บทที่ 4

เครื่องมือทางซอฟต์แวร์ (Software tool)

4.1 วัตถุประสงค์

ในบทนี้จะอธิบายถึง เครื่องมือทางซอฟต์แวร์ส่วนต่าง ๆ ที่นำมาประกอบกันเป็นโปรแกรม File Editor และ Screen Generator ที่ได้กล่าวแล้วในบทที่ 3 ซึ่งเครื่องมือเหล่านี้ถูกออกแบบ และพัฒนาขึ้นเอง เพื่อให้สามารถเรียกใช้จากโปรแกรมอื่น ๆ ได้

เครื่องมือทางซอฟต์แวร์ หมายถึง โปรแกรมย่อยที่เขียนขึ้นเพื่อให้ทำงาน เฉพาะหน้าที่หนึ่ง ๆ ตามที่ได้ออกแบบไว้ โดยให้มีความสามารถในการส่งผ่านข้อมูล ที่จำเป็นต้องใช้หรือข้อมูลที่จะได้กลับคืนมาหลังจากจบการทำงานนั้น ๆ ซึ่งการแยกโปรแกรมออกเป็นส่วน ๆ เช่นนี้ จะทำให้การพัฒนา และทดสอบโปรแกรมทำได้อย่างรวดเร็ว เนื่องจากสามารถแบ่งทดสอบได้เป็นส่วน ๆ เพื่อสามารถทำงานได้ตามฟังก์ชันที่ออกแบบไว้ จึงรวบรวมเป็นเครื่องมือทางซอฟต์แวร์ขึ้น

ในส่วนของเครื่องมือทางซอฟต์แวร์ ที่ใช้ประกอบกันเป็นโปรแกรม File Editor และ Screen Generator จะแบ่งออกเป็น 2 ส่วนใหญ่ ๆ ตามหน้าที่เฉพาะที่ออกแบบไว้ คือ

ก. Line Editor

ข. Screen Editor

หน้าที่หลักของเครื่องมือทางซอฟต์แวร์แต่ละส่วน อธิบายได้อย่างย่อๆ คือ

Line Editor เป็นเครื่องมือทางซอฟต์แวร์ ที่ใช้สำหรับ รับ และ/หรือ แก้ไข ข้อความไทย/อังกฤษ ภายใน 1 บรรทัด

Screen Editor เป็นเครื่องมือทางซอฟต์แวร์ ที่ใช้สำหรับ รับ และ/หรือ แก้ไข ข้อมูล (ไทย/อังกฤษ) ของระเบียนในแฟ้มข้อมูล 1 ระเบียน

นอกจากนี้ ยังมีส่วนช่วยเหลือ คือ Graphic Module ที่ช่วยให้การแสดงผลของโปรแกรมไม่ขึ้นกับตัวสร้างอักขระจากหน่วยความจำถาวร (ROM Character Generator) ของวงจรภาคแสดง (Display Adapter) โดยโปรแกรมส่วนนี้ไปแทนที่การแสดงผลของ

เทอร์โบปาสคาลเดิม เพื่อให้สามารถแสดงผลแบบกราฟฟิกได้ โปรแกรมส่วนนี้จะเปลี่ยนไปตามวงจรภาคแสดงแบบต่างๆ ที่ได้สร้างขึ้นนี้ใช้กับ Hercules Graphics Card [18] ส่วนนี้ไม่ใช่ส่วนหลักของบรรณกรแฟ้มข้อมูล จึงไม่ขอกล่าวรายละเอียดไว้ในวิทยานิพนธ์ฉบับนี้

เนื้อหาในบทนี้ จะแบ่งเป็น 3 หัวข้อตามส่วนหลัก ๆ ดังที่ได้กล่าวมาแล้ว โดยแต่ละส่วนจะกล่าวถึง

- หน้าที่อย่างละเอียด
- ส่วนประกอบของโปรแกรมที่เกี่ยวข้อง
- รายละเอียดของส่วนประกอบต่าง ๆ
- การนำไปใช้งาน

4.2 Line Editor

หน้าที่หลักของ Line Editor คือ รับ และ/หรือ แก้ไขข้อความ ไทย/อังกฤษ ภายใน 1 บรรทัด

ก่อนที่จะกล่าวถึงหน้าที่อย่างละเอียดของ Line Editor จะขอกล่าวถึงการรับข้อมูลโดยทั่ว ๆ ไปเสียก่อน เพื่อเป็นการเปรียบเทียบ ปกติการรับข้อมูล 1 ข้อมูลในภาษาปาสคาลจะมีลักษณะ ดังนี้

```
Write   ('Enter a line.');
Readln  (DataLine);
```

โดยที่ Dataline เป็นตัวแปรชนิดข้อความ (เช่น Dataline : String [80]) write และ readln เป็นคำสั่งการทำงานทางด้านการแสดง และรับข้อมูลพื้นฐานของภาษาปาสคาล

เมื่อโปรแกรมทำงานมาถึง 2 คำสั่งนี้ บนจอภาพจะปรากฏข้อความเตือนให้ผู้ใช้โปรแกรมนี้ (ต่อไปจะเรียกว่า ผู้ใช้) ใส่ข้อมูลได้แล้ว ('Enter a line') และโปรแกรมจะรอรับข้อมูลที่ผู้ใช้ป้อนผ่านแป้นพิมพ์ สมมติว่าผู้ใช้ป้อนข้อมูลเสร็จ ก็จะกดแป้น [Enter] หรือ [Return] แล้วแต่เครื่องคอมพิวเตอร์กำหนด (ในที่นี้ เครื่องหมาย [ ] ให้หมายถึงการกดแป้นพิมพ์ตามสัญลักษณ์นั้น 1 ครั้ง) โปรแกรมก็จะนำข้อมูลที่ผู้ใช้ป้อนผ่านแป้นพิมพ์ทั้งหมด ไปเก็บไว้ที่ตัวแปร Dataline ซึ่งขั้นตอนของการรับข้อมูลก็เป็นอันเสร็จสิ้น

ในระหว่างที่มีการป้อนข้อมูล ดังที่กล่าวมาแล้ว เช่น ผู้พิมพ์ป้อนข้อมูลว่า

This is a test Line.

แต่ยังไม่กด [Enter] ผู้ใช้มาตรวจทานแล้วพบว่า Line ควรจะเป็น line (มีการป้อนข้อมูลผิดจากอักษรตัวเล็กเป็นอักษรตัวใหญ่) ผู้ใช้สามารถแก้ไขตัวที่ผิดได้โดย กด [Back Space] ลบ .eniL ทีละตัวตามลำดับ แล้วพิมพ์ line เข้าไปใหม่ จากนั้น จึงกด [Enter] ข้อมูลที่ได้ก็เป็นอันถูกต้อง

ถ้าสมมติว่า ผู้ใช้ป้อนข้อมูล

This is a test Line.

และได้กด [Enter] ทันทีเมื่อป้อนเสร็จ ผู้ใช้จะไม่สามารถมาแก้ไขตัว Line ที่ผิดได้โดยตรง จะต้องมีวิธีการทางโปรแกรมอีกหลายคำสั่ง

ประเด็นที่กล่าวถึงมาแล้ว เป็นการรับข้อมูลภาษาอังกฤษเท่านั้น ซึ่งสามารถ รับและแก้ไขข้อมูลได้ถ้ายังไม่กด [Enter] แต่ถ้าป้อนข้อมูลผิด และกด [Enter] แล้วจะแก้ไขไม่ได้

ถ้าต้องการที่จะรับข้อมูลได้ทั้งไทย และอังกฤษ มีผู้ดัดแปลงไมโครคอมพิวเตอร์ ทั้งฮาร์ดแวร์ และ ซอฟต์แวร์ หลาย ๆ วิธี ทั้งนี้พอจะประมวลได้หลัก ๆ เป็น 3 วิธี คือ

ก. ภาษาไทยชนิด 8 บรรทัด

ข. ภาษาไทยชนิด 25 บรรทัด

ค. ภาษาไทยชนิดปรับจำนวนบรรทัดได้

ต่อไปจะขอกล่าวถึงวิธีการของภาษาไทยทั้ง 3 ชนิดนี้อย่างย่อ ๆ

วิธีการแรกคือ วิธีการของภาษาไทย 8 บรรทัด วิธีการนี้ สามารถทำได้โดย ดัดแปลงตัวสร้างอักขระในหน่วยความจำถาวรให้มีอักขระภาษาไทย (แก้ ROM character generator, ต่อไปจะเรียกเพียงสั้น ๆ ว่า "ดัดแปลงตัวสร้างอักขระ" หรือ "แก้รอม") โดยที่อักขระภาษาไทยที่แก้ไขนั้น จะมีรหัสในช่วงรหัสตั้งแต่ 160 ถึง 255 (หรือ A0 ถึง FF Hexadecimal) และแก้ระบบปฏิบัติการ (operating system) ในส่วนไบออส (BIOS, Basic Input Output System) การแสดงภาษาไทย 8 บรรทัดนี้ จะต้องมีการผสมอักขระของสระบนกับวรรณยุกต์ หรือสระบนกับไม้ทัณฑฆาต

วิธีการของภาษาไทย 25 บรรทัด วิธีการนี้จะต้องดัดแปลงแผงวงจรภาคแสดง (แก้ Hercules Monochrome Adapter) และแก้ระบบปฏิบัติการเช่นกัน แต่แตกต่างจากวิธีการ

แรก

วิธีการของภาษาไทยชนิดปรับบรรทัดได้ วิธีการนี้ใช้เทคนิคทางกราฟฟิค ซึ่งใช้โดยตรงได้กับแผงวงจร Hercules Graphic Adapter หรือ Colour Graphic Adapter

สำหรับรายละเอียด ข้อดี และข้อเสียของแต่ละวิธี จะได้กล่าวถึงต่อไป

วิธีการของภาษาไทย 8 บรรทัดนั้น จะให้ 1 บรรทัดของภาษาไทย เท่ากับการแสดงภาษาอังกฤษ 3 บรรทัด เพื่อแสดงอักขระภาษาไทยในระดับบน กลาง และล่างตามลำดับ ดังนั้นบนจอภาพปกติ (ภาษาอังกฤษ) ขนาด 25 บรรทัด จึงสามารถแสดงภาษาไทยได้จำนวน 8 บรรทัด ตัวอักขระภาษาไทยในระดับบนอาจจะเกิดจากการผสมอักขระของสระบน กับวรรณยุกต์ หรือสระบน กับไม้ทัณฑฆาต จึงต้องมีการผสมอักขระเหล่านี้เข้าเป็นรหัสตัวหนึ่งก่อน เพื่อแสดงออกจอภาพในขณะที่การเก็บรหัสสามารถแยกเก็บอักขระเป็นตัว ๆ ไป ความจำเป็นอันนี้ จึงต้องมีการแก้ไขระบบปฏิบัติการบางส่วน เพื่อให้การผสม และจัดแสดงบนจอภาพ ทำได้ภายใน 3 บรรทัด การดัดแปลงตัวสร้างอักขระ หรือแก้รอม ทำให้การแสดงตัวอักขระบนจอภาพด้วยวิธีนี้ ทำได้อย่างรวดเร็ว แต่มีข้อเสีย คือ สามารถแสดงข้อความได้น้อย เนื่องจากว่าสามารถแสดงได้เพียง 8 บรรทัด

วิธีการของภาษาไทย 25 บรรทัด ก็เป็นการแก้ไขข้อเสียอันหนึ่งของภาษาไทย 8 บรรทัด คือ ให้สามารถแสดงจำนวนบรรทัดบนจอภาพของภาษาไทย เท่ากับภาษาอังกฤษ (25 บรรทัด) (สำหรับวิธีการดัดแปลงแผงวงจรภาคแสดงนั้น จะไม่กล่าวถึง เนื่องจากมีผู้พัฒนาขึ้นมาหลายแบบ แต่หลักการที่ใช้ส่วนใหญ่ก็คือ เพิ่มจำนวนเส้นแสดงภาพ (Horizontal scan line) เป็น 2 เท่า และอาจใช้วิธีแสดงภาพซ้อน คืออักขระระดับกลางภาพหนึ่ง กับอักขระระดับบน-ล่าง อีกภาพหนึ่ง การผสมอักขระตัวบนอาจนำเอาสระล่างของบรรทัดก่อนมาร่วมด้วย) ข้อดีของวิธีนี้ก็คือ สามารถแสดงข้อความได้จำนวนมาก และการดัดแปลงโปรแกรมจากต่างประเทศ เพื่อให้สามารถแสดงผลเป็นภาษาไทยนั้น ทำได้ไม่ยากนัก เพราะมีจำนวนบรรทัดเท่ากัน แต่ข้อเสียก็คือ เนื่องจากมีการดัดแปลงวงจรภาคแสดง จึงต้องมีการดัดแปลงระบบปฏิบัติการบางส่วนควบคู่กันไปด้วย และเนื่องจากมีผู้ดัดแปลงแผงวงจรภาคแสดง และระบบปฏิบัติการกันหลายวิธีแตกต่างกัน จึงทำให้โปรแกรมที่ใช้ได้กับระบบหนึ่ง จะใช้กับอีกระบบหนึ่งไม่ได้ และการเพิ่มจำนวนเส้นแสดงภาพนี้เอง รวมทั้งรายละเอียดที่สูงของตัวอักขระภาษาไทยเกินความสามารถของจอภาพปกติ จึงทำให้ภาพที่ได้มีอาการสั่น (flicker) ซึ่งต้องใช้จอภาพที่สามารถลดอาการนี้ได้โดยเฉพาะ และตัวอักษรที่แสดงบนจอภาพอาจตัวเล็กเกินไปก็ได้

สำหรับวิธีการของภาษาไทยชนิดปรับจำนวนบรรทัดได้นั้น จะเป็นการปรับปรุงระหว่างวิธีการทั้ง 2 วิธีข้างต้นคือ มีขนาดตัวอักษรที่แสดงบนจอภาพเท่ากับชนิด 8 บรรทัด แต่สามารถจัดบรรทัดการแสดงได้มากกว่า 8 บรรทัด ตามลักษณะของข้อมูลที่แสดง เช่น ถ้าข้อมูลไม่มีอักขระบนล่างเลย ก็อาจจัดให้เป็น 25 บรรทัดเท่ากับปกติ หรือถ้ามีอักขระบนล่าง ก็จัดไม่ให้การแสดงผลอักขระบน และล่างแสดงทับกัน ข้อดีอีกอย่างหนึ่งก็คือ ไม่ต้องดัดแปลงวงจรภาคแสดง แต่ใช้เทคนิคทางกราฟฟิคแทน วิธีนี้ทำให้ต้องใช้กับภาคแสดงผลที่สามารถแสดงแบบกราฟฟิคได้เท่านั้น และการแสดงผลบนจอภาพอาจช้าลงบ้าง วิธีการนี้ จะเห็นได้ว่าไม่สามารถดัดแปลงโปรแกรมสำเร็จรูปอื่น ๆ จากต่างประเทศมาใช้กับวิธีการนี้ได้ จึงต้องเขียนโปรแกรมที่ต้องการใช้งานขึ้นมาเอง และภาษาไทยก็มีจุดแตกต่างกับภาษาอังกฤษ และปัญหาด้านต่าง ๆ อีกมาก จึงทำให้ควรเขียนโปรแกรมขึ้นเองดีกว่า ที่จะดัดแปลงโปรแกรมจากต่างประเทศ การแสดงผลด้วยวิธีนี้ เป็นการใช้ซอฟต์แวร์ล้วน ๆ จึงมีผลในแง่ความยืดหยุ่นทางด้านรหัสอักขระซึ่งก็ยังมีอยู่หลายแบบ การแก้ไขต่าง ๆ ทำได้ง่าย ดังนั้น การแสดงผลที่จะกล่าวถึงต่อไปทั้งหมด จะใช้การแสดงผลวิธีนี้

ถึงแม้ว่า จะมีการดัดแปลงไมโครคอมพิวเตอร์ให้มีความสามารถในการ รับ (และแสดง) ข้อมูลภาษาไทย/อังกฤษได้แล้ว แต่การที่จะแก้ไขข้อมูลที่รับเข้าไปแล้วนั้นก็ยังทำได้ยาก เพราะไบออสที่มีใช้กันอยู่ทั่วไป ไม่สามารถจัดการให้ได้ ตัวอย่างเช่น ข้อมูลที่ต้องการจริง คือ "นาย สมชาย" แต่ได้รับข้อมูลไปแล้วคือ "นาย สมมชาย" มีตัวอักษร "ม" เกินมาหนึ่งตัว ถ้าต้องการจะแก้ไขข้อมูลที่ใส่ผิด ปกติจะต้องพิมพ์อักษร "นาย สมชาย" เข้าไปใหม่ทั้งหมด เพราะไม่สามารถจะใช้วิธีเลื่อนเคอร์เซอร์ไปลบอักษร "ม" ที่เกินออกไปได้ ถ้าต้องการให้สามารถใช้เคอร์เซอร์ เลื่อนไปแก้ไขเพียงตัวเดียวได้ ก็จะต้องเขียนโปรแกรมขึ้นมาจัดการในอีกกรณีหนึ่ง ก็คือ ถ้าข้อมูลที่พิมพ์ผิดเป็น "นา สมชาย" จะเติมแทรก "ย" เพื่อแก้ไข "นา" ให้เป็น "นาย" หรือในกรณีที่พิมพ์ผิดเป็น "นาง สมชาย" หากต้องการจะพิมพ์ทับ "ง" ด้วยอักษร "ย" เพื่อแก้ไข "นาง" ให้เป็น "นาย" ปกติแล้วจะทำไม่ได้ จำเป็นต้องมีโปรแกรมมาจัดการเช่นเดียวกัน

เนื่องจาก มีความไม่สะดวกในการที่ต้องใส่ข้อมูลเข้าไปใหม่ทั้งหมด เมื่อมีการใส่ข้อมูลผิด จึงจำเป็นต้องมีโปรแกรมที่จะมาแก้ไขข้อมูลส่วนนั้นให้มีความถูกต้อง ทั้งนี้ให้สามารถแก้ไขตรงตัวอักษรบางตัวในข้อมูลนั้นได้อย่างสะดวก ซึ่ง Line Editor ที่พัฒนา และออกแบบขึ้นมานี้ สามารถที่จะจัดการกับการทำงานส่วนนี้ได้ และยังมีความสามารถด้านอื่น ๆ อีก ซึ่งจะ

ได้กล่าวถึงอย่างละเอียดต่อไป

ดังนั้น หน้าที่การทำงานที่จำเป็นของ Line Editor ต้องประกอบด้วย

1. มีความสามารถในการ รับ และแสดงข้อมูลได้ทั้ง ไทย และอังกฤษ
2. สามารถเลื่อนเคอร์เซอร์ไป ลบอักษรที่ผิดบางตัว หรือทั้งหมดได้
3. สามารถเติมแทรกตัวอักษรลงไปที่ตำแหน่งเคอร์เซอร์ได้ แทรกลงแทนที่อักษรเดิม
4. สามารถพิมพ์ทับตัวอักษรลงไปที่ตำแหน่งเคอร์เซอร์ทับอักษรเดิมได้

ส่วนที่เหลือของหัวข้อนี้ จะกล่าวถึงหน้าที่อย่างละเอียดของ Line Editor โดยจะกล่าวถึงสภาวะแวดล้อมของการทำงาน และสภาวะเริ่มต้นของการทำงานของ Line Editor ก่อน

4.2.1 <u>หน้าที่อย่างละเอียด</u>

สภาวะแวดล้อมของการทำงานจะหมายถึง สภาวะต่าง ๆ ที่ใช้บอกสถานะเพื่อควบคุมการทำงานของ Line Editor ให้สามารถทำงานได้อย่างที่กำหนดก่อน สภาวะแวดล้อมการทำงานของ Line Editor มี 2 ชนิดคือ

- ไทย/อังกฤษ
- เติมแทรก/พิมพ์ทับ

สภาวะแวดล้อม ไทย/อังกฤษ หมายถึง สภาวะที่ตัว Line Editor จะรับข้อมูลจากแป้นพิมพ์เป็นชนิด ภาษาไทย หรือภาษาอังกฤษ โดยโปรแกรมส่วนที่เรียกใช้ Line Editor จะต้องกำหนดการเริ่มต้นมาให้ก่อนว่า แป้นพิมพ์ในขณะนั้น จะทำหน้าที่เป็นแป้นพิมพ์ภาษาไทย หรือภาษาอังกฤษ ตัวอย่างเช่น ในการประยุกต์เพื่อจะรับข้อมูล รหัส และชื่อ ดังรูปที่ 4.1

รหัส : :

ชื่อ : :

รูปที่ 4.1 การใช้ Line Editor รับข้อมูล

เมื่อตำแหน่งเคอร์เซอร์ อยู่ที่ช่องรหัส โปรแกรมจะกำหนดให้แป้นพิมพ์เป็นภาษาอังกฤษ เพื่อรับรหัสภาษาอังกฤษ และในขณะที่เคอร์เซอร์อยู่ที่ช่องชื่อ โปรแกรมจะกำหนดให้แป้นพิมพ์เป็นภาษาไทย การกำหนดเช่นนี้ จะเป็นการสะดวกแก่ผู้ใช้ที่ไม่จำเป็นต้องมีการเปลี่ยนสภาวะแวดล้อม ไทย/อังกฤษเอง เนื่องจากสภาวะนั้นๆ สามารถที่จะกำหนดได้ล่วงหน้าอยู่แล้ว

ในขณะที่มีการรับข้อมูลภาษาไทย (หรือภาษาอังกฤษ) อยู่นั้น ผู้ใช้โปรแกรมสามารถจะสั่งเปลี่ยนแป้นพิมพ์ให้เป็นภาษาอังกฤษ (หรือภาษาไทย) หรือต้องการจะเปลี่ยนกลับอีก ก็จะทำได้เช่นเดียวกัน โดยการกดแป้นควบคุมที่กำหนดให้ (ในที่นี้ได้เลือกใช้ [ALT F10])

สภาวะแวดล้อมของการเติมแทรก/พิมพ์ทับ เพื่ออำนวยความสะดวกในการแก้ไขข้อมูล เช่นตัวอย่าง ข้อมูล "นา สมชาย" เราเลื่อนตำแหน่งพิมพ์มาที่หลังอักษร "า" แล้วพิมพ์ตัว "ย" เพื่อแทรกเข้าไปที่ตำแหน่งนั้น ก็ควรจะอยู่ในสภาวะเติมแทรก เพื่อแทรกตัวอักษรลงไปที่ตำแหน่งนั้น แล้วตัวอื่น ๆ ที่เหลือจะเลื่อนไปทางขวา 1 ตัวอักษร หรือถ้าข้อมูลเป็น "นาง สมชาย" เราต้องการเปลี่ยนตัว "ง" เป็นตัว "ย" ก็ควรเป็นสภาวะพิมพ์ทับ โดยเลื่อนตำแหน่งพิมพ์ไปที่ตำแหน่งตัว "ง" แล้วพิมพ์ตัว "ย" ทับลงไป โดยตัวอักษรที่อยู่ทางขวาของตำแหน่งพิมพ์ยังอยู่ที่เดิม การเปลี่ยนสภาวะแวดล้อมของการเติมแทรก และการพิมพ์ทับนี้ ผู้ใช้สามารถเปลี่ยนกลับไปมาได้ ระหว่าง 2 สภาวะนี้ โดยการกดปุ่มควบคุมปุ่มหนึ่งที่กำหนดให้ (ในที่นี้ได้เลือกใช้ [INS])

ต่อไปจะกล่าวถึง สภาวะเริ่มต้นของข้อมูลที่โปรแกรมส่วนที่เรียกใช้ Line Editor จะส่งสภาวะเริ่มต้นของข้อมูลนั้น รวมทั้งตัวข้อมูลนั้นมาให้ Line Editor จัดการ สภาวะเริ่มต้นของข้อมูล จะแบ่งเป็น 2 ชนิด คือ

- เมื่อเป็นบรรทัดว่าง (blank line)
- เมื่อมีข้อมูลในบรรทัดนั้นอยู่แล้ว

การทำงานของทั้ง 2 สภาวะจะแตกต่างกัน ขึ้นอยู่กับ ข้อมูลประกอบอื่น ๆ ที่ทางโปรแกรมผู้จะเรียกใช้ Line Editor จะกำหนดมาให้ล่วงหน้าอีก คือ

- ความยาวของบรรทัดของข้อมูลที่ต้องการ
- อักษรตัวแรกจากแป้นพิมพ์ที่ส่งมาให้แก่ Line Editor
- ตำแหน่งการเริ่มพิมพ์ในบรรทัดนั้น ที่นับสัมพัทธ์จากตำแหน่งแรกของบรรทัดข้อมูล
- สภาวะการตัดตัวช่องไฟที่ตามหลังข้อมูลทั้งหมด หรือไม่

ในการทำงานของ Line Editor เมื่อมีสภาวะเริ่มต้นของข้อมูล เป็นชนิดบรรทัดว่าง แสดงว่า โปรแกรมที่เรียกใช้ส่งข้อมูลบรรทัดว่างนั้นมาให้ คือข้อมูลเป็นช่องไฟทั้งหมด เพื่อเป็นการรับข้อมูลใหม่เข้าไป ดังนั้น ต้องกำหนดความยาวของช่องรับข้อมูลด้วย โดยที่มีตำแหน่งพิมพ์ที่กำหนดให้ล่วงหน้าเป็นตำแหน่งแรกภายในช่องรับข้อมูลที่กำหนดให้นั้น ความยาวของช่องรับข้อมูลที่กำหนดให้นี้ จะเป็นตัวกำหนดว่า ผู้เรียกใช้ Line Editor ต้องการข้อมูลเป็นจำนวนเท่าใด เมื่อมีการพิมพ์อักษรลงไปทีละตัว ตำแหน่งแป้นพิมพ์จะเลื่อนไปทางขวาทีละตำแหน่ง หรือไม่เลื่อน ขึ้นกับชนิดของตัวอักษรที่พิมพ์ เช่นเดียวกับในเครื่องพิมพ์ดีดทั่วไป เมื่อตำแหน่งพิมพ์เลื่อนไปจนเท่ากับความยาวบรรทัดที่กำหนด Line Editor จะทำการส่งข้อมูลที่ได้กลับไปให้แก่ผู้เรียกใช้ รวมทั้งเลือกตัดช่องไฟที่ต่อท้ายข้อมูลให้ตามสภาวะที่ได้กำหนดให้ล่วงหน้า และพร้อมกันนั้น ก็จะส่งสถานะการจบการทำงานของ Line Editor ให้ด้วย

สำหรับการทำงานของ Line Editor เมื่อมีสภาวะเริ่มต้นของข้อมูลเป็นชนิดมีข้อมูลในบรรทัดนั้นอยู่แล้ว ซึ่งจะเป็นกรณีเมื่อข้อมูลที่มีอยู่เดิมผิดแล้วต้องการแก้ไขใหม่ หรือให้มีข้อมูลที่มีอยู่เดิม ในกรณีนี้ โปรแกรมที่เรียกใช้ Line Editor ต้องส่งผ่านข้อมูลของบรรทัดนั้นและข้อมูลประกอบอื่น ๆ มาให้ด้วย เช่นเดียวกับเมื่อสภาวะเริ่มต้นของข้อมูลเป็นชนิดบรรทัดว่าง ตามที่ได้กล่าวมาแล้ว ตำแหน่งการเริ่มพิมพ์จะเป็นส่วนที่บอกให้ Line Editor จัดการกับตำแหน่งเคอร์เซอร์ ให้มีความสัมพันธ์ตรงกับข้อมูลที่ได้มา การจัดการต่าง ๆ โดยละเอียดจะกล่าวถึงในหัวข้อย่อย 4.2.2 ในเรื่องส่วนประกอบของโปรแกรมที่เกี่ยวข้อง

ตัวอย่างของการทำงาน Line Editor เช่น ในการนำไปใช้ในการรับข้อมูลที่ทำงานดังรูป

ที่ทำงาน : :

รูปที่ 4.2 แสดงช่องเติมข้อมูล

ตามหน้าจอภาพ จะมีข้อความว่า ที่ทำงาน และกรอบของช่องใส่ข้อมูล ส่วนนี้จะสร้างขึ้นมาจากโปรแกรมภายนอกที่เรียกใช้ Line Editor โดยการที่โปรแกรมภายนอกจะต้องส่งข้อมูลที่จำเป็นให้แก่ Line Editor ตามที่กล่าวมาแล้ว คือ สภาวะแวดล้อมขณะนั้น เช่น เป็นแป้นพิมพ์ชนิดภาษาไทย และอยู่ในภาวะเติมแทรก เป็นข้อมูลว่าง ๆ โดยที่มีความยาว

ช่องรับข้อมูล จำนวน 20 ช่อง และตำแหน่งเริ่มต้นการพิมพ์เป็น 1 (แต่อาจเป็นตำแหน่งอื่น ๆ ก็ได้) มีสถานะเป็นชนิดที่ตัดช่องไฟข้างหลัง โปรแกรมภายนอกจะกำหนดตำแหน่งของ ช่องรับข้อมูลให้ สมมติว่า ให้อยู่ที่พิกัด (10, 2) (ที่ตำแหน่ง คอลัมน์ ที่ 10 บรรทัดที่ 2 บนจอภาพ) จะมีเคอร์เซอร์ปรากฏบนจอภาพ ดังนี้

ที่ทำงาน :_ :

ตัวอักษรตัวแรกที่ส่งเข้าไป เป็น null (รหัส 0) ดังนั้น ในกรณีนี้จะเป็นการรับข้อมูลใหม่ (ในระหว่างที่ใส่ข้อมูลอยู่ สมมติว่าพิมพ์เกิน หรือพิมพ์ตก ก็สามารถเลื่อนเคอร์เซอร์ไปแก้ไขเฉพาะที่ผิดเสร็จแล้วจึงเลื่อนเคอร์เซอร์กลับไปใส่ข้อมูลจนหมด)

ถ้ามีข้อความอยู่แล้ว ก็ให้ส่งข้อความมาที่ หน่วยความจำที่จองไว้ใช้งานของ Line Editor ที่ได้กำหนดไว้ และกำหนดตำแหน่งเริ่มพิมพ์ให้สัมพัทธ์กับตำแหน่งช่องรับข้อมูล (ตำแหน่งแรกของบรรทัดรับข้อมูล) เช่น มีข้อความอยู่แล้วว่า "บริษัท ไมโคเอ็กซ์" ต้องการเติมอักษร "ร" ที่หลังตัวอักษร "ค" ตำแหน่งพิมพ์จะเป็นที่ตำแหน่ง 10 ตรงกับอักษร "เ" และส่งผ่านอักษร "ร" ให้ที่ภาวะเติมแทรก ตัว "ร" ก็จะไปแทรกที่ตำแหน่งอักษร "เ" เป็น "บริษัท ไมโครเอ็กซ์" การนับตำแหน่งพิมพ์นั้น จะไม่นับรวมสระบน หรือสระล่างด้วย วิธีการนี้ใช้สำหรับแก้ไขข้อมูล (หรือกำหนดตำแหน่งเริ่มต้นเป็นตำแหน่งแรก ส่งอักษรตัวแรกเป็น null แล้วให้ผู้ใช้เลื่อนเคอร์เซอร์ไปที่ตำแหน่งสัมพัทธ์ 10 แก้ไขโดยพิมพ์แทรกอักษร "ร")

จากตัวอย่างดังกล่าว จะเห็นได้ว่า ถ้านำ Line Editor มาประกอบกันหลาย ๆ บรรทัด ก็จะกลายเป็น Editor ชนิด Full Screen Editor ได้เช่นกัน สำหรับรายละเอียดวิธีใช้งานจะกล่าวถึงในหัวข้อย่อย 4.2.4 การนำไปใช้งาน

ได้กล่าวถึงการเพิ่มเติมข้อมูลไปแล้ว ต่อไปจะกล่าวถึงการลบข้อมูล ภายใน Line Editor ด้วย ใน Line Editor นั้น มีวิธีการลบข้อมูลอยู่ 2 แบบ คือ

ก. การลบตัวอักขระทางซ้ายมือ ของตำแหน่งเคอร์เซอร์ทีละ 1 อักขระ (ลบถอยหลัง หรือ backspace)

ข. การลบตัวอักขระที่ตำแหน่ง เคอร์เซอร์ ทีละ 1 ช่อง รวมอักขระบนล่างด้วย (ลบดูดกลืน หรือ delete)

การลบอักขระทางซ้ายมือของตำแหน่งเคอร์เซอร์ทีละ 1 อักขระนั้น จะแบ่งออกเป็นอีก 2 กรณีตามสภาวะแวดล้อมการทำงาน เติมแทรก/พิมพ์ทับ การลบอักขระด้วยวิธีนี้ ในที่นี้จะใช้แป้นควบคุม [BS] (back space) หรือ [CTRL H] ในกรณีสภาวะเติมแทรก เมื่อกดแป้นควบคุม [BS] ตัวอักขระทางซ้ายมือของเคอร์เซอร์จะถูกลบไปหนึ่งตัวตามลำดับที่ได้จัดเรียงไว้ คือ วรรณยุกต์ (หรือทัณฑฆาต) สระบน/ล่าง และตัวอักษรระดับกลาง เมื่อมีการลบอักษรระดับกลางออกแล้ว เคอร์เซอร์ จะเลื่อนไปทางซ้ายมือ 1 ตำแหน่ง พร้อมกันนี้ ตัวอักษรอื่น ๆ ทางขวามือ (ถ้ามี) ก็จะรันตามขึ้นไป 1 ตำแหน่งด้วย (เฉพาะบรรทัด หรือส่วนของบรรทัดที่อยู่ภายใต้การควบคุมของ Line Editor) ส่วนกรณีสภาวะพิมพ์ทับ เมื่อมีคำสั่งลบถอยหลัง ลำดับตัวอักขระที่จะถูกลบคือ วรรณยุกต์ (หรือทัณฑฆาต) สระบน/ล่าง และตัวอักษรระดับกลาง แต่เมื่อตัวอักษรในระดับกลางถูกลบออกไป เฉพาะเคอร์เซอร์เท่านั้นที่จะเลื่อนไปทางซ้าย 1 ตำแหน่ง ตัวอักษรอื่น ๆ ทางขวามือ จะไม่รันตามไปด้วย จะยังอยู่ที่เดิม เพียงแต่มีช่องไฟเกิดขึ้นมาแทนตัวอักษรที่ถูกลบออกไป

การลบตัวอักขระที่ตำแหน่งเคอร์เซอร์ทีละ 1 ช่อง รวมอักขระบนล่างด้วย (ลบดูดกลืน) วิธีนี้จะเป็นการดูดกลืนตัวอักษรที่ตำแหน่งเคอร์เซอร์หายไปทั้ง 3 ระดับ การลบแบบนี้ไม่ขึ้นกับสภาวะแวดล้อมการทำงาน เติมแทรก/พิมพ์ทับ (ในที่นี้ใช้แป้นควบคุม [del] หรือ [Ctrl G])

ตัวอย่างการลบอักขระเป็นดังนี้ สมมติว่า มีข้อมูลอยู่คือ "บรรทัดตัวอย่าง" และสมมติว่า ได้เลื่อนเคอร์เซอร์มาอยู่ที่ตำแหน่ง "ต" ถ้าต้องการลบให้เหลือเพียงข้อมูลว่า "ตัวอย่าง" ก็ให้กดปุ่มควบคุม [BS] 6 ครั้ง เพื่อลบอักขระทีละตัว รวมอักขระบน/ล่างด้วย ถ้าอยู่ในสภาวะเติมแทรก ผลลัพธ์ที่ได้จะเป็น "ตัวอย่าง" (ข้อความอยู่ชิดซ้ายสุด) แต่ถ้าอยู่ในสภาวะพิมพ์ทับ ผลลัพธ์จะเป็น "_____ตัวอย่าง" มีช่องไฟไปแทนที่ตำแหน่งของข้อความ "บรรทัด" แต่ถ้าต้องการให้เหลือเพียงข้อมูลว่า "บรรทัด" ก็ให้กดปุ่มควบคุม [del] 6 ครั้ง เพื่อดูดกลืน ั,ว,อ,ย่,า,ง ตามลำดับ

นอกจากวิธีการลบข้อมูลทั้ง 2 วิธีแล้ว ยังมีการลบอีกแบบหนึ่ง คือ การลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปจนจบบรรทัด (ที่อยู่ภายในความควบคุมของ Line Editor นั้น) การลบแบบนี้จะเกิดขึ้นเมื่ออยู่ในสภาวะเติมแทรก และกดปุ่มควบคุม [Enter] หรือ [Return] การลบแบบนี้ ข้อมูลจะไม่หายไปไหน เพียงแต่แยกออกเป็น 2 ส่วน คือ ส่วนของอักษรที่อยู่ก่อนหน้าเคอร์เซอร์ และส่วนของอักษรที่อยู่ตั้งแต่เคอร์เซอร์เป็นต้นไป 2 ส่วนนี้ จะส่งกลับไป

โปรแกรมที่เรียกใช้ แต่การแสดงบนจอภาพ จะเห็นเพียงส่วนแรกอย่างเดียว

ได้กล่าวถึงหน้าที่ของ Line Editor ไปแล้ว ต่อไปจะกล่าวถึงการออกจาก Line Editor (หรือจบการทำงานกับ Line Editor) ซึ่งมีวิธีการออกได้หลายวิธี ภายใน Line Editor นั้น จะมีการควบคุมฟังก์ชันการทำงานต่าง ๆ โดยการส่งผ่านแป้นควบคุม และแป้นควบคุมนี้ เราได้กำหนดไว้ล่วงหน้าแล้ว การกระทำใด ๆ ที่ยังอยู่ภายในความยาว ควบคุมที่ได้กำหนดไว้ ก็จะอยู่ในการทำงานของ Line Editor แต่ถ้ามีการกระทำใด ๆ ที่ออกนอกบรรทัด หรือความยาวควบคุมที่กำหนดไว้ เช่น เมื่อมีการกดแป้นควบคุมเลื่อนตำแหน่งพิมพ์ไปทางซ้าย หรือขวา ก็จะยังอยู่ในความควบคุมของ Line Editor แต่ถ้าตำแหน่งพิมพ์ถอยหลังมาเกินตำแหน่งเริ่มต้นที่กำหนดไว้ หรือไปทางซ้ายเกินความยาวที่กำหนดไว้ ไม่ว่าโดยวิธีเลื่อนตำแหน่งพิมพ์ หรือกดแป้นพิมพ์อักษรอื่น ๆ หรือมีการกดแป้นควบคุม นอกเหนือจากแป้นควบคุมการเลื่อนตำแหน่งพิมพ์ และแป้นควบคุมการลบแล้ว ก็จะเป็นการออกจาก Line Editor ทั้งสิ้น โดย Line Editor จะทำการส่งข้อมูลที่ได้มีการเปลี่ยนแปลงแล้วคืนให้แก่ผู้เรียกใช้ รวมทั้งรหัสข้อมูลสถานะการออกจาก Line Editor หรือรหัสจบงาน (Exit Code) ด้วยว่าออกมาด้วยวิธีใด สำหรับ รหัสข้อมูลสถานะการออกจาก Line Editor และวิธีการทำงานอย่างละเอียดจะกล่าวถึงในหัวข้อย่อย 4.2.2

หน้าที่ ที่สำคัญอีกอย่างหนึ่งของ Line Editor คือ การตรวจสอบความถูกต้องของภาษาไทย ในภาษาไทยนั้น มีการแบ่งอักษรออกเป็น 3 ระดับ คือ อักษรบน กลาง ล่าง ตามที่ได้กล่าวมาแล้วในบทที่ 2 เมื่อมีการพิมพ์อักษรกลาง ตำแหน่งพิมพ์จะเลื่อนไปทางขวา 1 ตำแหน่ง แต่อักษรบน และล่าง จะไม่มีการเลื่อนตำแหน่งพิมพ์ ทำให้อาจมีการพิมพ์ อักษรบน/ล่าง ซ้ำ ลงไปอีก ถ้าใน Line Editor ไม่สามารถตัดตัวอักษร บน/ล่าง ที่ซ้ำกันออกไปได้ จะทำให้ข้อมูลที่ได้ไม่ถูกต้อง คือที่เห็นบนจอภาพ อาจจะเหมือนกัน แต่ข้อมูลภายในจริงอาจจะไม่เหมือนกัน และในคำไทยจะมีกฎเกณฑ์ต่าง ๆ อยู่แล้ว เช่น จะไม่มีสระบน และสระล่างอยู่ที่ตำแหน่งเดียวกัน ตามที่ได้กล่าวในบทที่ 2 ดังนั้น Line Editor ก็จะมีความฉลาดในการตัดอักษรที่ซ้ำ และคำที่ไม่มีในคำไทยออก เพื่อความถูกต้องของข้อมูล และประหยัดที่เก็บข้อมูลด้วย

สรุปหน้าที่การทำงานทั้งหมดของ Line Editor

1. รับ และแสดงข้อมูลได้ทั้งไทย และอังกฤษ
2. มีความสามารถในการตรวจสอบความเป็นไปได้ของคำไทย

3. สามารถแก้ไขข้อมูลได้อย่างสะดวก
4. มีปุ่มควบคุมการทำงาน และกำหนดสภาวะแวดล้อมของ Line Editor ได้
5. สามารถเปลี่ยนแปลง Line Editor ให้ใช้กับรหัสต่าง ๆ ได้ โดยสามารถเปลี่ยนตำแหน่งแป้นพิมพ์ และรหัสออกจอภาพได้
6. สามารถควบคุมตำแหน่งเคอร์เซอร์ให้สัมพันธ์กับการรับข้อมูลขณะนั้นได้

การควบคุมการทำงานของ Line Editor จะมีแป้นควบคุมการทำงาน ซึ่งแบ่งได้เป็น 2 ชนิดคือ

1. แป้นควบคุมการทำงาน ภายใน
2. แป้นควบคุมการทำงานที่จะส่งรหัสออกจาก Line Editor (Exit Code)

แป้นควบคุมการทำงานภายใน จะประกอบด้วย

- แป้นควบคุมการเลื่อนเคอร์เซอร์ไปมาภายในช่องที่มีความยาวตามที่กำหนดไว้
- แป้นควบคุมการเปลี่ยนสภาวะแวดล้อมการทำงาน
- แป้นป้อนตัวอักษรต่าง ๆ

แป้นควบคุมการทำงานที่จะส่งรหัสออกจาก Line Editor จะเป็นตัวกำหนดการสิ้นสุดการทำงานของ Line Editor พร้อมส่งรหัสบางอย่างกลับคืน ผู้เรียกใช้ คือแป้นอื่น ๆ นอกจากแป้นควบคุมการทำงานภายใน เช่น

- แป้นฟังก์ชันต่าง ๆ [F1 - F10], [Ctrl F1 - F10],
  [Alt F1 - F10], [Shift F1 - F10]
- แป้น [PgUp], [PgDn], [UArr], [DArr], [Home], [End]
- แป้น [Ctrl - LArr, RArr, Home, End, PgUp, PgDn]
- แป้น [Alt - UArr, DArr, Ins, Del]
- แป้น [RETURN], [Ctrl - N, Y, I, T, B], [Tab]
- แป้น [ESC]

4.2.2 ส่วนประกอบของโปรแกรมที่เกี่ยวข้อง

การที่ต้องการให้ Line Editor สามารถทำงานได้ตามที่ได้กล่าวมาแล้ว คือ มีหน้าที่รับ และแก้ไขข้อมูล ตามสภาวะที่กำหนด และส่งรหัสควบคุมการทำงาน เมื่อจบการทำงาน ให้แก่ โปรแกรมผู้เรียกใช้ จะต้องมีโปรแกรมย่อยหลาย ๆ ส่วนประกอบกัน รวมทั้ง มีวิธีการทำงานต่าง ๆ เพื่อให้ได้หน้าที่ตามที่ต้องการ

ในหัวข้อนี้จะกล่าวถึง วิธีการทำงานอย่างละเอียดของ Line Editor และกล่าวถึงส่วนประกอบที่อยู่ภายใน ซึ่งจะอธิบายถึงการประกอบกันของส่วนประกอบเหล่านี้ โดยใช้เป็นคำสั่งเทียม (psudo code)

นอกจากส่วนประกอบที่เป็นโปรแกรมย่อยต่าง ๆ ยังมีองค์ประกอบอื่นที่ Line Editor จะต้องใช้ ได้แก่

- ข้อมูลที่จะส่งมาให้ Line Editor แก้ไข
- บัฟเฟอร์ซึ่งเป็นที่เก็บข้อมูลหลักในระหว่างที่ Line Editor กำลังทำงานอยู่
- ตำแหน่งรับข้อมูล หรือตำแหน่งเคอร์เซอร์
- รหัสตัวอักขระ หรือรหัสแป้นคำสั่งที่ Line Editor รับจากแป้นพิมพ์โดยตรง
- บัฟเฟอร์ชั่วคราว (patch buffer) ซึ่งเป็นที่เก็บข้อมูลส่วนท้ายตั้งแต่ตำแหน่งเคอร์เซอร์ ในกรณีที่อยู่ในภาวะเติมแทรก
- ข้อมูลที่จะได้จาก Line Editor เมื่อจบการทำงานของ Line Editor
- รหัสจบงานที่บอกว่า Line Editor จบการทำงานด้วยเงื่อนไขใดหรือแป้นคำสั่งใด

ลักษณะการทำงานของ Line Editor จะมีขั้นตอนการทำงานง่าย ๆ ดังนี้

0. Set Environment, unpack Line to buffer or clear buffer, send first character (if any).

1. Clear exit code, clear patch buffer.

2a. If funkey then

- left arrow.
- right arrow.
- toggle Thai/English.
- globble.
- toggle insert/overwrite.
- exit.

2b. If character then add char (or exit if beyond line length).

2c. If control then

- globble.
- backspace.
- press return.
- toggle insert/overwrite.
- exit.

3. If not exit then get_next_char and repeat step 2 until exit.

4. Packline and skip tailing blank.

5. End.

ขั้นตอนการทำงานของ Line Editor จะสามารถอธิบายได้ดังนี้

ขั้นที่ 0 เป็นขั้นก่อนการเรียกใช้ Line Editor มีการเตรียมสภาวะแวดล้อมการทำงานให้กับ Line Editor ซึ่งได้แก่ กำหนดแป้นพิมพ์ไทย หรืออังกฤษ กำหนดภาวะเติมแทรก หรือพิมพ์ทับ ส่งข้อมูลของข้อความที่ต้องการแก้ไข โดยจัดข้อมูลลงในบัฟเฟอร์แล้วแสดงข้อมูลทางจอภาพ ณ ตำแหน่งที่ต้องการ หรือถ้าเป็นการรับข้อมูลใหม่ ก็ทำบัฟเฟอร์ให้ว่างแล้วแสดงบรรทัดว่าง กำหนดตำแหน่งรับข้อมูลบนจอภาพ และแสดงลักษณะเคอร์เซอร์ตามสภาวะ

แวดล้อมการทำงาน ถ้ามีอักขระตัวแรกที่จะส่งให้กับ Line Editor ก็ส่งรหัสอักขระตัวนั้นมาด้วย มิฉะนั้นให้รหัสเป็น 0

หมายเหตุ ลักษณะเคอร์เซอร์ตามสภาวะแวดล้อมการทำงานมี 4 ลักษณะ ซึ่งแต่ละลักษณะเป็นเส้นกระพริบขนาดความหนาต่าง ๆ กัน คือ

หนาหนึ่งเส้น	สำหรับ (ภาษา)	อังกฤษ (ภาวะ)	เติมแทรก
หนาสองเส้น	สำหรับ	ไทย	เติมแทรก
หนาสี่เส้น	สำหรับ	อังกฤษ	พิมพ์ทับ
หนาแปดเส้น	สำหรับ	ไทย	พิมพ์ทับ

ความคิดที่ออกแบบลักษณะเคอร์เซอร์ตามสภาวะการทำงานดังที่ได้กล่าวมานี้ เป็นความคิดใหม่ เท่าที่ทราบมาไม่เคยมีใครเสนอมาก่อน ส่วนใหญ่จะแสดงสภาวะการทำงานบริเวณมุมของจอภาพ

หมายเหตุ ปกติอักขระตัวแรกที่จะส่งให้ Line Editor มักจะมีรหัสเป็น 0 แต่บางครั้งอาจมีรหัสอื่น ซึ่งมักจะเป็นอักขระตัวที่เกินมาจากบรรทัดก่อน หรืออาจเป็นรหัสแป้นควบคุม หรืออาจเป็นรหัสแป้นฟังก์ชัน

ขั้นที่ 1 เป็นขั้นแรกสุดที่ Line Editor ทำงาน ในขั้นนี้ มีการกำหนดรหัสจบงาน (Exit Code) ให้เป็น 0 ซึ่งหมายความว่า ยังไม่จบ และมีการเตรียมบัฟเฟอร์ชั่วคราวให้พร้อมโดยทำเป็นบัฟเฟอร์ว่าง

ขั้นที่ 2a-2c เป็นขั้นที่จำแนกการทำงานตามรหัสแป้นพิมพ์ต่าง ๆ ซึ่งจัดกลุ่มแป้นพิมพ์เป็น 3 กลุ่ม ได้แก่กลุ่มแป้นฟังก์ชัน กลุ่มแป้นตัวอักขระ และกลุ่มแป้นควบคุม ถ้าเป็นรหัสแป้นฟังก์ชัน ก็จะแยกการทำงานว่าเป็นการทำงานอะไร เช่น เลื่อนตำแหน่งพิมพ์ไปทางซ้าย 1 ตำแหน่ง หรือทางขวา 1 ตำแหน่ง, เปลี่ยนสภาวะการทำงาน ระหว่าง ไทยกับอังกฤษ, เปลี่ยนสถานะการทำงาน ระหว่างเติมแทรก กับพิมพ์ทับ ,สั่งลบข้อมูลแบบดูดกลืน ส่วนรหัสแป้นฟังก์ชันอื่น ๆ ที่นอกเหนือจากนี้ จะถูกนำไปกำหนดค่ารหัสจบงานให้สอดคล้องกับรหัสแป้นฟังก์ชันนั้น ๆ

ถ้าเป็นรหัสแป้นตัวอักขระ คือมีรหัสตั้งแต่ $20 ถึง $FF ก็จะนำรหัสอักขระตัวนั้นไปเพิ่มเติมในบัฟเฟอร์ (ถ้าไม่ขัดตามการทดสอบความถูกต้องของคำไทยที่ได้กำหนดไว้) และ

แสดงอักขระตัวนั้นออกทางจอภาพให้ถูกต้องด้วย

ถ้าเป็นรหัสแป้นควบคุม คือ รหัสตั้งแต่ $01 ถึง $1F ก็จะไปทำงานการควบคุมที่ได้กำหนดไว้ ซึ่งอาจจะซ้ำกับการทำงานของรหัสฟังก์ชั่นก็ได้ ส่วนของรหัสแป้นควบคุมที่ไม่ได้กำหนดหน้าที่ทำงานเอาไว้ ก็อาจถูกนำไปกำหนดตัวรหัสจบงานเมื่ออาจถูกทิ้งไป หรืออาจถูกบันทึกลงในบัฟเฟอร์

หมายเหตุ (ขั้นที่ 2a-2c) การนำรหัสอักขระไปเพิ่มในบัฟเฟอร์ (ขั้นที่ 2b) นั้น ถ้าอยู่ในภาวะพิมพ์ทับ อักขระใหม่จะไปแทนที่ตัวเดิม แต่ถ้าอยู่ในภาวะเติมแทรก อักขระตัวใหม่จะถูกแทรกที่ตำแหน่งเคอร์เซอร์ (สำหรับตัวที่อยู่ในระดับกลาง) ตัวที่ถูกแทรกก็จะร่นไปทางขวา แม้ว่าหลักการที่กล่าวถึงนี้ค่อนข้างตรงไปตรงมา แต่รายละเอียดการทำงานมีหลายขั้นตอน ซึ่งมีฟังก์ชั่นหรือโปรแกรมย่อยที่อาจจะถูกเรียกใช้ ได้แก่ split, cleartoend, getblock, putblock, patch

ขั้นที่ 3 เป็นขั้นที่จะรับข้อมูล (อักขระ) ใหม่ มาจากแป้นพิมพ์ซึ่งจะเกิดขึ้นก็ต่อเมื่อรหัสจบงานยังมีค่าเป็นศูนย์อยู่ เมื่อรับข้อมูลใหม่แล้วก็จะวนขึ้นไปทำซ้ำขั้นที่ 2a-2c ต่อไปจนกว่ารหัสจบงานไม่เป็นศูนย์

ขั้นที่ 4 เป็นขั้นที่ทำการแปลงข้อมูลจากบัฟเฟอร์ เพื่อส่งกลับให้กับผู้เรียกใช้ Line Editor และถ้ามีตัวข่องไฟอยู่ตอนท้ายก็ให้ตัดทิ้งด้วย

จากขั้นตอนการทำงานของ Line Editor จะมีส่วนประกอบที่สำคัญ ดังนี้

- getchar ใช้รับรหัสจากแป้นพิมพ์ ถ้าเป็นตัวอักษรก็จะตรวจสอบความเป็นไปได้ในการรับเบื้องต้น
- addchar นำตัวอักษรที่ได้ใส่ลงใน บัฟเฟอร์ (ผ่านโปรแกรมย่อย getblock)
- getblock ตรวจสอบความเป็นไปได้ในการรับภาษาไทยอย่างละเอียด ต่อจาก getchar และนำไปใส่ในบัฟเฟอร์
- putblock แสดงตัวอักษรทั้ง 3 ระดับออกทางจอภาพ
- split สั่งย้ายข้อมูลในบัฟเฟรอ์ ตั้งแต่จุดที่กำหนดไปจนหมด เพื่อเก็บไว้ในบัฟเฟอร์ชั่วคราว (patch buffer)

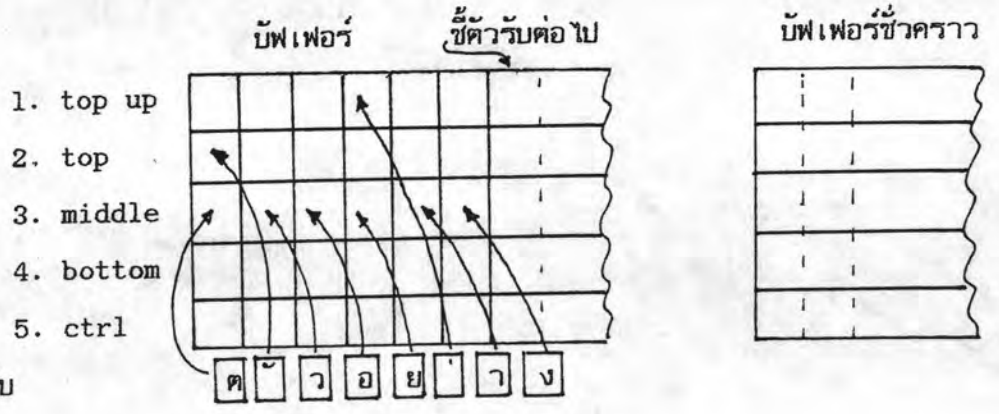
- patch สั่งย้ายข้อมูลจาก บัฟเฟอร์ชั่วคราว มาต่อกับ บัฟเฟอร์
- packline จัดลำดับตัวอักษรภายใน บัฟเฟอร์ให้เรียงตามลำดับบรรทัดเดียว
- unpackline นำตัวอักษรที่เรียงตามลำดับบรรทัดเดียว ไปจัดให้อยู่ในบัฟเฟอร์
- clrtoend ลบข้อความบนจอภาพตั้งแต่จุดที่กำหนดไปจนหมดความยาวของบรรทัด
- writeline แสดงข้อความ 1 บรรทัด

รายละเอียดของส่วนประกอบจะกล่าวในหัวข้อย่อย 4.2.3

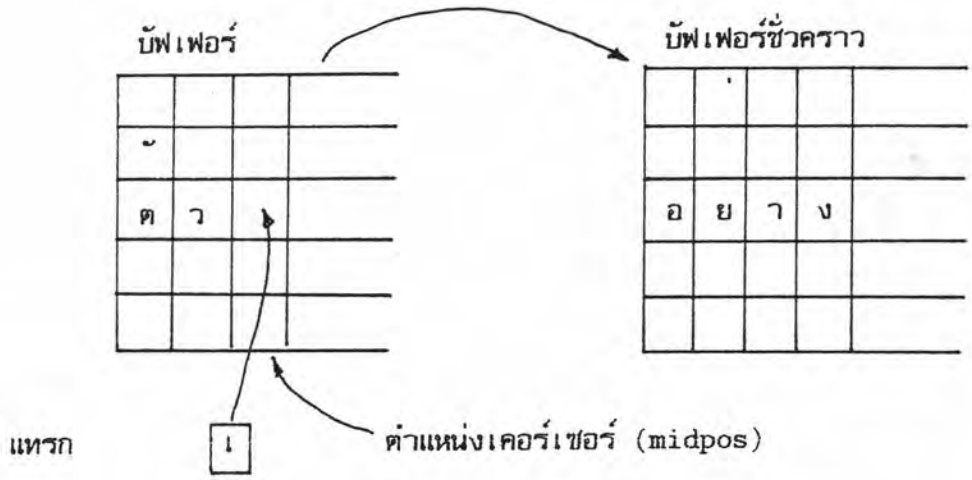
ต่อไปจะกล่าวถึงฟังก์ชันการทำงานต่าง ๆ ภายใน Line Editor โดยจะกล่าวถึงองค์ประกอบ ที่สำคัญก่อน คือ บัฟเฟอร์ (buffer), บัฟเฟอร์ชั่วคราว (patch buffer),อักขระที่รับ (c) และ บรรทัดการติดต่อ (line) ที่ใช้ภายใน

ภายในบัฟเฟอร์จะแบ่งตัวอักษรเป็น 4 แถวคือ

1. แถวบนสุด สำหรับ วรรณยุกต์ และไม้ทัณฑฆาต
2. แถวบน สำหรับ สระบนทั้งหมด
3. แถวกลาง สำหรับตัวพยัญชนะ สระหน้า สระหลัง อักษรภาษาอังกฤษ และเครื่องหมายต่าง ๆ
4. แถวล่าง สำหรับ สระล่าง
5. แถวควบคุม สำหรับเก็บรหัสควบคุมของอักษรช่องนั้น (ในที่นี้ไม่ได้ใช้แต่ได้ออกแบบเผื่อไว้สำหรับงาน word processor)



รูปที่ 4.3 แสดงการรับข้อมูลในบัฟเฟอร์ และระดับอักขระ



รูปที่ 4.4 แสดงบัฟเฟอร์ และบัฟเฟอร์ชั่วคราว

ถ้ามีการแทรกตัวอักษรเข้าไปจะมีการตัดข้อมูลใน บัฟเฟอร์ ตั้งแต่ตำแหน่งเคอร์เซอร์ไปที่ บัฟเฟอร์ชั่วคราว เมื่อเพิ่มหมดแล้ว จึงมีการย้ายข้อมูลจาก บัฟเฟอร์ชั่วคราว มาต่อที่ท้ายบัฟเฟอร์

ในการรับข้อมูล จะเป็นการนำตัวอักษรที่ได้รับทีละตัว มาตรวจสอบความเป็นไปได้กับข้อมูลในบัฟเฟอร์ ถ้าสามารถจะรับข้อมูลที่ได้ ก็จะนำอักษรตัวนั้นใส่ลงในบัฟเฟอร์ ถ้ารับไม่ได้จะส่งเสียงเตือนแล้วกลับไปรอรับใหม่ การใส่อักษรลงในบัฟเฟอร์ จะไปใส่ที่แถวต่าง ๆ ตามที่ได้กำหนดไว้ ถ้าเป็นแถวกลางจะมีการเลื่อนตำแหน่งรับข้อมูลไปอีก 1 ตำแหน่ง

ในกรณีที่อยู่ในสภาวะเติมแทรก โดยมีเคอร์เซอร์อยู่กลางข้อความ เมื่อมีการรับตัวอักษรมา จะมีการตรวจสอบความเป็นไปได้ของการรับเสียก่อน ถ้าสามารถรับได้ จะมีการย้ายข้อความที่เหลือ ตั้งแต่ตำแหน่งเคอร์เซอร์เป็นต้นไป ไปยังบัฟเฟอร์ชั่วคราว แล้วนำตัวอักขระที่ได้รับใหม่ใส่ลงในบัฟเฟอร์ที่ตำแหน่งเคอร์เซอร์ เมื่อมีการรับตัวอักษรหมดแล้ว ก็จะนำข้อความจากบัฟเฟอร์ชั่วคราวมาต่อที่ท้ายบัฟเฟอร์ทันที กรณีที่แสดงว่ารับตัวอักษรหมดแล้ว ก็คือ มีการกดแป้นฟังก์ชัน หรือแป้นควบคุม ถ้าเป็นแป้น [Return] ในภาวะเติมแทรกนั้น จะไม่มีการนำ ข้อมูลจากบัฟเฟอร์ชั่วคราวมาต่อแต่อย่างใด ในระหว่างการตัดต่อนั้น ต้องควบคุมการแสดงตัวอักษรบนจอภาพให้ถูกต้องเช่นเดียวกับข้อมูลที่มีอยู่ด้วย

ฟังก์ชันการทำงานที่เป็นการควบคุมภายในของ Line Editor มีดังนี้

1.	Left Arrow	เลื่อนตำแหน่งพิมพ์ไปทางซ้าย
2.	Right Arrow	เลื่อนตำแหน่งพิมพ์ไปทางขวา
3.	toggle Thai/English	เปลี่ยนสภาวะการทำงานระหว่าง ไทย และอังกฤษ
4.	toggle Insert/overwrite	เปลี่ยนสภาวะการทำงานระหว่างเติมแทรก/พิมพ์ทับ
5.	globble	การลบแบบตูดกลืน
6.	backspace	การลบถอยหลัง
7.	pressreturn	ลบตั้งแต่ตำแหน่งนั้นจนจบบรรทัด
8.	Exit	ใส่ค่าในตัวแปร Exit Code
9.	addchar	เพิ่มรหัสอักษรนั้นลงใน บัฟเฟอร์
10.	getnextchar	รับรหัสจากแป้นพิมพ์
11.	packline	เปลี่ยนโครงสร้างข้อมูลจาก buffer เป็น line
12.	skip tailing blank	ลบช่องไฟข้างท้ายออก

รายละเอียดฟังก์ชันการทำงานเป็นดังนี้

Left Arrow เป็นการทำงานเมื่อมีการสั่งให้เลื่อนเคอร์เซอร์ไปทางซ้าย ฟังก์ชันนี้จะทำการเลื่อนตำแหน่งตัวชี้ในบัฟเฟอร์ถอยหลังไป 1 ตำแหน่ง แต่ถ้าถอยจนหมดบัฟเฟอร์แล้ว จะให้ Exit Code ออกมาเป็น -1 เมื่อจบการทำงานของ Line Editor ทำให้บัฟเฟอร์ชั่วคราวว่างไว้

Right Arrow เป็นการทำงานเมื่อมีการสั่งให้เลื่อนแป้นพิมพ์ไปทางขวา ฟังก์ชันนี้จะทำการเลื่อนตำแหน่งตัวชี้ ในบัฟเฟอร์ไปข้างหน้าอีก 1 ตำแหน่ง แต่ถ้าเลื่อนไปเกินขอบเขตที่กำหนดด้วยตัวแปร fieldwidth จะให้ Exit Code เป็น 1 เมื่อจบการทำงานของ Line Editor ทำให้บัฟเฟอร์ชั่วคราวว่างไว้

toggle Thai/English เป็นการทำงาน เมื่อมีการเรียกคำสั่งให้เปลี่ยนสภาวะการทำงาน ไทย/อังกฤษ จะทำให้เกิดการเปลี่ยนค่าในตัวแปรตัวหนึ่ง ที่จำสภาวะนี้อยู่ (highbit = 0 หรือ 1) เพื่อนำไปรวมกับรหัสที่ได้จากแป้นพิมพ์ แล้วนำไปหาค่าที่แท้จริงในตารางรหัส พร้อมทั้งเปลี่ยนแปลงลักษณะเคอร์เซอร์ เพื่อให้เห็นถึงความแตกต่างระหว่างการรับข้อมูล ไทย/อังกฤษ

toggle Insert/overwrite เป็นการทำงานเมื่อมีการเรียกคำสั่งให้เปลี่ยนสภาวะการทำงาน เติมแทรก/พิมพ์ทับ จะทำการเปลี่ยนค่าในตัวแปรตัวหนึ่งที่เก็บสภาวะการทำงานนี้ และเปลี่ยนแปลงลักษณะเคอร์เซอร์ เพื่อให้เห็นความแตกต่าง ระหว่างสภาวะการทำงานแบบเติมแทรก หรือพิมพ์ทับ

globble เป็นการทำงานเมื่อมีคำสั่งลบข้อมูลแบบดูดกลืน จะทำการลบข้อมูลในบัฟเฟอร์ที่ตรงตำแหน่งตัวชี้บัฟเฟอร์นั้นออก แล้วทำการเปลี่ยนแปลงหน้าจอภาพให้แสดงได้ถูกต้อง โดยสั่ง cleartoend และ putblock ใหม่อีก การลบข้อมูลแบบดูดกลืนจะลบทั้ง 3 ระดับ

backspace เป็นการทำงาน เมื่อมีคำสั่งลบข้อมูลถอยหลัง การลบข้อมูลจะลบในบัฟเฟอร์ทีละตัวอักษร โดยถ้าขณะที่สั่งลบข้อมูลถอยหลัง ตำแหน่งชี้อยู่ที่ตำแหน่งที่ 1 จะให้ Exit Code เป็น -1 เช่นเดียวกับการเลื่อนแป้นพิมพ์ไปทางซ้ายบนสุด ที่ตำแหน่งตัวชี้บัฟเฟอร์อื่น ๆ ถ้าอยู่ในการเติมแทรก จะทำการแยกบัฟเฟอร์ ตั้งแต่ตำแหน่งตัวชี้นั้นไปไว้ที่บัฟเฟอร์ชั่วคราว แล้วทำการลบตัวอักษรทางซ้าย 1 ตัวอักษร ในการลบจะคำนึงถึงสภาวะแวดล้อมการทำงาน ถ้าเป็นพิมพ์ทับ และตัวที่ถูกลบอยู่ในระดับกลาง ตัวอักษรนั้นจะหายไปเป็นช่องไฟ โดยที่ข้อความข้างท้ายทั้งหมดไม่ได้ถอยร่นมาด้วย ถ้าอยู่ในภาวะเติมแทรก ข้อความข้างท้ายก็จะถอยร่นตามมา เมื่อสั่งลบแล้ว และตัวที่ถูกลบอยู่ในระดับกลาง ตำแหน่งพิมพ์จะถอยหลังไป 1 ตำแหน่ง หลังจากนั้น ข้อมูลในบัฟเฟอร์ชั่วคราวก็จะถูกนำมาต่อคืนให้

press return เป็นการทำงาน เมื่อกดแป้นพิมพ์ [return] ถ้าอยู่ในภาวะการทำงานเติมแทรก คำสั่งนี้ จะทำการแยกข้อมูลในบัฟเฟอร์ไปไว้ที่ บัฟเฟอร์ชั่วคราว แล้วลบข้อความบนจอภาพ ตั้งแต่ตำแหน่งแป้นพิมพ์นั้นไปจนจบบรรทัด

exit เป็นการใส่ค่าให้ตัวแปร เพื่อกำหนดรหัสการออกจาก Line Editor เป็น Exit Code

addchar เป็นการให้นำรหัสอักษรนั้น มาตรวจสอบความถูกต้องของคำไทยแล้วเพิ่มข้อมูลนั้นลงในบัฟเฟอร์

getnextchar เป็นการรอรับรหัสจากแป้นพิมพ์ โดยเรียกส่วนประกอบ getchar เพื่อรับรหัสจากแป้นพิมพ์ และตรวจสอบความถูกต้องของคำไทย ถ้าผิดก็จะไปรอรับใหม่

packline เป็นการเปลี่ยนโครงสร้างการเก็บข้อมูลจากบัฟเฟอร์เป็นบรรทัดเดียว เพื่อส่งเป็นผลลัพธ์ที่ได้

skip tailing blank เป็นการลบช่องไฟท้ายข้อมูลที่จะส่งกลับคืนให้กับผู้เรียกใช้ Line Editor

4.2.3 รายละเอียดของส่วนประกอบต่าง ๆ

ในหัวข้อนี้จะกล่าวถึงรายละเอียดของส่วนประกอบต่าง ๆ และวิธีการใช้งานของส่วนประกอบเหล่านี้ ในการจะใช้งานส่วนประกอบเหล่านี้ได้ จำเป็นต้องรู้จักตัวแปรที่จำเป็น และโครงสร้างข้อมูลของตัวแปรนั้นด้วย ที่ได้กล่าวถึงไปแล้วคือ บัฟเฟอร์ บัฟเฟอร์ร่วม ความกว้างการรับข้อมูล และ ตำแหน่งรับข้อมูล และในรายละเอียดของส่วนประกอบเหล่านี้จะกล่าวถึง ข้อมูลที่รับ ผลลัพธ์ที่ได้ หน้าที่ และการทำงาน

```
function  getchar (pos : integer):char;
```

ตัวแปร pos เป็นตำแหน่งตัวชี้ ในบัฟเฟอร์

ผลลัพธ์ รหัสที่ได้ ถ้าเป็นแป้นฟังก์ชัน จะให้ตัวแปร funckey = true มิฉะนั้น funckey = false รหัสจะส่งคืนให้กับตัวแปร getchar ถ้าผิดเงื่อนไขคำไทยจะส่ง Null คืน

หน้าที่ อ่านข้อมูลจากแป้นพิมพ์ และตรวจสอบคำไทยเบื้องต้น

การทำงาน

1. อ่านแป้นพิมพ์ ถ้าเป็นแป้นฟังก์ชัน อ่านรหัสของฟังก์ชันอีกทีหนึ่ง แล้วส่งค่ารหัสนั้นคืน จบการทำงาน

2. ถ้าไม่ใช่แป้นฟังก์ชัน จะทำการแปลงรหัส จากรหัสแป้นพิมพ์ เป็นรหัสตัวอักษร โดยประกอบเข้ากับตัวแปร กำหนดสภาวะ ไทย/อังกฤษ (highbit) และข้อมูลในตารางการแปลง

3. ตรวจสอบความเป็นไปได้ของคำไทย จากรหัสที่ได้กับรหัสข้อมูลที่มีอยู่แล้วภายในตัวแปร buffer ที่ตำแหน่ง pos ที่กำหนด ถ้ารับได้ก็ส่งรหัสนั้นคืน ถ้ารับไม่ได้ก็จะส่งรหัส Null คืน

หมายเหตุ หลักเกณฑ์ที่ใช้ตรวจสอบว่าจะรับรหัสใหม่ได้หรือไม่ ให้ใช้อักขระระดับกลางที่ตำแหน่ง pos-1 ช่วยตัดสิน เช่น ถ้าเป็นพยัญชนะก็รับรหัสใหม่ได้ทุกตัว ถ้าเป็นสระหน้า จะรับ

เฉพาะพยัญชนะ ถ้าเป็นสระหลัง จะรับ สระหน้า, พยัญชนะ, (ไม่รับสระหลังด้วยกัน ยกเว้น า ตามด้วย ะ)

procedure addchar;

ตัวแปร -

ผลลัพธ์ -

หน้าที่ นำตัวอักษรที่ได้ใส่ลงบัฟเฟอร์ และควบคุมการแสดงข้อมูลบนจอภาพ

การทำงาน

1. ถ้าสภาวะการทำงานเป็นชนิดเติมแทรก และจะสั่ง split ที่ตำแหน่งตัวชี้นั้น

2. ตรวจสอบการได้รหัสซ้ำ และจัดการจอภาพให้ถูกต้อง ถ้ามีรหัสเช่น รหัสบน, ล่าง มาซ้ำกัน หรือต้องแทนที่กัน ก็จะส่งสัญญาณเตือนสั้น ๆ โดยเรียกใช้ clrtoend, getblock, putblock, patch

3. ถ้าตำแหน่งแป้นพิมพ์อยู่ที่ตำแหน่งสุดท้ายของ fieldwidth ที่ตำแหน่งนี้ จะรอรับอักขระบน-ล่างได้อีก แต่ถ้าเพิ่มตัวอักขระระดับกลาง ลงไปอีก จะได้ Exit code เป็น 1 (และตัวอักขระนั้นจะถูกเก็บส่งต่อให้กับผู้เรียกใช้ Line Editor ด้วย)

procedure getblock (var c : char; pos : integer);

ตัวแปร c เป็นตัวแปรที่ใช้รับส่งข้อมูล

pos ตำแหน่งตัวชี้ใน บัฟเฟอร์

ผลลัพธ์ ถ้า c ไม่ถูกต้องจะส่ง c = Null คืน

หน้าที่ ตรวจสอบความถูกต้องของข้อมูลคำไทยอย่างละเอียด และจัดข้อมูลนั้นลงบัฟเฟอร์ที่ตำแหน่ง pos และแสดงผลบนจอภาพ

การทำงาน

1. ตรวจสอบชนิดข้อมูล c ที่ส่งมา

2. ถ้าถูกต้องกับคำไทย เก็บคำนั้นไว้ในบัฟเฟอร์ถ้าอักษรนั้นซ้ำ เช่นมีวรรณยุกต์อยู่แล้ว ใส่วรรณยุกต์อีกตัว จะมีเสียงเตือน แต่จะรับวรรณยุกต์ตัวใหม่แทนที่ตัวเก่า ถ้าไม่ถูกต้องส่งค่า c = Null คืน

3. แสดงผลบนจอภาพตามการเปลี่ยนแปลงที่เกิดขึ้น

หมายเหตุ หลักเกณฑ์ที่ใช้ตรวจสอบว่าจะรับรหัสใหม่ หรือแทนที่รหัสเดิม หรือไม่รับ ให้ใช้ข้อมูลในบัฟเฟอร์ที่ตำแหน่ง pos-1 ช่วยตัดสิน เช่น ถ้าเป็นพยัญชนะก็รับรหัสใหม่ได้ทุกตัว ถ้าเป็นสระบน-ล่าง จะถูกแทนที่ด้วยสระบน-ล่างใหม่ ถ้าเป็นวรรณยุกต์จะถูกแทนที่ด้วยวรรณยุกต์ใหม่ ถ้าเป็นสระหน้า จะไม่รับสระหน้าใหม่

procedure putblock (pos : integer);

ตัวแปร pos ตำแหน่งตัวชี้ในบัฟเฟอร์

ผลลัพธ์ -

หน้าที่ แสดงตัวอักษรในบัฟเฟอร์ที่ตำแหน่งตัวชี้ออกทางจอภาพ โดยมีการรวมสระบนและวรรณยุกต์หรือตัวที่ต้องต่อบนล่างอื่น ๆ

การทำงาน

1. ตรวจสอบชนิดของอักขระระดับกลาง เพื่อแยกประเภทควบคุมการออกจอภาพ
2. ถ้าเป็นตัวอักขระระดับ 1 และ 2 จะนำรหัสของทั้ง 2 ระดับมารวมเปลี่ยนเป็นรหัสออกจอภาพ หรือ ถ้าเป็นตัวอักขระระดับกลางที่มีการต่อบนล่างจะแสดงอักขระนั้นพร้อมตัวที่ต่อ เช่น ิ, ี, ึ, ุ, ู, ฺ

procedure split (pos, width : integer);

ตัวแปร pos ตำแหน่งที่จะเริ่มย้าย

width ความกว้างของการย้าย

ผลลัพธ์ -

หน้าที่ ใช้ย้ายข้อความจากบัฟเฟอร์ไปบัฟเฟอร์ชั่วคราวตั้งแต่ตำแหน่งที่กำหนด

วิธีทำงาน

1. ลบข้อมูลในบัฟเฟอร์ชั่วคราวออกทั้งหมด
2. ย้ายข้อมูลในบัฟเฟอร์ตั้งแต่ตำแหน่งที่ pos ไปที่บัฟเฟอร์ชั่วคราวทั้งหมด

procedure patch (pos, width : integer);

ตัวแปร pos ตำแหน่งที่จะนำมาต่อ

width ความกว้างช่องรับข้อมูล

ผลลัพธ์ –

หน้าที่ นำข้อมูล 2 เท่าของความกว้างช่องข้อมูลจากบัฟเฟอร์ชั่วคราวมาต่อท้ายบัฟเฟอร์ที่ตำแหน่ง pos และแสดงผลบนจอภาพ

วิธีทำงาน

1. ย้ายข้อมูลจำนวน 2 เท่าของความกว้างที่กำหนด จากบัฟเฟอร์ชั่วคราวมาต่อท้ายบัฟเฟอร์ที่ตำแหน่ง pos ส่วนที่เกิน 2 เท่าของความกว้างที่กำหนด จะถูกตัดทิ้ง

2. แสดงผลตั้งแต่ pos จนสุดความกว้างที่กำหนดใหม่ด้วย putblock

procedure packline (buffer : fourline; var line : strs; i : integer);

ตัวแปร buffer ตัวแปรข้อมูลภายในของ Line Editor

i จำนวนคอลัมน์ในบัฟเฟอร์ที่ต้องการจัดเป็นแถวเดียว

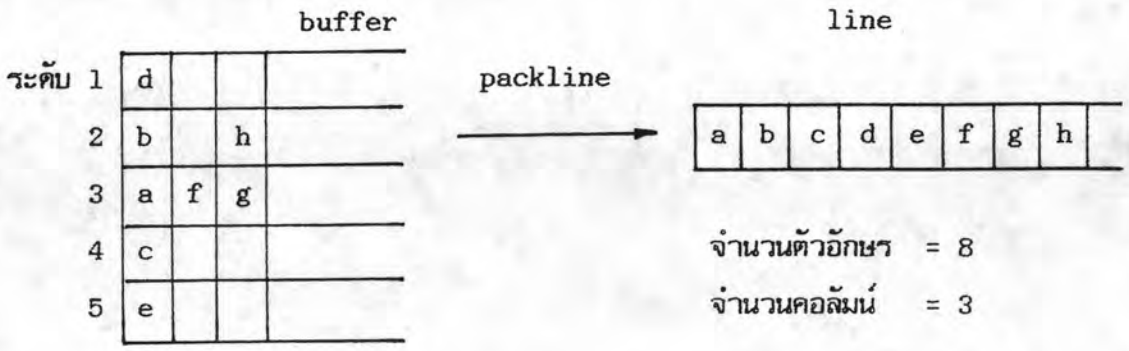
ผลลัพธ์ line ข้อมูลแถวเดียวที่ได้

หน้าที่ แปลงรูปข้อมูลจากบัฟเฟอร์เป็นแถวเดียวเพื่อประหยัดหน่วยความจำ เป็นการทำงานตรงข้ามกับ unpackline

วิธีทำงาน

1. เรียงลำดับข้อมูลอักขระภายในบัฟเฟอร์ ทีละระดับจาก 3,2,4,1,5 ตั้งแต่คอลัมน์แรก จนถึงคอลัมน์ที่กำหนดด้วย i จัดเป็นแถวเดียวลงในตัวแปร Line ระดับใดที่ไม่มีข้อมูลให้ข้ามไป พร้อมนับจำนวนตัวอักษรและจำนวนคอลัมน์ด้วย

2. จัดรูปแบบข้อมูลให้แก่ตัวแปร Line เพื่อส่งกลับ



รูปที่ 4.5 แสดงการจัดลำดับอักขระ

procedure unpackline (line : strs; var buffer : fourline);

ตัวแปร line ข้อมูลแถวเดียวที่รับเข้า

ผลลัพธ์ buffer ผลลัพธ์ชนิด buffer

หน้าที่ แปลงข้อมูลจากแถวเดียวเป็นแบบบัฟเฟอร์ เพื่อให้ Line editor สะดวกในการทำงาน เป็นการทำงานตรงข้ามกับ packline

วิธีทำงาน

1. ลบข้อมูลในบัฟเฟอร์ออกทั้งหมด
2. เรียงลำดับข้อมูลจากแถวเดียว ไปเป็นแบบบัฟเฟอร์ ตามความยาวของ line โดยพิจารณาจากชนิดของตัวอักขระที่อยู่ใน line

procedure clrtoend (pos, width : integer);

ตัวแปร pos ตำแหน่งในบัฟเฟอร์ที่ต้องการลบ

width ความกว้างของช่องของข้อมูล

ผลลัพธ์ -

หน้าที่ ใช้ลบข้อความบนจอภาพตั้งแต่ตำแหน่งที่กำหนดไปจนจบความกว้างที่กำหนด โดยลบเฉพาะเมื่อมีอักขระระดับบนล่างเท่านั้น เพื่อไม่ให้กระทบกระเทือนกับข้อมูลที่แสดงบนจอภาพของบรรทัดบนและล่างถัดไป

วิธีทำงาน

1. ตรวจสอบอักขระจากในบัฟเฟอร์ ตั้งแต่ช่องท้ายสุดของความกว้างที่กำหนดมาจนถึงตำแหน่งที่กำหนด และทำการลบอักขระบนจอภาพ และควบคุมตำแหน่งบนจอภาพให้ถูกต้อง
   - ลบอักขระระดับกลาง
   - ลบอักขระระดับบน เมื่อมีข้อมูลในระดับ 1,2 หรือระดับกลางเป็นชนิดต่อบน
   - ลบอักขระระดับล่าง เมื่อมีข้อมูลในระดับ 4 หรือระดับกลางเป็นชนิดต่อล่าง

procedure writeline (line : strs);

ตัวแปร line ข้อมูลแบบแถวเดียวที่ต้องการแสดงบนจอภาพ

ผลลัพธ์ -

หน้าที่ ใช้แสดงข้อมูลแถวเดียว จัดตำแหน่งบนจอภาพ โดยถือว่าข้อมูลใน line เรียง

ถูกต้องตามลำดับที่กำหนด

วิธีทำงาน

1. อ่านข้อมูลจาก line พิจารณาชนิดข้อมูลทีละตัวเพื่อผลในการรวมข้อมูลเป็นรหัสใหม่เพื่อแสดงผลออกทางจอภาพ ที่ตำแหน่งเคอร์เซอร์

2. ตำแหน่งเคอร์เซอร์ อยู่ที่เดิม

4.2.4 <u>การนำไปใช้งาน</u>

ในหัวข้อนี้จะกล่าวถึงการนำไปใช้งานของ Line Editor พร้อมทั้งยกตัวอย่างเพื่อให้ผู้ใช้สามารถเข้าใจได้โดยง่าย

การนำ Line Editor ไปใช้งาน ในที่นี้จะหมายถึงการนำ source code module ของส่วน line Editor ไปใช้รวมกับโปรแกรมผู้เรียกใช้ ในการออกแบบ Line Editor ให้เป็นเครื่องมือนี้ได้คำนึงถึงการออกแบบเป็นลักษณะโมดูล คือแต่ละส่วนทำงานตามหน้าที่เฉพาะอย่างเท่านั้น โดยมีการส่งผ่านข้อมูลที่จำเป็นถึงกัน

- ข้อมูลที่ Line Editor ต้องใช้ คือ

1. buffer, patch buffer เป็นตัวแปรที่ใช้รับข้อมูล และเก็บข้อมูลชั่วคราว โดยมีรูปแบบที่ให้ Line Editor ทำงานได้
2. first character เป็นตัวอักษรหรือ แป้นควบคุม ที่ส่งให้ Line Editor ก่อนควบคุมการทำงาน
3. position in buffer เป็นตำแหน่งในบัฟเฟอร์ ที่ต้องสัมพันธ์กับตำแหน่งบนจอภาพ
4. fieldwidth เป็นความยาวของข้อมูลที่ต้องการรับ
5. document flag เป็นสถานะบอกว่าต้องการข้อมูลแบบ document ซึ่งจะมีบรรทัดแสดงสถานะขึ้นบนจอภาพ

- ข้อมูลที่ Line Editor ส่งคืนผู้เรียกใช้

1. line เป็นข้อมูลที่ได้จากการรับหรือแก้ไขแล้ว
2. exit code เป็นรหัสการออกจาก Line Editor เพื่อให้โปรแกรมผู้เรียกใช้รู้ถึงสถานะการออกต่าง ๆ

- ข้อมูลที่ต้องเตรียมล่วงหน้า

1. กำหนดข้อมูลที่จะส่งมาให้อยู่ในรูปของบัฟเฟอร์เสียก่อน
2. เตรียมข้อมูลกำหนดสภาวะต่าง และตัวแปรต่างๆ Line Editor
3. กำหนดตำแหน่งเคอร์เซอร์ บนจอภาพให้สัมพันธ์กับตัวแปรตำแหน่งที่ส่งให้ Line Editor

เนื่องจากข้อมูลที่ใช้ทั่วไปเป็นชนิดแถวเดียว ก่อนเรียกใช้ Line Editor จึงต้องถ่ายข้อมูลนั้นไปที่บัฟเฟอร์ เพื่อความสะดวกในการใช้งาน ด้วยคำสั่ง unpackline กำหนดสภาวะแวดล้อมการทำงาน และเตรียมข้อมูลตัวอักษรที่ให้ Line Editor รับไปทำงานต่อ ไม่ว่าจะเป็นชนิด Function, Control หรือ Character ถ้าเป็น Null แล้ว Line Editor จะรอรับข้อมูลตัวแรกด้วย กำหนดความกว้างของข้อมูลไม่เกิน 80 ตัวอักษร และกำหนดตำแหน่งเคอร์เซอร์ให้ สัมพันธ์กับตำแหน่งตัวชี้ใน Buffer ด้วย ถ้าตำแหน่งของเคอร์เซอร์บนจอภาพกับตำแหน่งตัวชี้ของ buffer ไม่สัมพันธ์กัน จะทำให้การแสดงผลผิดพลาด หลังออกจาก Line Editor ด้วยวิธีใดวิธีหนึ่ง ผลลัพท์ที่ได้จะอยู่ที่ตัวแปร Line หรือ อาจนำไปจาก บัฟเฟอร์ หรือ บัฟเฟอร์ชั่วคราว ก็ได้ถ้าต้องการ โดยจะมี exit code กำกับว่าออกจาก Line Editor ด้วยวิธีใด

ข้อมูลที่ส่งออกจาก Line Editor ในบัฟเฟอร์ และ Line จะมีข้อมูลเหมือนกัน แต่ต่างกันที่โครงสร้างของตัวแปรเท่านั้น ส่วนในบัฟเฟอร์ชั่วคราวนั้น จะใช้ได้ก็ต่อเมื่อมีการลบคำจนหมดทั้งบรรทัดจากตำแหน่งเคอร์เซอร์ โดยกดปุ่มควบคุม (Enter) ในขณะที่สภาวะการทำงานเป็นแบบ เติมแทรก ข้อมูลตัวอักษรส่วนที่หายไปนั้น จะยังอยู่ในบัฟเฟอร์ชั่วคราว

```
procedure edline (c : char; var midpos : integer;
                  fieldwidth : integer;
                  var next : integer;  doc : boolean);
```

ตัวแปร c รหัสอักษรตัวแรก ประกอบกับตัวแปร funckey บอกชนิด funtion

ถ้า Funckey เป็น true c เป็นรหัสของ Function นั้น

ถ้า Funckey เป็น False c เป็นรหัสควบคุม หรือ รหัสอักขระ

และถ้า c เป็น Null จะรอรับแป้นใหม่

midpos เป็นตัวชี้ตำแหน่งในบัฟเฟอร์ เปลี่ยนตามตำแหน่งขณะนั้น

fieldwidth กำหนดความกว้างของช่องรับข้อมูล มีค่าน้อยกว่า 80 เพื่อความถูกต้องในการจัดการหน้าจอภาพ midpos > fildwidth จะได้ exit code

next เป็นตัวแปรที่ใช้ส่งคำ exit code คืนผู้เรียกใช้

doc เป็นตัวแปรบอกสภาวะการทำงาน เพื่อใช้ทำ word processor

ตัวแปรร่วม _shead : integer ใช้เปลี่ยนบัฟเฟอร์ของแป้นพิมพ์

buffer : fourline เป็นตัวแปรทำงานของ EdLine และจะเก็บค่านั้นไว้จนกว่าจะสั่งลบ

patchbuffer : fourline เป็นตัวแปรทำงานของ EdLineเก็บข้อมูลที่ย้ายจาก buffer เป็นที่เก็บข้อมูลส่วนที่ลบไปแล้ว เมื่อสั่งลบจนหมดบรรทัด

Kb,Con,typ : dta ตารางแปลงรหัส และตารางคุณสมบัติของรหัสนั้น

ผลลัพธ์ midpos ตำแหน่งตัวชี้ในบัฟเฟอร์ในขณะที่ออกจาก EdLine เป็นตำแหน่งแป้นพิมพ์ ที่นับสัมพันธ์กับตำแหน่งเริ่มต้นในบรรทัดนั้น

next เป็นรหัสการออกจาก EdLine มีค่าเมื่อเป็นกรณีต่าง ๆ ดังนี้

-2	[UArr]	101-110	[F1..F10]
2	[DArr]	181-190	[Shift-F1..F10]
-3	[PgUp]	191-200	[Ctrl-F1..F10]
3	[PgDn]	201-210	[Alt-F1..F10]
-4	[Home]	115-118	[Ctrl-LArr,RArr,Home,End]
4	[End]	132	[Ctrl-PgUp]
13	[Return]	14	[Ctrl-N]
25	[Ctrl-Y]	9	[TAB]
8	[Ctrl-T]	10	[Ctrl-B]
-1	เลื่อนซ้ายเกินขอบ	1	เลื่อนขวาตกขอบ
225	[ESC]		

ตัวอย่างการนำไปใช้งาน

```
{$I  TYPHEAD.EDT}
{$I  VARHEAD.EDT}
{$I  GEDITLN.EDT}
begin
        InitGraphic;                {ทำให้การแสดงผลเป็นชนิดกราฟฟิค}
        ReadTable;                  {อ่านตารางรหัสควบคุม}
        GetChar;                    {อ่านคำจากแป้นพิมพ์}
        SetLineValue;               {เตรียมข้อมูลในตัวแปร Line }
        UnpackLine;                 {เปลี่ยนข้อมูลให้อยู่ในรูปแบบของบัฟเฟอร์}
        SetScreenPosition;          {กำหนดตำแหน่งบนจอภาพให้สัมพันธ์กับตัวชี้}
        EdLine                      {เรียกใช้ Line Editor}
        :                           {ได้คำที่แก้ไขแล้วกลับมาในตัวแปร Line}
        LeaveGraphic                {จบการแสดงผลแบบกราฟฟิค}
end.
```

4.3 Screen Editor

หน้าที่หลักของ Screen Editor คือ รับ และ/หรือ แก้ไขข้อมูล (ไทย/อังกฤษ) ของระเบียนในแฟ้มข้อมูล โดยใช้ฟอร์ม เป็นตัวกำหนดตำแหน่งแสดงข้อมูลของระเบียน ทั้งนี้ผู้ใช้สามารถที่จะควบคุม การเลื่อนตำแหน่งเคอร์เซอร์ เพื่อแก้ไขข้อมูลในลักษณะที่เรียกว่าแก้ไขได้ทั้งจอ (Full Screen Editing)

เพื่อเป็นการเปรียบเทียบ วิธีการโดยทั่ว ๆ ไป กับวิธีการทำงานของ Screen Editor จะขอกล่าวถึงการรับข้อมูล 1 ระเบียน ในภาษาปาสคาลเสียก่อน ดังนี้

ข้อมูลของระเบียนจะต้องถูกกำหนดโครงสร้างของข้อมูลไว้ก่อน เช่น ระเบียนผลสอบของนิสิตที่โครงสร้างข้อมูลคือ

```
Type Student = record
                    ID : String [7];
                    Name : String [30];
                    Score : Integer;
                    Grade : Char;
                end;
var  Sdata : Student;
     Sfile : file of Student;
```

โครงสร้างข้อมูลของระเบียนผลสอบของนิสิต ประกอบด้วย รหัส, ชื่อ, คะแนน, เกรด โดยมีตัวแปรที่ใช้โครงสร้างข้อมูลนี้ คือ sdata และตัวแปรที่เก็บโครงสร้างแฟ้มข้อมูลของโครงสร้างข้อมูล Student คือ Sfile

ในการรับข้อมูล 1 ระเบียน ส่วนของโปรแกรม อาจเป็นดังนี้เช่น

```
Write ('ID:'); Readln  (Sdata.ID);
Write ('NAME:'); Readln (Sdata.Name);
Write ('SCORE:'); Readln (sadta. Score);
Write ('GRADE:'); Readln (Sdata Grade);
```

เมื่อโปรแกรมทำงานมาถึงคำสั่งเหล่านี้ ก็จะให้ผู้ใช้ใส่ข้อมูลทีละบรรทัด ในแต่ละบรรทัด ถ้าใส่ผิดก็ต้องแก้ไขให้ถูกต้องเสียก่อน ที่จะกด [Enter] ซึ่งหมายความว่า ผู้ใช้จะต้องใส่ข้อมูลอย่างระมัดระวังหากใส่ผิด และกด [Enter] ไปแล้ว จะแก้ไขยุ่งยาก ต้องมีส่วนของโปรแกรมเพิ่มเติมอีก นอกจากนี้ถ้าต้องการให้ตำแหน่งแสดงรับข้อมูลสวยงาม อาจเพิ่มเติมคำสั่ง GotoXY (X[I], Y[I]) โดยที่ X[I], Y[I] เป็นตำแหน่งบนจอภาพ (เช่น X[I] = 10, i = 1....4 และ Y[I] ถึง Y[4] = 3,5,7,9 ตามลำดับ) เพิ่มคำสั่งนี้ที่ทุกบรรทัดจะเป็นการจัดตำแหน่งแสดงรับข้อมูลบนจอภาพให้ตรงตามที่กำหนด

ในขณะที่กำลังรับข้อมูล ถ้าข้อมูลตัวที่ผ่านมาแล้ว บังเอิญใส่ผิดเอาไว้ แต่มารู้ภายหลัง คือหลังจากได้กด [Enter] ไปแล้ว หากต้องการให้สามารถย้อนกลับไปแก้ไขตัวที่ผิดได้ ผู้เขียนโปรแกรมจะต้องแทรกคำสั่งเพื่อควบคุมให้การทำงานย้อนกลับได้ เช่น ใส่ข้อมูล ID ผิด

กำลังที่จะใส่ข้อมูลของ Name ถ้ากดแป้นคำสั่งหนึ่ง (เช่น [Up Arrow] เพื่อสั่งให้ย้อนขึ้นไปแก้ไขข้อมูล ID เสียให้ถูกต้อง สมมุติว่า ย้อนกลับขึ้นไปแล้ว ถ้าเป็นวิธีการง่ายๆ ก็ให้ใส่ข้อมูล ID ใหม่ทั้งหมด แต่ถ้าจะให้สามารถเลื่อนเคอร์เซอร์ ไปเติมแทรกตัวที่พิมพ์ตก หรือ พิมพ์ทับตัวที่พิมพ์ผิด หรือลบตัวเกิน คำสั่งที่ใช้ก็จะซับซ้อนมากขึ้น

เนื่องจากการประยุกต์การใช้งาน คอมพิวเตอร์ในฐานข้อมูล มักจะมีเปลี่ยนแปลงแก้ไขข้อมูลอยู่เสมอ ๆ ถ้าหากว่าจะแก้ไขโดยวิธีใส่เข้าไปใหม่ทั้งหมดก็จะเป็นการไม่สะดวกทำงานล่าช้า ดังนั้นจึงควรมีฟังก์ชันที่ช่วยให้สามารถแก้ไขเฉพาะ ข้อมูลย่อยที่ต้องการ และเฉพาะตัวอักษรที่ต้องการแก้ไขเท่านั้น

จะเห็นได้ว่า วิธีการที่จะรับข้อมูล 1 ระเบียน ที่สามารถแก้ไขข้อมูลได้อย่างอิสระนั้นจะต้องใช้วิธีที่ซับซ้อน อย่างไรก็ตาม ในหัวข้อ 4.2 เราได้กล่าวถึง หน้าที่และการใช้งานของ Line Editor เราพบว่า การรับและแก้ไขข้อมูลย่อยของระเบียน สามารถใช้ Line Editor รับหรือแก้ไขข้อมูลนั้น ๆ เช่น เมื่อจะรับหรือแก้ไขข้อมูล ID ก็จะส่งถ่ายข้อมูลใน Sdata.ID ให้แก่ตัวแปร Line ของ Line Editor และกำหนดสภาวะแวดล้อมของการทำงาน ให้แป้นพิมพ์ทำหน้าที่เป็นภาษาอังกฤษ และให้อยู่ในภาวะเติมแทรก วางตำแหน่งเคอร์เซอร์ไปที่ตำแหน่งแรกของข้อมูล ให้ข้อมูลที่มีความยาว 7 ตัว และให้ Line Editor ทำหน้าที่รับหรือแก้ไขข้อมูลของตัวแปร Line จนเป็นที่พอใจ เมื่อมีคำสั่งจบการทำงานของ Line Editor แล้ว ข้อมูลของตัวแปร Line ก็จะถูกส่งถ่ายคืนให้แก่ Sdata.ID

การทำงานกับข้อมูล Sdata.Name ก็มีลักษณะเช่นเดียวกับ Sdata.ID อาจจะแตกต่างกันตรงที่กำหนดให้แป้นพิมพ์ ทำหน้าที่เป็นภาษาไทย

สำหรับข้อมูลของ Sdata.Score ซึ่งเป็นชนิด Integer ก่อนที่จะส่งถ่ายให้กับตัวแปร Line ซึ่งเป็น Array of char ต้องแปลงข้อมูลนี้ให้เป็นชนิดเดียวกันเสียก่อน และในทางกลับกัน เมื่อ Line Editor ทำงานเสร็จแล้ว ก็จะแปลงข้อมูลของตัวแปร Line ส่งถ่ายกลับมาเป็นชนิด Integer

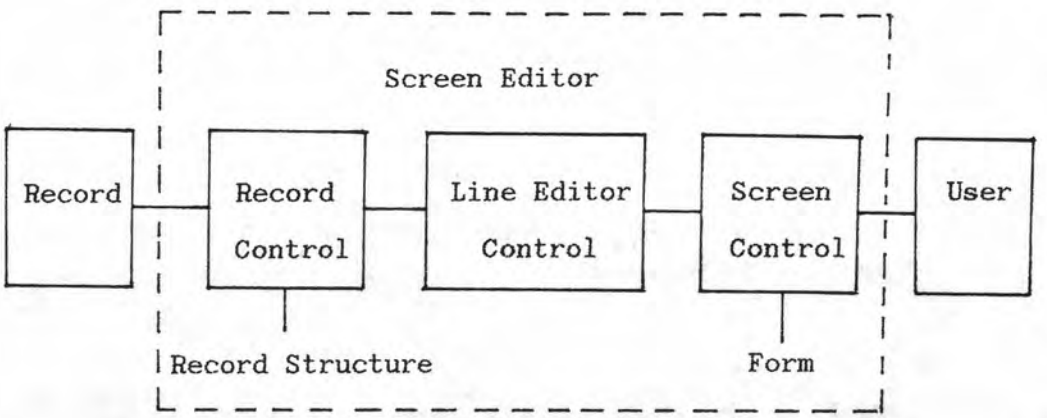
สำหรับข้อมูล Sdata.Grade ก็มีลักษณะเช่นเดียวกัน

โดยการทำงานวิธีนี้ การจัดการของข้อมูลย่อย เป็นหน้าที่ของ Line Editor การจัดการของข้อมูลทั้งระเบียนเป็นหน้าที่ของ Screen Editor โดย Scren Editor มีหน้าที่ 3 ส่วนคือ ส่วนที่เกี่ยวกับระเบียน, ส่วนที่เกี่ยวกับ Line Editor และส่วนที่เกี่ยวกับการแสดงผลบนจอภาพ

ส่วนที่เกี่ยวกับระเบียน จะทำหน้าที่ส่งถ่ายข้อมูลย่อยแต่ละตัวในระเบียนกับตัวแปร Line ส่งคุณสมบัติของข้อมูลนั้น เช่น ชนิดของข้อมูล, ความยาวของข้อมูล และตรวจสอบความต้องการของข้อมูลนั้น

ส่วนที่เกี่ยวกับ Line Editor ทำหน้าที่กำหนดสภาวะการทำงาน และรับผลของการทำงานนั้น (Exit code)

ส่วนที่เกี่ยวกับการแสดงผลบนจอภาพ ส่วนนี้ทำหน้าที่แสดงฟอร์ม, ควบคุมตำแหน่งเคอร์เซอร์บนจอ



รูปที่ 4.6 ความสัมพันธ์ระหว่างส่วนประกอบภายใน Screen Editor

จากรูปที่ 4.6 จะเป็นการแสดงความสัมพันธ์ระหว่างส่วนต่าง ๆ ทั้ง 3 ภายใน Screen Editor โดยทางผู้ใช้สามารถติดต่อกับระเบียนแต่ละระเบียนได้ โดยผ่านทาง Screen Editor โดยมีข้อมูลที่ต้องรู้ก่อนคือ ฟอร์ม และโครงสร้างข้อมูล

ต่อไปจะกล่าวถึงหน้าที่อย่างละเอียดของ Screen Editor โดยกล่าวถึงสภาวะแวดล้อม การทำงานก่อน

4.3.1 <u>หน้าที่อย่างละเอียดของ Screen Editor</u>

สภาวะแวดล้อมของการทำงาน ของ Screen Editor ประกอบด้วย

- ฟอร์ม
- โครงสร้างข้อมูล

ฟอร์ม หมายถึง ข้อมูลที่เก็บอยู่ในแฟ้มใดแฟ้มหนึ่ง ที่บอกลักษณะของข้อมูลที่ปรากฏบนจอภาพ ซึ่งประกอบด้วยข้อความถาวร และตำแหน่งที่แสดงบนจอภาพ, ตำแหน่งของช่องรับข้อมูลพร้อมความยาวของช่องรับข้อมูล และตำแหน่งแสดงข้อความพิเศษ ดังรายละเอียดในหัวข้อที่ 3.2

โครงสร้างข้อมูล หมายถึง สิ่งที่บอกชนิดของข้อมูลย่อย แต่ละตัวในระเบียน เช่น ระเบียน Sdata ที่เป็นตัวอย่างที่ได้กล่าวไปแล้ว จะมีโครงสร้างของข้อมูลเป็น 'S7T23IC'

จากข้อมูลของฟอร์มที่กำหนดมาให้ Screen Editor ก็จะมาสร้างจอภาพเพื่อเป็นสื่อติดต่อกับผู้ใช้ เช่น จากตัวอย่าง ระเบียนผลสอบของนิสิต ที่กล่าวมาข้างต้น จะสร้างจอภาพ ดังรูป

ผลสอบของนิสิต

รหัส : :

ชื่อ : :

คะแนน : :

เกรด : :

รูปที่ 4.7 แสดงตัวอย่างหน้าจอภาพระเบียนผลสอบของนิสิต

จากข้อกำหนดโครงสร้างข้อมูล Screen Editor จะนำมาใช้แบ่งเขตข้อมูลย่อยสำหรับส่งถ่ายข้อมูลย่อยเหล่านั้น ให้กับ Line Editor และใช้กำหนดควบคู่ไปกับช่องรับข้อมูล ตัวอย่างการแบ่งเขตข้อมูลย่อยของระเบียนผลสอบของนิสิต ดังรูปที่ 4.8

สถานะการลบ	รหัส	ชื่อ	คะแนน	เกรด
B	S7	T23	I	C
0	1	9	40	42 43

รูปที่ 4.8 แสดงการแบ่งเขตข้อมูลย่อยภายในระเบียน

ตำแหน่งแรกของระเบียน เป็นสถานะการลบ ชนิดตรรกะขนาด 1 ไบต์
ตำแหน่งที่ 2 ของระเบียน เป็นรหัส ชนิดข้อความ 7 ตัวอักษร ขนาด 8 ไบต์
ตำแหน่งที่ 3 ของระเบียน เป็นชื่อ ชนิดข้อความภาษาไทย 23 คอลัมน์ ขนาด 31 ไบต์
ตำแหน่งที่ 4 ของระเบียน เป็นคะแนน ชนิดเลขจำนวนเต็ม ขนาด 2 ไบต์
ตำแหน่งที่ 5 ของระเบียน เป็นเกรด ชนิดตัวอักษร ขนาด 1 ไบต์
ในระเบียนนี้มีข้อมูล 43 ไบต์ สถานะการลบจะมีปรากฏอยู่เองโดยไม่ต้องกำหนด

ต่อไปจะกล่าวถึงสภาวะเริ่มต้นการทำงานของ Screen Editor คือ

- เมื่อแสดงข้อมูลในระเบียน
- เมื่อเป็นฟอร์มว่าง
- เมื่อแสดงข้อมูลเดิมที่มีอยู่ในหน่วยความจำแล้ว

ขณะแสดงข้อมูลในระเบียน ต้องเตรียมระเบียน และสภาวะแวดล้อมการทำงานให้แก่ Screen Editor สภาวะแวดล้อมการทำงานคือ ฟอร์ม และโครงสร้าง ข้อมูลต้องสัมพันธ์กัน โครงสร้างข้อมูลต้องตรงกับช่องรับข้อมูล หรือช่องแสดงข้อมูลพิเศษ รหัสที่ใช้ในโครงสร้างข้อมูลจะกำหนดการตรวจสอบความถูกต้องในช่องรับข้อมูลและกำหนดสภาวะแวดล้อมการทำงานไทย/อังกฤษ ให้แก่ Line Editor

เมื่อเป็นฟอร์มว่าง ต้องการให้ Screen Editor รับข้อมูลใหม่โดยใช้ฟอร์มเดิม ให้กำหนดชื่อฟอร์ม เพื่อเรียกใช้ฟอร์มว่างตามสภาวะแวดล้อมการทำงานที่กำหนดไว้แล้ว วิธีการนี้ประหยัดเวลาในการอ่านข้อมูลฟอร์มจากแฟ้มข้อมูลอีก

เมื่อแสดงข้อมูลเดิมที่มีอยู่ในหน่วยความจำ วิธีใช้ออกจาก Soreen Editor และกลับเข้า Screen Editor โดยมีฟอร์มและข้อมูลที่แสดงเหมือนเดิม ก็ไม่จำเป็นต้องเตรียมข้อมูลใหม่ เพียงกำหนดชื่อฟอร์ม เพื่อแสดงฟอร์มและระเบียนเดิม ทำให้ประหยัดเวลาอ่านข้อมูลฟอร์มจากแฟ้มข้อมูล

ต่อไปจะกล่าวถึงการควบคุมตำแหน่งเคอร์เซอร์บนจอภาพ จากตัวอย่างฟอร์มผลสอบของนิสิต การวางช่องรับข้อมูลเป็นแบบแนวตั้ง ดังนั้นการเลื่อนเคอร์เซอร์จะวิ่งลงทีละช่อง จากรหัส ชื่อ คะแนน เกรด ตามลำดับ เมื่อถึงช่องล่างสุดจะกลับไปช่องบนสุดอีกครั้ง แต่ในการควบคุมตำแหน่งเคอร์เซอร์บนจอภาพ ยังมีลักษณะอื่นอีก เช่น ในกรณีที่ฟอร์มผลสอบนิสิตเป็นดังรูปที่ 4.9

ผลสอบของนิสิต

รหัส : : ชื่อ : :

คะแนน : : เกรด : :

รูปที่ 4.9 แสดงการจัดหน้าฟอร์มผลสอบของนิสิต อีกแบบหนึ่ง

การจัดช่องรับข้อมูลแบบนี้ เพื่อให้สามารถเข้าถึงช่องรับข้อมูล แต่ละช่องได้ โดยรวดเร็ว จะแบ่งชนิดการเข้าถึงช่องรับข้อมูล เป็น 2 ชนิดคือ

- การเข้าถึง ช่องรับข้อมูลในแนวตั้ง
- การเข้าถึง ช่องรับข้อมูลในแนวนอน

การเข้าถึงช่องรับข้อมูลในแนวตั้ง เป็นการเลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลที่ใกล้ที่สุดในแนวตั้ง ไม่ว่าจะเป็นการเลื่อนขึ้น หรือเลื่อนลง ตัวอย่างเช่น ตามรูปที่ 4.9 เมื่อเลื่อนเคอร์เซอร์ลง เริ่มต้นเคอร์เซอร์อยู่ที่ช่องรหัส จะเลื่อนมาที่ช่องคะแนน และกลับไปช่องรหัสอีก หรือถ้าเริ่มต้นที่ช่องชื่อ จะมาที่ช่องเกรด และกลับไปช่องชื่ออีก เนื่องจากเป็นช่องที่ใกล้ที่สุดในแนวตั้ง สำหรับการเลื่อนขึ้นก็มีลักษณะเช่นเดียวกัน การเลื่อนเคอร์เซอร์ขึ้นกำหนดให้ใช้แป้นควบคุม [Up Arrow] และ [Down Arrow] สำหรับเลื่อนเคอร์เซอร์ลง

การเข้าถึงช่องรับข้อมูลในแนวนอน เป็นการเลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลถัดไป เมื่อเลื่อนมาถึงช่องสุดท้ายทางมุมล่างขวาของจอภาพ ก็จะเลื่อนไปที่ช่องแรกทางมุมบนซ้ายอีก หรือกลับกัน เช่น เมื่อสั่งให้เลื่อนเคอร์เซอร์ไปทางขวา เริ่มที่ช่องรหัส ก็จะเลื่อนไปที่ช่อง ชื่อ, คะแนน, เกรด และกลับมาที่ช่องรหัสอีกตามลำดับ หรือจะกลับลำดับกันเมื่อสั่งให้เลื่อนเคอร์เซอร์ไปทางซ้าย ในที่นี้ได้กำหนดให้แป้นควบคุม ([Tab]) ในการเลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลถัดไปทางขวา นอกจากนี้การเลื่อนเคอร์เซอร์ไปยังช่องทางขวาอาจเกิดได้จากกรณีที่ออกจาก Line Editor ด้วยการเลื่อนเคอร์เซอร์ภายในช่องรับข้อมูลไปจนสุดช่องรับข้อมูลทางขวา สำหรับการเลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลถัดไปทางซ้าย เกิดในกรณีที่ออกจาก Line Editor ด้วยการเลื่อนเคอร์เซอร์ภายในช่องรับข้อมูลนั้นไปจนสุดช่องรับข้อมูลทางซ้าย

หน้าที่ที่สำคัญอีกอย่างหนึ่งของ Screen Editor คือการตรวจสอบความถูกต้องของข้อมูล (Intrigity) ทำให้ข้อมูลไม่ผิดพลาด เช่น ช่องที่เป็นตัวเลข ไม่ควรมีตัว

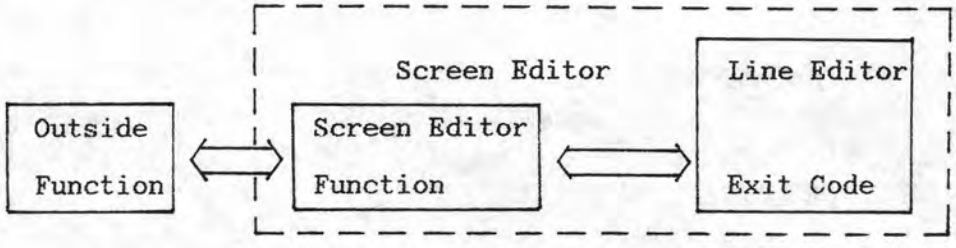
อักษรปน การตรวจสอบนี้จะทำ 2 ครั้งคือ เมื่อเคอร์เซอร์เลื่อนออกจากช่องนั้น และเมื่อสั่งจบการทำงานของ Screen Editor

เมื่อเคอร์เซอร์เลื่อนออกจากช่องรับข้อมูล Screen Editor จะทำการตรวจสอบข้อมูลในช่องนั้นว่าตรงตามโครงสร้างข้อมูลที่ได้กำหนดไว้หรือไม่ ถ้าไม่ถูกต้อง จะมีสัญญาณเตือน และเคอร์เซอร์จะกลับไปที่ช่องรับข้อมูลนั้นอีก

เมื่อสั่งจบการทำงานของ Screen Editor จะมีการตรวจสอบความถูกต้องของข้อมูลทุกช่องอีกครั้งหนึ่ง ถ้าพบว่าช่องใดไม่ถูกต้อง จะมีสัญญาณเตือน และเคอร์เซอร์จะเลื่อนไปยังช่องรับข้อมูลที่ข้อมูลไม่ถูกต้องนั้น

การควบคุมการทำงานของ Screen Editor จะแบ่งเป็น 3 ระดับคือ การควบคุมภายในของ Line Editor การควบคุมในภายใน Screen Editor และการควบคุมภายนอก Screen Editor ดังรูปที่ 4.10

ส่วนควบคุมภายในของ Screen Editor ควบคุมการเลื่อนเคอร์เซอร์ไปตามช่องรับข้อมูลต่างๆ แต่ถ้ามีคำสั่งของ Line Editor ก็จะให้ Line Editor นั้นทำงาน คำสั่งอื่น ๆ นอกจากของ Line Editor และ Screen Editor จะถูกส่งออกไปให้ตัวควบคุมการทำงานภายนอกของ Screen Editor อีกทีหนึ่ง



รูปที่ 4.10 แสดงความสัมพันธ์ระหว่างฟังก์ชันการทำงานภายใน Screen Editor

สรุปหน้าที่การทำงานของ Screen Editor

1. ควบคุมการเรียกใช้ฟอร์มคู่กับระเบียนต่าง ๆ
2. ควบคุมการเรียกใช้ Line Editor
3. ควบคุมการเลื่อนตำแหน่งเคอร์เซอร์ไปยังช่องรับข้อมูลต่าง ๆ
4. ตรวจสอบความถูกต้องของข้อมูล ในช่องรับข้อมูลต่าง ๆ

4.3.2 <u>ส่วนประกอบของโปรแกรมที่เกี่ยวข้อง</u>

หัวข้อนี้จะกล่าวถึงวิธีการทำงานอย่างละเอียด และส่วนประกอบของโปรแกรมที่เกี่ยวข้องเพื่อประกอบกันเป็นโปรแกรม Screen Editor และตัวแปรที่ใช้

ใน Screen Editor ได้กำหนดตัวแปรที่ใช้งานร่วมกันชื่อ InputData ชนิดกลุ่มข้อความ แทนช่องแสดงข้อมูลแต่ละช่อง เพื่อใช้เป็นตัวกลางในการรับ หรือแก้ไขข้อมูลกับฟอร์มที่กำหนด ตัวแปรนี้จะได้กล่าวถึงต่อไป

ขั้นตอนการทำงานของ Screen Editor

1. กำหนดฟอร์ม และโครงสร้างของข้อมูล
2. ถ่ายข้อมูลระเบียน และข้อความพิเศษ ไปที่ InputData
3. แสดงฟอร์มว่าง
4. แสดงข้อมูล InputData ตามฟอร์มที่กำหนด
5. ใช้ Line Editor ในการรับข้อมูลของช่องรับข้อมูลต่าง ๆ
6. ตรวจสอบความถูกต้องของข้อมูลภายในช่องรับข้อมูล ถ้าไม่ถูกต้องไปทำขั้นที่ 5 ถ้าถูกต้อง เลื่อนเคอร์เซอร์ไปช่องรับข้อมูลถัดไป ตามที่กำหนดไว้
7. ถ้าจบการทำงานของ Screen Editor จะตรวจสอบความถูกต้องของข้อมูลภายในช่องรับข้อมูลทุกช่องอีก ถ้าไม่ถูกต้องไปทำขั้นที่ 5
8. ถ่ายข้อมูลจาก InputData ไปเป็นข้อมูลของระเบียนนั้น

รายละเอียดแต่ละขั้นตอนเป็นดังนี้

กำหนดฟอร์มและโครงสร้างข้อมูล เป็นการอ่านข้อมูลฟอร์มจากแฟ้มข้อมูล โดยผู้เรียกใช้จะกำหนดชื่อฟอร์มมาให้ และชื่อฟอร์มนี้จะกำหนดการทำงานดังนี้

'R' แสดงฟอร์มเดิมใหม่อีกครั้งพร้อมทั้งแสดงข้อมูลที่มีในหน่วยความจำนั้นด้วย

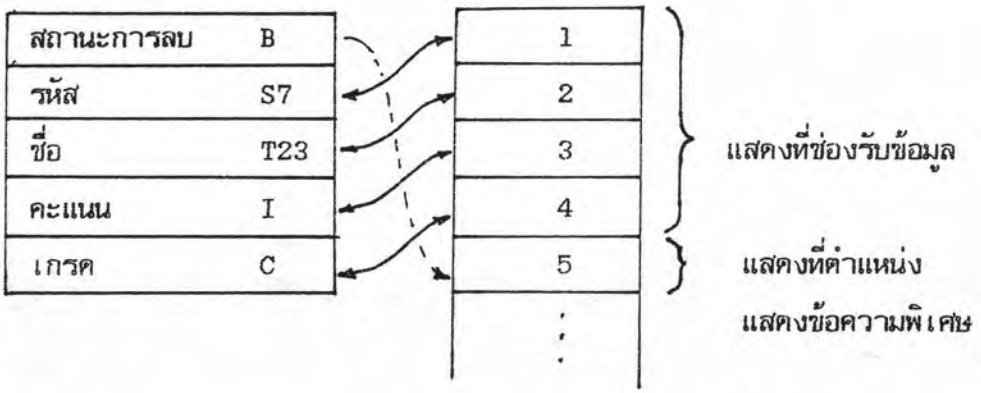
'B' แสดงฟอร์มว่าง

อื่น ๆ อ่านข้อมูลฟอร์มจากแฟ้มข้อมูลตามชื่อที่กำหนด และแสดงฟอร์มนั้น

โครงสร้างข้อมูล มีหลายหน้าที่คือ กำหนดชนิดข้อมูลที่แสดงออกทางช่องรับข้อมูลหรือข้อความพิเศษ ถ้าเป็นอักษรตัวใหญ่ สำหรับช่องรับข้อมูล อักษรตัวเล็กสำหรับข้อความพิเศษ

ชนิดตัวเลขแสดงชิดขวา ถ้าเป็นชนิดตัวอักษร จะทำการตัดช่องไฟหน้าและท้ายข้อมูล และแสดงชิดซ้าย และยังใช้ในการตรวจสอบความถูกต้องของข้อมูลด้วย ดังนั้นฟอร์มและโครงสร้างข้อมูลที่กำหนดมาต้องตรงกัน

แปลงข้อมูลในระเบียนและข้อความพิเศษ ให้อยู่ใน InputData เพื่อง่ายในการทำงาน เช่นระเบียนผลสอบนิสิต ทำการเปลี่ยนข้อมูลทุกตัวให้มีรูปเป็นชนิดข้อความทั้งหมดแล้วเก็บในตัวแปร InputData ตามลำดับดังรูปที่ 4.11



รูปที่ 4.11 แสดงความสัมพันธ์ระหว่างตัวแปร InputData กับ ระเบียน

แสดงฟอร์มว่าง ตามการกำหนดในข้อ 1 โดยเรียกใช้ฟังก์ชัน DisplayScr แสดงฟอร์มว่างบนจอภาพ

แสดงข้อมูลในหน่วยความจำชั่วคราวที่ใช้ทำงานคือ InputData ลงไปในฟอร์มว่างนั้น โดยเรียกใช้ฟังก์ชัน DisplaySys

เรียกใช้ Line Editor รับข้อมูลในช่อง โดย Screen Editor จะจัดเตรียมภาวะแวดล้อมการทำงานให้แก่ Line Editor ตามโครงสร้างข้อมูลที่กำหนด คือ ถ้าเป็นชนิด 'T' ก็จะเปลี่ยนภาวะแวดล้อมการทำงานให้เป็นชนิดแป้นพิมพ์ภาษาไทย ถ้าเป็นชนิดอื่น ๆ ก็เป็นแป้นพิมพ์ภาษาอังกฤษ โดยให้อยู่ในภาวะเติมแทรกทั้งหมด การทำงานในการรับหรือแก้ไขข้อมูลจะอยู่ในความควบคุมของ Line Editor จนกว่าจะมีคำสั่งให้จบการทำงานของ Line Editor ด้วยวิธีใดวิธีหนึ่ง

ทำการตรวจสอบความถูกต้องของข้อมูลภายในช่องรับข้อมูลนั้น โดยเรียกฟังก์ชัน CatchError ซึ่งฟังก์ชันนี้จะทำการตรวจสอบข้อมูลตามชนิดข้อมูลที่กำหนดพร้อมทั้งจัดรูปแบบข้อความในช่องรับข้อมูลนั้นตามกำหนด ถ้าข้อมูลไม่ถูกต้องจะไปเรียก Line Editor ในขั้นที่ 5 ใหม่ โดยยังอยู่ที่ช่องรับข้อมูลเดิม แต่ถ้าถูกต้องและคำสั่งที่จบการทำงานของ Line Editor เป็นคำสั่งควบคุมการเลื่อนตำแหน่งรับข้อมูลของ Screen Editor ก็จะไปเรียกฟังก์ชัน JumpUp, JumpDown หรือเลื่อนไปที่ช่องข้างเคียง เพื่อไปแก้ไขข้อมูลในตัวแปร InputData ถัดไป

เมื่อมีคำสั่งจบการทำงาน Screen Editor จะทำการตรวจสอบความถูกต้องของข้อมูลทั้งหมด โดยเรียก CatchError อีกครั้งหนึ่ง ถ้าไม่ถูกต้องให้ไปทำขั้นที่ 5 อีก ถ้าถูกต้อง ก็ถ่ายข้อมูลจาก InputData ลงสู่ระเบียนที่ผู้เรียกใช้ส่งมาให้

จากขั้นตอนการทำงาน จะแบ่งออกเป็นส่วนประกอบที่ทำให้เป็น Screen Editor ได้ดังนี้

1. EditLine เป็นฟังก์ชันการทำงานในหน้าที่ของ Line Editor
2. CatchError ใช้ตรวจสอบความถูกต้องและจัดรูปแบบ ตามชนิดของข้อมูล
3. DisplayScr แสดงฟอร์มว่าง
4. DisplaySys แสดงส่วนข้อความพิเศษ
5. JumpUp เลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลข้างบนที่ใกล้ที่สุด
6. JumpDown เลื่อนเคอร์เซอร์ไปยังช่องรับข้อมูลข้างล่างที่ใกล้ที่สุด
7. FiletoSep อ่านข้อมูลฟอร์มมาจากแฟ้มข้อมูล

ในหัวข้อนี้ได้กล่าวถึงการทำงานร่วมกันของส่วนประกอบต่าง ๆ เป็นขั้นตอนการทำงานของ Screen Editor และต่อไปจะกล่าวถึง รายละเอียดของส่วนประกอบต่าง ๆ เหล่านี้

4.3.3 รายละเอียดของส่วนประกอบต่าง ๆ

หัวข้อนี้จะกล่าวถึงรายละเอียดการทำงานของส่วนประกอบต่างๆ และการเรียกใช้ส่วนประกอบเหล่านี้จาก Screen Editor เพื่อให้ทราบถึงขั้นตอนการทำงานของ Screen Editor อย่างละเอียด โดยกล่าวถึงส่วนประกอบ EditLine, CatchError, DisplayScr, DisplaySys, JumpUp, JumpDown และ FileToSep เนื่องจากส่วนประกอบเหล่านี้ไม่มีความจำเป็นต้องเรียกใช้จากโปรแกรมภายนอก

```
Procedure EditLine ( C : Char; Var MidPos : Integer;
                     FieldWidth : Integer; Var Next : Integer;
                     Doc : Boolean );
```

หน้าที่ - ใช้รับและแก้ไขข้อมูล 1 บรรทัด โดยทำหน้าที่เป็น Line Editor นั่นเอง

อ้างถึง - หัวข้อ 4.2.4

```
Function CatchError ( Var Line : StrS;
                      Var datatype : Char;
                      InputLineNo : Integer ) : boolean;
```

หน้าที่ - ใช้ตรวจสอบความถูกต้องของข้อมูล ในตัวแปร Line ตามชนิดข้อมูลที่กำหนดใน DataType และกำหนดตำแหน่งช่องเก็บข้อมูล InputLineNo รวมทั้งตัดช่องไฟหน้าหลัง จัดวางตำแหน่งการแสดงข้อมูล ให้ข้อความชิดซ้าย ตัวเลขชิดขวา

การตรวจสอบตามชนิดข้อมูล

I : Integer ข้อมูลเป็นเลขจำนวนเต็มโดยมีค่าอยู่ระหว่าง -32768 ถึง 32767 แสดงโดยไม่มีจุดทศนิยม และชิดขวา

1..9 : Range ข้อมูลเป็นตัวเลขอยู่ในช่วง 1 ถึง 9

R : Real ข้อมูลเป็นตัวเลขชนิดตัวเลขจำนวนจริง แสดงมีจุดทศนิยม 5 ตำแหน่ง และชิดขวา

C : Char ข้อมูลเป็นตัวอักษร A..Z หรือ a..z หรือ 0..9

B : Boolean ข้อมูลเป็นค่าตรรกะ 0, 1, T, F, X, _, Y, N

T : Thai ตัด space หน้าหลัง แสดงชิดซ้าย

S : String ตัด space หน้าหลัง แสดงชิดซ้าย

F : Form date มีเครื่องหมาย /, -, \ คั่นระหว่างวันที่ เดือน ปี

U : Upper case แปลงให้เป็นอักษรตัวใหญ่ แสดงชิดซ้าย

D : Date ข้อมูลมีค่าระหว่าง 1 ถึง 31

M : Month ข้อมูลมีค่าระหว่าง 1 ถึง 12

Y : Year ข้อมูลมีค่าระหว่าง 0 ถึง 99

$: Money ข้อมูลเป็นตัวเลขจำนวนจริง แสดงจุดทศนิยม 2 ตำแหน่ง ชิดขวา

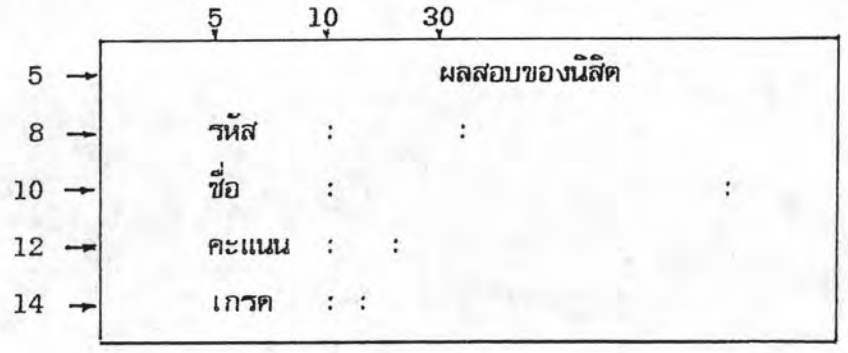
Procedure DisplayScr (Var Screen : Seperate; Var Co : BlankSep;
Var Sys : SysBlank; Var SysWords : SysText);

หน้าที่ - ทำการแสดงหน้าฟอร์มว่าง โดย Screen เป็นตำแหน่ง และข้อความถาวร Co เป็นข้อมูลของช่องรับข้อมูล แสดงตำแหน่ง และความกว้างของช่องรับข้อมูล เช่น

ข้อมูลของ Screen เป็น [30,5] 'ผลสอบของนิสิต' [5,8] 'รหัส'
[5,10] 'ชื่อ' [5,12] 'คะแนน'
[5,14] 'เกรด'

ข้อมูลของ Co เป็น [10,8] (7) [10,10] (23)
[10,12] (3) [10,14] (1)

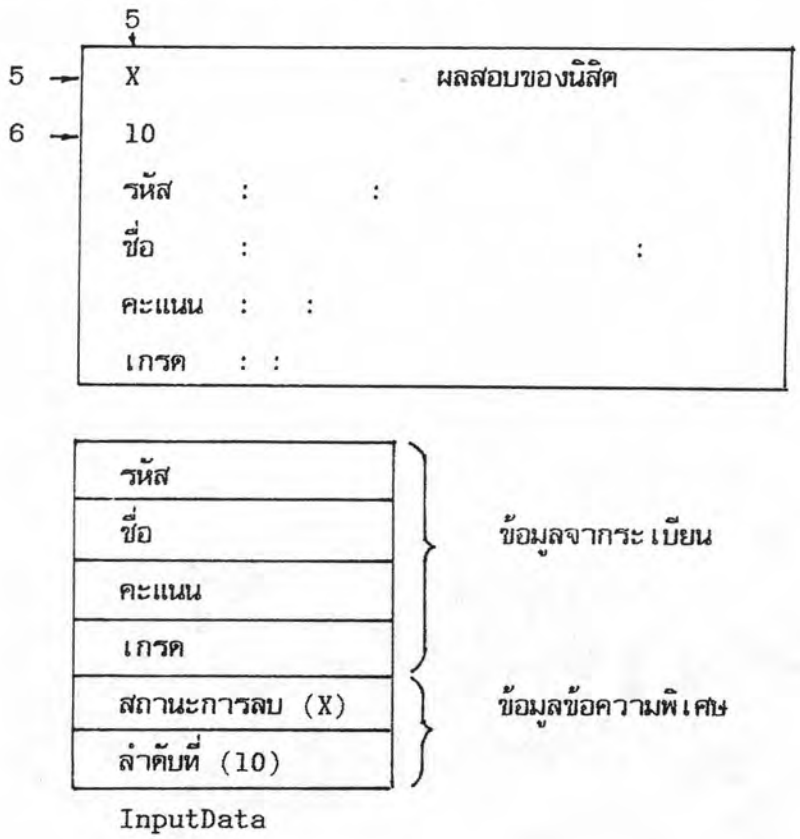
จะได้หน้าจอภาพเป็น



รูปที่ 4.12 แสดงหน้าจอภาพที่สร้างขึ้นโดย DisplayScr

Procedure DisplaySys;

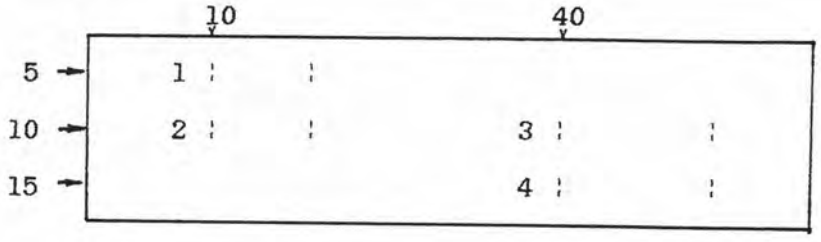
หน้าที่ - ทำการแสดงข้อความพิเศษในตำแหน่งที่กำหนด ในตัวแปร Sys โดยใช้ข้อมูล InputData ที่ได้เตรียมไว้แล้ว ตัวอย่างข้อมูลใน Sys เป็น [5,5], [5,6] และเตรียมข้อมูลให้ InputData ต่อจากข้อมูลในระเบียนเป็น X และ 10 สำหรับระเบียนผลสอบนิสิตจะเป็นดังนี้



รูปที่ 4.13 แสดงฟอร์มที่สร้างด้วย DisplaySys และข้อมูลภายในตัวแปร InputData

Procedure JumpUp;

หน้าที่ - ทำการคำนวนตำแหน่งถัดไปทางด้านบนของช่องเก็บข้อมูลในแนวตั้งที่ใกล้ที่สุด แต่ถ้าอยู่ที่ตำแหน่งบนสุดแล้ว ตำแหน่งถัดไป จะเป็นตำแหน่งล่างสุด เช่น หน้าจอภาพมีช่องรับข้อมูลดังนี้



รูปที่ 4.14 แสดงการวางช่องรับข้อมูล

ข้อมูลในตัวแปร Co จะเป็น [10,5], [10,10], [40,10], [40,15] ตรงกับช่อง 1,2,3 และ 4 ตามลำดับ

ถ้าเริ่มต้นที่ช่อง 1 การเลื่อนขึ้นจะเลื่อนไปตามช่อง 4,3,1,4,3,1 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 2 การเลื่อนขึ้นจะเลื่อนไปตามช่อง 1,4,3,1,4,3 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 3 การเลื่อนขึ้นจะเลื่อนไปตามช่อง 1,4,3,1,4,3 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 4 การเลื่อนขึ้นจะเลื่อนไปตามช่อง 3,1,4,3,1,4 ตามลำดับ

สังเกตว่า ถ้าเริ่มต้นที่ช่อง 2 โดยวิธีเลื่อนขึ้นอย่างเดียวจะไม่สามารถกลับมาที่ช่อง 2 ได้อีก ต้องใช้เลื่อนซ้ายหรือขวาประกอบด้วย

Procedure JumpDown;

หน้าที่ - ทำการคำนวณตำแหน่งถัดไปทางด้านล่างของช่องเก็บข้อมูลในแนวตั้ง ที่ใกล้ที่สุด แต่ถ้าอยู่ที่ตำแหน่งล่างสุดแล้ว ตำแหน่งถัดไปจะเป็นตำแหน่งบนสุด คล้ายกับ procedure JumpUp

ตำแหน่งช่องรับข้อมูลเช่นเดียวกับรูป 4.14

ถ้าเริ่มต้นที่ช่อง 1 การเลื่อนลงจะเลื่อนไปตามช่อง 2,4,1,2,4 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 2 การเลื่อนลงจะเลื่อนไปตามช่อง 4,1,2,4,1 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 3 การเลื่อนลงจะเลื่อนไปตามช่อง 4,1,2,4,1 ตามลำดับ
ถ้าเริ่มต้นที่ช่อง 4 การเลื่อนลงจะเลื่อนไปตามช่อง 1,2,4,1,2 ตามลำดับ

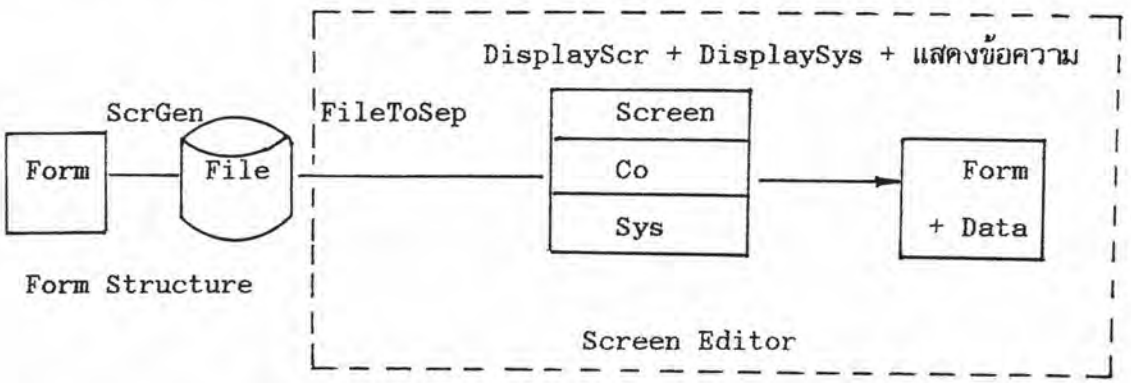
สังเกตว่า ถ้าเริ่มต้นที่ช่อง 3 โดยวิธีเลื่อนลงอย่างเดียวจะไม่สามารถกลับมาที่ช่องที่ 3 ได้อีก ต้องใช้เลื่อนซ้ายหรือขวาประกอบด้วย

Function FileToSep (Var Screen : Seperate;

Var Co : BlankSep;

Var Sys : SysBlank) : Boolean;

หน้าที่ - ทำการอ่านข้อมูลของฟอร์มที่เก็บอยู่ในแฟ้มข้อมูลที่มีโครงสร้างของฟอร์ม ตามที่กำหนดในหัวข้อ 3.2 มาเก็บไว้ในตัวแปร Screen, Co และ Sys โดย Screen เก็บข้อความถาวร Co เก็บตำแหน่งช่องรับข้อมูล และ Sys เก็บตำแหน่งแสดงข้อความพิเศษ การแสดงหน้าฟอร์มต่าง ๆ จึงนำจากตัวแปรเหล่านี้ไปสร้างฟอร์มใหม่ได้



รูปที่ 4.15 แสดงตัวแปรที่ใช้ในการแสดงฟอร์ม

ตัวแปร Screen ประกอบด้วย ตำแหน่งบนจอภาพ, ความยาวของข้อความ, จำนวนคอลัมน์ของข้อความ และข้อความนั้น

ตัวแปร Co ประกอบด้วย ตำแหน่งบนจอภาพ, ความยาวช่องรับข้อมูล

ตัวแปร Sys ประกอบด้วย ตำแหน่งบนจอภาพ

4.3.4 <u>การนำไปใช้งาน</u>

ในหัวข้อนี้จะกล่าวถึงการนำ Screen Editor ไปใช้งาน พร้อมทั้งยกตัวอย่างด้วย สำหรับการใช้ Screen Editor นั้น จะนำ source code medule ไปรวมเข้ากับโปรแกรมของผู้ใช้ โดยถ้าต้องการให้มีฟังก์ชันการทำงานเพิ่มเติมไปอีก ก็ให้เปลี่ยนแปลงส่วน source code นั้นให้เหมาะสม

Function EditInput (Var ScreenName, InputType : Strings) : integer;

หน้าที่ - รับ และแก้ไขข้อมูลของระเบียนในแฟ้มข้อมูล โดยใช้ฟอร์มกำหนดตำแหน่ง

ตัวแปร - ScreenName เป็นชื่อแฟ้มข้อมูลของฟอร์ม

InputType เป็นรหัสโครงสร้างข้อมูลที่กำหนด เพื่อกำหนดรูปแบบของข้อมูลที่แสดง

ผลลัพธ์ - EditInput เป็นรหัสการออกจาก Screen Editor

อ้างถึง - ScrGen, EditLine

รายละเอียด

การกำหนดฟอร์มและโครงสร้างข้อมูล ต้องมีความสัมพันธ์กันเพื่อความถูกต้องในการแสดงบนจอภาพ สำหรับชื่อฟอร์ม จะแบ่งเป็น 3 ชนิดคือ

ScreenName = 'R' แสดงฟอร์มเดิม พร้อมข้อมูลใหม่

'B' แสดงฟอร์มว่าง

อื่น ๆ อ่านข้อมูลฟอร์มจากแฟ้มข้อมูลตามชื่อที่กำหนด

InputType เป็นโครงสร้างของข้อมูลที่กำหนด ชนิดช่องรับข้อมูล และข้อความพิเศษ ประกอบด้วยรหัสแทนชนิดของข้อมูลเรียงต่อกัน จากชนิดช่องรับข้อมูลช่องแรก จนถึงช่องสุดท้าย และตามด้วย ชนิดของข้อความพิเศษจากตำแหน่งแรกถึงตำแหน่งสุดท้าย เช่นเดียวกันตัวอย่างเช่น สำหรับฟอร์มผลการสอบของนิสิต ให้ชื่อว่า 'EXAM.SEP' ดังรูปที่ 4.16

```
>                          ผลสอบของนิสิต
หน้าที่ >
>
        รหัส :       :
        ชื่อ  :                    :
        คะแนน :     :
        เกรด : :
>
        [F2] = บันทึก                    [ESC] = ยกเลิก
```

รูปที่ 4.16 แสดงฟอร์มผลการสอบของนิสิต

เครื่องหมาย : : เป็นช่องรับข้อมูล และ > เป็นตำแหน่งแสดงข้อความพิเศษที่ไม่ปรากฏบนจอภาพ

ชนิดของข้อมูล ในช่องรับข้อมูลเป็นชนิดข้อความ 7 ตัวอักษร (S7) ชื่อเป็นชนิดข้อความไทย 23 ตัว (T23) คะแนนเป็นชนิดตัวเลขจำนวนเต็ม (I) และเกรดเป็นชนิดตัวอักษร (C)

สำหรับชนิดของข้อความพิเศษ ซึ่งมีอยู่ 4 ตำแหน่ง ให้ตำแหน่งบนสุดแสดงสถานะการลบ ชนิดตรรกะ(B) ตำแหน่งที่ 2 แสดงหมายเลขหน้าชนิดตัวเลขจำนวนเต็ม (I) อีก 2 ตำแหน่งที่เหลือให้แสดงข้อความ ดังนั้น

InputType = 'STICBISS'

โดย 4 ตัวอักษรแรกคือ 'STIC' สำหรับช่องรับข้อมูล และ 4 ตัวอักษรถัดมาเป็น 'BISS' สำหรับแสดงข้อความพิเศษ

รหัสต่างๆ ที่ใช้แสดงชนิดของข้อมูล เช่นเดียวกับในฟังก์ชัน Catch Error ในหัวข้อย่อย 4.3.3 มีดังนี้

S	ข้อความอังกฤษ	T	ข้อความไทย
U	ตัวอักษรใหญ่	I	เลขจำนวนเต็ม
R	เลขจำนวนจริง	$	เลขทศนิยม 2 ตำแหน่ง
1.9	เลขตรวจช่วง	C	ตัวอักษร 1 ตัว
B	ค่าตรรกะ	F	แบบวันที่

รหัสเหล่านี้ใช้แสดงชนิดข้อมูล กำหนดรูปแบบของข้อมูล และใช้ในการตรวจรูปแบบด้วย เช่น รหัส U ตัวอักษรใหญ่ เมื่อรับข้อความในช่องรับข้อมูลแล้วจะเปลี่ยนเป็นตัวอักษรใหญ่ทั้งหมด เพื่อแสดงข้อความนั้น

สำหรับรหัสการออกจาก EditInput นั้น จะเป็นรหัสเดียวกับที่ออกจาก Line Editor ยกเว้นรหัสควบคุมภายในของ Editinput

รหัสควบคุมภายในของ Edit Input มีดังนี้

-1 เลื่อนไปช่องถัดไปทางซ้าย (เคอร์เซอร์เลื่อนเกินซ้ายสุดของช่อง)

13,9,1 เลื่อนไปช่องถัดไปทางขวา (กดแป้น [Enter], [Tab] หรือเคอร์เซอร์เกินขวาสุดของช่อง)

-2 เลื่อนขึ้น (กดแป้น [UArr])

2 เลื่อนลง (กดแป้น [DArr])

อื่น ๆ ออกจาก EditInput โดยส่งรหัสนี้ออกมาด้วย (ปุ่มควบคุมอื่น ๆ ที่ให้รหัสออกจาก Line Editor) ดูรหัสที่ออกจาก Line Editor ในหัวข้อย่อย 4.2.4

ตัวอย่างขั้นตอนการเรียกใช้ EditInput

```
{$I TYPHEAD.EDT}
{$I VARHEAD.EDT}
{$I GEDITLN.EDT}
{$I GEDITINP.EDT}
 :
Var Command : integer;
Begin
       :
       InitBuffer;                { เตรียมข้อมูลที่จะแสดง }
       Command = EditInput ('EXAM.SEP', 'STICBISS');
       :
End.
```

การเตรียมข้อมูลที่จะแสดง เป็นการจัดข้อมูลลงในตัวแปร InputData ตามตำแหน่งของข้อความนั้น เช่น สำหรับนิสิตรหัส '1234567' ชื่อ 'นายสมชาย' คะแนน 90 เกรด A เป็นหน้าแสดงที่ 10 มีเครื่องหมายการลบ X และข้อความอื่น ๆ อีก 2 ข้อความ จะเป็นดังรูปที่ 4.17

ข้อความใน INPUTDATA จะตรงกับโครงสร้างข้อมูล 'STICBISS' ก่อนเรียกใช้จึงต้องเตรียมข้อมูลให้เรียบร้อยก่อน และต้องให้สัมพันธ์กับฟอร์มและโครงสร้างข้อมูล

การแสดงข้อความพิเศษ จะมีการกำหนดจำนวนตำแหน่งที่จะแสดงได้ โดยเฉพาะเมื่อเป็นชนิดตัวเลขจะมีการแสดงชิดขวาเป็นกัน โดยมีการกำหนดตำแหน่งที่แสดงได้ดังนี้

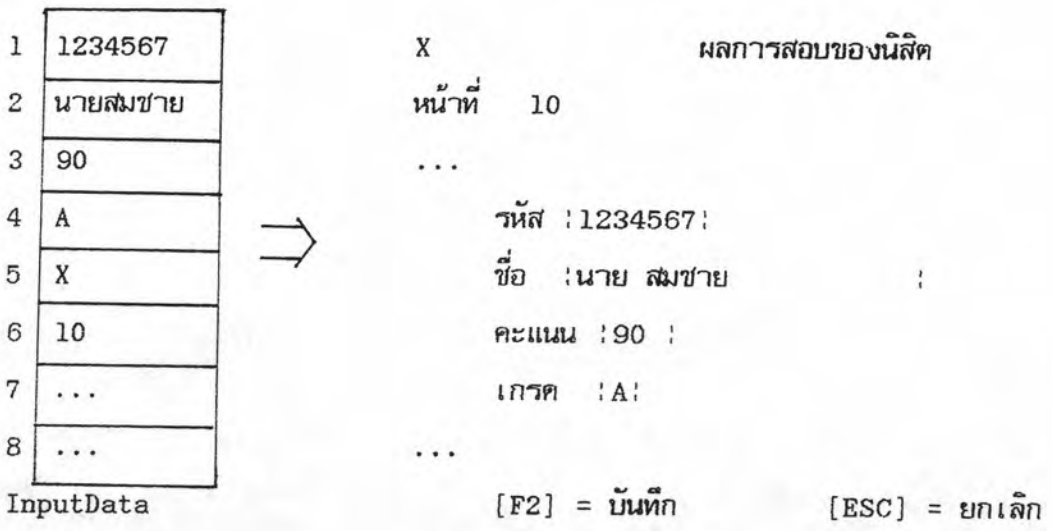
$, R ใช้ 11 ตำแหน่ง

I ใช้ 5 ตำแหน่ง

C, B ใช้ 1 ตำแหน่ง

1..9, D, M, Y ใช้ 2 ตำแหน่ง

F,T,S ไม่จำกัดตำแหน่ง เนื่องจากมีแสดงชิดซ้าย ตรงกับตำแหน่งแสดงพอดี



รูปที่ 4.17 แสดงผลที่ได้จากการเรียกใช้ EditInput

4.4 สรุป

ในบทนี้ได้กล่าวถึงเครื่องมือทางซอฟแวร์ส่วนใหญ่ๆ 2 ส่วนคือ Line Editor และ Screen Editor โดยกล่าวถึงหน้าที่อย่างละเอียด ส่วนประกอบ และการทำงานของโปรแกรม โดยอธิบายเป็นขั้นตอนการทำงาน รายละเอียดของส่วนประกอบต่างๆ และการนำเครื่องมือนี้ไปใช้งาน เพื่อให้เข้าใจการทำงานของแต่ละส่วน ซึ่งทั้ง 2 ส่วนนี้เป็นส่วนประกอบหลักของบรรณการแฟ้มข้อมูล ทำให้เข้าใจการทำงานของบรรณการแฟ้มข้อมูลได้อย่างละเอียด นอกจากนี้ เครื่องมือทางซอฟต์แวร์เหล่านี้ยังสามารถนำไปใช้ประกอบกับโปรแกรมอื่นๆ ได้อีกที่

ต้องการหน้าที่การทำงานที่คล้ายกัน โดยที่รายละเอียดของโปรแกรมที่เขียนขึ้นด้วยภาษาปาสคาล สำหรับ เทอร์โบปาสคาล บนเครื่องไมโครคอมพิวเตอร์ ไอบีเอ็ม พีซี มิได้รวมอยู่ในวิทยานิพนธ์ฉบับนี้ แต่ได้แยกเล่มไว้สำหรับห้องปฏิบัติการวิจัยระบบเชิงเลข ดู [22]