับวัดระยะทางและมุม
   6.6 การย่อ-ขยายขนาดการแสดงผล
   6.7 การลบ การคัดลอกกับคลิปบอร์ด เฉพาะรูปแบบข้อมูลของโปรแกรมที่จะสร้างขึ้นเท่านั้น

<u>เครื่องมือที่ใช้</u>

1. เครื่องไมโครคอมพิวเตอร์ไอบีเอ็มหรือไอบีเอ็มคอมแพททิเบิล ที่มีไมโครโปรเซสเซอร์เบอร์ 80286 ขึ้นไป มีหน่วยความจำตั้งแต่ 2 เมกะไบต์ และมีหน่วยขับจานแม่เหล็กขนาด 1.2(5-1/4") หรือ 1.44(3-1/2") เมกะไบต์ 1 หน่วย

2. ซอฟต์แวร์ไมโครซอฟต์วินโดวส์รุ่น 3.x โดยจะต้องทำงานในภาวะสแตนดาร์ดหรือเอ็นฮานซ์เท่านั้น

3. ซอฟต์แวร์เอทีเอ็มรุ่น 1.1 ขึ้นไป

4. จอภาพและวงจรแสดงผลแบบวีจีเอ ที่สามารถแสดงสีได้ 16 สีขึ้นไป

5. ไมโครซอฟต์เมาส์และแผงแป้นอักขระ

6. ซอฟต์แวร์ไมโครซอฟต์วินโดวส์ไรท์ รุ่น 3.1 เป็นซอฟต์แวร์ที่ใช้ทดสอบการพิมพ์ตัวอักษร

7. เครื่องพิมพ์แบบจุดและเครื่องพิมพ์แบบเลเซอร์

8. ภาษาซีโดยใช้ซอฟต์แวร์ไมโครซอฟต์ควิกซีฟอร์วินโดวส์ รุ่น 1.0 เป็นตัวแปลภาษา

การออกแบบโปรแกรม

1. การออกแบบจอภาพ

จอภาพเป็นส่วนแสดงผลหลักที่สามารถย่อ-ขยายและเคลื่อนย้ายได้ โดยแบ่งออกเป็น 8 ส่วนสำคัญ(ดูรูปที่ 5.1)ดังนี้

1.1 ส่วนแสดงผลตัวอักษรโครงร่าง เป็นส่วนที่แสดงให้เห็นถึงตัวอักษรแบบโครงร่างที่ผู้ใช้สามารถใช้เมาส์เพื่อทำการแก้ไข ตกแต่ง ออกแบบได้ตามความต้องการ นอกจากนี้ยังแสดงผลของค่าพารามิเตอร์ต่างๆ เช่น ค่าของอไลเมนท์โซนที่กำหนด จอภาพในส่วนนี้จะมีแท่งเลื่อนเพื่อให้สามารถเลื่อนการแสดงผลไปทางซ้าย-ขวา บน-ล่างและสามารถย่อ-ขยายขนาดการแสดงผลได้

1.2 แท่งรายการเลือก เป็นส่วนที่ใช้แสดงรายการเลือกของโปรแกรม

1.3 ไตเติลบาร์ เป็นส่วนที่ใช้แสดงชื่อโปรแกรมและชื่อแฟ้มข้อมูลโปรแกรมแบบอักษรที่กำลังแก้ไข

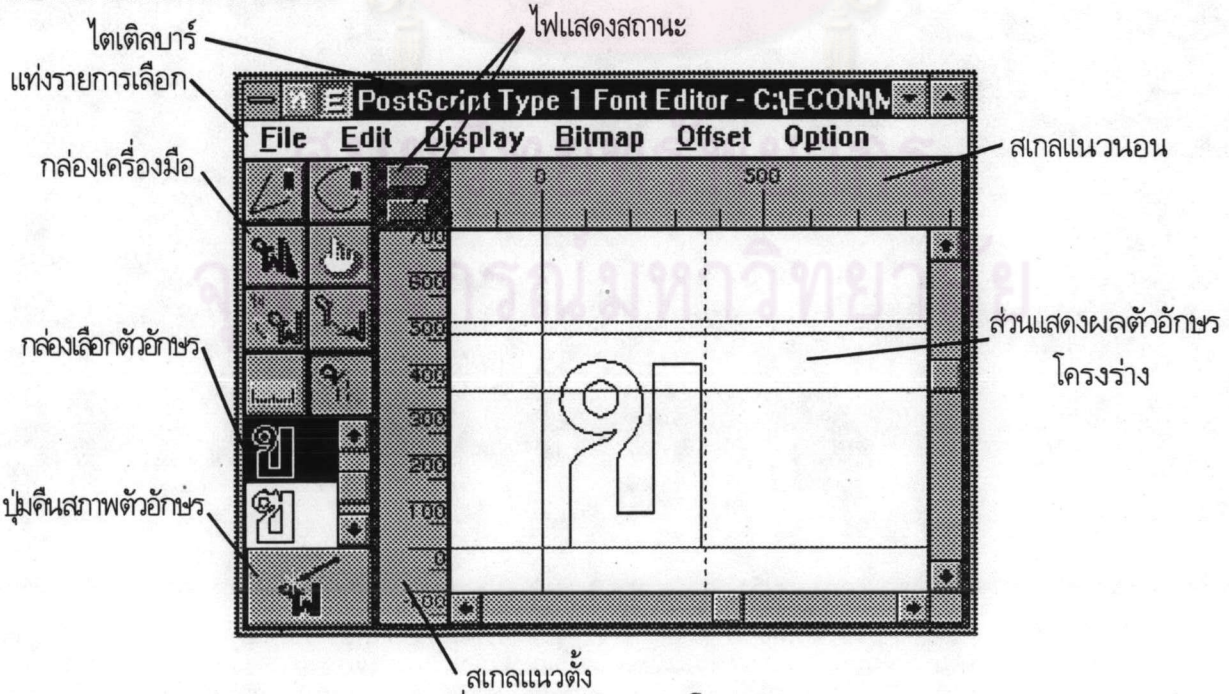
1.4 กล่องเครื่องมือ เป็นส่วนที่รวบรวมเครื่องมือต่างๆที่โปรแกรมจัดเตรียมไว้ให้ผู้ใช้เลือกใช้ โดยแสดงผลเป็นสัญญลักษณ์

1.5 กล่องเลือกตัวอักษร เป็นส่วนที่ใช้แสดงตัวอักษรต่างๆของแฟ้มข้อมูลโปรแกรมแบบอักษรที่ต้องการแก้ไข โดยผู้ใช้สามารถใช้เมาส์เลือกตัวอักษรใดตัวอักษรหนึ่งจากส่วนนี้เพื่อแก้ไขรูปร่าง ในส่วนนี้จะมีแท่งเลื่อนเพื่อให้สามารถเลื่อนการแสดงผลขึ้น-ลงได้

1.6 ปุ่มคืนสภาพตัวอักษร เป็นส่วนที่ใช้ยกเลิกตัวอักษรที่ได้แก้ไขแล้วให้กลับสู่สภาพเดิม

1.7 สเกลแนวนอนและแนวตั้ง เป็นส่วนที่แสดงสเกลของส่วนแสดงผลหลัก และผู้ใช้สามารถใช้เมาส์กดและลากเส้นสเกลเพื่อให้ปรากฏในส่วนแสดงผลหลักได้

1.8 ไฟแสดงสถานะ เป็นส่วนที่บ่งบอกถึงสถานะของการแก้ไขตัวอักษรว่าเป็นการแก้ไขตัวอักษรปรกติหรือตัวอักษรที่เป็นโปรแกรมย่อย



รูปที่ 5.1 แสดงจอภาพของโปรแกรม

2. การออกแบบตัวประสานกับผู้ใช้

การออกแบบในส่วนนี้จะเป็นไปตามรูปแบบการเขียนโปรแกรมบนไมโครซอฟต์วินโดวส์ ดังนี้

2.1 ระบบรายการเลือก เป็นแบบรายการเลือกแบบดึงลง ผู้ใช้สามารถใช้แผงแป้นอักขระหรือเมาส์เลือกรายการเลือก และจะมีแป้นลัด สำหรับบางรายการเลือกที่มีการใช้บ่อย

2.2 กล่องเครื่องมือ เป็นปุ่มกดที่รวบรวมเครื่องมือสำคัญที่โปรแกรมได้เตรียมไว้ โดยต้องใช้เมาส์ในการเลือก

2.3 กล่องเลือกตัวอักษร เป็นกล่องที่รวบรวมตัวอักษรแบบโครงร่างของแฟ้มข้อมูลโปรแกรมแบบอักษรที่กำลังแก้ไข สำหรับให้ผู้ใช้ใช้เมาส์เพื่อเลือกตัวอักษรที่ต้องการแก้ไขได้

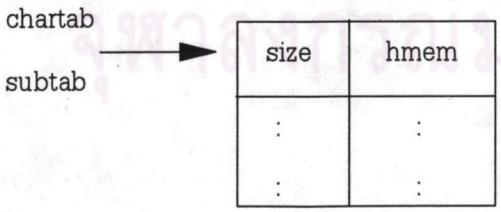
2.4 กล่องคำโต้ตอบ สำหรับการติดต่อกับผู้ใช้ในการรอรับค่าพารามิเตอร์หรือแสดงข้อมูลสำคัญ

3. การออกแบบโครงสร้างข้อมูล

ส่วนสำคัญที่สุดของการสร้างกราฟิกโปรแกรมก็คือการออกแบบโครงสร้างข้อมูลที่ดี มีความเหมาะสมตรงกับความต้องการใช้ ซึ่งจะทำให้โปรแกรมทำงานได้อย่างรวดเร็วและมีประสิทธิภาพ โครงสร้างข้อมูลสำคัญที่ออกแบบเพื่อใช้กับโปรแกรม มีดังนี้

3.1 โครงสร้างข้อมูล chartab และ subtab

เป็นโครงสร้างข้อมูลสำคัญที่ใช้เก็บข้อมูลส่วน charstring ของตัวอักษรในโปรแกรมแบบอักษรที่อ่านจากแฟ้มข้อมูลหรือที่แก้ไขแล้ว โดยเป็นตัวชี้ที่ชี้ไปยังตัวแปรแถวลำดับที่ใช้เก็บค่าแฮนเดิล(handle)ที่กำหนดตำแหน่งของหน่วยความจำที่เก็บ charstring ซึ่งอยู่ในรูปแบบคำสั่งโพสต์สคริปต์ประเภทที่ 1 ที่ยังไม่ได้เข้ารหัสลับรวมทั้งขนาดของ charstring นั้นๆ โดยขนาดของตัวแปรแถวลำดับที่ chartab ชี้อยู่ จะมีขนาด 256 แถว ใช้เป็นที่เก็บ charstring ของตัวอักษรต่างๆ ในขณะที่ตัวแปรแถวลำดับที่ subtab ชี้อยู่นั้น จะมีขนาดเริ่มต้นตามจำนวนของโปรแกรมย่อยของแฟ้มข้อมูลโปรแกรมแบบอักษรนั้นๆแต่สามารถเพิ่มหรือลดขนาดได้ตามการกำหนดของผู้ออกแบบแก้ไขโปรแกรมแบบอักษร(โดยการเรียกใช้ฟังก์ชัน LocalReAlloc ของวินโดวส์เพื่อเพิ่ม/ลดขนาดตามต้องการ) โครงสร้างข้อมูลสามารถแสดงได้ดังนี้(ดูรูปที่ 5.2)



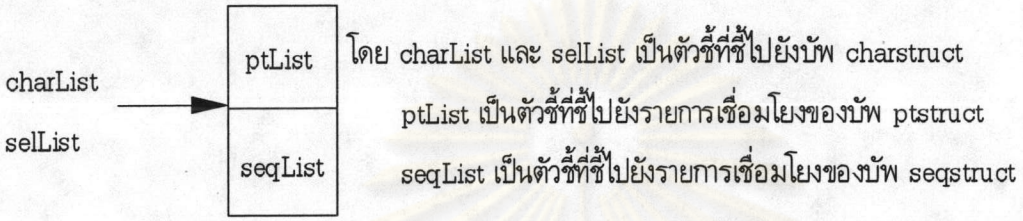
โดย size คือขนาดของ charstring

hmem คือค่าแฮนเดิลของตำแหน่งหน่วยความจำที่เก็บ charstring

รูปที่ 5.2 แสดงโครงสร้างข้อมูลของ chartab และ subtab

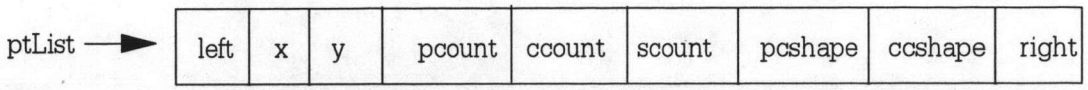
3.2 โครงสร้างข้อมูล charstruct

เป็นโครงสร้างส่วนที่สำคัญที่สุด เนื่องจากเป็นโครงสร้างข้อมูลที่ใช้เก็บข้อมูลคำสั่งการบรรยายตัวอักษรแบบโครงร่างเพื่อแสดงผลในส่วนแสดงผลหลัก ซึ่งการประมวลผลจะเป็นการประมวลผลคำสั่งและพารามิเตอร์ที่มีการแทรก ลบ แยก จึงจำเป็นที่จะต้องใช้โครงสร้างข้อมูลแบบรายการเชื่อมโยง โดยมีโครงสร้างดังนี้(ดูรูปที่ 5.3)



รูปที่ 5.3 แสดงโครงสร้างข้อมูลของ charstruct

โครงสร้างข้อมูลของบัพ ptstruct จะเชื่อมต่อกันเป็นรายการเชื่อมโยง ptList แบบรายการโยงคู่ โดยจะใช้เก็บข้อมูลของคู่ลำดับxและyต่างๆที่ใช้สำหรับการบรรยายการสร้างตัวอักษรนั้นนอกจากนี้จะเก็บข้อมูลจำนวนของบัพ cmdstruct ที่ใช้ค่าพิกัด x และ y ของบัพนี้ ทั้งนี้ก็เพื่อใช้เป็นค่าบ่งบอกถึงจำนวนของคำสั่งที่ใช้คู่ลำดับนี้(pcount ccount และ scount) นอกจากนั้นยังใช้ในการแสดงผลว่าณ.คู่ลำดับ x และ y นั้น เป็นจุดปลายของเส้นที่เป็นจุดเปิด (เส้นไม่บรรจบกัน)หรือจุดปิด(เส้นบรรจบกัน) และสำหรับในการแสดงผลเมื่อผู้ใช้เลือกเครื่องมือสำหรับการเปลี่ยนรูปร่างจะแสดงผลจุดปลายของเส้นและจุดควบคุมความโค้งของเส้นโค้งเฉพาะเส้นที่เลือกไว้เท่านั้น ดังนั้นจึงต้องมีจำนวนนับสำหรับจุดปลายและจุดควบคุมของเส้นที่เลือกไว้แยกต่างหากด้วย รายละเอียดของโครงสร้าง ptstruct แสดงได้ดังนี้(ดูรูปที่ 5.4)



โดย left เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ก่อนหน้า

x เป็นค่าพิกัดในแนวแกน x

y เป็นค่าพิกัดในแนวแกน y

pcount เป็นจำนวนของบัพ cmdstruct ที่ใช้ค่าพิกัด x และ y ของบัพนี้เป็นจุดปลายของเส้นตรงและเส้นโค้ง

ccount เป็นจำนวนของบัพ cmdstruct ที่ใช้ค่าพิกัด x และ y ของบัพนี้เป็นจุดควบคุมความโค้งของเส้นโค้ง

scount เป็นจำนวนของบัพ cmdstruct ที่ใช้ค่าพิกัด x และ y ของบัพนี้ เป็นจุดเริ่มต้นของการเรียกใช้โปรแกรมย่อย

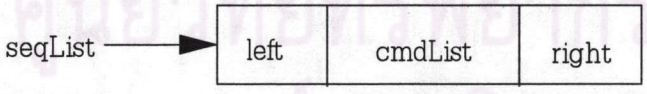
pcshape เป็นจำนวนของบัพ cmdstruct ที่ถูกเลือกและใช้ค่าพิกัด x และ y ของบัพนี้เป็นจุดปลายของเส้นตรงและเส้นโค้ง

ccshape เป็นจำนวนของบัพ cmdstruct ที่ถูกเลือกและใช้ค่าพิกัด x และ y ของบัพนี้เป็นจุดควบคุมความโค้งของเส้นโค้ง

right เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ถัดไป

รูปที่ 5.4 แสดงโครงสร้างข้อมูลของ ptstruct

โครงสร้างข้อมูลของบัพ seqstruct จะเชื่อมต่อกันเป็นรายการเชื่อมโยง seqList แบบรายการโยงคู่ โดยจะใช้บ่งบอกถึงรายการเชื่อมโยง cmdList ที่ประกอบขึ้นจากบัพ cmdstruct ที่แทนเส้นต่างๆที่เชื่อมต่อกัน ซึ่งหมายความว่ารายการเชื่อมโยง cmdList ของแต่ละบัพ seqstruct จะแทนกลุ่มของเส้นของตัวอักษรที่ไม่ต่อเนื่องกัน รายละเอียดของโครงสร้าง seqstruct แสดงได้ดังนี้(ดูรูปที่ 5.5)



โดย left เป็นตัวชี้ที่ชี้ไปยังบัพ seqstruct ก่อนหน้า

cmdList เป็นตัวชี้ที่ชี้ไปยังรายการเชื่อมโยง cmdList

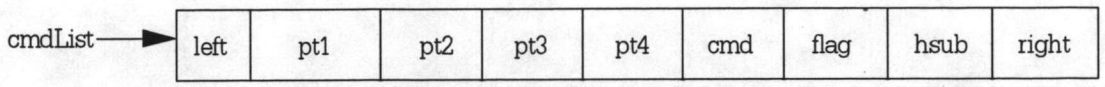
right เป็นตัวชี้ที่ชี้ไปยังบัพ seqstruct ถัดไป

รูปที่ 5.5 แสดงโครงสร้างข้อมูลของ seqstruct

โครงสร้างข้อมูลของบัพ cmdstruct จะเชื่อมต่อกันเป็นรายการเชื่อมโยง cmdList แบบรายการโยงคู่หมุนวนทั้งนี้เพื่อให้การเข้าหาบัพต้นและท้ายสำหรับตรวจสอบว่าเป็นกลุ่มของเส้นที่เชื่อมต่อกันหมดหรือไม่ และจะสามารถทำการเชื่อมต่อกับกลุ่มใหม่ของอีก seqstruct ให้กลายเป็นกลุ่มเดียวกันได้หรือไม่ เป็นไปอย่างรวดเร็ว โครงสร้างข้อมูลของบัพ cmdstruct นี้ ถือว่าเป็นส่วนสำคัญของโปรแกรม เพราะว่าเป็นส่วนที่ใช้เก็บข้อมูลคำสั่ง

ที่ใช้ในการบรรยายลักษณะของตัวอักษร เช่น คำสั่งให้ลากเส้นตรง เส้นโค้งตลอดจนการเรียกใช้โปรแกรมย่อย โดยจุดคู่ลำดับของคำสั่งเหล่านี้จะแทนด้วยตัวชี้ที่ชี้ไปยังบัพ ptstruct ที่เก็บคู่ลำดับ x และ y สาเหตุที่ต้องออกแบบเช่นนี้ ก็เพื่อลดความซ้ำซ้อนของการจัดเก็บคู่ลำดับ เพราะว่าลักษณะของตัวอักษรแบบโครงร่างนั้น จะมีจุดคู่ลำดับซ้ำกันที่จุดปลายและจุดเริ่มเสมอ นอกจากนั้นยังอาจจะมีการใช้จุดคู่ลำดับร่วมกันของจุดควบคุมความโค้งของเส้นโค้งอีกด้วย นอกจากนี้ยังใช้ช่วยในการแสดงผลจุดปลายของเส้นว่าปิดหรือเปิด แสดงจุดควบคุมความโค้งของเส้นโค้งเมื่อผู้ใช้เลือกเครื่องมือการเปลี่ยนแปลงรูปร่าง และช่วยให้การทำงานของฟังก์ชันที่ช่วยในการลากเส้นให้เชื่อมต่อกับจุดปลายของเส้นง่ายขึ้น รวมทั้งยังช่วยเพิ่มความเร็วของโปรแกรมโดยลดระยะเวลาที่ใช้ในการคำนวณจุดใหม่สำหรับฟังก์ชันการเปลี่ยนแปลง เช่น การย้าย การหมุนเป็นต้น

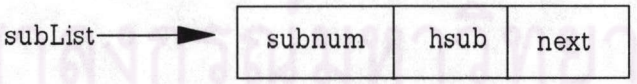
นอกจากนี้จะต้องมี flag เพื่อใช้บอกสถานะของบัพว่าเป็นบัพที่ถูกเลือก(flag=0)หรือไม่เลือก(flag=1)ตามความต้องการของผู้ใช้ และ hsub เป็นตัวที่บ่งบอกว่าบัพนั้นๆเป็นบัพปรกติ(ค่าเป็น 0)หรือเป็นบัพของโปรแกรมย่อย(ค่ามากกว่า 0) โดยการเก็บค่าแฮนเดิลของโปรแกรมย่อย สาเหตุที่ต้องใช้ค่าแฮนเดิลแทนที่จะใช้หมายเลขของโปรแกรมย่อยโดยตรงก็เพราะว่าโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 นั้น กำหนดให้หมายเลขของโปรแกรมย่อยเริ่มจาก 0 เสมอ และไม่มีข้อจำกัดในจำนวนของโปรแกรมย่อย การใช้หมายเลขของโปรแกรมย่อยจะต้องเตรียมเนื้อที่ไว้ให้มากๆ เพื่อที่จะสามารถรองรับกับแฟ้มข้อมูลโปรแกรมแบบอักษรที่มีโปรแกรมย่อยมากๆได้ นอกจากนี้ในแต่ละตัวอักษรอาจมีการเรียกใช้โปรแกรมย่อยเดียวกันหลายๆครั้งได้ การใช้หมายเลขของโปรแกรมย่อยเป็นตัวบ่งบอกจะเกิดปัญหาความยุ่งยากในการแยกแยะโดยเฉพาะอย่างยิ่งกรณีที่นำโปรแกรมย่อยเดียวกันมาเชื่อมต่อกันเป็นรายการเชื่อมโยงcmdListเดียวขั้นตอนการทำงานจะยุ่งยากเช่นในการเลือกส่วนของโปรแกรมย่อยส่วนใดส่วนหนึ่งจะต้องทำการเลือกบัพ cmdstruct ทุกบัพที่เป็นโปรแกรมย่อยนั้นด้วย ถ้าใช้หมายเลขของโปรแกรมย่อยจะต้องตรวจสอบว่าพบคำสั่ง CALL หรือยัง หรือพบบัพ cmdstruct ที่ไม่ใช่โปรแกรมย่อยจึงหยุดการทำงาน ในขณะที่การใช้ค่าแฮนเดิลก็จะตรวจสอบแค่ค่าแฮนเดิลว่าเปลี่ยนหรือยังเท่านั้น ดังนั้นจึงกำหนดให้ใช้ค่าแฮนเดิลแทน ซึ่งการเรียกใช้โปรแกรมย่อยแต่ละครั้งจะให้ค่าแฮนเดิลไม่ซ้ำกัน โดยในการจัดการค่าแฮนเดิลและหมายเลขของโปรแกรมย่อยนั้น จะจัดเก็บในรูปของรายการเชื่อมโยง subList สำหรับรายละเอียดของโครงสร้าง cmdstruct แสดงได้ดังนี้(ดูรูปที่ 5.6)



โดย left เป็นตัวชี้ที่ชี้ไปยังบัพ cmdstruct ก่อนหน้า
pt1 เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ที่เก็บคู่ลำดับที่เป็นจุดเริ่มต้น
pt2 เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ที่เก็บคู่ลำดับที่เป็นจุดควบคุมความโค้งที่ 2 ของเส้นโค้ง
pt3 เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ที่เก็บคู่ลำดับที่เป็นจุดควบคุมความโค้งที่ 3 ของเส้นโค้ง
pt4 เป็นตัวชี้ที่ชี้ไปยังบัพ ptstruct ที่เก็บคู่ลำดับที่เป็นจุดสิ้นสุดของเส้น
cmd เป็นรหัสของคำสั่งพื้นฐานมีค่าได้ดังนี้
1 คำสั่ง LINE (ลากเส้นตรง)
2 คำสั่ง CURVE (ลากเส้นโค้ง)
3 คำสั่ง CALL (เรียกใช้โปรแกรมย่อย)
flag เป็นค่าที่ใช้บอกสถานะของบัพนี้ว่าถูกเลือกหรือไม่ (0=ไม่เลือก, 1=เลือก)
hsub เป็นค่าแฮนเดิลของโปรแกรมย่อย
right เป็นตัวชี้ที่ชี้ไปยังบัพ cmdstruct ถัดไป

รูปที่ 5.6 แสดงโครงสร้างข้อมูลของ cmdstruct

สำหรับโครงสร้างข้อมูลของบัพ substruct จะเชื่อมต่อกันเป็นรายการเชื่อมโยง subList แบบรายการโยงเดี่ยว โดยจะใช้เก็บข้อมูลหมายเลขของโปรแกรมย่อยที่เรียกใช้กับค่าแฮนเดิลของโปรแกรมย่อย แนวทางการให้ค่าของแฮนเดิลสำหรับโปรแกรมย่อยนั้น จากการศึกษาโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 ที่มีอยู่ในปัจจุบัน พบว่าแต่ละตัวอักษรจะมีการเรียกใช้โปรแกรมย่อยไม่มากนัก ดังนั้นเพื่อความสะดวกและง่ายต่อการออกแบบ จึงจำกัดค่าของแฮนเดิลสูงสุดไว้ที่ 255 โดยค่าของแฮนเดิลจะเริ่มจาก 1 และเพิ่มทีละ 1 เมื่อมีการใช้โปรแกรมย่อย วิธีการให้ค่าแฮนเดิลแบบนี้ จะมีปัญหากรณีที่มีการเรียกใช้และยกเลิกโปรแกรมย่อยมากๆ หรือเรียกใช้โปรแกรมย่อยมากถึง 255 โปรแกรม ก็จะทำให้ไม่สามารถให้ค่าแฮนเดิลและใช้งานโปรแกรมต่อไปได้ (ซึ่งโอกาสที่จะเกิดกรณีนี้ขึ้นมีน้อยมาก) รายละเอียดของโครงสร้าง substruct แสดงได้ดังนี้(ดูรูปที่ 5.7)

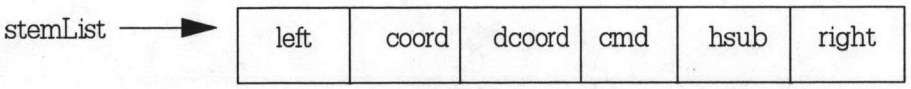


โดย subnum คือหมายเลขของโปรแกรมย่อย
hsub คือค่าแฮนเดิลของโปรแกรมย่อย
next คือตัวชี้ที่ชี้ไปยังบัพ substruct ถัดไป

รูปที่ 5.7 แสดงโครงสร้างข้อมูลของ substruct

3.3 โครงสร้างข้อมูล stemstruct

เป็นโครงสร้างข้อมูลที่เชื่อมต่อกันเป็นรายการเชื่อมโยง stemList แบบรายการโยงคู่ ที่ใช้เก็บข้อมูลตำแหน่งและความกว้างสเตมของตัวอักษร โดยมีโครงสร้างดังนี้(ดูรูปที่ 5.8)



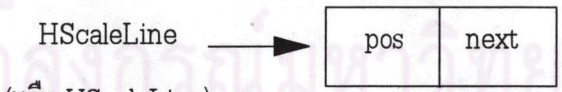
โดย left คือตัวชี้ที่ชี้ไปยังบัพ stemstruct ก่อนหน้า
coord คือค่าพิกัดของ x หรือ y ที่เป็นตำแหน่งเริ่มต้นของสเตม
dcoord คือค่าความกว้างของสเตม
cmd คือรหัสคำสั่งซึ่งมีค่าได้ดังนี้
0 หมายถึงสเตมแนวนอน
1 หมายถึงสเตมแนวตั้ง
2 หมายถึงการเรียกใช้โปรแกรมย่อยของสเตม
hsub คือค่าแฮนเดิลของโปรแกรมย่อย
right คือตัวชี้ที่ชี้ไปยังบัพ stemstruct ถัดไป

รูปที่ 5.8 แสดงโครงสร้างข้อมูลของ stemstruct

สาเหตุที่ต้องแยกข้อมูลของสเตมออกจากโครงสร้างข้อมูล charstruct ซึ่งเป็นโครงสร้างหลักที่ใช้ในการบรรยายการสร้างตัวอักษรนั้น ทั้งนี้ก็เพราะว่าโปรแกรมที่พัฒนาขึ้น จะไม่สนับสนุนการเปลี่ยนแปลงสเตมภายในตัวอักษร ดังนั้นการกำหนดสเตมจะไม่มีความสัมพันธ์กับตำแหน่งของส่วนของตัวอักษรที่เรียกใช้สเตม การแยกโครงสร้างออกมาจะทำให้การออกแบบและพัฒนาโปรแกรมทำได้ง่าย และนอกจากนี้เพื่อให้การปรับปรุงโปรแกรมในอนาคต มีความสามารถในการกำหนดสเตมโดยอัตโนมัติโดยที่ผู้ใช้ไม่ต้องกำหนดเอง สามารถทำได้โดยง่ายโดยไม่ต้องเปลี่ยนแปลงโครงสร้างของ charstruct อีก

3.4 โครงสร้างข้อมูล scalestruct

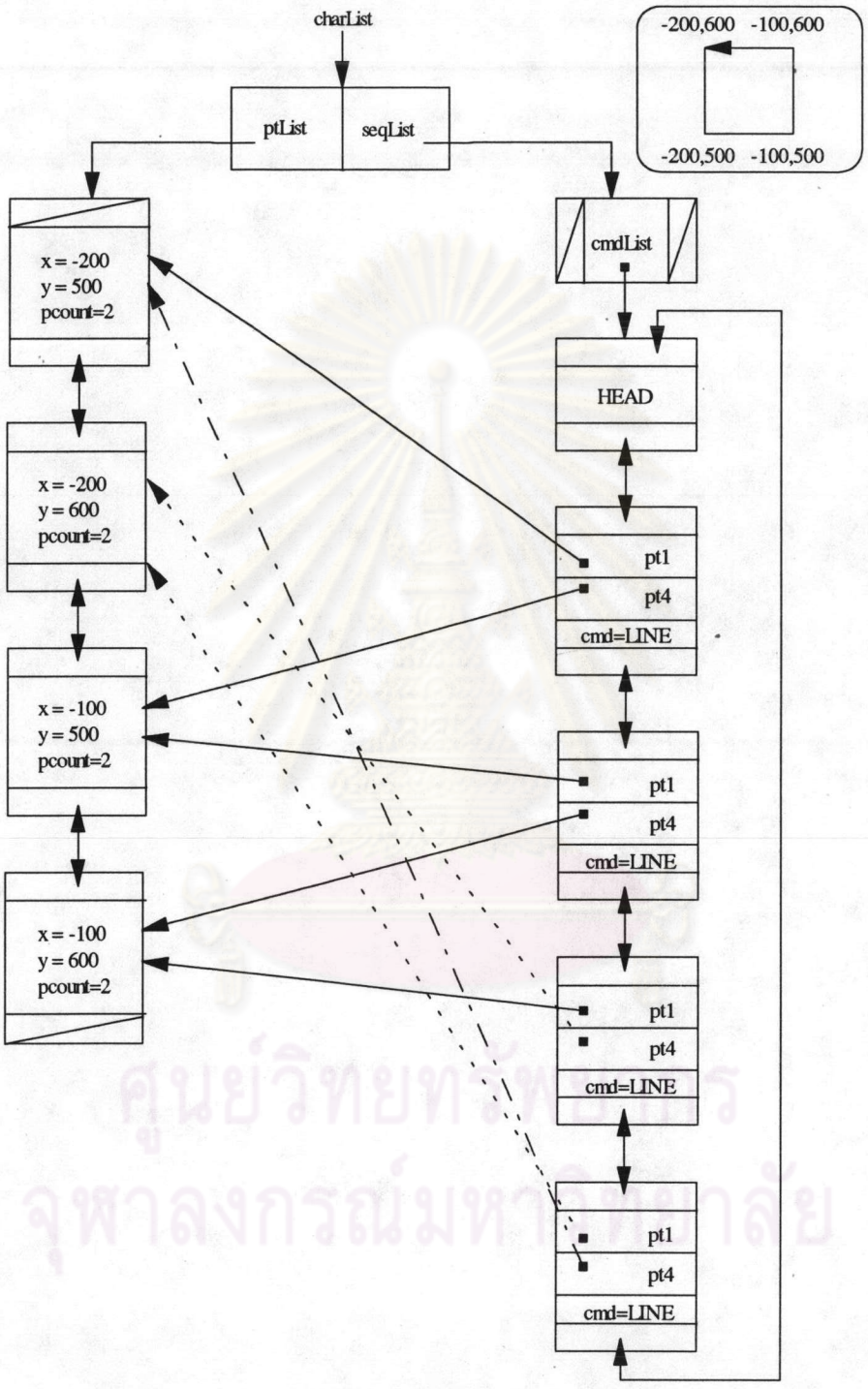
เป็นโครงสร้างข้อมูลที่เชื่อมต่อกันเป็นรายการเชื่อมโยง HScaleLine หรือ VScaleLine แบบรายการโยงเดี่ยว วัตถุประสงค์เพื่อใช้เก็บพิกัดของตำแหน่งเส้นสเกลตามที่ผู้ใช้กำหนด โดยมีโครงสร้างดังนี้(ดูรูปที่ 5.9)



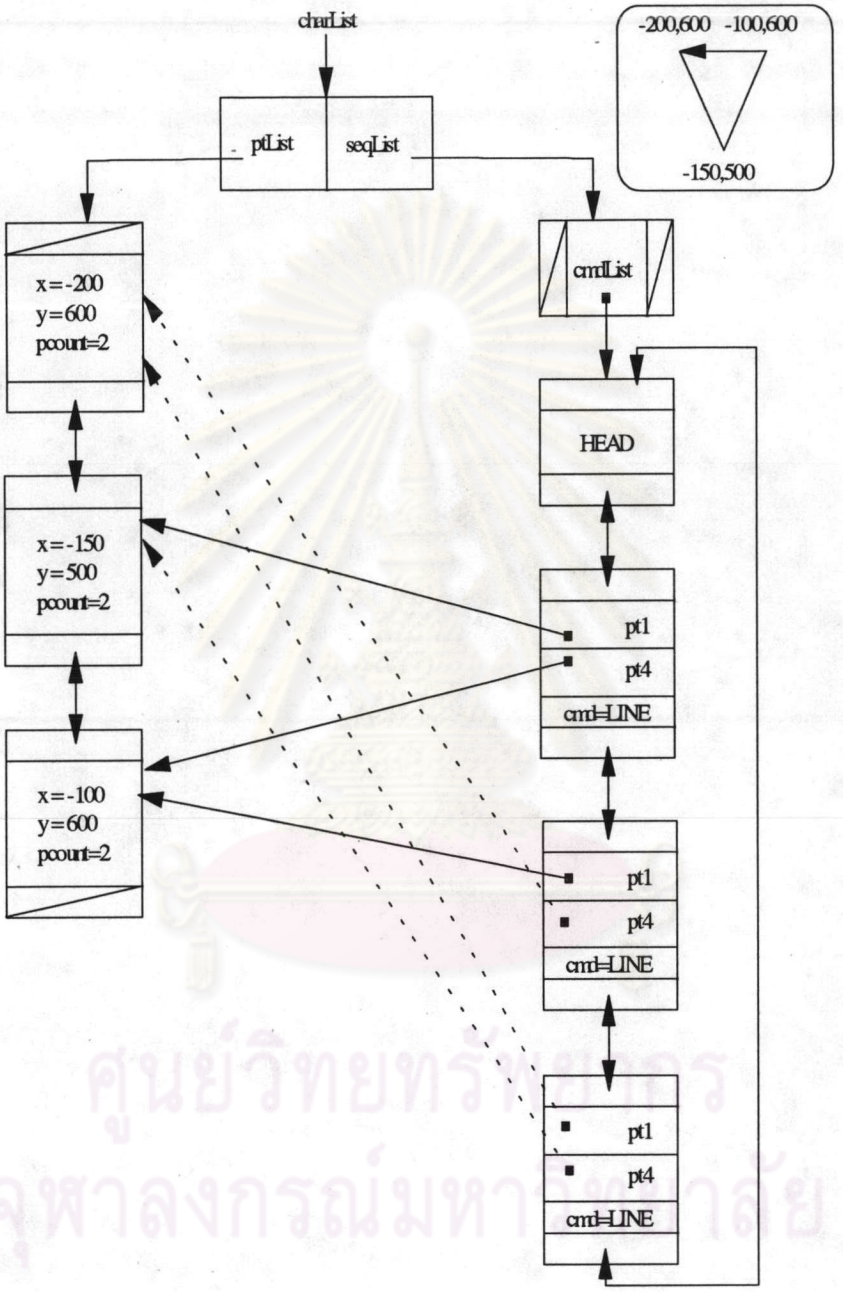
โดย pos คือพิกัดในแนวแกน x หรือ y ที่เป็นตำแหน่งของเส้นสเกล
next คือตัวชี้ที่ชี้ไปยังบัพ scalestruct ถัดไป

รูปที่ 5.9 แสดงโครงสร้างข้อมูลของ scalestruct

ตัวอย่างรายการเชื่อมโยงของโครงสร้างข้อมูล charstruct แสดงดังรูปที่ 5.10 และเมื่อมีการเปลี่ยนแปลง เช่น ลบเส้นใดเส้นหนึ่งออกแล้ว รายการเชื่อมโยงจะมีการเปลี่ยนแปลงไปดังรูปที่ 5.11



รูปที่ 5.10 แสดงตัวอย่างรายการเชื่อมโยงของโครงสร้างข้อมูล charstruct



รูปที่ 5.11 แสดงตัวอย่างรายการเชื่อมโยงของโครงสร้างข้อมูล charstruct เมื่อเปลี่ยนรูปร่างของตัวอักษร

4. การออกแบบข้อความและการประมวลผล

โปรแกรมที่สร้างขึ้นจะประกอบด้วย window procedure หลักและข้อความที่ประมวลผลดังตารางที่ 5.1

ตารางที่ 5.1 แสดงข้อความและการประมวลผลของวินโดว์ต่างๆ

ข้อความ	การประมวลผล
1. WndProc	
WM_COMMAND	จัดการเกี่ยวกับการเลือกรายการเลือกของผู้ใช้รวมทั้งการเลือกเครื่องมือของกล่องเครื่องมือ
WM_CREATE	สร้างวินโดว์ลูกของกล่องเครื่องมือ ส่วนแสดงผลหลัก สเกลแนวตั้งและแนวนอน กล่องเลือกตัวอักษร
WM_DRAWITEM	วาดภาพกล่องเครื่องมือและภาพเครื่องมือ
WM_SIZE	คำนวณขนาดของวินโดว์ลูก
WM_INITMENUPOPUP	จัดสถานะของรายการเลือกหลักและรายการเลือกย่อย
WM_TIMER	ปรับสถานะของไฟแสดงภาวะการแก้ไข
WM_PAINT	วาดภาพไฟแสดงภาวะการแก้ไข
WM_QUIT	จัดเก็บแฟ้มข้อมูลแบบอักษรที่แก้ไข แฟ้มข้อมูลสถานะของโปรแกรมและการทำลายวินโดว์ต่างๆ
2. DBoxProc	
WM_CREATE	กำหนดขนาดเริ่มต้นของส่วนแสดงผลหลัก
WM_SIZE	คำนวณขนาดเชิงตรรกของส่วนแสดงผลหลัก กำหนดช่วงของแท่งเลื่อน
WM_LBUTTONDOWN	หาตำแหน่งของเมาส์ ปรับสถานะการแสดงผลระยะทางและมุม - กรณีเลือกการเปลี่ยนรูปทรง จะหาเส้นที่ต้องการเปลี่ยน - กรณีเลือกการเลือก จะหาว่าเป็นการเลือกเส้นสเกลหรือไม่ - กรณีเลือกการเปลี่ยนรูปร่าง จะหาว่าเป็นการเปลี่ยนแบบใด
WM_LBUTTONUP	ปรับสถานะการแสดงผลระยะทางและมุม - กรณีเลือกการเพิ่มเส้นตรง เส้นโค้งหรือการเปลี่ยนรูปทรง จะเพิ่มบัพ cmdstruct เข้าไปใน charList - กรณีเลือกการเลือก ถ้าเป็นการเลือกเส้นสเกลจะเพิ่มบัพ scalestruct เข้าไปใน HScale หรือ VScale แต่ถ้าเป็นการเลือกส่วนของตัวอักษรจะกำหนดค่า flag ไว้ที่บัพ cmdstruct ที่เลือก - กรณีเลือกไม้บรรทัด จะลบเส้นไม้บรรทัด - กรณีเลือกการย่อ-ขยาย คำนวณขนาดเชิงตรรก ส่ง WM_PAINT ไปยัง

ตารางที่ 5.1 แสดงข้อความและการประมวลผลของวินโดว์ต่างๆ(ต่อ)

ข้อความ	การประมวลผล
	DBoxProc, HScaleProc, VScaleProc
WM_MOUSEMOVE	ปรับสถานะการแสดงผลระยะทางและมุม วาดภาพตามการเคลื่อนที่ของเมาส์
WM_VSCROLL	จัดการเกี่ยวกับแท่งเลื่อนและเลื่อนส่วนแสดงผลทางแนวตั้ง
WM_HSCROLL	จัดการเกี่ยวกับแท่งเลื่อนและเลื่อนส่วนแสดงผลทางแนวนอน
WM_PAINT	วาดภาพโครงร่างของตัวอักษร เส้นต่างๆที่เกี่ยวข้องเช่น เส้นสเกล
3. CSBoxProc	
WM_CREATE	สร้างวินโดว์ลูกสำหรับกล่องเลือกตัวอักษรและปุ่มคืนสภาพตัวอักษร
WM_SIZE	คำนวณขนาดของวินโดว์ใหม่
WM_COMMAND(lParam)	
BN_PAINT	วาดภาพปุ่มคืนสภาพตัวอักษร
BN_HILITE	วาดภาพการกดปุ่มคืนสภาพตัวอักษร
BN_UNHILITE	วาดภาพการปล่อยปุ่มคืนสภาพตัวอักษร
WM_LBUTTONDOWN	หาจุดที่กดปุ่มซ้ายของเมาส์
WM_LBUTTONUP	จัดเก็บตัวอักษรที่แก้ไข สร้าง charList ของตัวอักษรที่เลือก ส่ง WM_PAINT ไปยัง DBoxProc และ IDM_BMPUPDATE ให้ DBmpProc
WM_SIZE	คำนวณขนาดและจำนวนตัวอักษรที่จะแสดงในกล่องเลือกตัวอักษร
WM_VSCROLL	กำหนดช่วงแท่งเลื่อนแนวตั้ง
WM_PAINT	วาดภาพตัวอักษรในกล่องเลือกตัวอักษร
4. HScaleProc	
WM_LBUTTONDOWN	กำหนดให้ข้อความของเมาส์ส่งมาที่ HScaleProc เท่านั้น
WM_LBUTTONUP	เพิ่มบัพ scalestruct เข้าไปใน HScale
WM_MOUSEMOVE	วาดภาพเส้นสเกลตามการเคลื่อนที่ของเมาส์ในส่วนแสดงผลตัวอักษรโครงร่าง
WM_PAINT	วาดภาพสเกลแนวนอน
5. VScaleProc	
WM_LBUTTONDOWN	กำหนดให้ข้อความของเมาส์ส่งมาที่ VScaleProc เท่านั้น
WM_LBUTTONUP	เพิ่มบัพ scalestruct เข้าไปใน VScale
WM_MOUSEMOVE	วาดภาพเส้นสเกลตามการเคลื่อนที่ของเมาส์ในส่วนแสดงผลตัวอักษรโครงร่าง
WM_PAINT	วาดภาพสเกลแนวตั้ง

5. การออกแบบขั้นตอนวิธี

ขั้นตอนวิธีของโปรแกรมที่สำคัญแบ่งออกได้เป็น 3 กลุ่ม ดังนี้

5.1 ขั้นตอนการทำงานเกี่ยวกับการจัดการรายการเชื่อมโยงที่เก็บคำสั่งการบรรยายตัวอักษร

5.1.1 การเพิ่มบัพใหม่เข้าไปยังรายการเชื่อมโยงหลัก charList จะทำโดยทำการเพิ่มบัพที่ต้องการเข้าไปยังรายการเชื่อมโยง selList ก่อน แล้วจึงทำการรวมรายการเชื่อมโยง selList เข้ากับรายการเชื่อมโยง charList ดังนี้

จากขั้นตอนวิธี COMBINE_SEL_2_CHARLIST จะทำการรวมรายการเชื่อมโยง ptList ของ selList เข้าไปยัง รายการเชื่อมโยง ptList ของ charList โดยการเรียกขั้นตอนวิธี COMBINE_PT ก่อน จากนั้นจึงนำรายการเชื่อมโยง seqList ของ selList รวมกับรายการเชื่อมโยง seqList ของ charList โดยเรียกขั้นตอนวิธี COMBINE_SEQ แล้วทำการปรับโครงสร้างรายการเชื่อมโยง charList เพื่อเชื่อมต่อบัพ cmdstruct ที่แทนส่วนของเส้นต่างๆที่เชื่อมต่อกัน โดยเรียกขั้นตอนวิธี ADJUST_CHARLIST

การทำงานของขั้นตอนวิธี COMBINE_PT นั้น จะทำทีละบัพโดยเริ่มจากบัพ ptstruct แรก ทำการค้นหาบัพ ptstruct ในรายการเชื่อมโยง ptList ของ charList ว่ามีบัพที่มีคู่ลำดับเดียวกันหรือไม่ ถ้ามีก็จะรวมค่าของเขตข้อมูล pcount ccount scount pcshape และ ccshape ของบัพ ptstuct ทั้งสอง จากนั้นจะต้องทำการเปลี่ยนค่าตัวชี้ของบัพ cmdstruct ที่ชี้บัพ ptstuct เดิมอยู่ให้ชี้ไปยังบัพ ptstruct ที่ค้นพบทุกบัพ แต่ในกรณีที่ค้นไม่พบจะทำการตัดบัพออกแล้วแทรกในรายการเชื่อมโยง ptList ของ charList แล้ววนกลับไปทำบัพถัดไปจนกว่าจะหมด

การทำงานของขั้นตอนวิธี COMBINE_SEQ นั้น ทำโดยการตัดรายการเชื่อมโยงออกจากรายการเชื่อมโยง selList แล้ว แทรกในตำแหน่งหน้าสุดของรายการเชื่อมโยง selList ของ charList

การทำงานของขั้นตอนวิธี ADJUST_CHARLIST นั้น จะเริ่มจากบัพ seqstruct แรกนำไปพิจารณาเทียบกับบัพ seqstruct ถัดไปทุกบัพ ว่าจุดเริ่มหรือสิ้นสุดของเส้นของทั้งสอง seqstruct นั้นเป็นจุดเดียวกันหรือไม่ ถ้าใช่ก็จะเรียกขั้นตอนวิธี CONNECT_SEQ เพื่อทำการเชื่อมต่อสอง seqstruct ให้เป็นบัพเดียว ถ้าไม่ใช่ก็จะข้ามไปยังบัพถัดไปจนหมด จึงจะเริ่มนำบัพ seqstruct ถัดไปมาพิจารณา โดยทั้งนี้แต่ละบัพ seqstruct ที่นำมาพิจารณา จะต้องตรวจสอบก่อนว่าจุดเริ่มต้นและจุดสุดท้ายของเส้นของบัพ seqstruct นั้นๆเป็นจุดเดียวกันหรือไม่(ดูรูปที่ 5.12 ถึง 5.13) รวมทั้งต้องพิจารณาด้วยว่า seqstruct นั้น ไม่ได้เป็นส่วนกลางของโปรแกรมย่อย(ดูรูปที่ 5.14) ซึ่งถ้าใช่แสดงว่าเส้นต่างๆของบัพ seqstruct นั้น ไม่สามารถเชื่อมต่อกับเส้นอื่นได้อีกก็จะข้ามการพิจารณาไปยังบัพถัดไปเลย

การทำงานของขั้นตอนวิธี CONNECT_SEQ นั้น จะเริ่มจากการพิจารณาว่าเป็นการเชื่อมต่อของบัพ cmdstruct ณ.ตำแหน่งใด

ถ้าเชื่อมต่อบัพcmdstructท้ายของseqstructแรกกับบัพcmdstructแรกของseqstructหลัง จะสามารถเชื่อมต่อกันได้เลยโดยเรียกขั้นตอนวิธี PROCESS1(ดูรูปที่ 5.15)

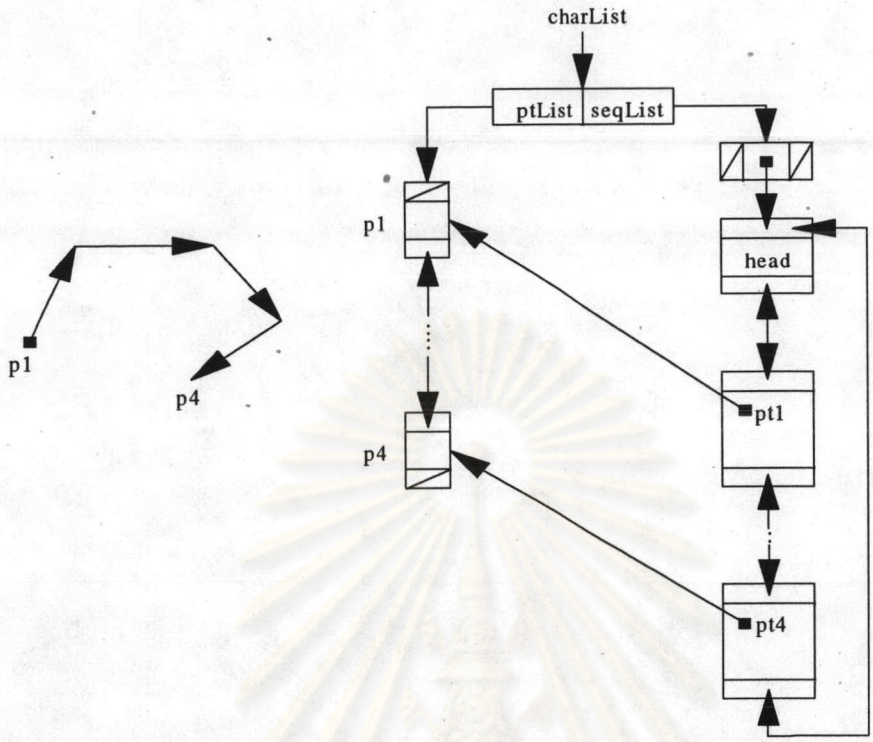
ถ้าเชื่อมต่อบัพcmdstructแรกของseqstructแรกกับบัพcmdstructท้ายของseqstructหลัง จะสามารถเชื่อมต่อกันได้เลยโดยเรียกขั้นตอนวิธี PROCESS2

ถ้าเชื่อมต่อบัพcmdstructแรกของseqstructแรกกับบัพcmdstructแรกของseqstructหลัง จะต้องทำการกลับทิศทางของบัพ cmdstruct ของ seqstruct (โดยการสลับตำแหน่งของบัพและคู่ลำดับ) โดยถ้าสามารถกลับทิศทางของบัพ cmdstruct ของ seqstruct หลังได้ก็จะกลับทิศทางแล้วเรียกขั้นตอนวิธี PROCESS2 หากไม่ได้ก็จะทำการกลับทิศทางของบัพ cmdstruct ของ seqstruct แรกแล้วเรียกขั้นตอนวิธี PROCESS1 แต่ถ้ากลับไม่ได้อีกแสดงว่าไม่สามารถจะเชื่อมต่อกันได้ ก็จะส่งค่าความผิดพลาดกลับไป

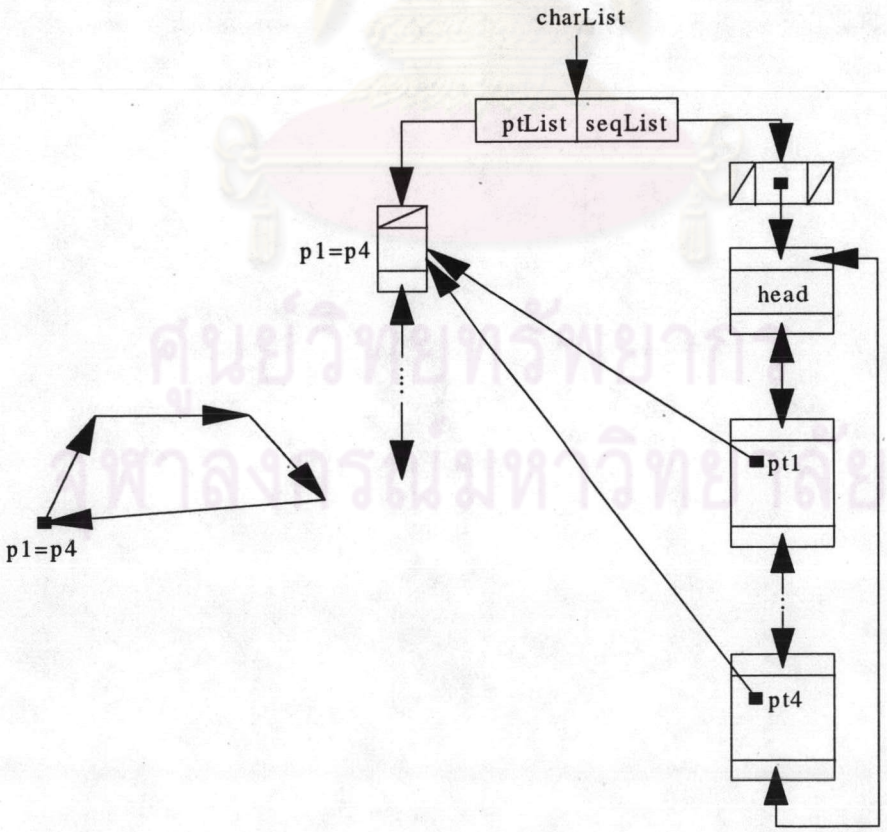
ถ้าเชื่อมต่อบัพcmdstructท้ายของseqstructแรกกับบัพcmdstructท้ายของseqstructหลัง(ดูรูปที่ 5.17) จะต้องทำการกลับทิศทางของบัพ cmdstruct ของ seqstruct (โดยการสลับตำแหน่งของบัพและคู่ลำดับ ดูรูปที่ 5.18) โดยถ้าสามารถกลับทิศทางของบัพ cmdstruct ของ seqstruct หลังได้ ก็จะกลับทิศทางแล้วเรียกขั้นตอนวิธี PROCESS1 หากไม่ได้ก็จะทำการกลับทิศทางของบัพ cmdstruct ของ seqstruct แรกแล้วเรียกขั้นตอนวิธี PROCESS2 แต่ถ้ากลับไม่ได้อีกแสดงว่าไม่สามารถจะเชื่อมต่อกันได้ ก็จะส่งค่าความผิดพลาดกลับไป(ดูรูปที่ 5.19)

การทำงานของขั้นตอนวิธี PROCESS1 เริ่มจากการหากลุ่มของบัพ seqstruct หลัง (กรณีที่มีการเรียกใช้โปรแกรมย่อยที่มีหลายส่วน บัพ seqstruct จะอยู่เรียงติดต่อกันไป การตัด-ย้ายจะต้องทำทั้งกลุ่ม ดูรูปที่ 5.14) แล้วทำการตัดกลุ่มนี้ออกจาก seqList แล้วแทรกไว้หลังบัพ seqstruct แรก จากนั้นจึงนำบัพ cmdstruct แรกของ seqstruct หลัง มาเชื่อมต่อกับบัพสุดท้ายของ seqstruct แรก แล้วเปลี่ยนตัวชี้ให้ถูกต้อง seqstruct แรกและ seqstruct หลังก็จะถูกเชื่อมต่อเป็น seqstruct เดียว(ดูรูปที่ 5.16)

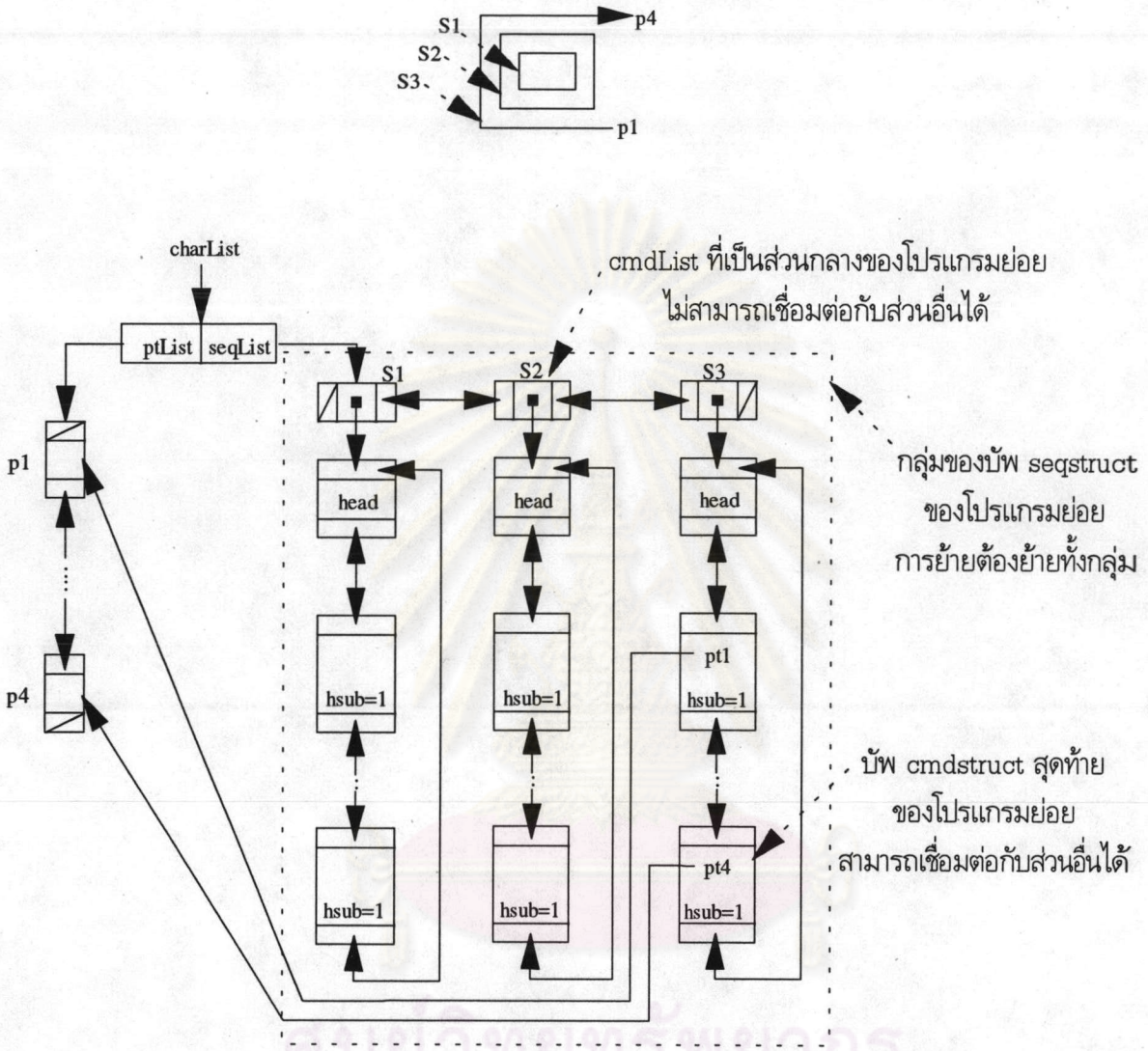
การทำงานของขั้นตอนวิธี PROCESS2 เริ่มจากการหากลุ่มของบัพ seqstruct หลัง แล้วตัดกลุ่มนี้ออกจาก seqList แล้วแทรกไว้หน้าบัพ seqstruct แรก จากนั้นจึงนำบัพ cmdstruct แรกของ seqstruct แรก มาเชื่อมต่อกับบัพสุดท้ายของ seqstruct หลัง แล้วเปลี่ยนตัวชี้ให้ถูกต้อง seqstruct แรกและ seqstruct หลังก็จะถูกเชื่อมต่อเป็น seqstruct เดียว และต้องทำการกำหนดค่า seqptr ใหม่โดยให้มาชี้ที่บัพ seqstruct หลังที่ย้ายมาแล้ว



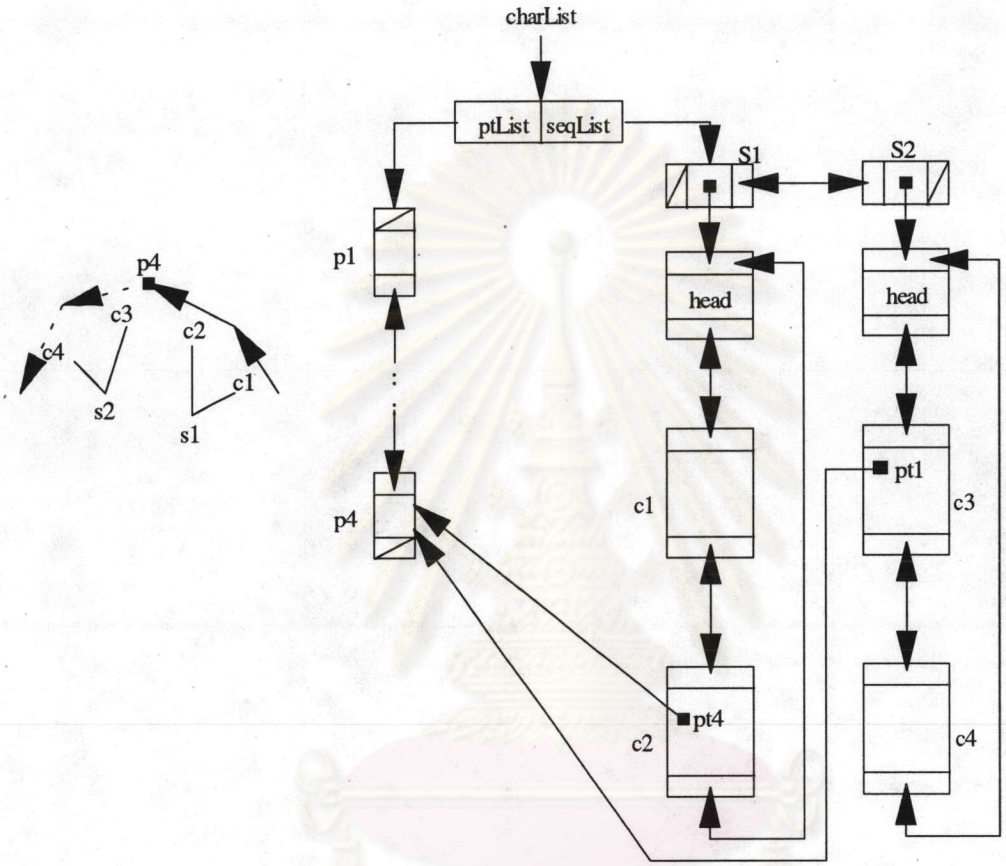
รูปที่ 5.12 แสดงรายการเชื่อมโยง cmdList ที่จุดเริ่มต้นและจุดสุดท้ายไม่ได้เป็นจุดเดียวกัน



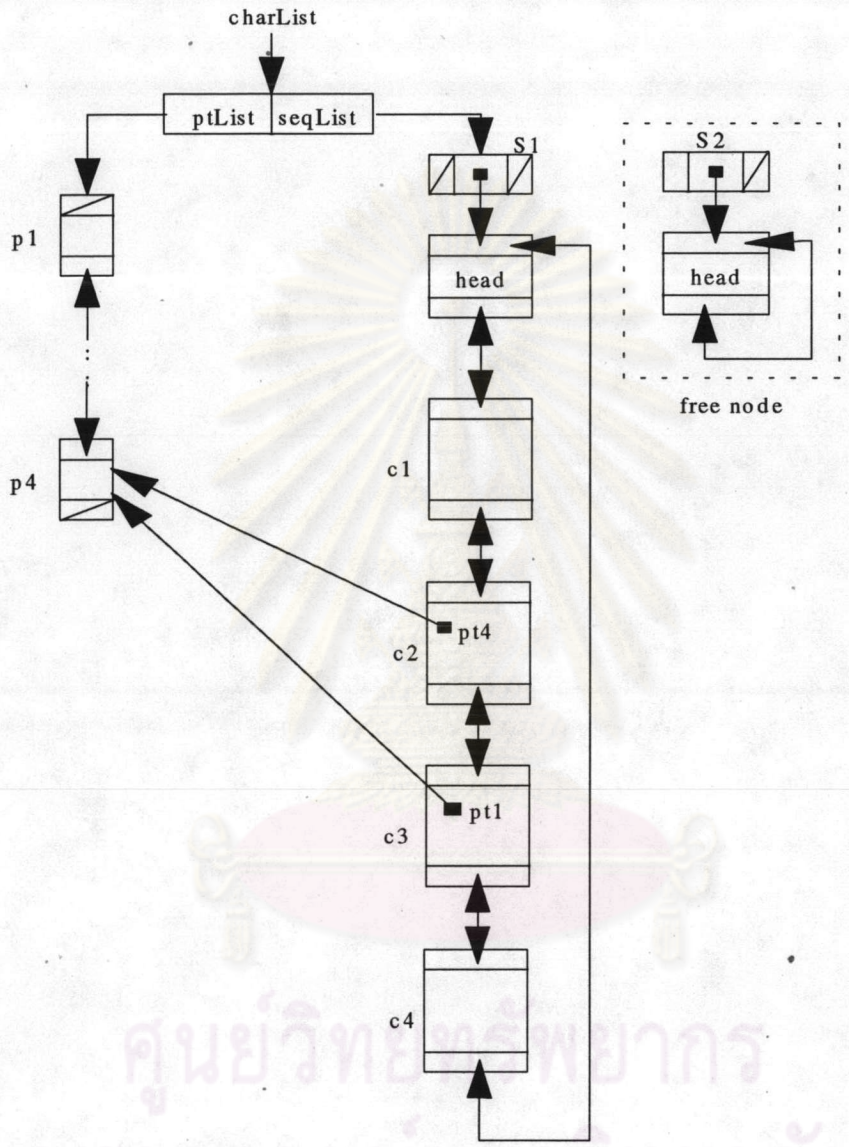
รูปที่ 5.13 แสดงรายการเชื่อมโยง cmdList ที่จุดเริ่มต้นและจุดสุดท้ายเป็นจุดเดียวกัน



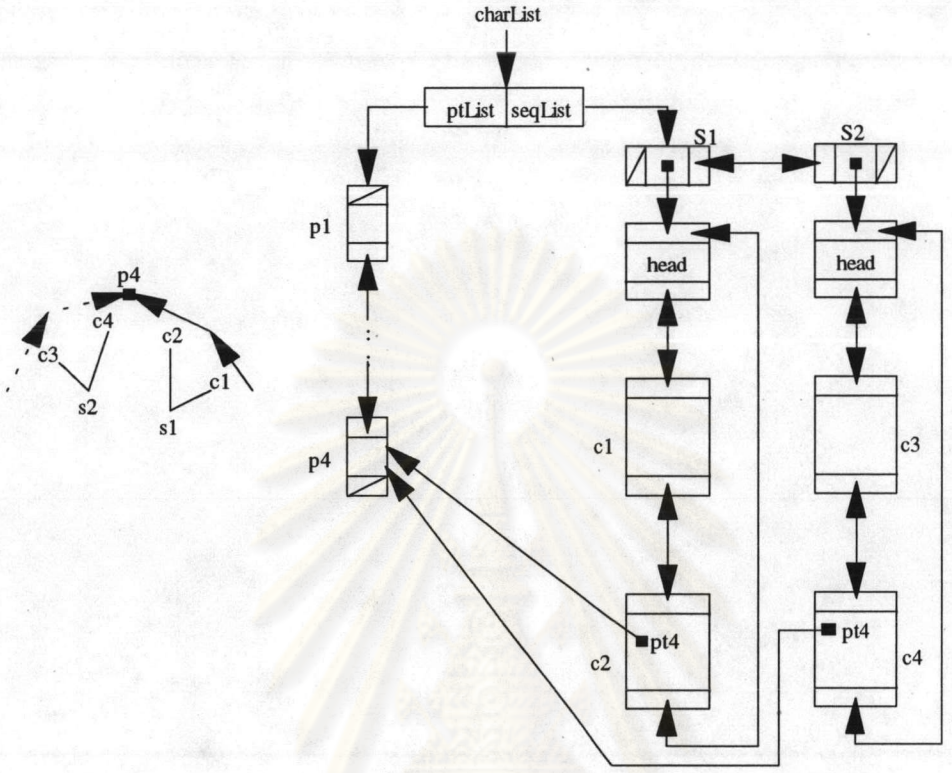
รูปที่ 5.14 แสดงรายการเชื่อมโยง cmdList ของส่วนของตัวอักษรที่เป็นโปรแกรมย่อย
ซึ่งประกอบด้วยส่วนของตัวอักษร 3 ส่วน



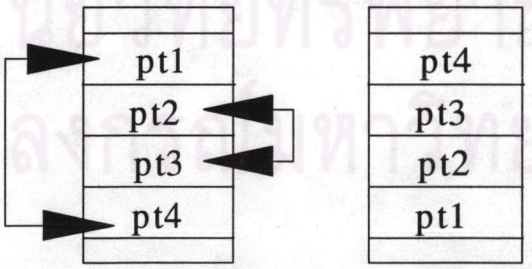
รูปที่ 5.15 แสดงรายการเชื่อมโยง cmdList ที่สามารถเชื่อมต่อกันได้ระหว่าง
บัพ cmdstruct ท้ายของ seqstruct แรกกับบัพ cmdstruct แรกของ seqstruct หลัง



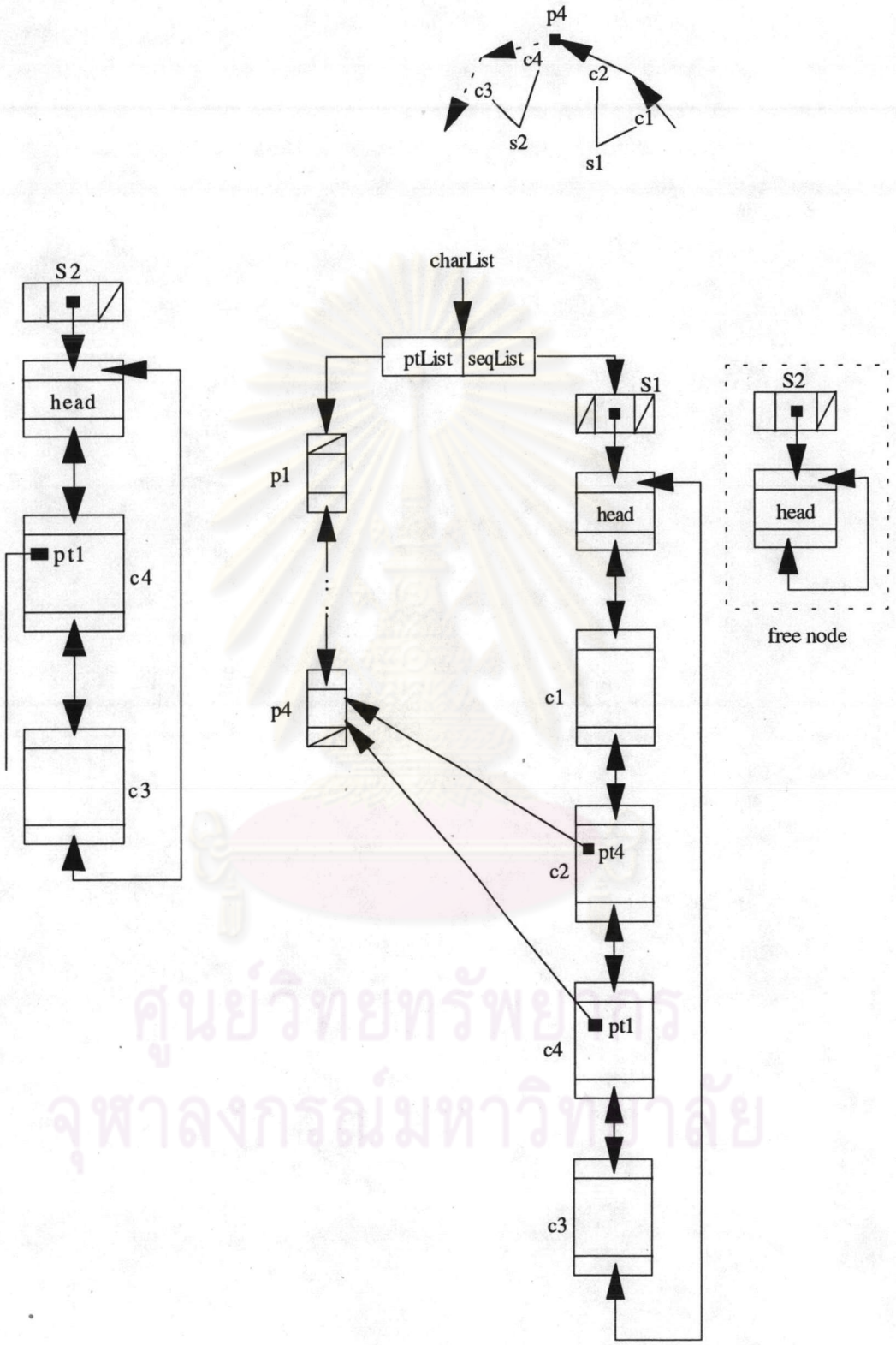
รูปที่ 5.16 แสดงรายการเชื่อมโยง cmdList หลังการเชื่อมต่อ
บัพ cmdstruct ท้ายของ seqstruct แรกกับบัพ cmdstruct แรกของ seqstruct หลัง



รูปที่ 5.17 แสดงรายการเชื่อมโยง cmdList ที่สามารถเชื่อมต่อกันได้ระหว่าง
บัพ cmdstruct ท้ายของ seqstruct แรกกับบัพ cmdstruct ท้ายของ seqstruct หลัง



รูปที่ 5.18 แสดงการกลับค่าตัวชี้ pt1 pt2 pt3 และ pt4 ของบัพ cmdstruct



รูปที่ 5.19 แสดงการกลับทิศทางของบัพ cmdstruct และรายการเชื่อมโยงหลังการเชื่อมต่อ

Algorithm COMBINE_SEL_2_CHARLIST(charstruct **charList*, charstruct **selList*)

COMBINE_PT(*charList, selList*);
COMBINE_SEQ(*charList, selList*);
ADJUST_CHARLIST(*charList*);

Algorithm COMBINE_PT(charstruct *charList*, charstruct *selList*)

ptstruct **ptptr, *oldptptr*;

set *ptptr* to first ptstruct node of *selList*;
while (not end of ptList)
{
find *oldptptr* of ptList of *charList* which has the same as coordinate of *ptptr* node;
if (not found)
cut *ptptr* node from *selList* and insert into ptList of *charList*;
else
{
add *pcount, ccount, scount, pcshape, ccshape* of *ptptr* node to *oldptptr* node;
change all point reference of cmdstruct node in *selList* from *ptptr* to *oldptptr*;
free *ptptr* node;
}
next ptstruct node;
}

Algorithm COMBINE_SEQ(charstruct **charList*, charstruct **selList*)

cut seqList from *selList* and insert at head of seqList of *charList*;

Algorithm ADJUST_CHARLIST(charstruct **charList*)

seqstruct **seqptr, *testseqptr*;

set *seqptr* to first seqstruct node of *charList*;
while (not end of seqList)
{
if (starting point of cmdstruct node not equal ending point of last cmdstruct node of this *seqptr*)
{
set *testseqptr* to next node;
while (not end of seqList)
{
if (starting point of cmdstruct node not equal ending point of last cmdstruct node of this seqptr)
{
if (starting point of first cmdstruct node or ending point of last cmdstruct node of *seqptr* is equal to starting point of first cmdstruct node or ending point of last cmdstruct node of *testseqptr*)

```
                    if (can't CONNECT_SEQ(charList, seqptr, testseqptr))
                      return ERR;
                  }
                }
              next testseqptr node;
             }
           next seqptr node;
          }
```

Algorithm CONNECT_SEQ(charstruct **charList*, seqstruct **seqptr*, seqstruct **testseqptr*)

```
if (connect end cmdstruct node of seqptr with first cmdstruct node of testseqptr)
   PROCESS1(charList, seqptr, testseqptr);
else if (connect first cmdstruct node of seqptr with last cmdstruct node of testseqptr)
        PROCESS2(charList, seqptr, testseqptr);
else if (connect first cmdstruct node of seqptr with first cmdstruct node of  testseqptr)
        {
         if (can reverse order of cmdstruct node of testseqptr)
           {
            REVERSE_CMD(testseqptr);
            PROCESS2(charList, seqptr, testseqptr);
           }
         else if (can reverse order of cmdstruct node of seqptr)
                 {
                  REVERSE_CMD(seqptr);
                  PROCESS1(charList, seqptr, testseqptr);
                 }
         else return ERR;
        }
else
       {
       if (can reverse order of cmdstruct node of testseqptr)
         {
          REVERSE_CMD(testseqptr);
          PROCESS1(charList, seqptr, testseqptr);
         }
       else if (can reverse order of cmdstruct node of seqptr)
              {
               REVERSE_CMD(seqptr);
               PROCESS2(charList, seqptr, testseqptr);
              }
       else return ERR;
       }
```

Algorithm PROCESS1(charstruct **charList*, seqstruct **seqptr*, seqstruct **testseqptr*)

seqstruct **begseqptr*, **endseqptr*;

FIND_GROUP_SEQ(*testseqptr*, *begseqptr*, *endseqptr*);
cut *begseqptr* - *endseqptr* from seqList of *charList* and insert behind *seqptr*;
connect first cmdstruct node of *begseqptr* to last cmdstruct node of *seqptr* and change all link reference(left & right);
free *begseqptr*;

Algorithm PROCESS2(charstruct **charList*, seqstruct **seqptr*, seqstruct **testseqptr*)

seqstruct **begseqptr*, **endseqptr*;

FIND_GROUP_SEQ(*testseqptr*, *begseqptr*, *endseqptr*);
cut *begseqptr* - *endseqptr* from seqList of *charList* and insert before *seqptr*;
connect first cmdstruct node of *seqptr* to last cmdstruct node of *endseqptr* and change all link reference(left & right);
free *seqptr*;
set *seqptr* and *testseqptr* to *begseqptr*;

5.1.2 การดึงบัพ cmdstruct ที่เลือกไว้ออกจากรายการเชื่อมโยง charList จะเป็นการดึงเอาบัพดังกล่าวออกจากรายการเชื่อมโยง charList แล้วนำไปเก็บไว้ที่รายการเชื่อมโยง selList (ดูรูปที่ 5.19 และ 5.20) มีรายละเอียดดังนี้

จากขั้นตอนวิธี DEL_CHAR_2_SEL การทำงานเริ่มจากการนำบัพ seqstruct ของรายการเชื่อมโยง charList มาพิจารณาทีละบัพจนกว่าจะหมด โดยแต่ละบัพนั้น จะทำการหาบัพ cmdstruct ที่ได้เลือกไว้ (flag=1) เมื่อพบจะทำการเก็บค่าตัวชี้ของบัพนั้นไว้ แล้วตรวจสอบบัพถัดไปจนกว่าจะพบบัพที่ไม่ได้เลือกหรือหมดบัพในรายการเชื่อมโยง cmdLsit ของ seqstruct นั้น สำหรับบัพ cmdstruct ที่พบว่าได้เลือกไว้ ถ้าเป็นบัพของคำสั่งที่เป็นส่วนของโปรแกรมย่อยจะทำการเปลี่ยนตัวชี้ของจุดที่ชี้บัพptstructของรายการเชื่อมโยงcharListให้ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง selList แทน โดยการเรียกขั้นตอนวิธี CHANGE_PT_LINK แต่ถ้าไม่ใช่บัพของคำสั่งที่เป็นส่วนของโปรแกรมย่อย จะเรียกขั้นตอนวิธี DEC_SHAPE เพื่อลดตัวนับ pcshape และ/หรือ ccshape ของบัพ ptstruct เดิม เปลี่ยนตัวชี้ของจุดที่ชี้บัพ ptstruct ของรายการเชื่อมโยง charList ให้ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง selList แทน โดยการเรียกขั้นตอนวิธี CHANGE_PT_LINK แล้วจึงเรียกขั้นตอนวิธี INC_SHAPE เพื่อเพิ่มตัวนับ pcshape และ/หรือ ccshape ของบัพ ptstruct ใหม่ จากนั้นจะพิจารณาแยกเป็น 3 กรณีดังนี้

ถ้าบัพ cmdstruct ที่จะดึงบัพแรกเป็นบัพแรกสุดของรายการเชื่อมโยง cmdList และบัพ cmdstructบัพท้ายเป็นบัพท้ายสุดของรายการเชื่อมโยงcmdListจะทำการตัดบัพseqstructออกจากรายการเชื่อมโยง seqList ของ charList

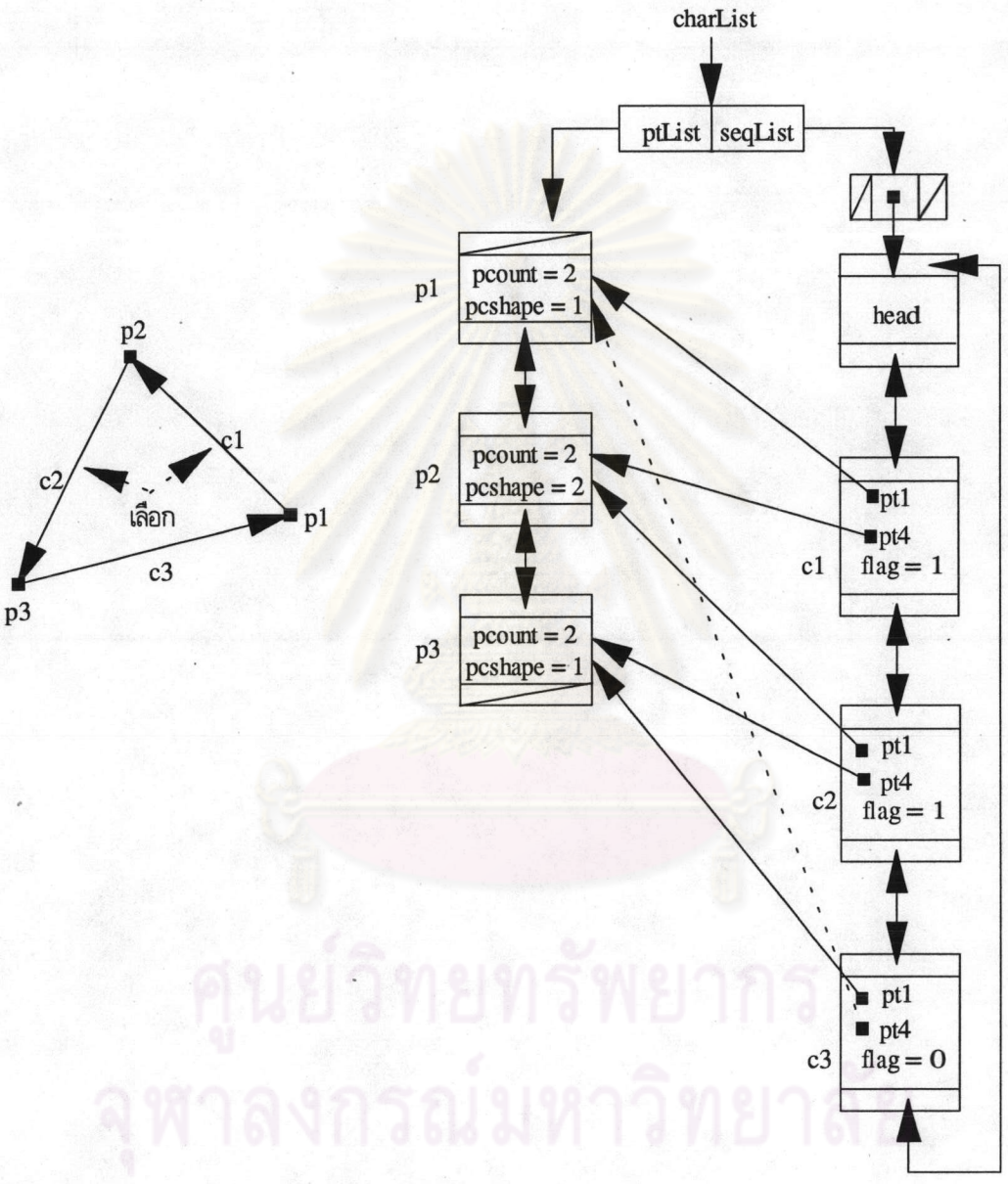
ถ้าบัพ cmdstruct ที่จะดึงบัพแรกเป็นบัพแรกสุดของรายการเชื่อมโยง cmdList จะทำการสร้างบัพ seqstruct ใหม่ แล้วนำบัพ cmdstruct แรกจนถึงบัพท้ายไปเพิ่มในรายการเชื่อมโยง cmdList ของบัพ seqstruct ใหม่ ส่วนบัพ cmdstruct ที่เหลืออยู่จะทำการเปลี่ยนตัวชี้ให้ถูกต้อง

ถ้าไม่ใช่ทั้ง 2 กรณีดังกล่าว จะพิจารณาว่าถ้าบัพ cmdstruct ท้ายไม่ใช่บัพสุดท้ายของรายการเชื่อมโยง cmdList จะต้องสร้างบัพ seqstruct ใหม่ เพื่อนำบัพ cmdstruct ที่ต่อกับบัพ cmdstruct ที่จะดึงบัพท้ายทั้งหมดไปต่อกับบัพ seqstruct ใหม่ แล้วแทรกไว้ที่หลังบัพ seqstruct ปัจจุบันในรายการเชื่อมโยง seqList ของ charList ก่อน จากนั้นสร้างบัพ seqstruct ใหม่แล้วนำบัพ cmdstruct ที่ต้องการดึงไปเพิ่มในรายการเชื่อมโยง cmdList ของบัพ seqstruct ใหม่

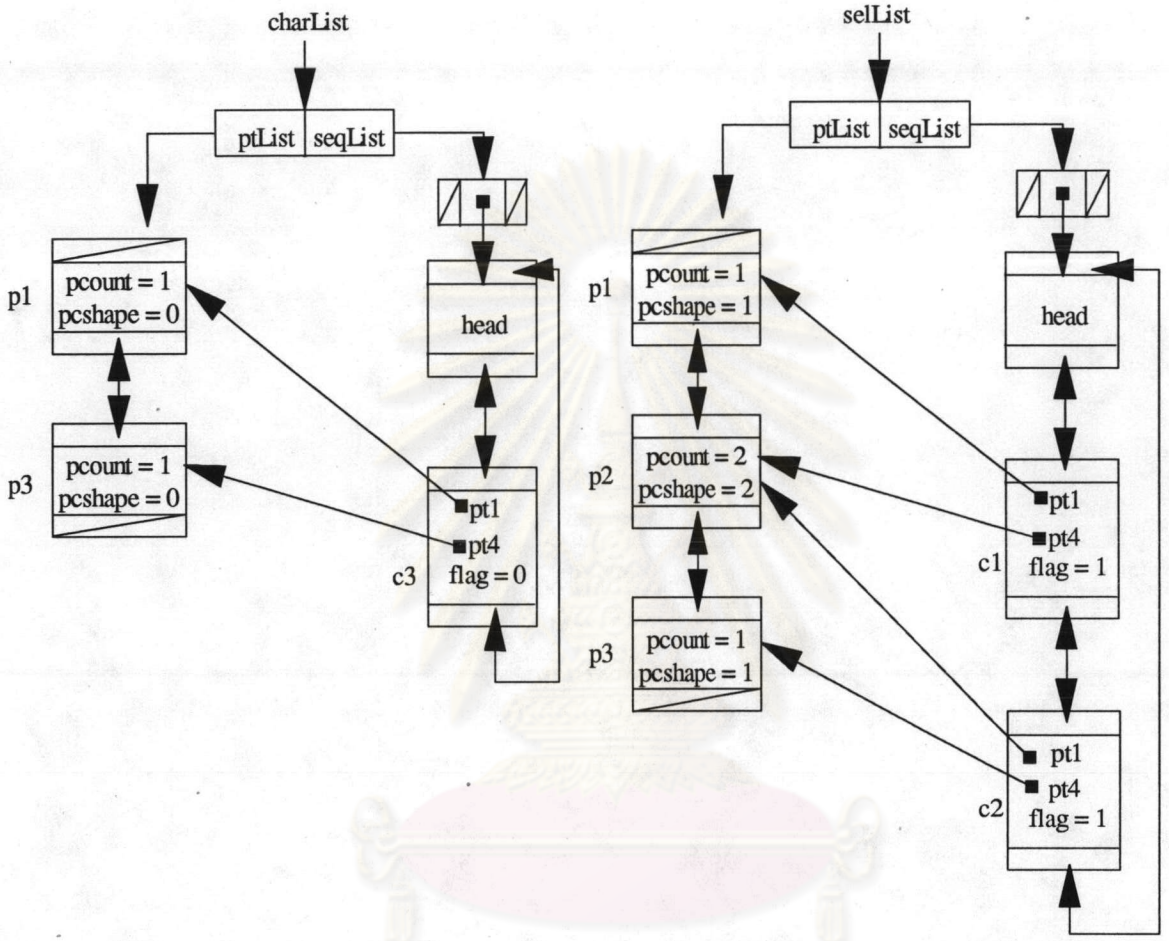
หลังจากการดึงบัพที่เลือกไว้ออกจากรายการเชื่อมโยง charList แล้ว จึงทำการแทรกบัพ seqstruct ของรายการเชื่อมโยง cmdList ที่ต้องการไปเพิ่มที่ตำแหน่งหลังสุดของรายการเชื่อมโยง seqList ของ selList จากนั้นถ้าบัพ cmdstruct ของบัพ seqstruct ปัจจุบันยังไม่หมด ก็จะกลับไปตรวจสอบต่อจนหมด จากนั้นก็จะนำบัพ seqstruct ถัดไปมาพิจารณา

สำหรับขั้นตอนการทำงานของขั้นตอนวิธี **CHANGE_PT_LINK** เป็นการทำงานเพื่อเปลี่ยนตัวชี้ที่ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง ptList ของ charList ให้ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList และเพิ่ม/ลดตัวนับ pcount, ccount, scount โดยมีขั้นตอนการทำงานดังนี้ พิจารณาคำสั่งของบัพ cmdstruct ว่าเป็นคำสั่งใด ถ้าเป็นคำสั่ง CALL จะค้นหาหรือสร้างบัพ ptstruct ในรายการเชื่อมโยง ptList ของ selList ที่มีคู่ลำดับ x, y เหมือนกับบัพ ptstruct ที่ชี้อยู่เดิม ทำการเพิ่มค่า scount ของบัพ ptstruct ใหม่ ลดค่า scount ของบัพ ptstruct เดิม และถ้าผลรวมของ pcount, ccount, scount เป็น 0 แสดงว่าไม่มีบัพ cmdstruct ใดๆชี้มาที่บัพนี้ ก็จะทำการทำลายบัพ ptstruct นี้ เพื่อคืนหน่วยความจำให้ระบบ แล้วเปลี่ยนค่า pt1 ของบัพ cmdstruct ให้ชี้ไปที่บัพ ptstruct ของ selList

แต่ถ้าเป็นคำสั่ง LINE, CURVE หรือ CLOSEPATH จะค้นหาหรือสร้างบัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList สำหรับคู่ลำดับที่เป็นจุดเริ่มต้นและจุดสิ้นสุดของบัพ ยกเว้นกรณีคำสั่ง CURVE จะเพิ่มจุดควบคุมที่ 2 และ 3 ด้วย โดยถ้าเป็นจุดเริ่มหรือจุดสิ้นสุด จะเพิ่ม/ลดค่า pcount ของบัพ ptstruct ใหม่/เก่า และถ้าเป็นจุดควบคุมจะเพิ่ม/ลดค่า ccount ของบัพ ptstruct ใหม่/เก่า และถ้าผลรวมของ pcount, ccount, scount ของบัพ ptstruct เดิมเป็น 0 ก็จะทำลายบัพ ptstruct นั้นเพื่อคืนหน่วยความจำให้ระบบ แล้วเปลี่ยนค่า pt1, pt2, pt3, pt4 ให้ชี้ไปที่บัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList



รูปที่ 5.20 แสดงรายการเชื่อมโยง charList ก่อนการดึงบัพ cmdstruct ที่เลือกไว้ออก



รูปที่ 5.21 แสดงรายการเชื่อมโยง charList และ selList หลังการดึงบัพ cmdstruct ที่เลือกไว้ออก

Algorithm DEL_CHAR_2_SEL(charstruct **charList*, charstruct **selList*)

```
seqstruct *seqptr;
cmdstruct *begcmdptr, *endcmdptr;

set seqptr to first seqstruct node of seqList of charList;
while (not end of seqList of charList)
      {
      set cmdptr to first cmdstruct node of cmdList of seqptr;
      while (not end of cmdList)
            {
            if (cmdptr node is select(flag=1))
               {
               save pointer of cmdptr to begcmdptr;
               do {
                  if (not a subroutine(hsub=0))
                    {
                    DEC_SHAPE(cmdptr);
                    CHANGE_PT_LINK(charList, selList, cmdptr);
                    INC_SHAPE(cmdptr);
                    }
                  else
                    CHANGE_PT_LINK(charList, selList, cmdptr);
                  next cmdstruct node;
                  } while (not end of cmdList && cmdptr node is select)
               save pointer of previous cmdptr to endcmdptr;
               if (begcmdptr is a first node && endcmdptr is a last node of cmdList)
                  cut seqptr out of seqList of charList;
               else if (begcmdptr is a first node)
                       {
                       create new seqstruct node, add begcmdptr-endcmdptr node to it;
                       change link of cmdstruct node behind endcmdptr;
                       }
               else
                  {
                  if (endcmdptr is not a last node)
                     create new seqstruct node, cut cmdstruct node behind endcmdptr, connect to it
                     and insert in seqList behind seqptr;
                  create new seqstruct, add begcmdptr-endcmdptr node to it;
                  change link of cmdstruct node before begcmdptr;
                  }
               add seqstruct node that has cmdList of begcmdptr-endcmdptr to the
               end of seqList of selList;
               break;
               }
            next cmdstruct node;
            }
      next seqstruct node;
      }
```

Algorithm CHANGE_PT_LINK(charstruct **charList*, charstruct **selList*, cmdstruct **cmdptr*)

ptstruct **pt1ptr, *pt2ptr, *pt3ptr, *pt4ptr*;

if (*cmdptr* is a CALL command node)
{
find or create ptstruct node of ptList of *selList* which has the same as coordinate *cmdptr->pt1->x, cmdptr->pt1->y*;
set *pt1ptr* point to it;
increment *scount* of ptstruct node that point by *pt1ptr* by 1;
decrement *scount* of ptstruct node that point by *cmdptr->pt1* by 1;
if (sum of *pcount,ccount,scount* of ptstruct node that point by *cmdptr->pt1* == 0)
free ptstruct node that point by *cmdptr->pt1*;
change *pt1 of cmdptr* to point at *pt1ptr*;
}
else
{
find or create ptstruct node of ptList of *selList* which has the same as coordinate *cmdptr->pt1->x, cmdptr->pt1->y*;
set *pt1ptr* point to it;
find or create ptstruct node of ptList of *selList* which has the same as coordinate *cmdptr->pt4->x, cmdptr->pt4->y*;
set *pt4ptr* point to it;
increment *pcount* of ptstruct node that point by *pt1ptr* and *pt4ptr* by 1;
decrement *pcount* of ptstruct node that point by *cmdptr->pt1 and cmdptr->pt4* by 1;
if (sum of *pcount,ccount,scount* of ptstruct node that point by *cmdptr->pt1* == 0)
free ptstruct node that point by *cmdptr->pt1*;
if (sum of *pcount,ccount,scount* of ptstruct node that point by *cmdptr->pt4* == 0)
free ptstruct node that point by *cmdptr->pt4*;
if (*cmdptr* is a CURVE command node)
{
find or create ptstruct node of ptList of *selList* which has the same as coordinate *cmdptr->pt2 ->x, cmdptr->pt2->y*;
set *pt2ptr* point to it;
find or create ptstruct node of ptList of *selList* which has the same as coordinate *cmdptr->pt3->x, cmdptr->pt3->y*;
set *pt3ptr* point to it;
increment *ccount* of ptstruct node that point by *pt2ptr* and *pt3ptr* by 1;
decrement *ccount* of ptstruct node that point by *cmdptr->pt2* and *cmdptr->pt3* by 1;
if (sum of *pcount,ccount,scount* of ptstruct node that point by *cmdptr->pt2* == 0)
free ptstruct node that point by *cmdptr->pt2*;
if (sum of *pcount,ccount,scount* of ptstruct node that point by *cmdptr->pt3* == 0)
free ptstruct node that point by *cmdptr->pt3*;
}
change *pt1, pt2, pt3, pt4* of *cmdptr* to point at *pt1ptr, pt2ptr, pt3ptr, pt4ptr*;
}

5.1.3 การสร้างรายการเชื่อมโยง charList หรือ selList จาก charstring byte เป็นการนำข้อมูลการบรรยายการสร้างตัวอักษรที่เก็บอยู่ในรูปแบบของโพสต์สคริปต์ประเภทที่ 1 มาสร้างเป็นรายการเชื่อมโยงเพื่อทำการตกแต่ง แก้ไขตามต้องการ ขั้นตอนการทำงานแสดงดังขั้นตอนวิธี GEN_CHARLIST ดังนี้

การทำงานเริ่มจากการนำ charstring byte มาพิจารณาทีละไบต์ ถ้าค่ามากกว่า 0x1F แสดงว่าเป็นไบต์ข้อมูลตัวเลข ซึ่งจะต้องทำการ decode ให้เป็นตัวเลขที่ใช้ได้แล้วเก็บลง operand stack แต่ถ้าน้อยกว่าแสดงว่าเป็นไบต์คำสั่ง ซึ่งแยกการทำงานดังนี้

กรณีคำสั่ง HSTEM และ VSTEM แสดงว่าเป็นคำสั่งการกำหนดสเตมฮินท์จะดึงค่าออกจาก operand stack แปลงค่าให้เป็นตำแหน่งคู่ลำดับโดยบวกกับตำแหน่งของ left side bearing point แล้วสร้างบัพ stemstruct กำหนดค่าแล้วเพิ่มบัพดังกล่าวในรายการเชื่อมโยง stemList

กรณีคำสั่ง RMOVETO, VMOVETO, HMOVETO แสดงว่าเป็นคำสั่งการเปลี่ยนตำแหน่งปัจจุบันเพื่อวาดส่วนของตัวอักษรใหม่ จะดึงค่าออกจาก operand stack ตามจำนวนพารามิเตอร์ที่ต้องการ แปลงค่าจากค่าสัมพัทธ์ให้เป็นตำแหน่งคู่ลำดับโดยบวกกับตำแหน่งปัจจุบันและกำหนดตำแหน่งปัจจุบันใหม่ สร้างบัพ seqstruct เพิ่มในรายการเชื่อมโยง seqList

กรณีคำสั่ง RLINETO, VLINETOและ HLINETO แสดงว่าเป็นคำสั่งการลากเส้นตรง จะดึงค่าออกจาก operand stack ตามจำนวนพารามิเตอร์ที่ต้องการ แปลงค่าจากค่าสัมพัทธ์ให้เป็นตำแหน่งคู่ลำดับโดยบวกกับตำแหน่งปัจจุบัน สร้างหรือค้นหาบัพ ptstruct จากรายการเชื่อมโยง ptList ที่มีตำแหน่งของคู่ลำดับตรงกัน เพิ่มค่า pcount ของบัพ ptstruct สร้างบัพ cmdstruct กำหนดค่าต่างๆ แล้วเพิ่มในรายการเชื่อมโยง cmdList ในตำแหน่งท้ายสุด และกำหนดตำแหน่งปัจจุบันให้มาอยู่ที่จุดสิ้นสุดของเส้นนั้น

กรณีคำสั่ง RRCURVETO, VHCURVETO และ HVCURVETO แสดงว่าเป็นคำสั่งการลากเส้นโค้ง การทำงานจะเหมือนกรณีเส้นตรง เว้นแต่จะเพิ่มการสร้างหรือค้นหาบัพ ptstruct จากรายการเชื่อมโยง ptList ที่มีตำแหน่งคู่ลำดับตรงกันกับจุดควบคุมที่ 2 และ 3 และเพิ่มค่า ccount ของบัพ ptstruct นั้น

กรณีคำสั่ง CLOSEPATH แสดงว่าเป็นคำสั่งจบส่วนของตัวอักษร และให้ลากเส้นตรงจากจุดสุดท้ายไปยังจุดเริ่มต้นของส่วนของตัวอักษรนั้นในกรณีที่จุดเริ่มและจุดสุดท้ายไม่ใช่จุดเดียวกัน การทำงานจะเหมือนกับการลากเส้นตรงทุกอย่าง เว้นแต่จะไม่ทำการเปลี่ยนแปลงตำแหน่งของจุดปัจจุบัน

กรณีคำสั่ง HSBW แสดงว่าเป็นคำสั่งกำหนดจุด left side bearing และขนาดความกว้างของตัวอักษร จะดึงค่าออกจาก operand stack ตามจำนวนพารามิเตอร์ที่ต้องการ เพื่อกำหนดค่าให้กับตัวแปรดังกล่าว

กรณีคำสั่ง RETURN แสดงว่าเป็นการจบ charstring byte ของโปรแกรมย่อย จะคืนการทำงานไปยังส่วนที่เรียกมา และในกรณีที่เป็นโปรแกรมย่อยที่กำหนดสเตมฮินท์เท่านั้น จะส่งค่า STEMONLY กลับไปยังส่วนที่เรียก

กรณีคำสั่ง CALLSUBR แสดงว่าเป็นการเรียกใช้โปรแกรมย่อย จะดึงหมายเลขของโปรแกรมย่อยนั้นออกจาก operand stack ถ้าค่าแฮนเดิลปัจจุบันเป็น 0 แสดงว่าตัวอักษรปรกติเรียกใช้โปรแกรมย่อย จะทำการสร้างค่าแฮนเดิลใหม่ แต่ถ้าไม่ใช่ 0 แสดงว่าโปรแกรมย่อยเรียกใช้โปรแกรมย่อย จะไม่มีการสร้างค่าแฮนเดิลใหม่ จากนั้นจะเรียกขั้นตอนวิธี GEN_CHARLIST เพื่อสร้างรายการเชื่อมโยงสำหรับโปรแกรมย่อยนั้น จนกว่าจะมีการส่งการทำงานกลับมาจะตรวจสอบค่าที่ส่งกลับ ถ้าเป็น FLEX แสดงว่า charstring byte ที่ตามมาเป็นการเรียกใช้เฟลกซ์ จะทำการเปลี่ยน charstring byte นั้นให้เป็นบัพ cmdstruct ที่บรรยายการลากเส้นโค้ง 2 เส้น แต่ถ้าค่าเป็น STEMONLY แสดงว่าโปรแกรมย่อยที่เรียกใช้ เป็นการกำหนดสเตมฮินท์เพียงอย่างเดียว จะสร้างบัพ stemstruct สำหรับคำสั่ง CALL แล้วแทรกไว้ในตำแหน่งแรกสุดของบัพ stemstruct โปรแกรมย่อยนั้นในรายการเชื่อมโยง stemList แต่ถ้ากรณีที่เรียกใช้โปรแกรมย่อยจากตัวอักษรปรกติ จะสร้างบัพ cmdstruct สำหรับคำสั่ง CALL แทรกไว้ในตำแหน่งแรกสุดของบัพ cmdstruct ของโปรแกรมย่อยนั้นในรายการเชื่อมโยง cmdList

กรณีคำสั่ง ENDCHAR แสดงว่าเป็นการจบ charstring byte ของตัวอักษรนั้น ถ้าเป็นการสร้างรายการเชื่อมโยงสำหรับโปรแกรมย่อย จะมีการสร้างบัพ cmdstruct สำหรับคำสั่ง ENDCHAR แล้วเพิ่มเข้าเป็นบัพสุดท้ายของรายการเชื่อมโยง cmdList

กรณีคำสั่ง ESCAPE แสดงว่า charstring byte ไบต์ถัดไปเป็นไบต์คำสั่งดังต่อไปนี้

กรณีคำสั่ง SBW แสดงว่าเป็นคำสั่งกำหนดจุด left side bearing ทั้งแนวแกน x และ y และกำหนดความกว้างและความสูงของตัวอักษร จะดึงค่าออกจาก operand stack ตามจำนวนพารามิเตอร์ที่ต้องการ เพื่อกำหนดค่าให้กับตัวแปรดังกล่าว

กรณีคำสั่ง DIV แสดงว่าเป็นคำสั่งการหาร จะดึงค่าออกจาก operand stack 2 ค่า แล้วหารค่าแรกด้วยค่าที่สอง ผลลัพธ์เฉพาะเลขจำนวนเต็มจะถูกเก็บลง operand stack

กรณีคำสั่ง CALLOTHERSUBR แสดงว่าเป็นการเรียกใช้ความสามารถพิเศษบางอย่าง จะดึงหมายเลขของโปรแกรมย่อยอื่นๆออกจาก operand stack ถ้าเป็นหมายเลข 3 แสดงว่าเรียกใช้การเปลี่ยนสเตมฮินท์ซึ่งโปรแกรมไม่สนับสนุน จะดึงค่าที่ไม่ใช้ออกจาก operand stack ทิ้งไป แต่ถ้าไม่ใช่แสดงว่าเป็นการเรียกใช้เฟลกซ์ จะส่งค่า FLEX และคืนการทำงานกลับไปยังส่วนที่เรียกมา เพื่อทำการแปลงให้เป็นคำสั่งการลากเส้นโค้งต่อไป

กรณีคำสั่ง SEAC แสดงว่าเป็นการเรียกใช้การสร้างตัวอักษรจากตัวอักษรที่กำหนดแล้ว 2 ตัว จะดึงค่าออกจาก operand stack ตามจำนวนพารามิเตอร์ที่ต้องการ โดยค่าของตัวอักษรที่ใช้เป็นเบสและแอ็คเซนท์จะต้องแปลงจากรหัสตัวอักษรมาตรฐานของบริษัทอโดบีให้เป็นรหัส ISO ก่อน* จากนั้นก็จะเรียกขั้นตอนวิธี GEN_CHARLIST เพื่อสร้างรายการเชื่อมโยงสำหรับตัวอักษรเบสและแอ็คเซนท์ตามลำดับ

Algorithm GEN_CHARLIST(charstruct **charList*, LPSTR **lpbyte*, int *nbyte*, BYTE *hsub*, POINT *lsbp*, POINT *currentpoint*, int *charwidth*)

```
while (nbyte > 0)
    {
    if (*lpbyte > 0x1F)
      decode number and push to  operand stack ;
    else
      {
      switch (*lpbyte)
            {
            case HSTEM:
            case VSTEM:
                  pop operand from  operand stack ;
                  convert first operand to absolute coordinate by plus lsbp;
                  create & initialize new stemstruct node, add to stemList;
                  break;

            case RMOVETO:
            case VMOVETO:
            case HMOVETO:
                  pop operand from  operand stack ;
                  convert to absolute coordinate by plus currentpoint;
                  set currentpoint to it;
                  create & add new seqstruct node at end of seqList;
                  break;

            case RLINETO:
            case VLINETO:
            case HLINETO:
                  pop operand from  operand stack ;
                  convert to absolute coordinate by plus currentpoint;
                  find or create ptstruct node of starting & ending point in ptList;
                  increment pcount of ptstruct node by 1;
                  set currentpoint to ending point;
                  create & add new cmdstruct node at end of cmdList;
                  break;
```

---

*ดูรายละเอียดเพิ่มเติมที่ภาคผนวก ข, หน้า 129.

```
case RRCURVETO:
case VHCURVETO:
case HVCURVETO:
      pop operand from  operand stack ;
      convert to absolute coordinate by plus currentpoint;
      find or create ptstruct node of starting & ending point in ptList;
      increment pcount of ptstruct node by 1;
      find or create ptstruct node of control point in ptList;
      increment ccount of ptstruct node by 1;
      set currentpoint to ending point;
      create & add new cmdstruct node at end of cmdList;
      break;

case CLOSEPATH:
      if (strting point of first cmdstruct node not equal ending point of
          last cmdstruct node of cmdList)
          {
           create cmdstruct node and initialize by set starting point to
           endingpoint of last node & ending point to starting point of first node,
           set cmd to LINE;
           increment pcount of ptstruct node that is a starting & ending point by 1;
           add cmdstruct node at end of cmdList;
          }
      break;

case HSBW:
      pop operand from  operand stack ;
      set x-left side bearing point(lsbp.x) to first operand;
      set charwidth to second operand;
      break;

case RETURN:
      if (generate only HSTEM or VSTEM command)
          return STEMONLY;
      else return;

case CALLSUBR:
      pop subroutine number from  operand stack ;
      if (current subroutine handle(hsub) is 0)
          generate new subroutine handle;
      set lpbyte&nbyte to subroutine charstring byte&size;
      generate linklist by add to charlist of this subroutine by GEN_CHARLIST;
      if (return value is FLEX)
          convert charstring byte that follow to primitive CURVE command;
      else if (return value is STEMONLY)
              create & insert stemstruct "CALL" node at the first of stemstruct node
              of this subroutine;
      else if (current subroutine handle is 0)
              create & insert cmdstruct "CALL" node at the first of cmdstruct node
              of this subroutine;
      break;
```

```
        case ENDCHAR:
            if (subroutine handle is not 0)
                create & insert cmdstruct "ENDCHAR" node at end of cmdList;
            return;

        case ESCAPE:
            next lpbyte;
            decrement nbyte;
            switch (*lpbyte)
                {
                case SBW:
                        pop operand from  operand stack ;
                        set left side bearing point(lsbp) to first, second operand;
                        set charwidth to second operand;
                        break;

                case DIV:
                        pop operand from  operand stack ;
                        divide first openrand by second operand;
                        push quotient to  operand stack ;
                        break;

                case CALLOTHERSUBR:
                        pop othersubroutine number from  operand stack ;
                        if (othersubroutine number is 3)
                          {
                           pop 2 operand from  operand stack ;
                           skip to next 2 charstring byte;
                          }
                        else
                          return FLEX;
                        break;

                case SEAC:
                        pop character code of accent&base from  operand stack ;
                        pop character origin point of accent from  operand stack ;
                        pop x-left side bearing point of accent from operan stack;
                        convert character code to ISO code;
                        generate charList for base&accent character by GEN_CHARLIST;
                        break;
                }
            break;
            }
    }
next charstring byte;
decrement nbyte;
}
```

5.2 ขั้นตอนการทำงานเกี่ยวกับแฟ้มข้อมูล

5.2.1 ขั้นตอนการเปิดแฟ้มข้อมูล

ขั้นตอนการทำงานนี้แสดงดังขั้นตอนวิธี **FILEOPEN** ซึ่งการทำงานเริ่มจากการเปิดแฟ้มข้อมูล PFM ตวจสอบความถูกต้องของแฟ้มโดยการตรวจสอบรหัสที่ต้นแฟ้มข้อมูล แล้วอ่านข้อมูลเข้าเก็บในโครงสร้างข้อมูล PFM จากนั้นปิดแฟ้มข้อมูล แล้วเปิดแฟ้มข้อมูล PFB ตรวจสอบความถูกต้องของแฟ้มโดยการตรวจสอบรหัสที่ต้นแฟ้มข้อมูล รหัสของส่วนข้อมูลที่เข้ารหัส ถ้าถูกต้องจึงทำการอ่านแฟ้มข้อมูล จัดเก็บข้อมูลส่วนแอสกี ส่วนแรก แล้วจึงทำการถอดรหัสส่วน eexec เมื่อถอดรหัสแล้วพิจารณาแต่ละ token ถ้าเป็นส่วนของโปรแกรมย่อยหรือตัวอักษร จะทำการถอดรหัสส่วน charstring และตัดคำสั่งที่โปรแกรมไม่สนับสนุนออก จากนั้นจึงคัดลอกไปเก็บยังหน่วยความจำที่จองไว้ เก็บค่าตำแหน่งของหน่วยความจำและขนาดไว้ในตัวแปรแถวลำดับ subtab หรือ chartab ตามลำดับของตัวอักษร แต่ถ้าเป็นคำสำคัญอื่นๆก็จะคัดลอกเก็บในโครงสร้างข้อมูล ทำซ้ำจนกว่าจะพบคำสำคัญ "end" ซึ่งแสดงว่าหมดข้อมูลแล้ว ก็ทำการปิดแฟ้มข้อมูล PFB

Algorithm FILEOPEN()

```
if (cannot open pfm file)
   return ERR;
read pfm file to pfm data structure;
close pfm file;
if (cannot open pfb file)
   return ERR;
read pfb file;
copy ASCII part 1 to pfb data structure;
decryption eexec byte;
skip to first token;
while (token != end)
   {
    if (token == "/Subr" || token == "/CharStrings")
       {
        decryption charstring byte;
        remove unsupport command&parameter;
        copy charstring byte to allocated memory;
        store address&size of memory to subtab or chartab;
       }
    else
       copy data to pfb data structure;
    skip to next token;
   }
close pfb file;
```

5.2.2 ขั้นตอนการจัดเก็บแฟ้มข้อมูล

ขั้นตอนการจัดเก็บแฟ้มข้อมูลแสดงดังขั้นตอนวิธี FILESAVE การทำงานเริ่มจากการสร้างแฟ้มข้อมูล PFM คัดลอกข้อมูลจากโครงสร้างข้อมูล PFM ไปยังบัฟเฟอร์ ทำการเขียนข้อมูล แล้วปิดแฟ้มข้อมูล PFM จากนั้นจึงเริ่มสร้างแฟ้มข้อมูล PFB โดยคัดลอกข้อมูลแอสกีส่วนแรกไปยังบัฟเฟอร์ ตามด้วยส่วน private dictionary ในกรณีที่มีการกำหนดโปรแกรมย่อยจะต้องทำการเข้ารหัสส่วน charstring ก่อนคัดลอก ซึ่งเหมือนกับส่วน charstring dictionary จากนั้นจึงทำการเข้ารหัสอีกครั้งตั้งแต่ส่วน private dictionary เป็นต้นมา คัดลอกเลข 0 จำนวน 512 ตัวไปยังบัฟเฟอร์ เขียนข้อมูลแล้วปิดแฟ้มข้อมูล PFB

Algorithm FILESAVE()

```
create pfm file;
copy pfm data structure to buffer;
write buffer to pfm file;
close pfm file;
create pfb file;
copy ASCII part 1 to buffer;
copy private dictionary to buffer;
if (use subroutine)
   {
    while (not end of subroutine)
         {
         copy&encrypt subroutine charstring byte to buffer;
         next subroutine;
         }
   }
while (not end of character)
     {
      copy&encrypt character charstring byte to buffer;
      next character;
     }
encrypt eexec byte(from private to character charstring byte);
copy 512 ASCII ZERO to buffer;
write pfb file;
close pfb file;
```

5.3 ขั้นตอนการทำงานอื่นๆ

5.3.1 ขั้นตอนการทำงานการจัดเก็บข้อมูลการบรรยายการสร้างตัวอักษรจากรายการเชื่อมโยงให้อยู่ในรูปแบบของโพสต์สคริปต์ประเภทที่ 1 ซึ่งแสดงดังขั้นตอนวิธี GEN_CHARSTRING โดยการทำงานเริ่มจากการพิจารณารายการเชื่อมโยง charList ว่ามีรายการเชื่อมโยง ptList และ seqList หรือไม่ ถ้าไม่มีจะพิจารณา 2 กรณี คือ การสร้าง charstring สำหรับตัวอักษรว่างเปล่าโดยการตรวจสอบจากค่าความกว้างของตัวอักษร ถ้าค่าความกว้างของตัวอักษรมากกว่า 0 จะทำการเข้ารหัสค่าพารามิเตอร์ของคำสั่ง HSBW และเข้ารหัสคำสั่ง HSBW ตามด้วย ENDCHAR แล้วส่งการทำงานกลับ ส่วนกรณีที่ 2 คือ การสร้าง charstring สำหรับโปรแกรมย่อย จะแยกพิจารณา

2 กรณีคือ ถ้ารายการเชื่อมโยง stemList ไม่มีและต้องการสร้างคำสั่ง CLOSEPATH และ/หรือ ENDCHAR ก็จะเข้ารหัสคำสั่ง CLOSEPATH และ/หรือ ENDCHAR ตามด้วย RETURN แล้วส่งการทำงานกลับ แต่ถ้ามีรายการเชื่อมโยง stemList จะทำการสร้าง charstring สำหรับการกำหนดสเตมฮินท์ โดยเรียกขั้นตอนวิธี SAVE_STEM ตามด้วยการเข้ารหัสคำสั่ง RETURN แล้วส่งการทำงานกลับ

ในกรณีที่มีรายการเชื่อมโยง ptList และ seqList แสดงว่าจะต้องมีการสร้าง charstring ที่เก็บคำสั่งการลากเส้นด้วย จะเริ่มพิจารณาว่าเป็นการสร้างโปรแกรมย่อยหรือไม่ ถ้าใช่จะตรวจสอบว่าต้องการสร้างคำสั่งพิเศษหรือไม่ ถ้าใช่จะเข้ารหัสคำสั่ง CLOSEPATH หรือพารามิเตอร์และคำสั่ง RMOVETO ก่อน แต่ถ้าไม่ใช่การสร้างโปรแกรมย่อยจะตรวจสอบว่าเป็นตัวอักษรแบบแอ็คเซนท์หรือไม่ถ้าใช่จะหาค่าและเข้ารหัสx-leftside bearing และความกว้างตัวอักษร ตามด้วยการเข้ารหัสคำสั่ง HSBW, ENDCHAR แล้วส่งการทำงานกลับ

จากนั้นจะตรวจสอบรายการเชื่อมโยง stemList ถ้ามีจะทำการกำหนดสเตมฮินท์ โดยเรียกขั้นตอนวิธี SAVE_STEM เสร็จแล้วจะเริ่มทำการสร้าง charstring สำหรับคำสั่งต่างๆ โดยเริ่มจากบัพ seqstruct แรกของรายการเชื่อมโยง seqList และบัพถัดไปจนกว่าจะหมด

ในแต่ละบัพ seqstruct นั้น จะนำบัพแต่ละบัพ cmdstruct มาพิจารณา โดยถ้าเป็นบัพแรกของ cmdList ที่ไม่ใช่การเรียกใช้โปรแกรมย่อยที่เริ่มต้นด้วยคำสั่ง MOVE(RMOVETO, VMOVETO, HMOVETO) หรือยังไม่ได้สร้างคำสั่ง MOVE จะทำการหาระยะทางสัมพัทธ์จากจุดปัจจุบันไปถึงจุดเริ่มต้นของบัพ cmdList นั้น เข้ารหัสระยะทางและคำสั่ง RMOVETO, HMOVETO, VMOVETO ที่เหมาะสม จากนั้นนำบัพ cmdstruct มาพิจารณา ถ้าเป็นบัพคำสั่ง CALL จะหาหมายเลขโปแกรมย่อยจากรายการเชื่อมโยง subList เข้ารหัสหมายเลขโปรแกรมย่อย และคำสั่ง CALLSUBR ถ้าเป็นบัพที่เป็นส่วนของโปรแกรมย่อย(hsub > 0) และเป็นคำสั่ง LINE หรือ CURVE จะกำหนดจุดปัจจุบันใหม่เป็นจุดสุดท้ายของบัพนั้น ถ้าเป็นบัพคำสั่ง ENDCHAR จะส่งการทำงานกลับ ถ้าเป็นคำสั่ง CLOSEPATH จะกำหนดตัวแปร bclosepath ให้เป็นจริง แต่ถ้าเป็นบัพของคำสั่งของตัวอักษรปรกติ (hsub = 0) และเป็นคำสั่ง LINE จะพิจารณาหาคำสั่งที่เหมาะสม(RLINETO หรือ VLINETO หรือ HLINETO) ที่เหมาะสมเข้ารหัสระยะทางและคำสั่งดังกล่าว แล้วกำหนดจุดปัจจุบันใหม่เป็นจุดสุดท้ายของบัพ ถ้าเป็นคำสั่ง CURVE จะพิจารณาหาคำสั่งที่เหมาะสม(RRCURVETO หรือ VHCURVETO หรือ HVCURVETO) หาระยะทางสัมพัทธ์เข้ารหัสระยะทางและคำสั่ง แล้วกำหนดจุดปัจจุบันใหม่เป็นจุดสุดท้ายของบัพ

เมื่อพิจารณาบัพ cmdstruct ครบแล้วถ้ายังไม่มีคำสั่ง CLOSEPATH (bclosepath เป็นเท็จ) และเป็นการสร้าง charstring ของตัวอักษรปรกติหรือโปรแกรมย่อยที่ยังไม่ใช่บัพ seqstruct สุดท้ายของรายการเชื่อมโยง seqList จะเข้ารหัสคำสั่ง CLOSEPATH แล้วพิจารณาบัพ seqstruct ถัดไป

เมื่อพิจารณาบัพ seqstruct ครบแล้ว ถ้าเป็นการสร้าง charstring ของตัวอักษรปรกติ จะเข้ารหัสคำสั่ง ENDCHAR แล้วส่งการทำงานกลับ ถ้าเป็นการสร้าง charstring ของโปรแกรมย่อย

และต้องการสร้างคำสั่งพิเศษCLOSEPATHและ/หรือENDCHARจะเข้ารหัสคำสั่งเหล่านี้จากนั้นจะเข้ารหัสคำสั่งRETURN แล้วส่งการทำงานกลับ

สำหรับขั้นตอนการทำงานของขั้นตอนวิธี SAVE_STEM จะเป็นการสร้าง charstring ของการกำหนดสเตมฮินท์ โดยนำบัพ stemstruct จากรายการเชื่อมโยง stemList มาพิจารณา ถ้าเป็นบัพของคำสั่ง CALL แสดงว่ามีการเรียกใช้โปรแกรมย่อยสำหรับการกำหนดสเตมฮินท์เพียงอย่างเดียว จะทำการหาหมายเลขของโปรแกรมย่อยจากรายการเชื่อมโยง subList เข้ารหัสหมายเลขโปรแกรมย่อยและคำสั่ง CALLSUBR ถ้าไม่ใช่บัพของคำสั่ง CALL และไม่ใช่บัพที่เป็นส่วนของโปรแกรมย่อย จะทำการหาระยะทางสัมพัทธ์จากจุด left side bearing ถึงจุดเริ่มต้นของสเตม เข้ารหัสระยะทางและความกว้างสเตม จากนั้นเข้ารหัสคำสั่งที่เหมาะสม (VSTEM หรือ HSTEM) แล้วส่งการทำงานกลับ

Algorithm GEN_CHARSTRING(charstruct **charList*, substruct **subList*, stemstruct **stemList*, LPSTR *lpbyte*, POINT *lsbp*, POINT *currentpoint*, int *charwidth*, BYTE *basecode*, BYTE *accentcode*)

```
seqstruct *seqptr;
cmdstruct *cmdptr;
BOOL bclosepath;

if (null seqList && null ptList)
  {
  if (SAVESUBROUTINE)
    {
    if (null stemList)
       {
       if (want to save CLOSEPATH and/or ENDCHAR)
          place CLOSEPATH and/or ENDCHAR follow with RETURN command in charstring byte;
       }
    else
       {
       SAVE_STEM(stemList, subList, lpbyte, lsbp);
       place RETURN command in charstring byte;
       }
    }
  else if (charwidth > 0)
       {
       encoding number 0 & charwidth;
       place number follow with HSBW and ENDCHAR command in charstring byte;
       }
  return lpbyte;
  }
if (SAVESUBROUTINE)
  {
  if (want to save CLOSEPATH at first command)
    place CLOSEPATH command in charstring byte;
  else if (want to save MOVETO at first command)
```

```
            {
             encoding number of distance x,y;
             place number follow with RMOVETO command in charstring byte;
            }
    }
else if (SAVENORMALCHARACTER)
        {
        if (this is a accent character)
          {
          find x left side bearing of accent character;
          encoding x left side bearing, origin of accent character;
          convert accent&base character code to PostScript standard;
          encoding accent&base character code;
          place encoding number follow with SEAC command in charstring byte;
          return lpbyte;
          }
        encoding x left side bearing & character width;
        place encoding number & HSBW command in charstring byte;
        }
if (not null stemList)
   SAVE_STEM(stemList, subList, lpbyte, lsbp);
set seqptr to first seqstruct node of seqList;
while (not end of seqList)
        {
        bclosepath = FALSE;
        set cmdptr to first node of cmdList;
        if (cmdptr is not a subroutine(hsub=0) || cmdptr is a "CALL" subroutine that has not moveto at
           first command)
           {
           find & encoding relative distance x, y from current point to starting point of first cmdptr node;
           place encoding number & MOVETO command in charstring byte;
           }
        while (not end of cmdList)
              {
              if (cmdptr is a "CALL" command)
                {
                get subroutine number from subList;
                encoding subroutine number;
                place subroutine number & CALLSUBR command in charstring byte;
                }
              else if (cmdptr is a command of subroutine(hsub!=0))
                    {
                    switch (command of cmdptr)
                          {
                          case LINE:
                          case CURVE:
                               set current point to ending point of cmdstruct node;
                               break;
```

```
                        case CLOSEPATH:
                            bclosepath = TRUE;
                            break;

                        case ENDCHAR:
                            return lpbyte;
                        }
                    }
                else
                    {
                    switch (command of cmdptr)
                            {
                            case LINE:
                                determine appropriate RLINETO, VLINETO, HLINETO command;
                                find and encoding relative distance x, y of parameter of command;
                                place encoding number & appropriate command;
                                set current point to ending point of cmdstruct node;
                                break;

                            case CURVE:
                                determine appropriate RRCURVETO, VHCURVETO, HVCURVETO
                                command;
                                find and encoding relative distance x, y of parameter of command;
                                place encoding number & appropriate command;
                                set current point to ending point of cmdstruct node;
                                break;
                            }
                    }
                next cmdptr node;
                }
        if (!bclosepath &&
            (SAVENORMALCHARACTER || (SAVESUBROUTINE && not a last seqstruct in
            seqList)))
          place CLOSEPATH command in charstring byte;
          next seqstruct node;
        }
if (SAVENORMALCHARACTER)
  {
  place ENDCHAR command in charstring byte;
  return lpbyte;
  }
if (SAVESUBROUTINE)
  {
  if (want to save CLOSEPATH at last)
     place CLOSEPATH command in charstring byte;
  if (want to save ENDCHAR at last)
     place ENDCHAR command in charstring byte;
  }
place RETURN command in charstring byte;
return lpbyte;
```

Algorithm SAVE_STEM(stemstruct **stemList*, substruct **subList*, LPSTR *lpbyte*, POINT *lsbp*)

stemstruct **stemptr*;

set *stemptr* to first stemstruct node of *stemList*;
while (not end of *stemList*)
{
if (*stemptr* is a CALL command node)
{
get subroutine number from *subList*;
encoding subroutine number;
place encoding number & CALLSUBR command;
}
else if (*stemptr* is not a part of subroutine(*hsub*=0))
{
find x or y coordinate of stem relative to lsbp and encoding it;
encoding stem width or height;
palce encoding number & appropriate stem command(HSTEM or VSTEM) in
charstring byte;
}
next stemstruct node;
}

5.3.2 ขั้นตอนการทำงานการตรวจสอบและปรับทิศทางของเส้นให้ถูกต้องตามข้อกำหนดของโปรแกรมแบบอักษรโพสต์สคริปต์ประเภทที่ 1 การทำงานขั้นตอนนี้แสดงดังขั้นตอนวิธี TUNE_DIRECTION ซึ่งจะทำการตรวจสอบและปรับทิศทางของส่วนของตัวอักษรต่างๆโดยอัตโนมัติ ดังนี้

เริ่มการทำงานจากการหาจำนวนของบัพ seqstruct ของรายการเชื่อมโยง seqList ของ charList เพื่อใช้เป็นขนาดของตัวแปรแถวลำดับ seqtable ที่สร้างขึ้นเพื่อใช้เก็บข้อมูลเฉพาะของแต่ละบัพ seqstruct ประกอบด้วยทิศทางปัจจุบัน(ตามเข็มหรือทวนเข็ม) จำนวนของส่วนของตัวอักษรที่ล้อมรอบอยู่และขนาดกรอบสี่เหลี่ยมใหญ่สุดของของส่วนของตัวอักษรของบัพ seqstruct นั้นๆ แล้วทำการสร้างตัวแปรแถวลำดับสำหรับเก็บจุดคู่ลำดับของแต่ละส่วนของตัวอักษร และสำหรับเก็บจำนวนของจุดคู่ลำดับของแต่ละส่วนของตัวอักษร จากนั้นจะนำบัพ seqstruct มาพิจารณาทีละบัพจนกว่าจะหมด

ขั้นตอนการพิจารณาแต่ละบัพ seqstruct นั้น จะเริ่มจากการหาจุดเริ่มต้นและจุดสิ้นสุดของบัพ cmdstruct ของรายการเชื่อมโยง cmdList ของ seqstruct นั้นๆ โดยถ้าเป็นเส้นตรงจะใช้จุดเริ่มต้นและจุดสิ้นสุด แต่ถ้าเป็นเส้นโค้งจะต้องคำนวณหาชุดของเส้นตรงที่ใช้แสดงเส้นโค้งก่อนเก็บจุดต่างๆเหล่านั้นลงในตัวแปรแถวลำดับสำหรับเก็บจุดคู่ลำดับ และเก็บจำนวนของจุดเหล่านี้ในตัวแปรแถวลำดับสำหรับเก็บจำนวนของจุด(โดยข้อมูลดังกล่าวของแต่ละบัพ seqstruct จะเก็บเรียงติดต่อกันไป) จากจุดคู่ลำดับที่หาได้จะนำมาหากรอบสี่เหลี่ยมใหญ่สุดของส่วนของตัวอักษรเก็บลงในตัวแปรแถวลำดับseqtableหาจุดคู่ลำดับที่มีค่าของพิกัด x ที่มีค่าน้อยที่สุดในบรรดาค่าพิกัด x ของจุดเหล่านั้น แล้วนำจุดก่อนหน้าและจุดที่ตามหลัง

มาพิจารณาจะได้เป็นเส้นตรง 2 เส้น ที่เชื่อมต่อกันตรงจุดดังกล่าว แล้วหาค่าของการครอสเวคเตอร์ของเส้นตรงทั้งสอง ถ้าค่าที่ได้น้อยกว่า 0 แสดงว่าส่วนของเส้นตรงนี้ มีทิศทางตามเข็มนาฬิกา แต่ถ้ามากกว่า 0 แสดงว่ามีทิศทางทวนเข็มนาฬิกา แต่ถ้าค่าที่ได้เท่ากับ 0 จะต้องพิจารณาจากค่าจุดเริ่มต้นและสิ้นสุดของค่าพิกัด y ถ้าค่าเพิ่มจากน้อยไปหามาก แสดงว่ามีทิศทางตามเข็มนาฬิกา และมีทิศทางทวนเข็มนาฬิกาในทางตรงกันข้าม เก็บผลลัพธ์ที่ได้ในตัวแปรแถวลำดับ seqtable

เมื่อได้ข้อมูลต่างๆครบทุกบัพ seqstruct แล้ว จะเริ่มพิจารณาหาตำแหน่งของส่วนของตัวอักษรและทิศทางที่ถูกต้อง โดยกำหนดตัวแปร seqptr ให้ชี้ไปยังบัพ seqstruct แรกของรายการเชื่อมโยง seqList นำข้อมูลของตัวแปรแถวลำดับ seqtable แถวแรกโดยยึดเป็นแถวหลัก นำมาพิจารณาเปรียบเทียบกับข้อมูลแถวอื่นๆ เพื่อหาจำนวนของส่วนของวัตถุที่ล้อมรอบอยู่ ดังนี้

ถ้าผลลัพธ์ของการ intersect กรอบสี่เหลี่ยมของ 2 แถว ทำให้เกิดสี่เหลี่ยมว่างเปล่า จะนำตัวแปรแถวลำดับ seqtable แถวถัดไปมาพิจารณา แต่ถ้าไม่ จะทำการทดสอบการอยู่ใน-นอกของจุดของแถวหลักกับรูปหลายเหลี่ยมของจุดที่กำหนดโดยแถวอื่นๆโดยใช้วิธีการนับคู่-คี่ ถ้าผลลัพธ์เป็นคี่แสดงว่าจุดของส่วนของตัวอักษรของแถวหลักอยู่ภายในรูหลายเหลี่ยม จะทำการเพิ่มค่า nCount ของแถวหลักนั้น และนำแถวถัดไปมาพิจารณาจนหมด ค่าของ nCount ของแถวหลักที่ได้ จะเป็นจำนวนของส่วนของวัตถุที่ล้อมรอบอยู่ ซึ่งถ้ามีค่าเป็นจำนวนคู่ ทิศทางของเส้นจะต้องทวนเข็มนาฬิกา แต่ถ้าเป็นจำนวนคี่ ทิศทางของเส้นจะต้องตามเข็มนาฬิกา

จากผลที่ได้นำไปเปรียบเทียบกับทิศทางปัจจุบันที่หาไว้ก่อนหน้าแล้ว(direction ของ seqtable ที่เป็นแถวหลัก) ถ้ามีค่าไม่ตรงกันจะต้องทำการกลับทิศทางทั้งบัพ cmdstruct ของรายการเชื่อมโยง cmdList ของบัพ seqstruct ที่ชี้โดย seqptr รวมทั้งจุดที่เก็บอยู่ในตัวแปรแถวลำดับของจุดคู่ลำดับ แต่ถ้าไม่สามารถกลับทิศทางได้ แสดงว่าผู้ใช้ได้เรียกใช้โปรแกรมย่อยที่มีทิศทางผิด จะแสดงข้อความเตือนขึ้นมา จากนั้นจึงเลื่อน seqptr ให้ชี้ไปยังบัพ seqstruct ถัดไปเปลี่ยนตัวแปรแถวลำดับ seqtable แถวหลักให้เป็นแถวถัดมาแล้วกลับไปเริ่มพิจารณาใหม่จนกว่าจะหมด

Algorithm TUNE_DIRECTION(charstruct **charList*)

```
struct seqtab {
                int direction;
                int nCount;
                RECT rect;
            } *seqtable;
int i, j;
seqstruct *seqptr;

find number of seqstruct;
allocate seqtable array;
if (not show bitmap character)
```

```
        {
        allocate array of number of point;
        allocate array of point;
        }
i=0;
while (not end of seqList)
        {
        find&store vertice point of LINE or CURVE(in series of LINE) in array of point;
        store number of point in array of number of point;
        find&store RECTANGLE of these point in rect of seqtable[i];
        find point in array of point that x-coordinate is the left most;
        from this point, find line that join at this point;
        find cross product of these two line;
        if (result < 0)
           store CLOCLWISE in direction of seqtable[i];
      else
           store COUNTERCLOCLWISE in direction of seqtable[i];
        next seqstruct node;
        next i;
        }
set seqptr to first seqstruct node of seqList;
for (i=0;i<number of sequence;i++)
    {
    for (j=0;j<number of sequence;j++)
        {
        if (j!= i && INTERSECTRECTANGLE(seqtable[i].rect, seqtable[j].rect) is not empty)
          {
          test each point of sequence i is in or out polygon that form by point of sequence j(use count
          even-odd method);
          if (result is in)
             increment nCount of seqtable[i] by 1;
          }
        }
    if (seqtab[i].nCount % 2 == 0)
        require direction is COUNTERCLOCKWISE;
    else
        require direction is CLOCKWISE;
    if (seqtab[i].direction != require direction)
        {
        if (can reverse order of cmdstruct node of seqptr)
          {
          REVERSE_CMD(seqptr);
          reverse array of point of sequence i;
          }
        else display message "invalid direction";
        }
    next seqstruct node;
    }
free seqtable array;
if (not show bitmap character)
```

```
{
free array of point;
free array of number of point;
}
```

5.3.3 ขั้นตอนการทำงานการเพิ่มเส้นตรงหรือเส้นโค้งเข้าไปยังรายการเชื่อมโยง charList จะต้องทำการเพิ่มบัพ cmdstruct เข้าไปในรายการเชื่อมโยง selList ก่อน แล้วจึงรวมรายการเชื่อมโยง selList เข้าไปยัง charList ดังขั้นตอนวิธี ADD_LINE_CURVE ซึ่งมีขั้นตอนดังนี้

เริ่มจากการค้นหาหรือสร้างบัพ ptstruct จากรายการเชื่อมโยง ptList ของ selList สำหรับแต่ละจุดเริ่มต้น จุดสิ้นสุด และ/หรือจุดควบคุมที่ 2 และ 3 ในกรณีเป็นเส้นโค้งก่อน ทำการสร้างบัพ cmdstruct กำหนดค่าตัวชี้ pt1 pt2 pt3 และ pt4 ให้ชี้ไปยังบัพ ptstruct ที่ได้ กำหนดคำสั่ง LINE หรือ CURVE และเพิ่มค่า pcount ของบัพ ptstruct ที่ชี้โดย pt1 และ pt4 และ/หรือ ccount ของบัพ ptstruct ที่ชี้โดย pt2 และ pt3 ในกรณีเป็นเส้นโค้ง แล้วเพิ่มบัพ cmdstruct เข้าไปยังรายการเชื่อมโยง cmdList ของบัพ seqstruct ที่สร้างขึ้นใหม่ในรายการเชื่อมโยง selList จากนั้นจึงเรียกขั้นตอนวิธี COMBINE_SEL_2_CHARLIST เพื่อรวมบัพต่างๆให้เข้าไปอยู่ในรายการเชื่อมโยง charList

Algorithm ADD_LINE_CURVE(charstruct **charList*, charstruct **selList*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*, int *x4*, int *y4*)

ptstruct **pt1ptr*, **pt2ptr*, **pt3ptr*, **pt4ptr*;

find or create ptstruct node that point by *pt1ptr* in ptList of *selList* which has the same coordinate *x1*, *y1*;
find or create ptstruct node that point by *pt4ptr* in ptList of *selList* which has the same coordinate *x4*, *y4*;
if (add curve)
  {
  find or create ptstruct node that point by *pt2ptr* in ptList of *selList* which has the same coordinate *x2*, *y2*;
  find or create ptstruct node that point by *pt3ptr* in ptList of *selList* which has the same coordinate *x3*, *y3*;
  }
create new cmdstruct node;
set *pt1*, *pt2*, *pt3*, *pt4* of cmdstruct node to *pt1ptr*, *pt2ptr*, *pt3ptr*, *pt4ptr*;
set *cmd* of cmdstruct node to LINE or CURVE;
increment *pcount* of ptstruct node that point by *pt1ptr* and *pt4ptr*;
if (add curve)
  increment *ccount* of ptstruct node that point by *pt2ptr* and *pt3ptr*;
create new seqstruct node and add cmdstruct node into it;
add seqstruct node to seqList of *selList*;
COMBINE_SEL_2_CHARLIST(*charList*, *selList*);

5.3.4 ขั้นตอนการทำงานการเปลี่ยนแปลงรูปร่างของเส้นต่างๆที่ได้เลือกไว้ การทำงานจะต้องหาบัพ cmdstruct ของเส้นที่ต้องการจากตำแหน่งของเมาส์ ดึงบัพออกจากรายการเชื่อมโยง charList ไปไว้ที่

selList แล้วจึงเปลี่ยนคู่ลำดับ x และ y ของจุดที่ต้องการตามการเคลื่อนที่ของเมาส์(ดูรูปที่ 5.22 และ 5.23) แล้วจึงรวมรายการเชื่อมโยง selList เข้าไปใน charList การทำงานการดึงบัพ cmdstruct ของเส้นที่ต้องการ แสดงดังขั้นตอนวิธี **SHAPE_SEGMENT** ดังนี้

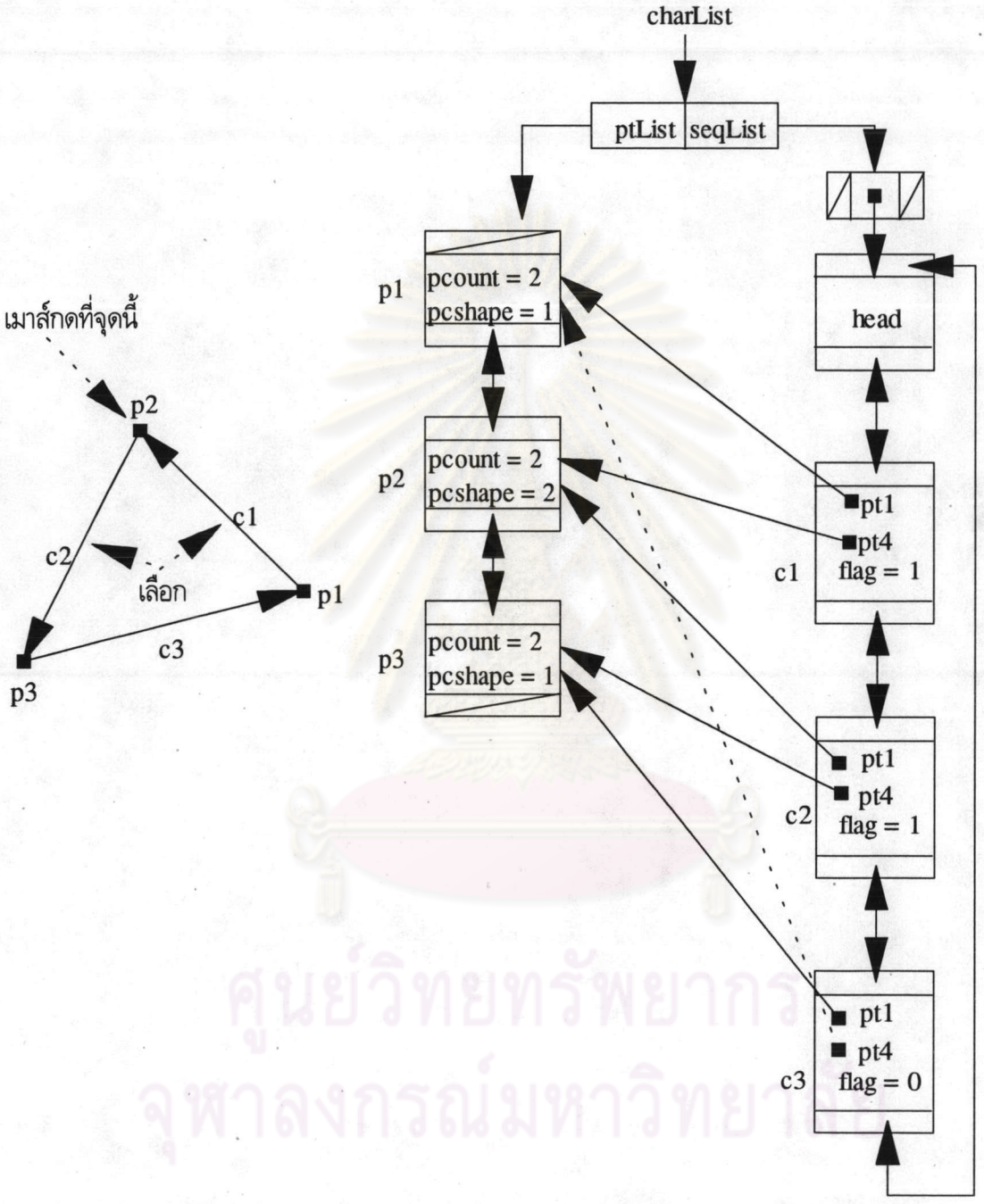
นำบัพ cmdstruct ทุกบัพเฉพาะที่เลือกไว้(flag=1) และไม่ใช่ส่วนของโปรแกรมย่อย (hsub=0) มาพิจารณาว่า ตำแหน่งของเมาส์ชี้ไปที่ตำแหน่งเริ่มต้นหรือสุดท้าย หรือจุดควบคุมที่ 2 หรือ 3 ในกรณีเป็นเส้นโค้งหรือไม่ ถ้าไม่ใช่จะนำบัพ cmdstruct ถัดไปมาพิจารณาจนกว่าจะหมด แล้วเริ่มนำบัพ cmdstruct ของ seqstruct ถัดไปมาพิจารณาจนกว่าจะพบหรือหมด seqList ซึ่งแสดงว่าไม่พบบัพใดใดที่ต้องการเปลี่ยนแปลงก็จะส่งการทำงานกลับ

ถ้าพบว่าเมาส์ชี้ไปที่ตำแหน่งเริ่มต้น จะทำการสร้างและเพิ่มบัพ seqstruct เข้าไปในรายการเชื่อมโยง seqList ของ selList เปลี่ยนตัวชี้ของ pt1 ให้ชี้ไปยังบัพ ptstruct ใหม่ของselList โดยเรียกขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** จากนั้นจะทำการตรวจสอบบัพ cmdstruct บัพก่อนหน้าว่าถูกเลือกและไม่ใช่โปรแกรมย่อยหรือไม่ ถ้าใช่จะเรียกขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** เพื่อเปลี่ยนตัวชี้ของ pt4 ให้ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList ดึงบัพ cmdstruct บัพก่อนหน้าออกจากรายการเชื่อมโยง cmdList ของ charList และเพิ่มใน cmdList ของ selList ก่อน จากนั้นจึงดึงบัพ cmdstruct ออกและเพิ่มเป็นบัพสุดท้ายต่อกับบัพก่อนหน้า แล้วส่งการทำงานกลับ

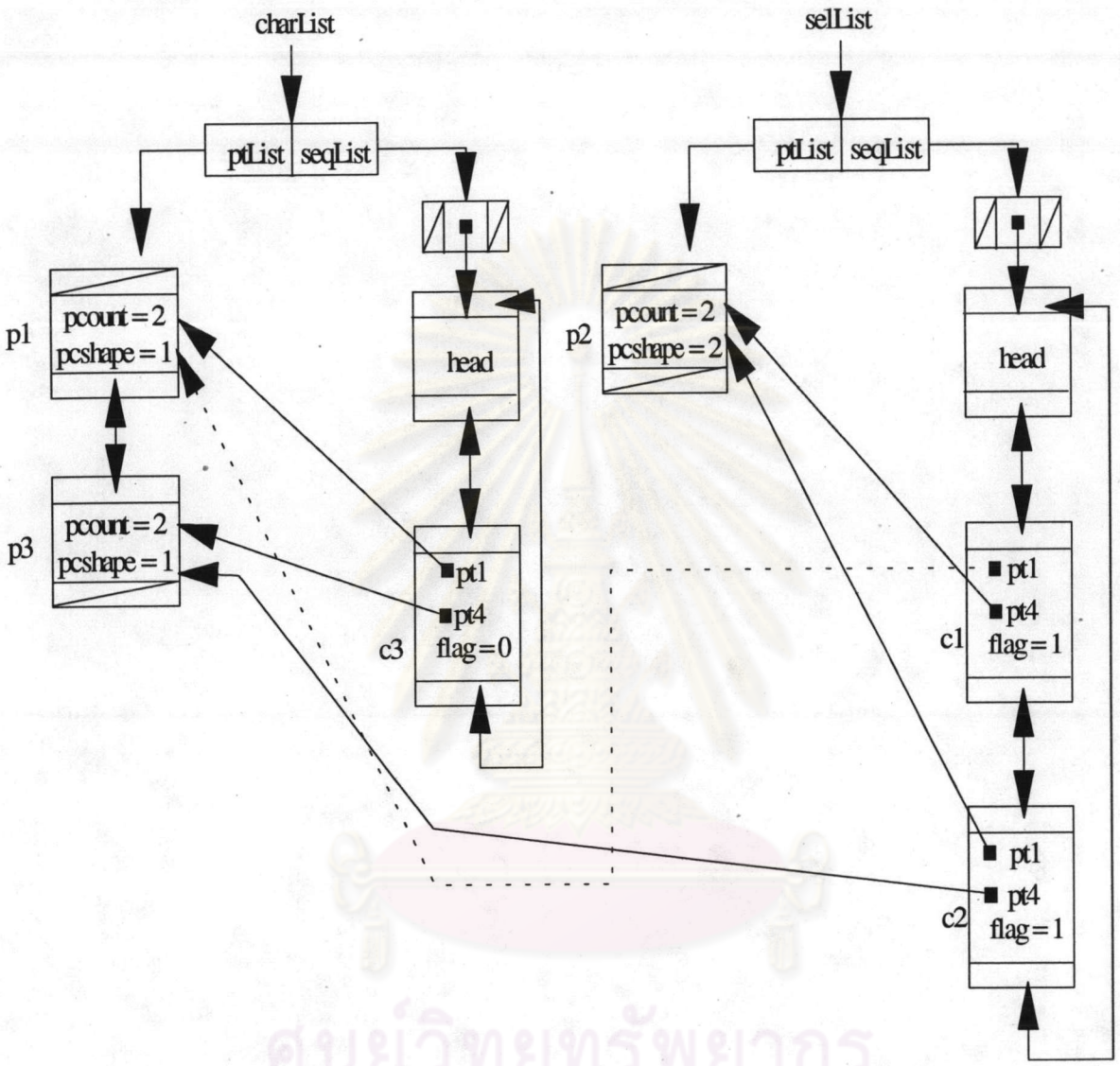
ถ้าเมาส์ชี้ไปที่ตำแหน่งสุดท้าย จะทำการสร้างและเพิ่มบัพ seqstruct เข้าไปในรายการเชื่อมโยง seqList ของ selList เปลี่ยนตัวชี้ของ pt4 ให้ชี้ไปยังบัพ ptstruct ใหม่ของ selList โดยเรียกขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** ดึงบัพ cmdstruct ออกจากรายการเชื่อมโยง cmdList ของ charList และเพิ่มใน cmdList ของ selList จากนั้นจะทำการตรวจสอบบัพ cmdstruct บัพถัดไปว่าถูกเลือกและไม่ใช่โปรแกรมย่อยหรือไม่ ถ้าใช่จะเรียกขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** เพื่อเปลี่ยนตัวชี้ของ pt1 ให้ชี้ไปยังบัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList ดึงบัพ cmdstruct ออกจากรายการเชื่อมโยง cmdList ของ charList และเพิ่มใน cmdList ของ selList เป็นบัพสุดท้าย แล้วส่งการทำงานกลับ

ถ้าบัพ cmdstruct เป็นบัพของคำสั่ง CURVE จะตรวจสอบตำแหน่งของเมาส์กับจุดควบคุมที่ 2 หรือ 3 ถ้าเมาส์ชี้ไปที่จุดดังกล่าว จะทำการสร้างและเพิ่มบัพ seqstruct เข้าไปในรายการเชื่อมโยง seqList ของ selList เปลี่ยนตัวชี้ของ pt2 หรือ pt3 ให้ชี้ไปยังบัพ ptstruct ใหม่ของ selList โดยเรียกขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** ดึงบัพ cmdstruct ออกจากรายการเชื่อมโยง cmdList ของ charList และเพิ่มใน cmdList ของ selList แล้วส่งการทำงานกลับ

สำหรับการทำงานของขั้นตอนวิธี **CHANGE_SHAPE_PT_LINK** เริ่มจากการค้นหาบัพ ptstruct ในรายการเชื่อมโยง ptList ของ selList ที่มีคู่ลำดับ x และ y ตรงกับที่เมาส์ชี้ ถ้าไม่พบจะทำการสร้างและเพิ่มบัพ



รูปที่ 5.22 แสดงรายการเชื่อมโยง charList ก่อนการดึงบัพ cmdstruct เพื่อทำการเปลี่ยนแปลงรูปร่าง



รูปที่ 5.23 แสดงรายการเชื่อมโยง charList และ selList เมื่อกดเมาส์ที่จุด p2

ptstruct เข้าไปในรายการเชื่อมโยง ptList ก่อน จากนั้นจะทำการเพิ่มค่าตัวนับ pcount หรือ pcshape ในกรณีที่เป็นจุดเริ่มต้นหรือจุดสิ้นสุด หรือ ccount หรือ ccshape กรณีที่เป็นจุดควบคุมที่ 2 หรือ 3 ของบัพ ptstruct ในรายการเชื่อมโยง ptList ของ selList และลดค่าตัวนับดังกล่าวของบัพ ptstruct เดิมของรายการเชื่อมโยง ptList ของ charList และถ้าผลรวมของ pcount ccount และ scount ของบัพ ptstruct เดิม มีค่าเท่ากับ 0 แสดงว่าไม่มีบัพ cmdstruct ใดใดของ charList ชี้มาที่บัพ ptstruct นั้น จะทำการทำลายบัพ ptstruct เพื่อคืนหน่วยความจำให้ระบบ จากนั้นจึงทำการเปลี่ยนตัวชี้ของ pt1 หรือ pt2 หรือ pt3 หรือ pt4 ของบัพ cmdstruct แล้วแต่กรณี

Algorithm SHAPE_SEGMENT(charstruct **charList*, charstruct **selList*, POINT *shapepoint*)

```
seqstruct *seqptr;
cmdstruct *cmdptr;

set seqptr to first seqstruct node of seqList of charList;
while (not end of seqList)
      {
      set cmdptr to first cmdstruct node of cmdList;
      while (not end of cmdList)
            {
            if (cmdptr is select(flag=1) && is not a subroutine(hsub=0))
              {
              if (shapepoint point at starting point)
                 {
                 create & add new seqstruct node in seqList of selList;
                 CHANGE_SHAPE_PT_LINK(charList, selList, cmdptr, firstpoint);
                 if (previous cmdstruct node is select && is not a subroutine)
                   {
                   CHANGE_SHAPE_PT_LINK(charList, selList, cmdptr, endpoint);
                   cut&add previous cmdstruct node into new seqstruct node;
                   }
                 cut&add cmdptr node into new seqstruct node;
                 return;
                 }
              else if (shapepoint point at ending point)
                      {
                      create & add new seqstruct node in seqList of selList;
                      CHANGE_SHAPE_PT_LINK(charList, selList, cmdptr, endpoint);
                      cut&add cmdptr node into new seqstruct node;
                      if (next cmdstruct node is select && is not a subroutine)
                        {
                        CHANGE_SHAPE_PT_LINK(charList, selList, cmdptr, firstpoint);
                        cut&add previous cmdstruct node into new seqstruct node;
                        }
                      return;
                 }
              else if (cmdptr is "CURVE" cmd)
                      {
                      if (shapepoint point at second or third control point)
```

```
                    {
                    create & add new seqstruct node in seqList of selList;
                    CHANGE_SHAPE_PT_LINK(charList, selList, cmdptr, endpoint);
                    cut&add cmdptr node into new seqstruct node;
                    return;
                    }
                  }
               }
            next cmdstruct node;
            }
         next seqstruct node;
        }
```

Algorithm CHANGE_SHAPE_PT_LINK(charstruct **charList*, charstruct **selList*, cmdstruct **cmdptr*, char *pos*)

find or create new ptstruct node of ptList of *selList* that has the same as coordinate of ptstruct node that cmdstruct node point to;
if (*pos = first point // end point*)
  {
  increment *pcount, pcshape* of new ptstruct node;
  decrement *pcount, pcshape* of ptstruct node that point by cmdstruct node;
  if (*pcount+ccount+scount* = 0)
    free ptstruct node that point by cmdstruct node;
  change *pt1* or *pt4 of cmdptr* point to new ptstruct node;
  }
else
  {
  increment *ccount, ccshape* of new ptstruct node;
  decrement *ccount, ccshape* of ptstruct node that point by cmdstruct node;
  if (*pcount+ccount+scount* = 0)
    free ptstruct node that point by cmdstruct node;
  change *pt2* or *pt3 of cmdptr* point to new ptstruct node;
  }

5.3.5 ขั้นตอนการทำงานการเลือกเส้นที่ต้องการ แสดงดังขั้นตอนวิธี SELECT โดยการทำงานเริ่มจากนำบัพ cmdstruct แต่ละบัพของรายการเชื่อมโยง charList มาทำการทดสอบ กรณีการเลือกแบบจุดจะทดสอบระยะห่างจากจุดที่เมาส์ชี้อยู่ถึงส่วนของเส้นที่กำหนดโดยบัพ cmdstruct (กรณีเส้นโค้งจะทำการหาจุดต่างๆ ของเส้น เพื่อให้ได้เป็นชุดของเส้นตรงก่อน) ถ้าระยะห่างอยู่ในขอบเขตที่กำหนด แสดงว่าผู้ใช้เลือกหรือยกเลิกการเลือกเส้นที่กำหนดโดยบัพ cmdstruct นี้ ซึ่งถ้าบัพ cmdstruct นี้เป็นส่วนของโปรแกรมย่อย(hsub>0) จะเรียกขั้นตอนวิธี **DO_SUB_SELECT** เพื่อทำการเลือกบัพทุกบัพที่เป็นโปรแกรมย่อยนี้ แต่ถ้าไม่ใช่จะทำการ exclusive OR (XOR) ค่า flag กับ 1 เพื่อใช้แสดงถึงการเลือกหรือยกเลิกการเลือก แล้วทำการเพิ่มหรือลดค่าของ pcshape และ/หรือ ccshape ตามค่าของ flag และคำสั่งของบัพ cmdstruct นั้น แล้วส่งการทำงานกลับ

แต่ถ้าเป็นการเลือกแบบลากกรอบสี่เหลี่ยมล้อมรอบ จะทดสอบจุดเริ่มและจุดสิ้นสุดของเส้นที่กำหนดโดยบัพ cmdstruct ว่าอยู่ภายในกรอบนั้นหรือไม่(กรณีเส้นโค้งจะทำการหาจุดต่างๆของเส้น เพื่อให้ได้เป็นชุดของเส้นตรงก่อน)ซึ่งถ้าอยู่ภายในกรอบแสดงว่าผู้ใช้เลือกหรือยกเลิกการเลือกเส้นที่กำหนดโดยบัพ cmdstruct นี้ จากนั้นขั้นตอนการทำงานจะเหมือนกับการเลือกแบบจุด แต่จะยังไม่ส่งการทำงานกลับจนกว่าจะทำการทดสอบบัพทุกบัพของ cmdstruct แล้ว

สำหรับการทำงานของขั้นตอนวิธี **DO_SUB_SELECT** นั้น จะนำบัพ cmdstruct ทุกบัพที่มีค่าแฮนเดิลตรงกับบัพ cmdstruct ที่พบ มาทำ exclusive OR(XOR) ค่า flag กับ 1 จนครบ แล้วจึงส่งการทำงานกลับ

ในการทำการเพิ่ม/ลดค่าของ pcount และ/หรือ ccshape นั้น แสดงดังขั้นตอนวิธี **INC_SHAPE** และ **DEC_SHAPE** โดยถ้าค่า flag เป็น 1 (หมายถึงการเลือก) จะเรียกใช้ขั้นตอนวิธี **INC_SHAPE** และเรียกใช้ขั้นตอนวิธี **DEC_SHAPE** ในทางตรงกันข้าม การทำงานจะพิจารณาว่าถ้าบัพ cmdstruct เป็นการลากเส้นตรงจะทำการเพิ่ม/ลดค่าของpcshapeของจุดเริ่มและสิ้นสุด แต่ถ้าเป็นการลากเส้นโค้งจะทำการเพิ่ม/ลดค่าของ ccshape ของจุดควบคุมที่ 2 และ 3 ด้วย

Algorithm SELECT(charstruct **charList*, POINT *pt1*, POINT *pt2*)

```
seqstruct *seqptr;
cmdstruct *cmdptr;

set seqptr to first seqstruct node of seqList;
while (not end of seqList)
      {
      set cmdptr to first cmdstruct node of cmdList;
      while (not end of cmdList)
            {
            if (POINTSELECTION)
               {
               if (line or curve that represent by cmdstruct node is near pt1)
                  if (not a subroutine)
                     {
                     XOR flag of cmdstruct node with 1;
                     if (flag = 0(NOTSELECT))
                        DEC_SHAPE(cmdptr);
                     else
                        INC_SHAPE(cmdptr);
                     return;
                     }
                  else
                     {
                     DO_SUB_SELECT(charList, cmdptr->hsub);
                     return;
```

```
                }
            }
          else
            {
            if (start & end point of line or curve is in rectangle define by pt1&pt2)
              if (not a subroutine)
                {
                XOR flag of cmdstruct node with 1;
                if (flag = 0(NOTSELECT))
                  DEC_SHAPE(cmdptr);
                else
                  INC_SHAPE(cmdptr);
                }
              else
                DO_SUB_SELECT(charList, cmdptr->hsub);
            }
          next cmdstruct node;
          }
      next seqstruct node;
      }
```

Algorithm DO_SUB_SELECT(charstruct *_charList_, BYTE _hsub_)

```
for (all cmdstruct node that has hsub = hsub)
      XOR flag of cmdstruct node with 1;
```

Algorithm DEC_SHAPE(cmdstruct *_cmdptr_)

```
if (cmdptr is LINE command)
   decrement pcshape of starting&ending point by 1;
else
   {
   decrement pcshape of starting&ending point by 1;
   decrement ccshape of control point 2&3 by 1;
   }
```

Algorithm INC_SHAPE(cmdstruct *_cmdptr_)

```
if (cmdptr is LINE command)
  increment pcshape of starting&ending point by 1;
else
  {
  increment pcshape of starting&ending point by 1;
  increment ccshape of control point 2&3 by 1;
  }
```

5.3.6 ขั้นตอนการทำงานการเปลี่ยนแปลงทำได้โดยง่าย โดยการดึงบัพ cmdstruct ที่เลือกไว้ออกจากรายการเชื่อมโยง cmdList ของ charList และนำไปใส่ในรายการเชื่อมโยง selList โดยเรียกขั้นตอนวิธี DEL_CHAR_2_SEL จากนั้นทำการเปลี่ยนแปลงค่าคู่ลำดับจากบัพ ptstruct ของรายการเชื่อมโยง ptList ของ selList ตามต้องการ แล้วรวมบัพกลับเข้าไปยังรายการเชื่อมโยง charList ตามเดิม โดยเรียกขั้นตอนวิธี COMBINE_SEL_2_CHARLIST ดังแสดงในขั้นตอนวิธี TRANSFORM

Algorithm TRANSFORM(charstruct **charList*, charstruct **selList*)

ptsruct **ptptr*;

DEL_CHAR_2_SEL(*charList, selList*);
set *ptptr* to first node of ptList of *selList*;
while (not end of ptList)
    multiply coordinate with translation matrix;
COMBINE_SEL_2_CHARLIST(*charList, selList*);

5.3.7 ขั้นตอนการคัดลอกส่วนของตัวอักษรที่เลือกไปยังคลิปบอร์ด แสดงดังขั้นตอนวิธี COPY โดยเริ่มการทำงานจากการดึงบัพ cmdstruct ที่เลือกไว้ออกจากรายการเชื่อมโยง charList และนำไปใส่ในรายการเชื่อมโยง selList โดยเรียกขั้นตอนวิธี DEL_CHAR_2_SEL แล้วทำการสร้าง charstring byte จากรายการเชื่อมโยง selList โดยเรียกขั้นตอนวิธี GEN_CHARSTRING คัดลอกข้อมูล charstring byte ไปยังคลิปบอร์ด แล้วรวมบัพกลับเข้าไปยังรายการเชื่อมโยง charList ตามเดิม โดยเรียกขั้นตอนวิธี COMBINE_SEL_2_CHARLIST

Algorithm COPY(charstruct **charList*, charstruct **selList*)

delete select segment from *charList* to *selList* by CALL DEL_CHAR_2_SEL;
generate charstring byte from *selList* by CALL GEN_CHARSTRING;
copy charstring byte to clipboard;
combine *selList* to *charList* by CALL COMBINE_SEL_2_CHARLIST;

5.3.8 ขั้นตอนการทำงานการตัดส่วนของตัวอักษรที่เลือกไปยังคลิปบอร์ด แสดงดังขั้นตอนวิธี CUT ซึ่งมีการทำงานเหมือนกับขั้นตอนวิธี COPY เพียงแต่ไม่ต้องเรียกขั้นตอนวิธี COMBINE_SEL_2_CHARLIST เพื่อรวมบัพกลับ แต่ทำการทำลายบัพเหล่านั้นใน selList เพื่อคืนหน่วยความจำให้กับระบบ

Algorithm CUT(charstruct **charList*, charstruct **selList*)

delete select segment from *charList* to *selList* by CALL DEL_CHAR_2_SEL;
generate charstring byte from *selList* by CALL GEN_CHARSTRING;
copy charstring byte to clipboard;

5.3.9 ขั้นตอนการทำงานการลบส่วนของตัวอักษรที่เลือก แสดงดังขั้นตอนวิธี **CLEAR** ซึ่งมีการทำงานเหมือนกับขั้นตอนวิธี COPY เพียงแต่ไม่ต้องเรียกขั้นตอนวิธี GEN_CHARSTRING เพื่อสร้าง charstring byte

Algorithm CLEAR(charstruct **charList*, charstruct **selList*)

delete select segment from *charList* to *selList* by CALL DEL_CHAR_2_SEL;

5.3.10 ขั้นตอนการทำงานการคัดลอกข้อมูลจากคลิปบอร์ด แสดงดังขั้นตอนวิธี PASTE โดยเริ่มการทำงานจากการคัดลอก charstring byte จากคลิปบอร์ด สร้างรายการเชื่อมโยง selList จาก charstring byte นั้น โดยเรียกขั้นตอนวิธี GEN_CHARLIST แล้วจึงเรียกขั้นตอนวิธี COMBINE_SEL_2_CHARLIST เพื่อรวมรายการเชื่อมโยง selList เข้าไปยัง charList

Algorithm PASTE(charstruct **charList*, charstruct **selList*)

copy charstring byte from clipboard;
generate *selList* from charstring byte by CALL GEN_CHARLIST;
combine *selList* to *charList* by CALL COMBINE_SEL_2_CHARLIST;