บรรณานุกรม

Angermeyer, J. and K. Jaeger, MS-DOS Developer's Guide, p.93-142 Howard W. Sam & Co., 1986.

Btrieve Version 3.0 : User's Guide, Softcraft, Inc., 1984.

Claybrook, B.G., File Management Techniques, p.119-124, John Wisley & Sons, 1983.

Duncan, R., ADVANCED MSDOS Version 1.1-3.2, p.10-181, Microsoft Press, Washington, 1986.

Gates, W.H., jr., MS-DOS (Version 1.1-3.2) : Technical Reference Encyclopidia, p.479-840, Microsoft Press, Washington, 1986.

Michael, J.F. and B. Zoellick, File Structures : A Conceptual Toolkit, p.249-357, Addison-Wesly, 1987.

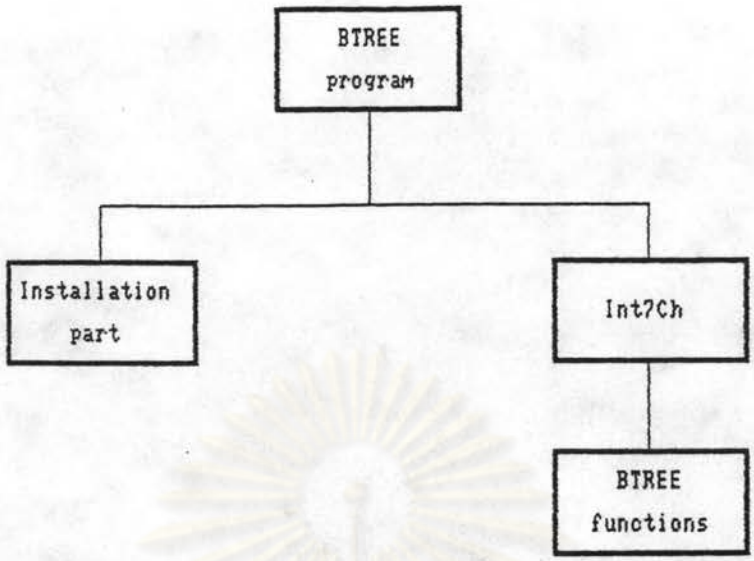
Microsoft C Compiler : User's Guide, Microsoft Corporation.

Microsoft FORTRAN Compiler : User's Guide, Microsoft Corporation.

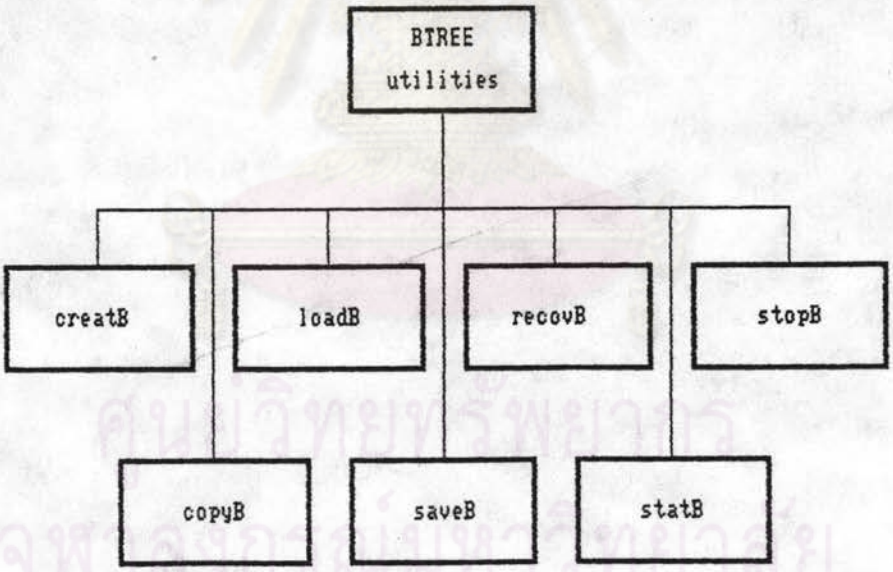
Microsoft Pascal Compiler : User's Guide, Microsoft Corporation.

ภาคผนวก ก

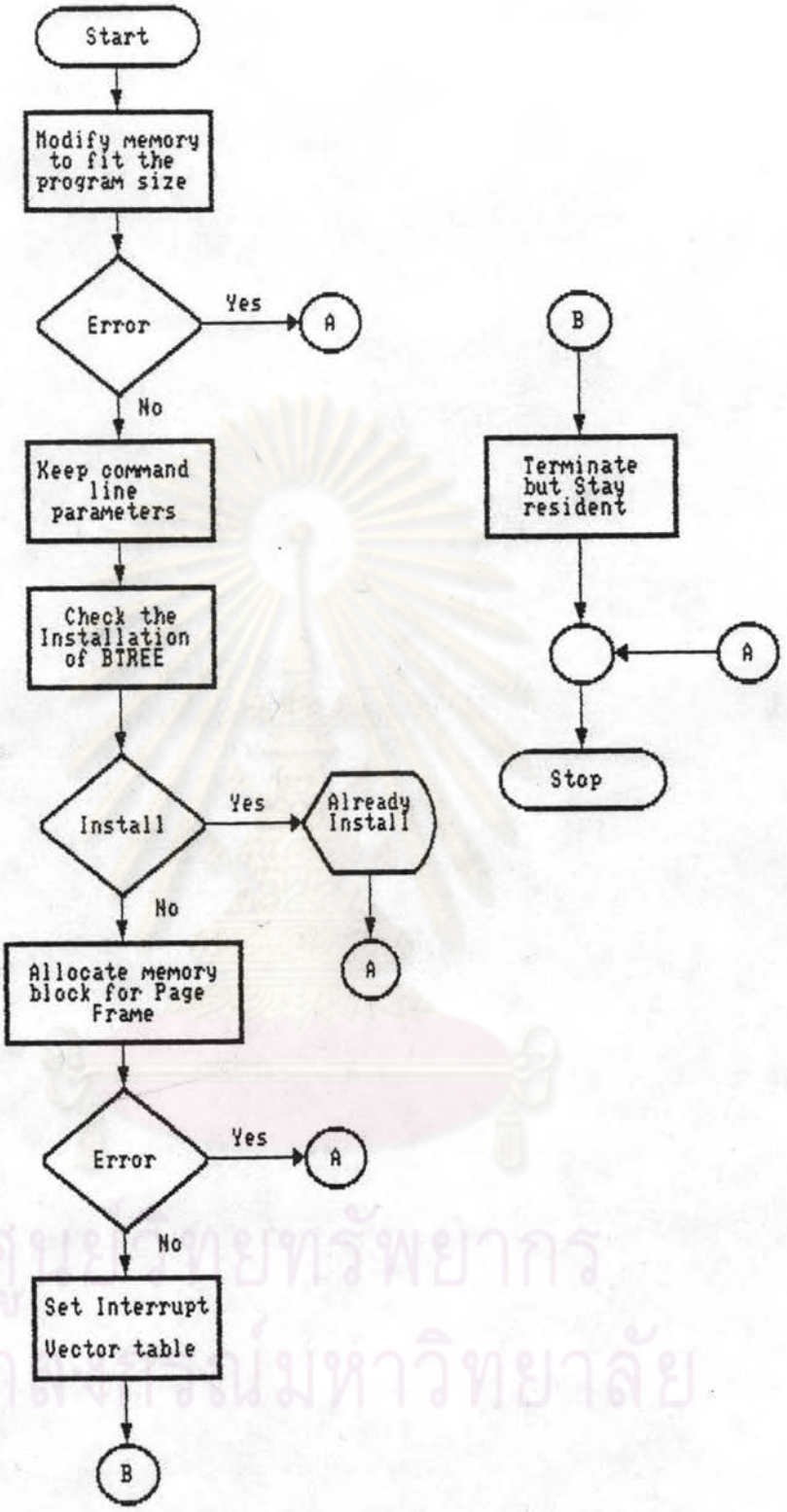
ผังงานของโปรแกรม



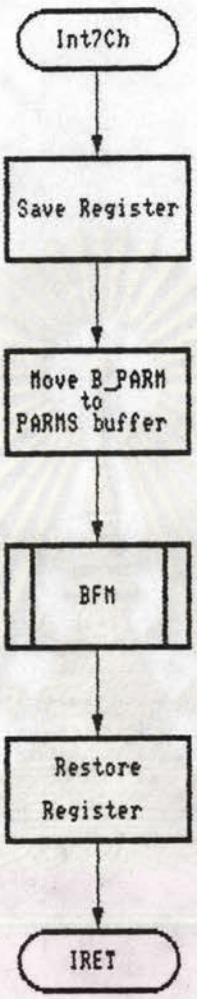
ภาพที่ ก.1 ผังการทำงานของโปรแกรม BTREE



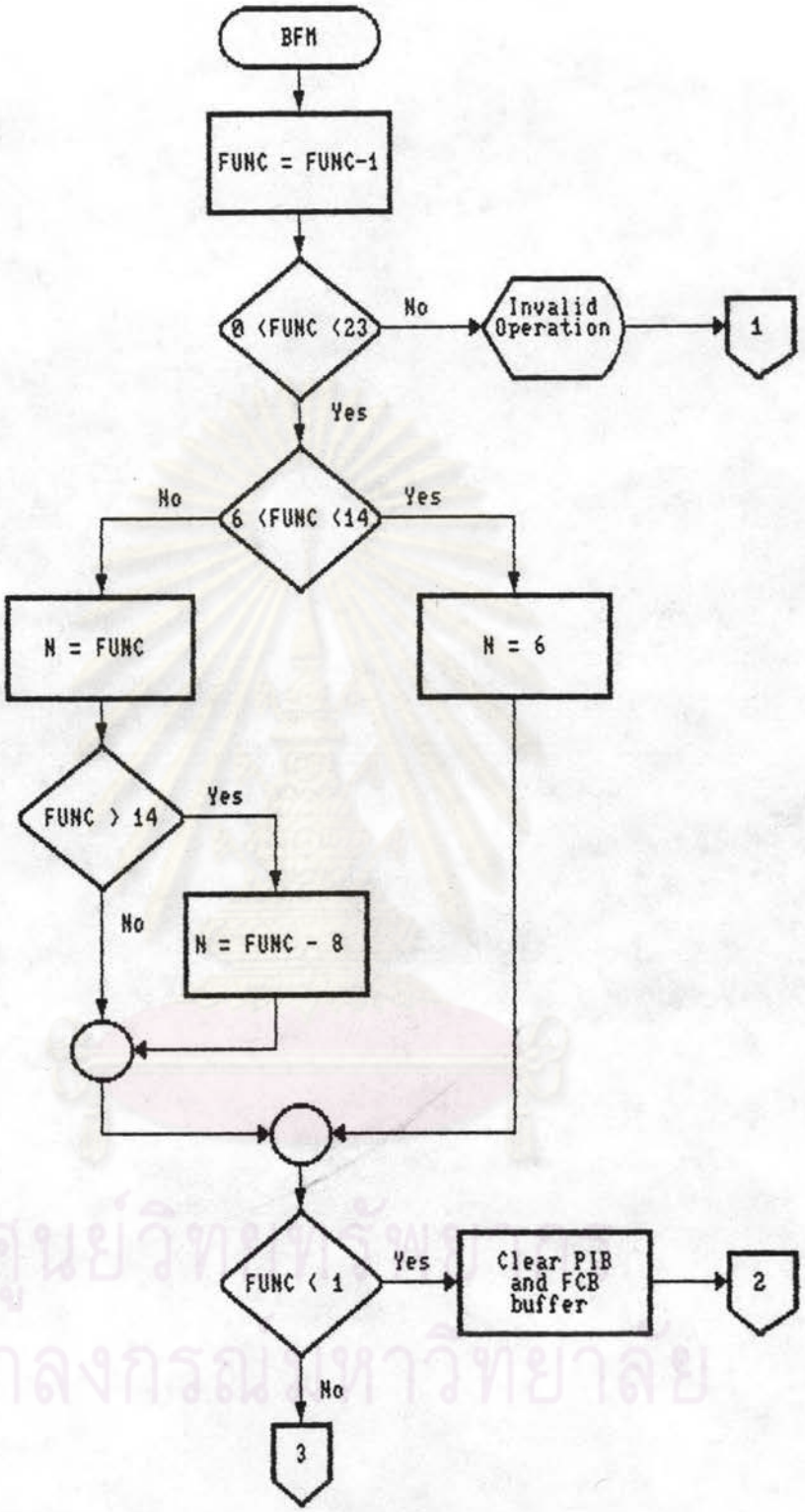
ภาพที่ ก.2 ผังการทำงานของโปรแกรมอรรถประโยชน์สำหรับโปรแกรม BTREE



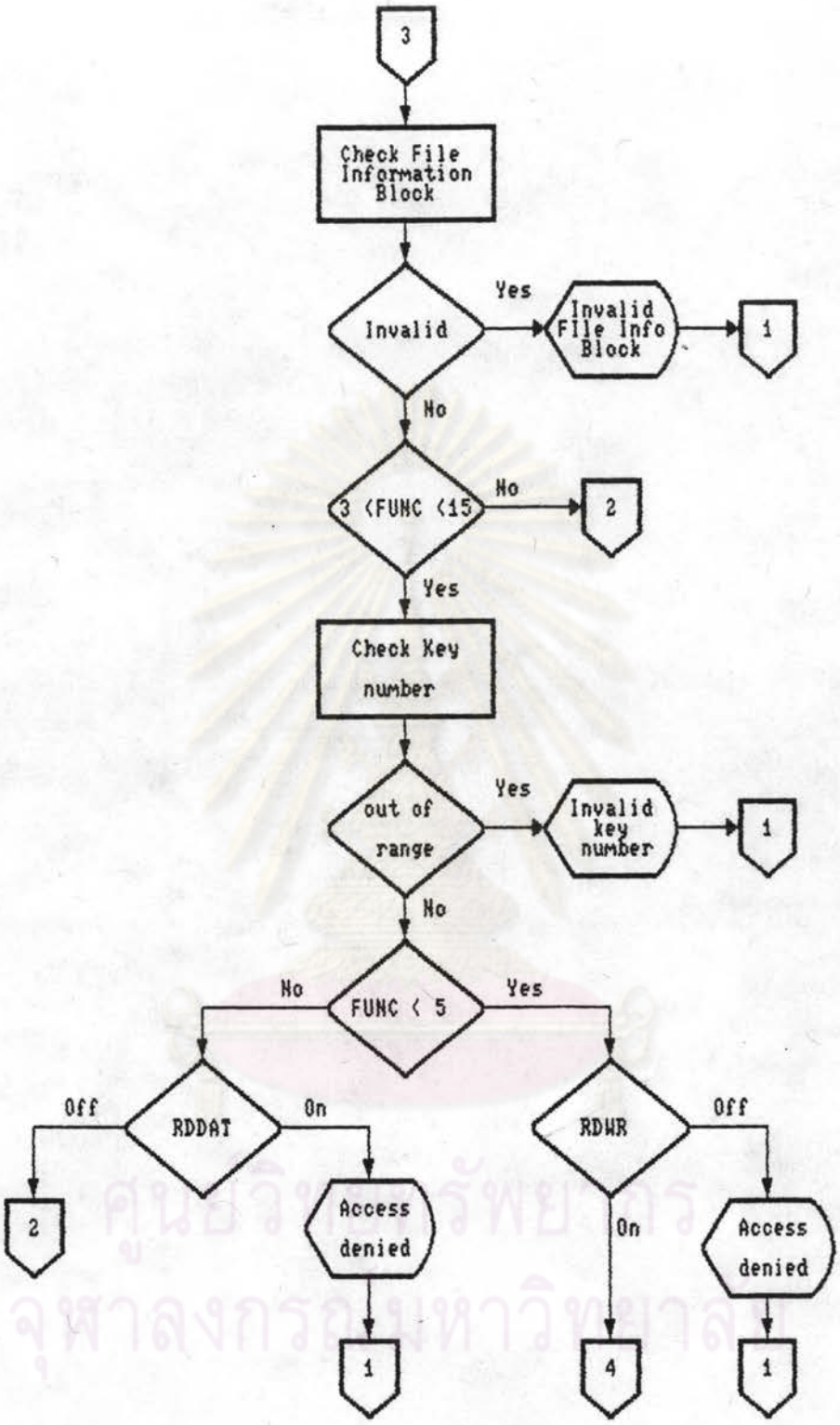
ภาพที่ ก.3 ผังงานของโปรแกรม BTREE



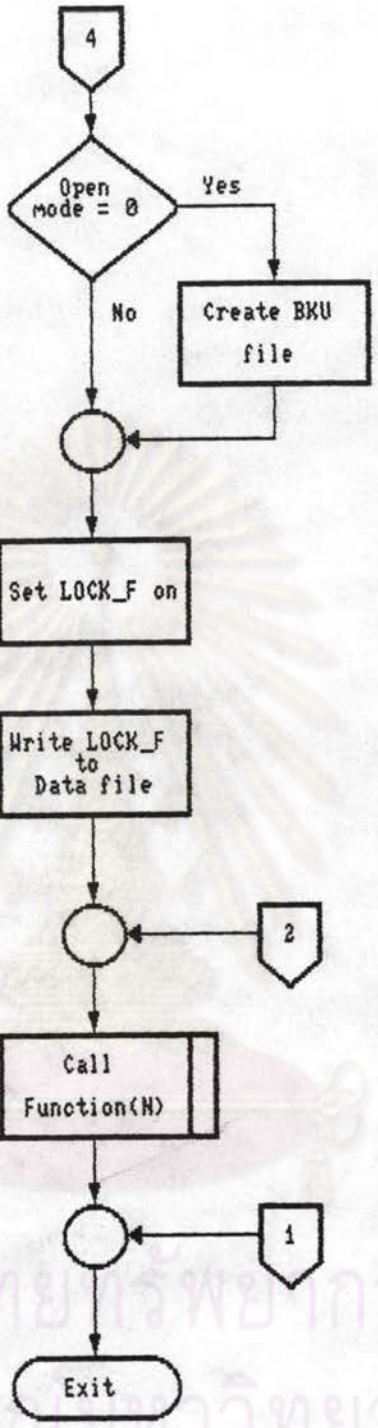
ภาพที่ ก.4 ผังงานของฟังก์ชันเริ่มต้นการทำงานเมื่อถูกเรียกใช้ (Int7Ch)



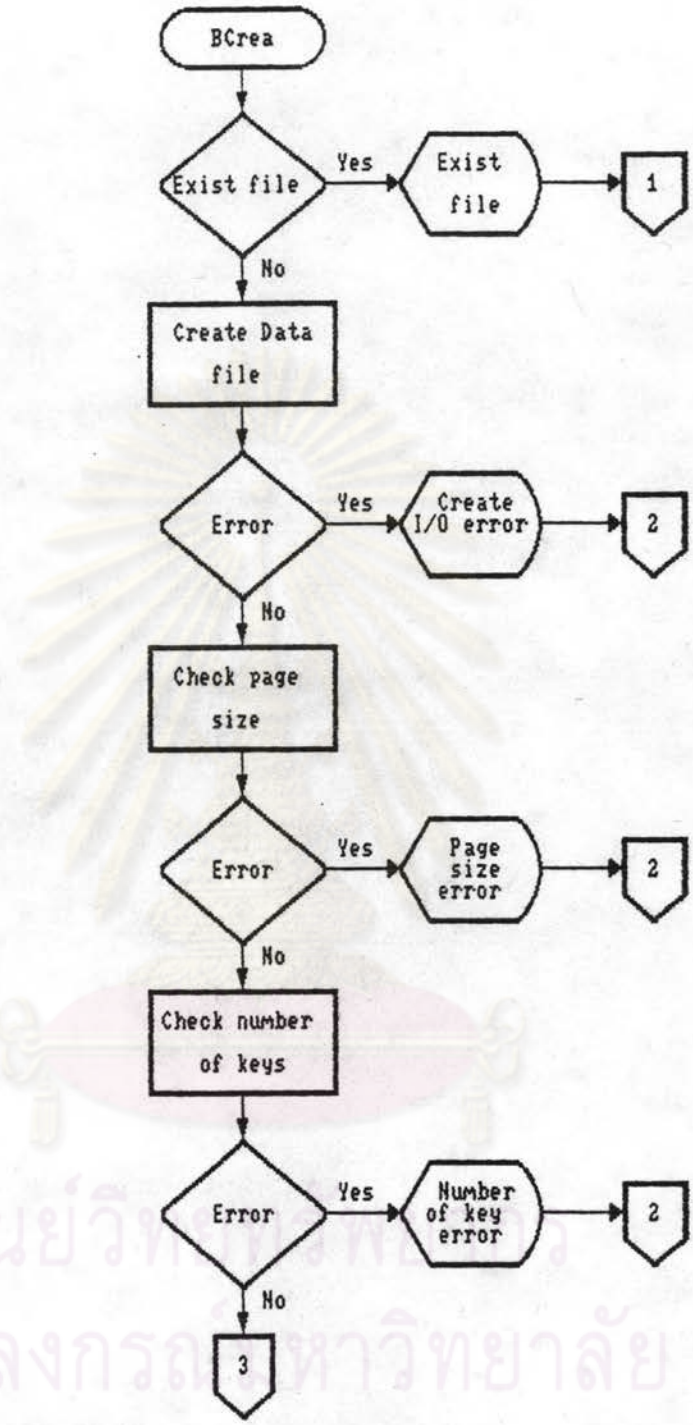
ภาพที่ ก.5 ผังงานของฟังก์ชัน BFM



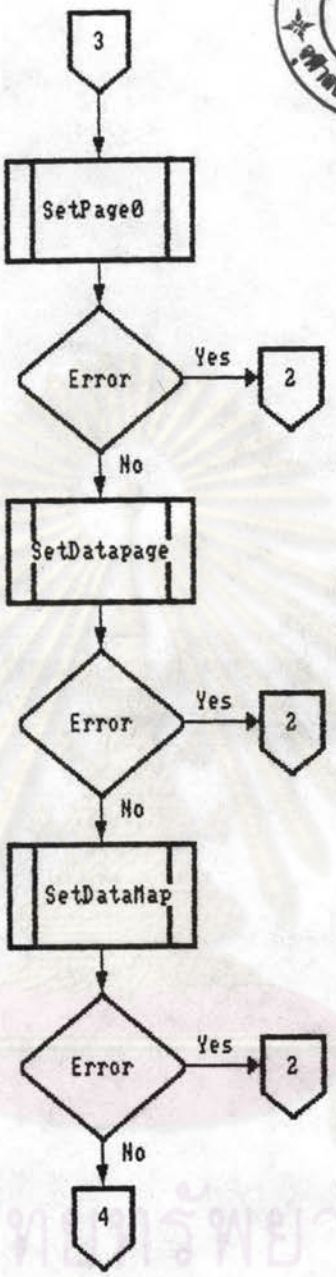
ภาพที่ ก.5 ผังงานของฟังก์ชัน BFM (ต่อ)



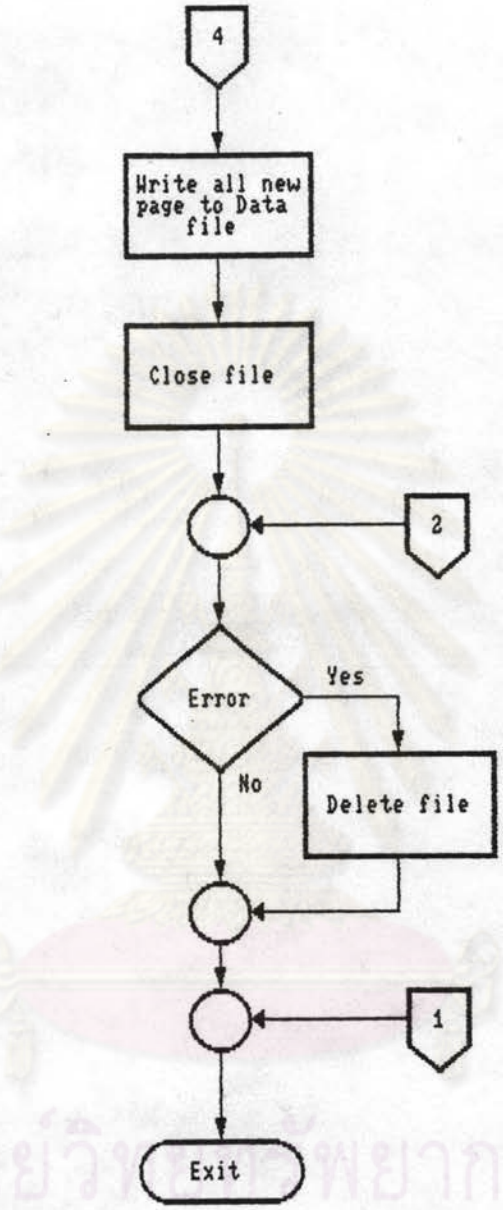
ภาพที่ ก.5 ผังงานของฟังก์ชัน BFM (ต่อ)



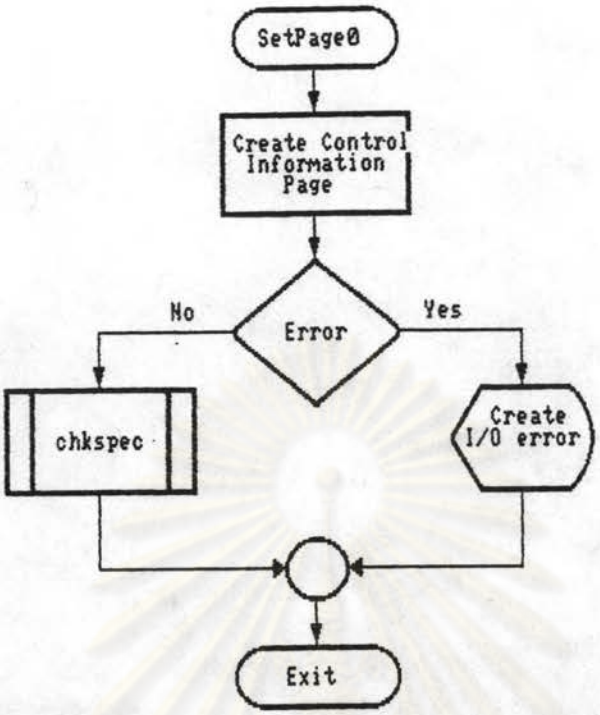
ภาพที่ ก.6 ผังงานของฟังก์ชัน BCrea



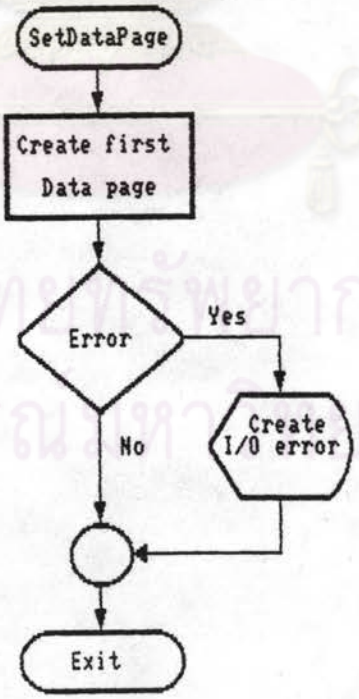
ภาพที่ ก.6 ผังงานของฟังก์ชัน BCrea (ต่อ)



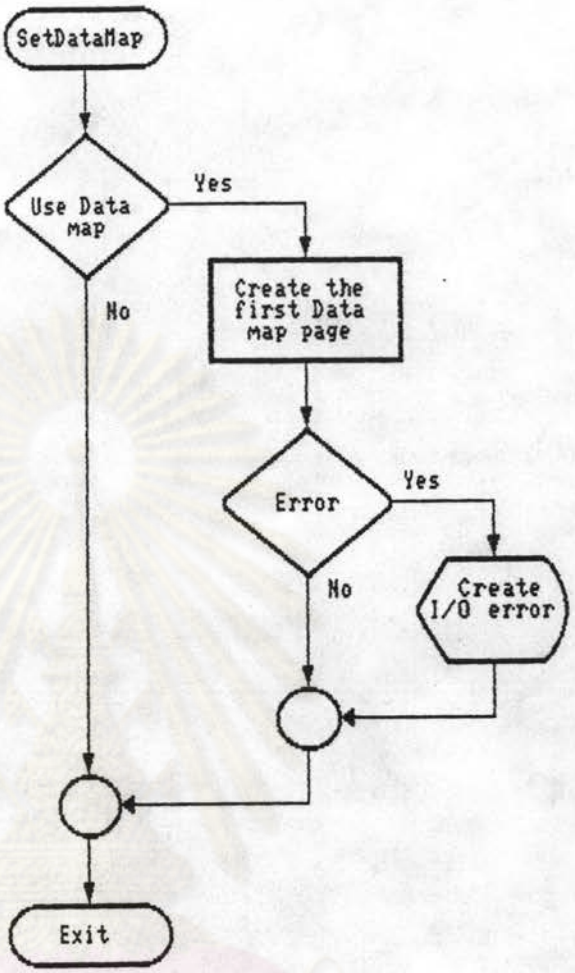
ภาพที่ ก.6 ผังงานของฟังก์ชัน BCrea (ต่อ)



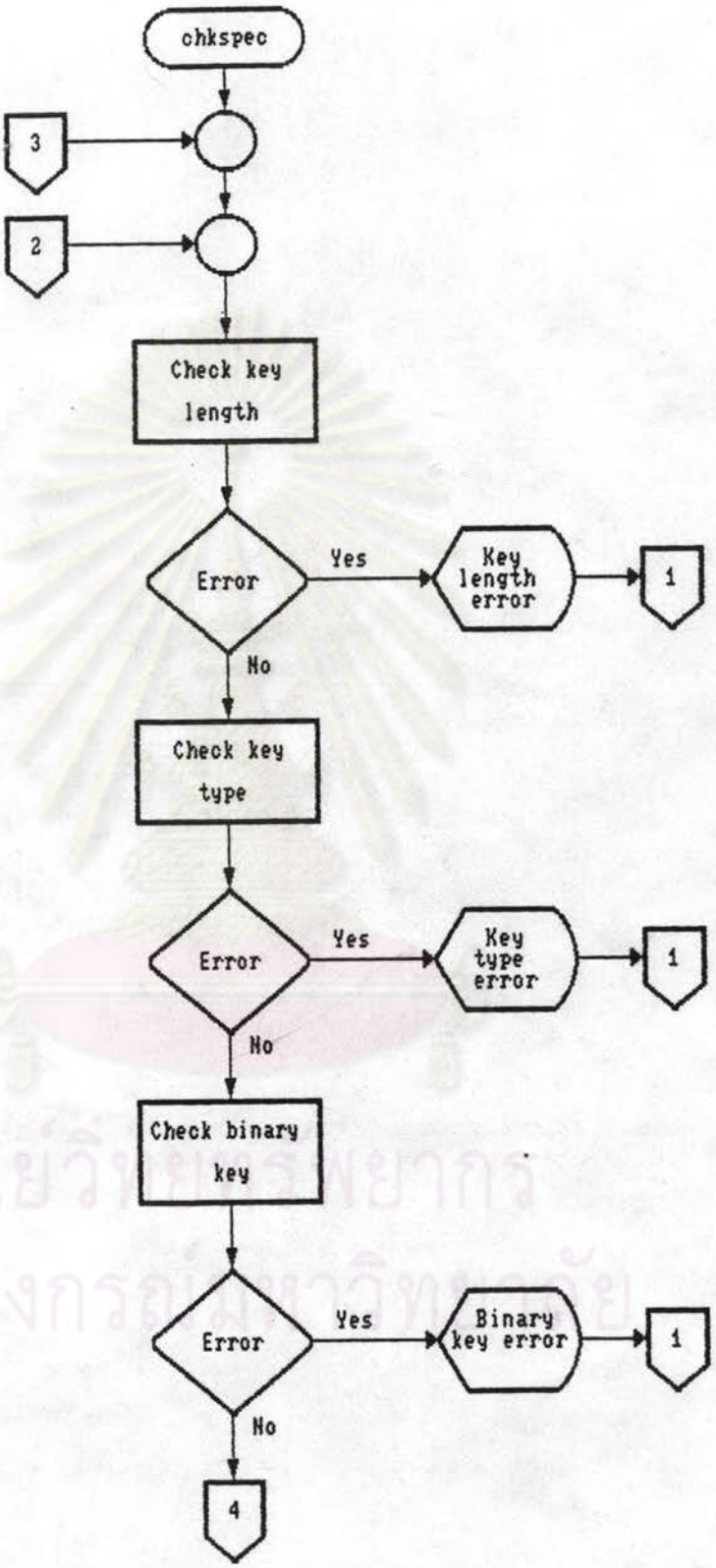
ภาพที่ ก.7 ผังงานของฟังก์ชัน SetPage0



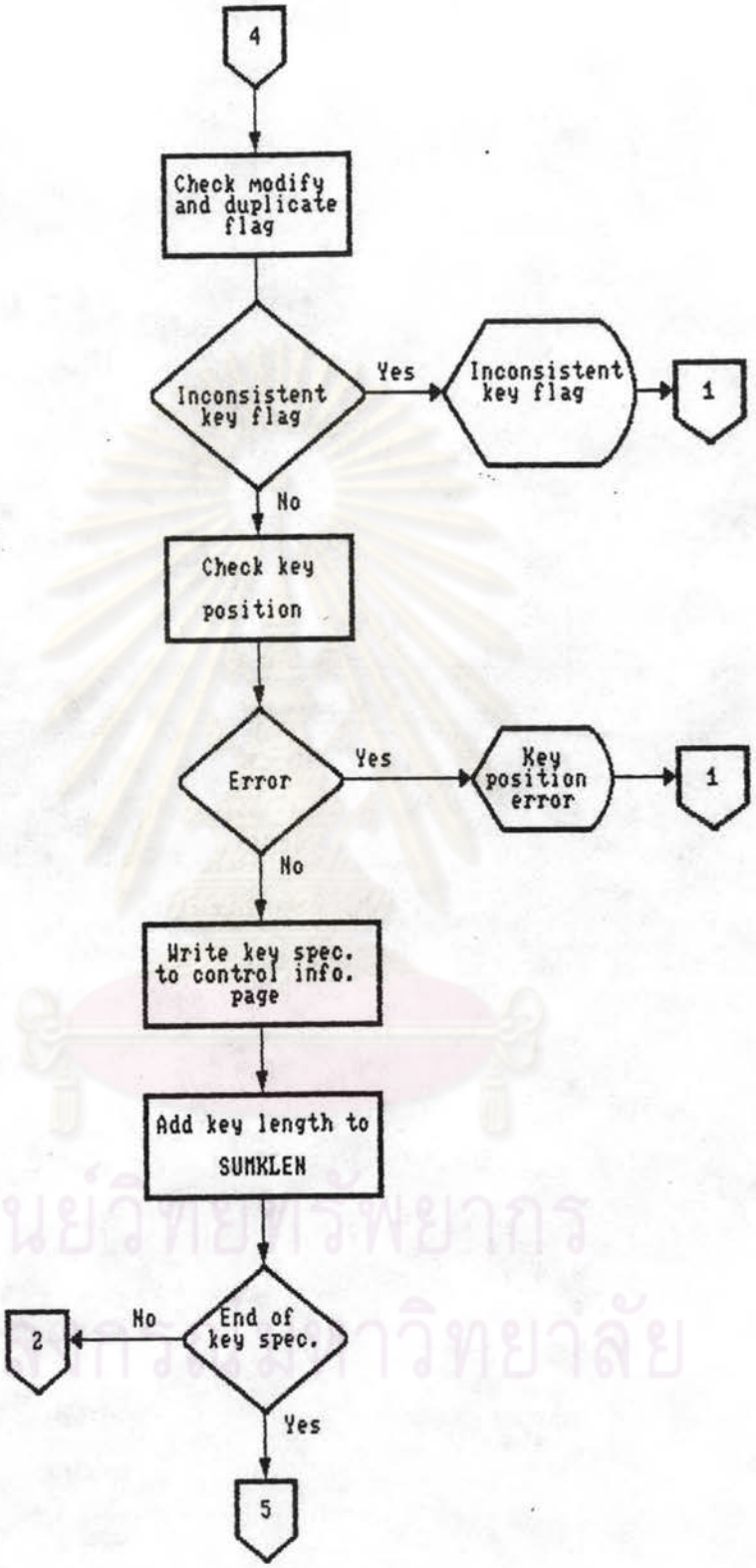
ภาพที่ ก.8 ผังงานของฟังก์ชัน SetDataPage



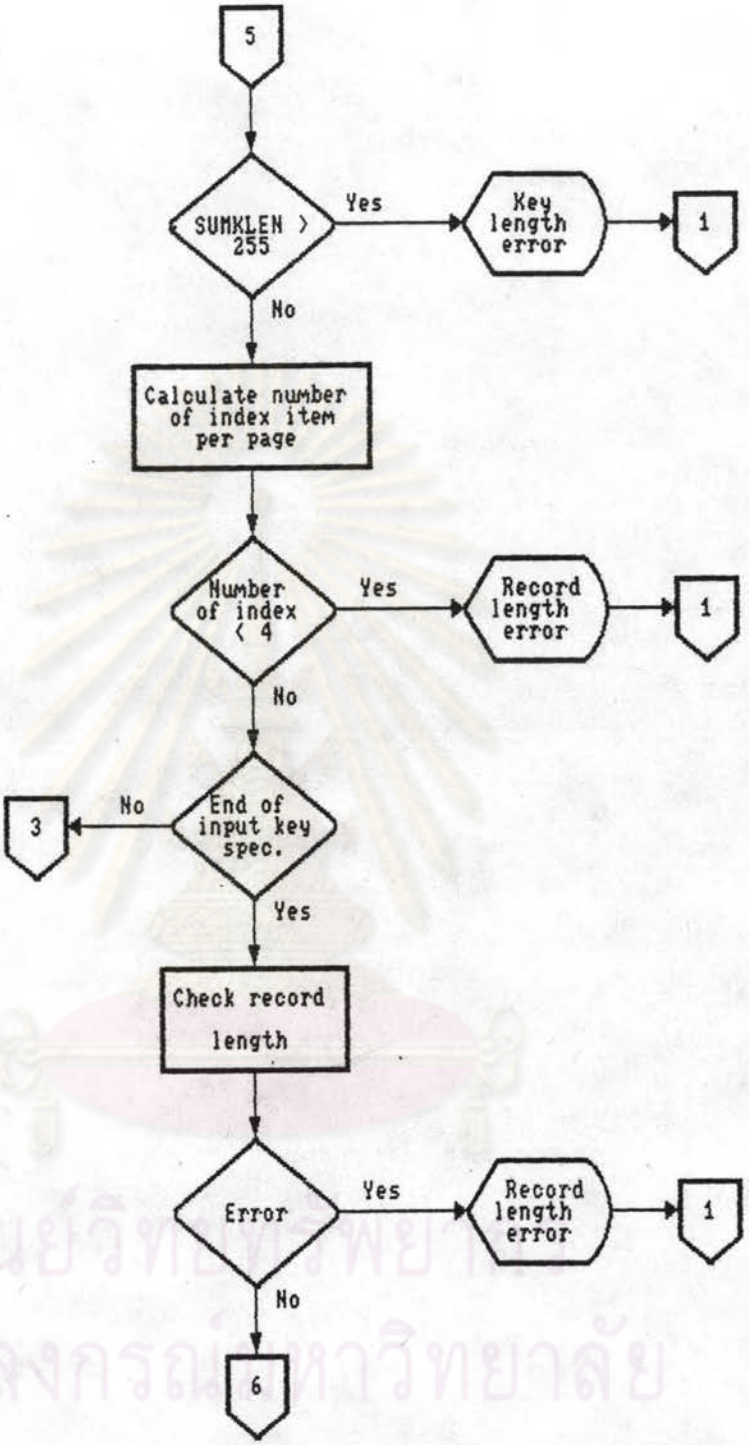
ภาพที่ ก.9 ผังงานของฟังก์ชัน SetDataMap



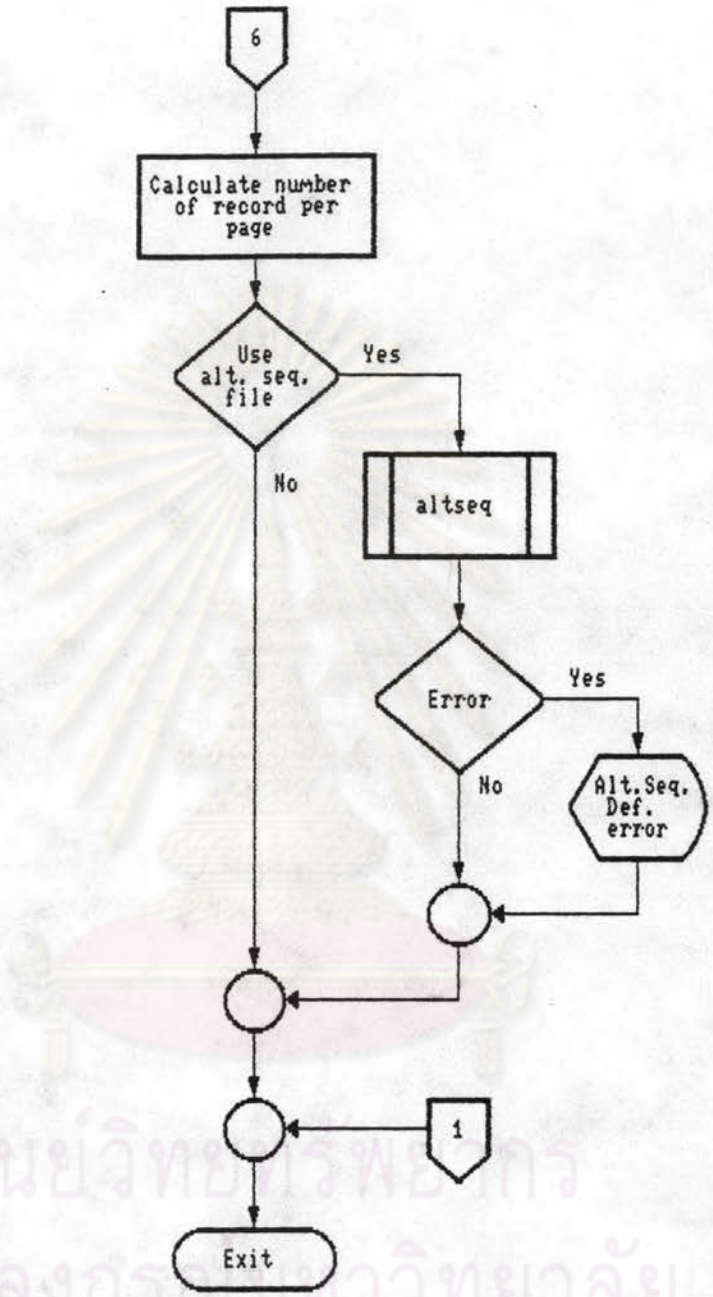
ภาพที่ ก.10 ผังงานของฟังก์ชัน chkspec



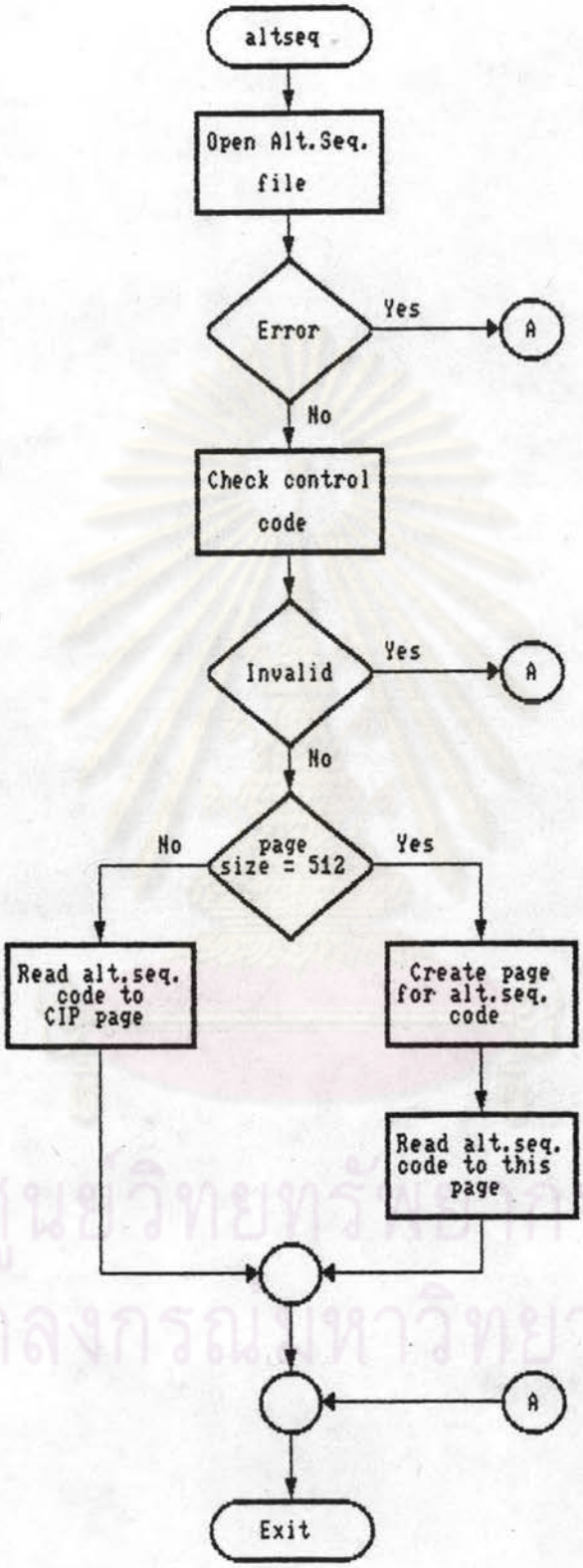
ภาพที่ ก.10 ผังงานของฟังก์ชัน chkspec (ต่อ)



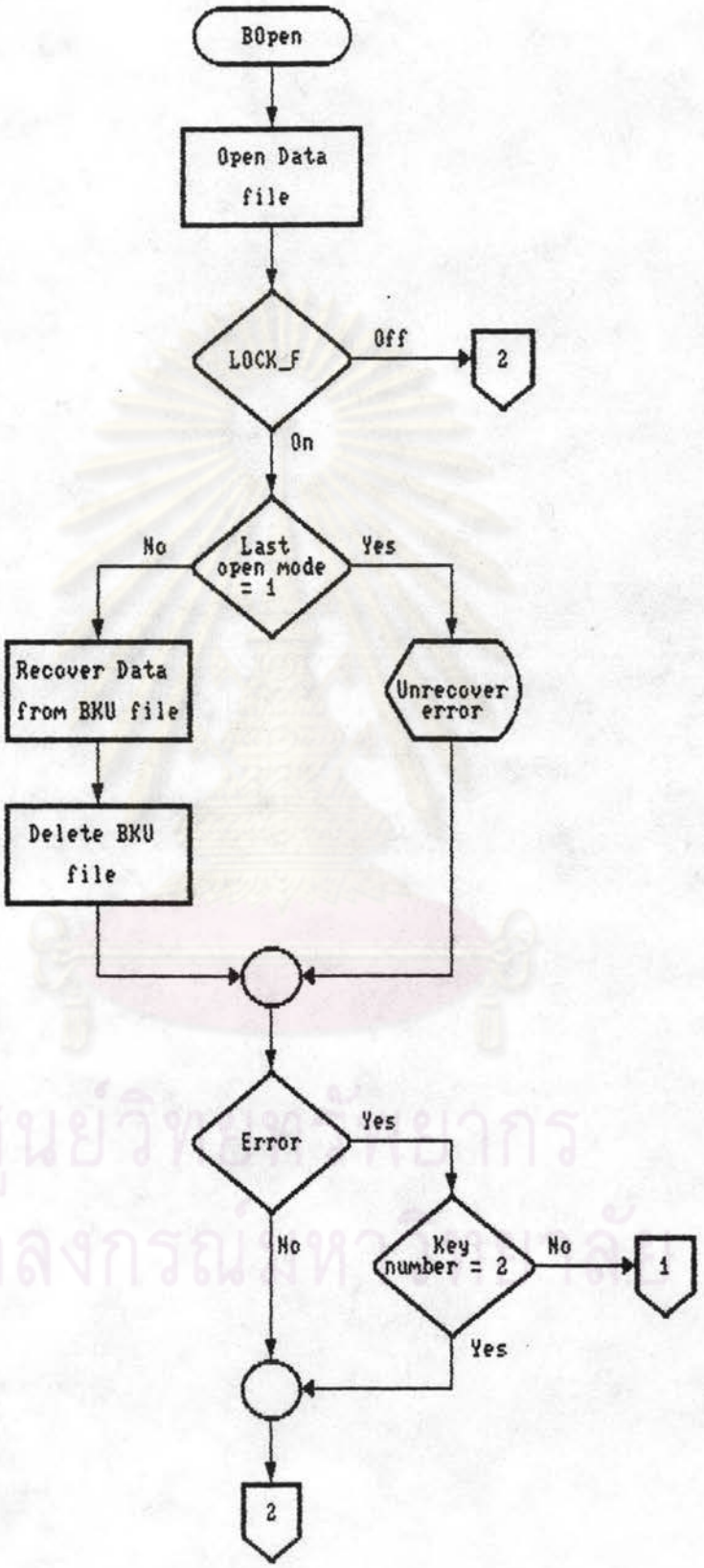
ภาพที่ ก.10 ผังงานของฟังก์ชัน chkspec (ต่อ)



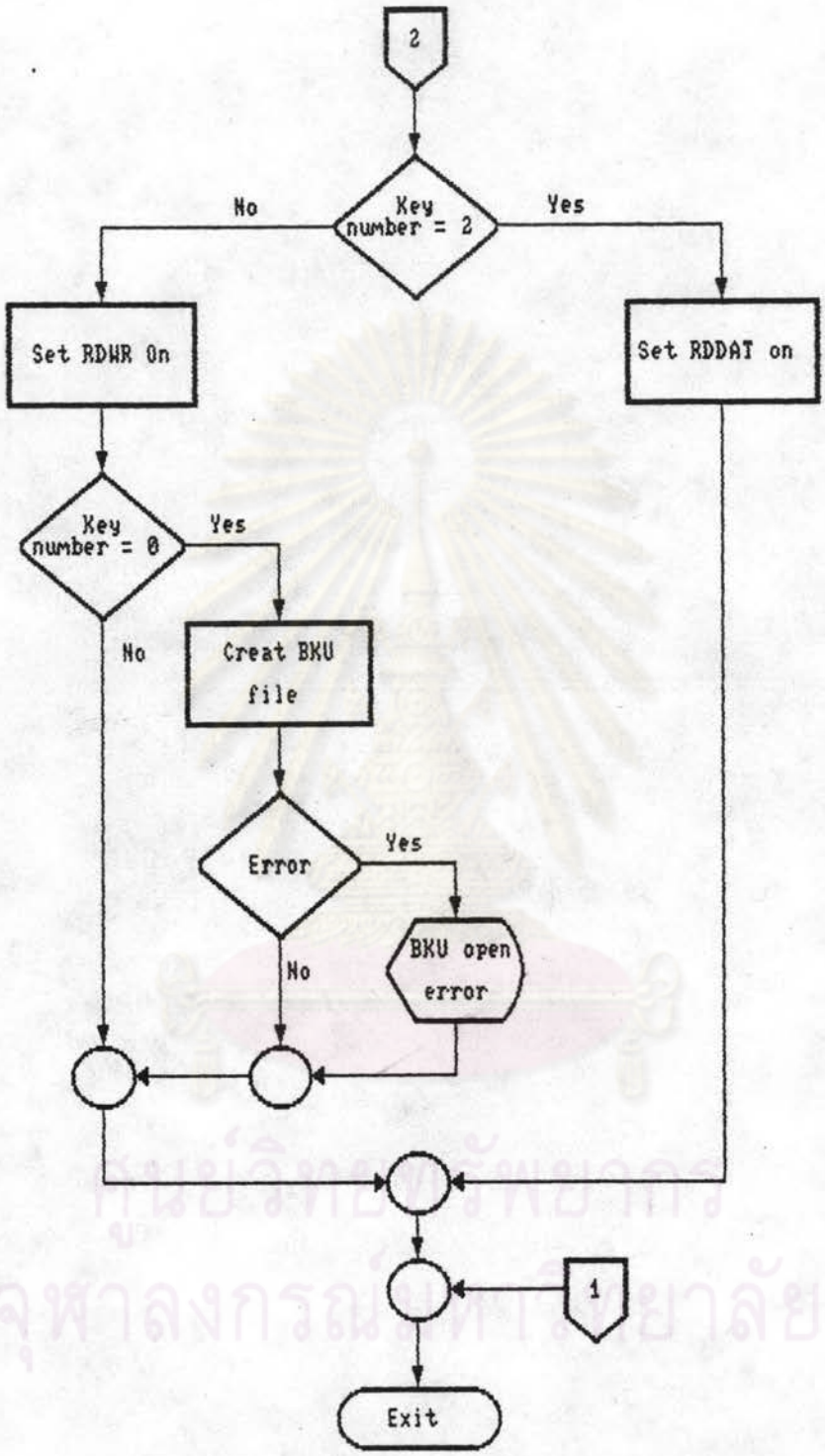
ภาพที่ ก.10 ผังงานของฟังก์ชัน chkspec (ต่อ)



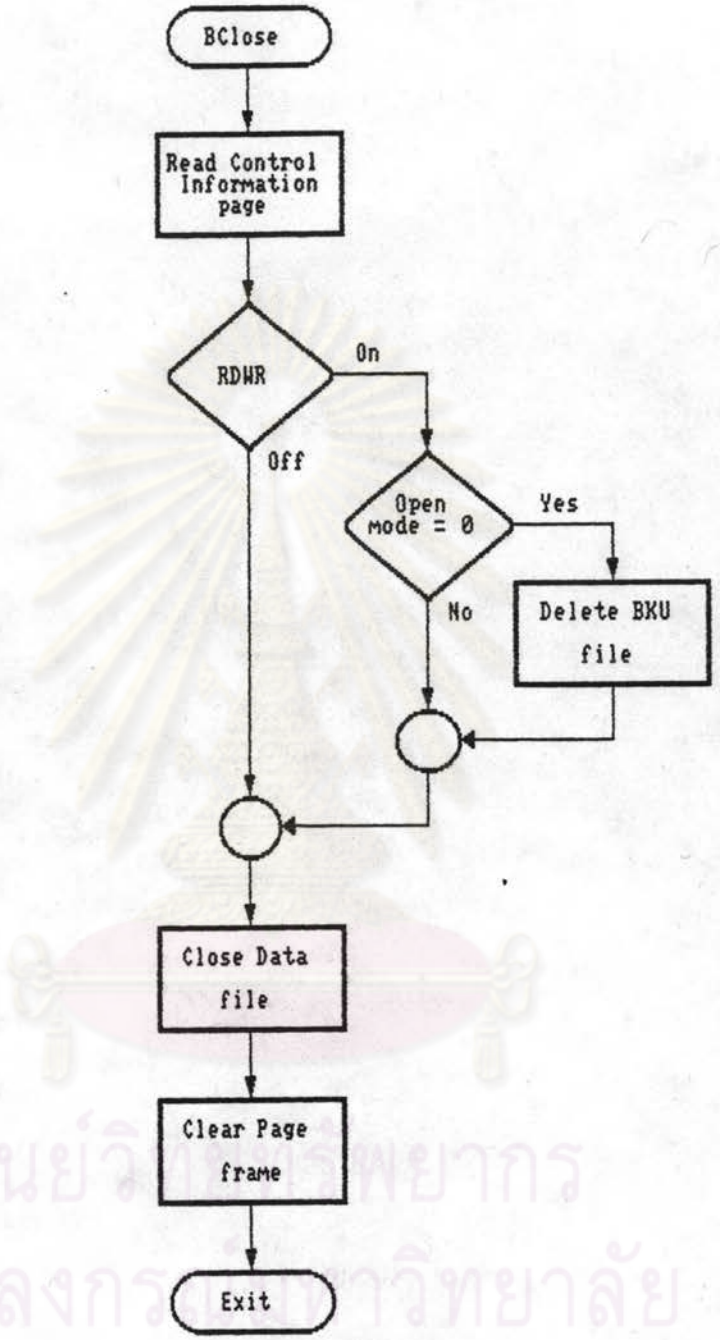
ภาพที่ ก.11 ผังงานของฟังก์ชัน altseq



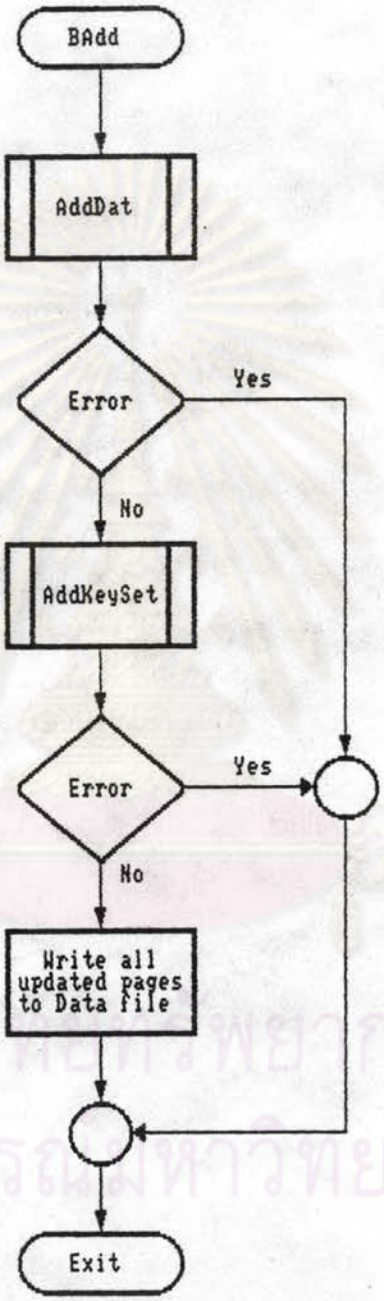
ภาพที่ ก.12 ผังงานของฟังก์ชัน BOpen



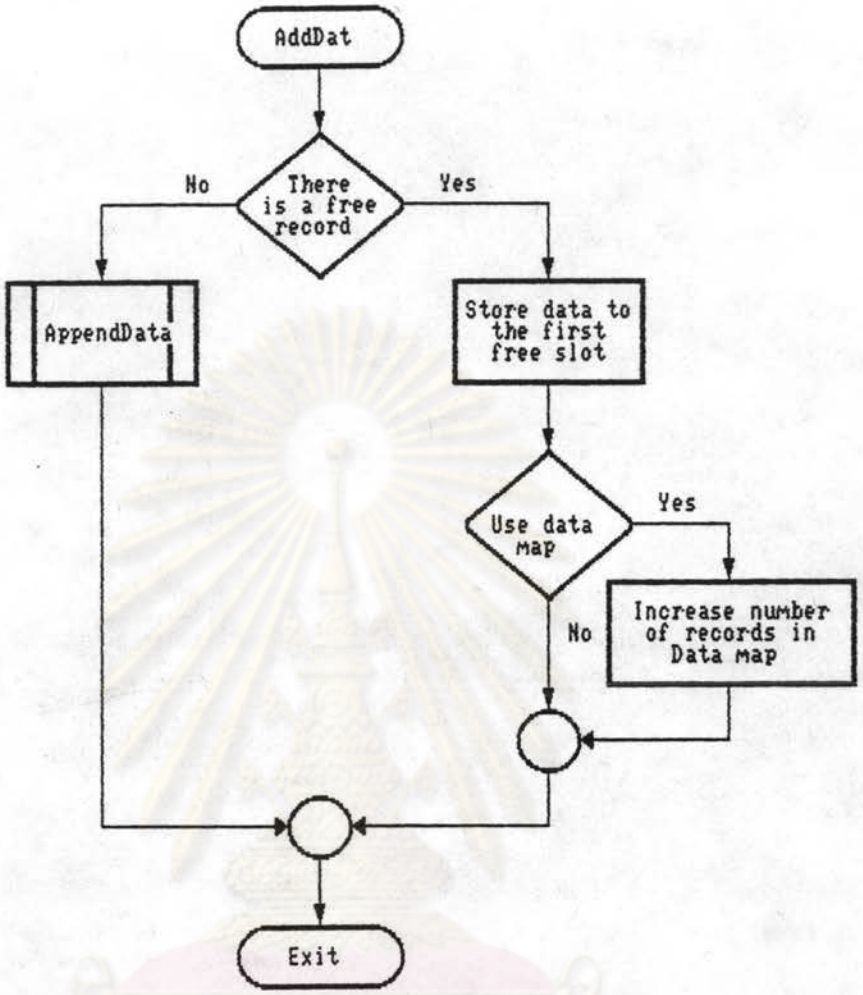
ภาพที่ ก.12 ผังงานของฟังก์ชัน BOpen (ต่อ)



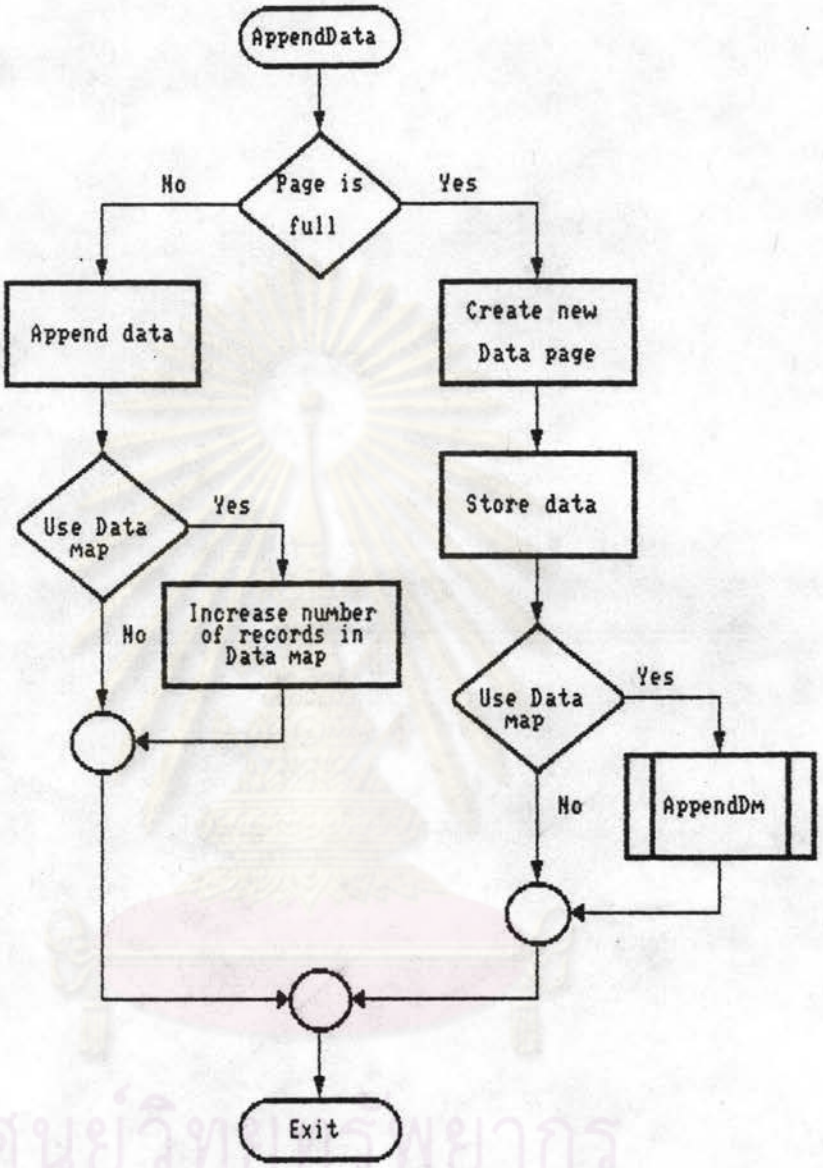
ภาพที่ ก.13 ผังงานของฟังก์ชัน BClose



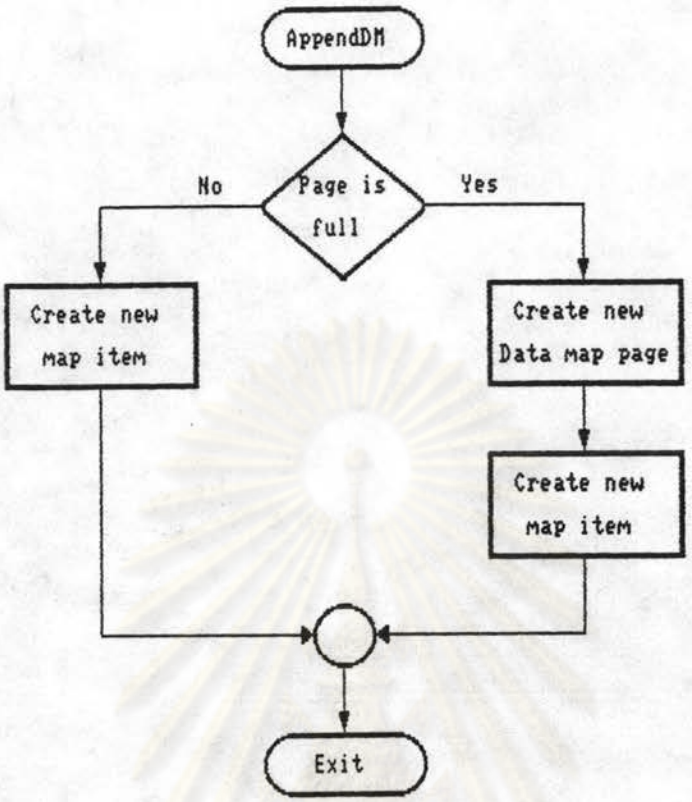
ภาพที่ ก.14 ผังงานของฟังก์ชัน BAdd



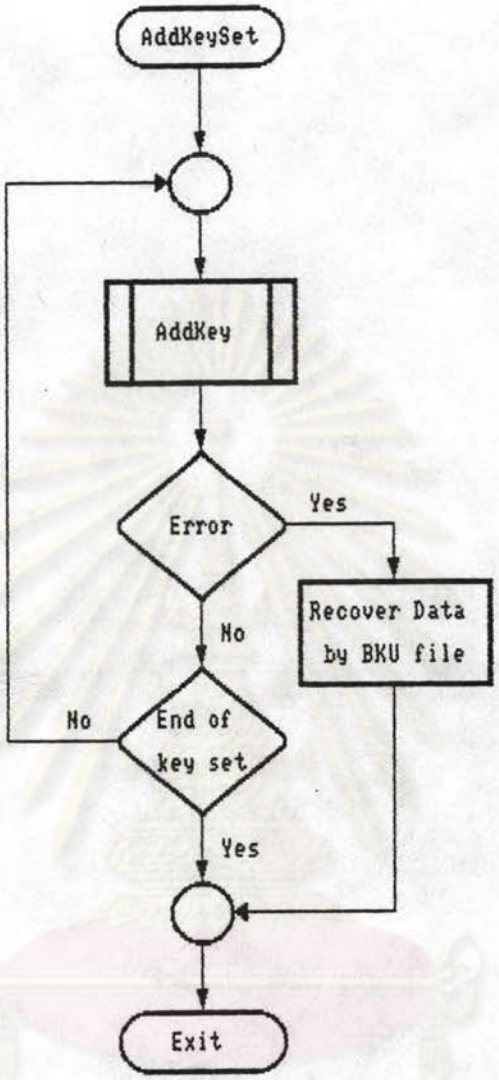
ภาพที่ ก.15 ผังงานของฟังก์ชัน AddDat



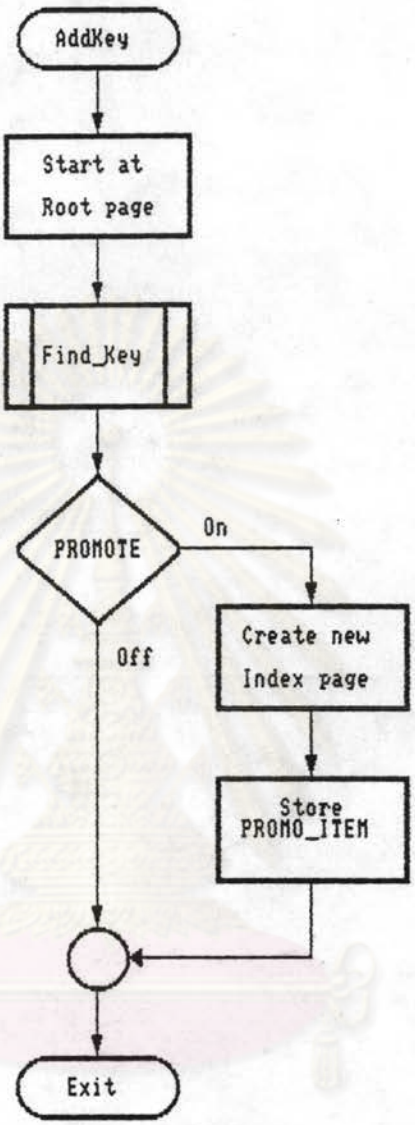
ภาพที่ ก.16 ผังงานของฟังก์ชัน AppendData



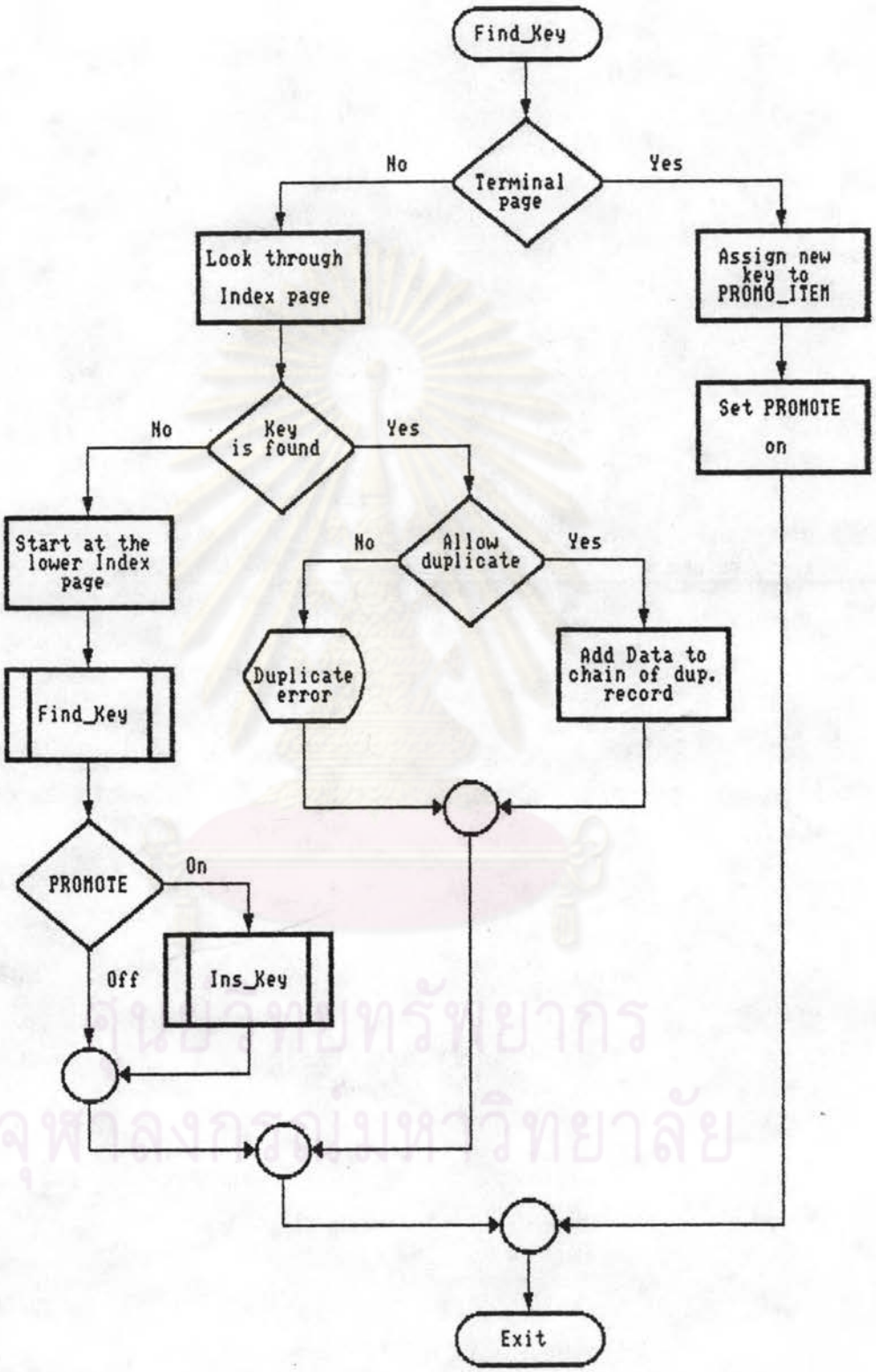
ภาพที่ ก.17 ผังงานของฟังก์ชัน AppendDM



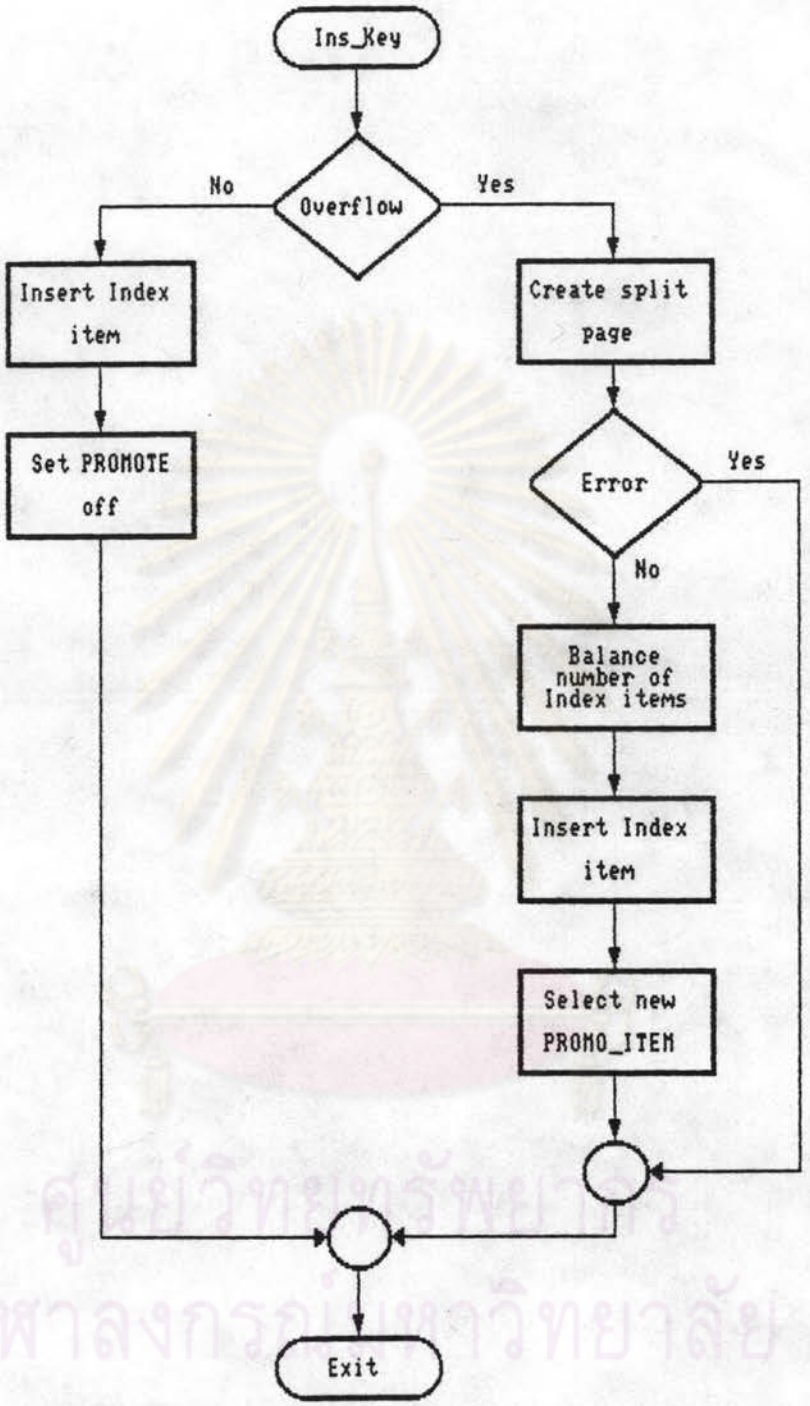
ภาพที่ ก.18 ผังงานของฟังก์ชัน AddKeySet



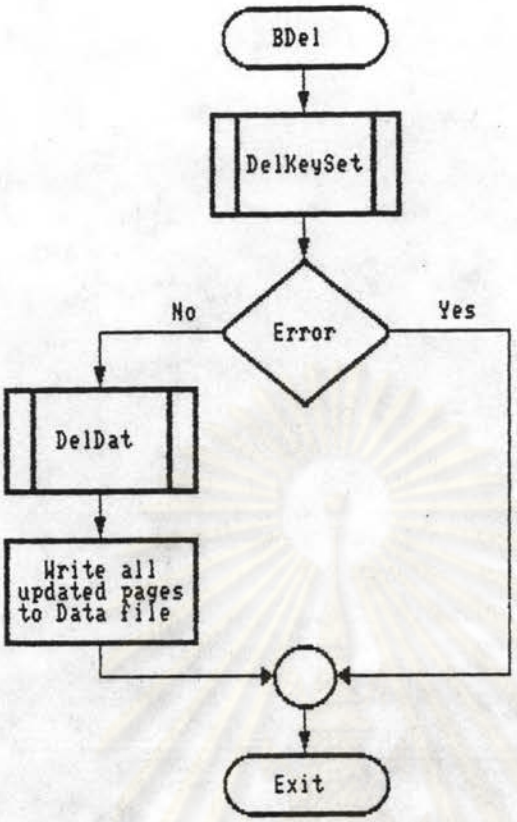
ภาพที่ ก.19 ผังงานของฟังก์ชัน AddKey



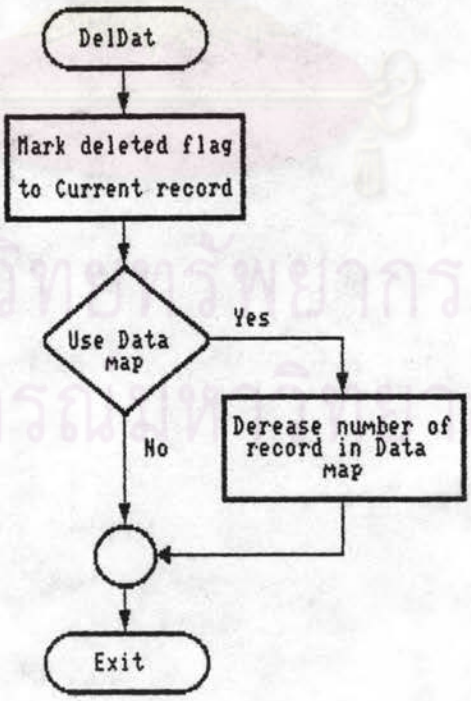
ภาพที่ ก.20 ผังงานของฟังก์ชัน Find_Key



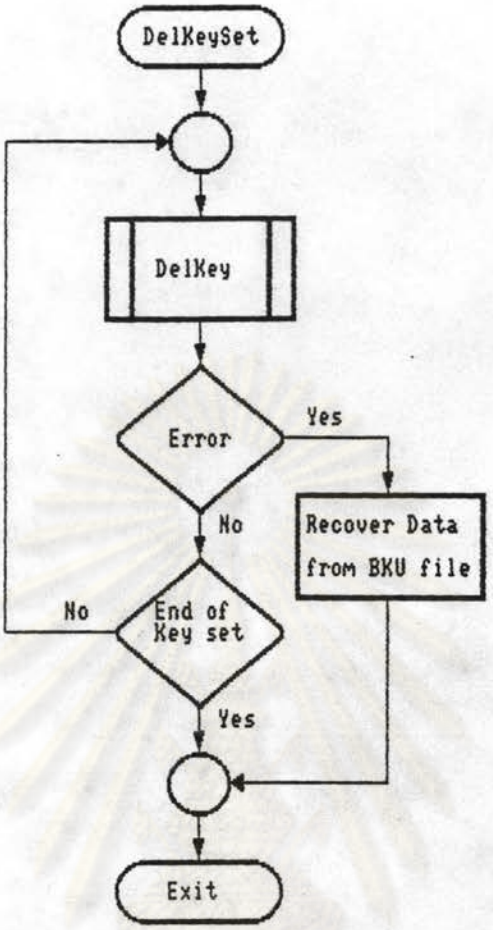
ภาพที่ ก.21 ผังงานของฟังก์ชัน Ins_Key



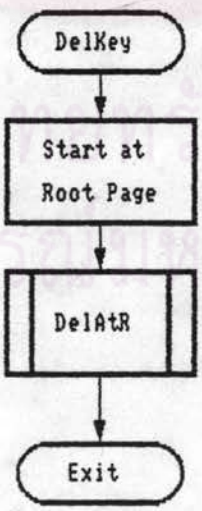
ภาพที่ ก.22 ผังงานของฟังก์ชัน BDel



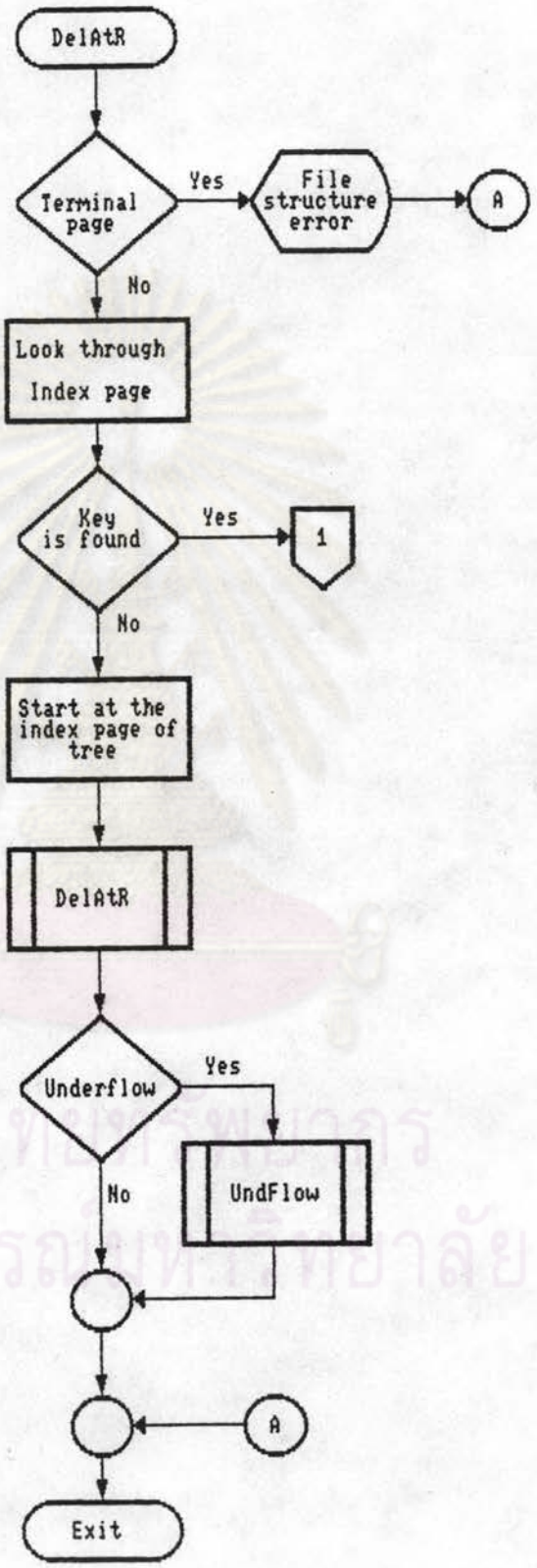
ภาพที่ ก.23 ผังงานของฟังก์ชัน DelDat



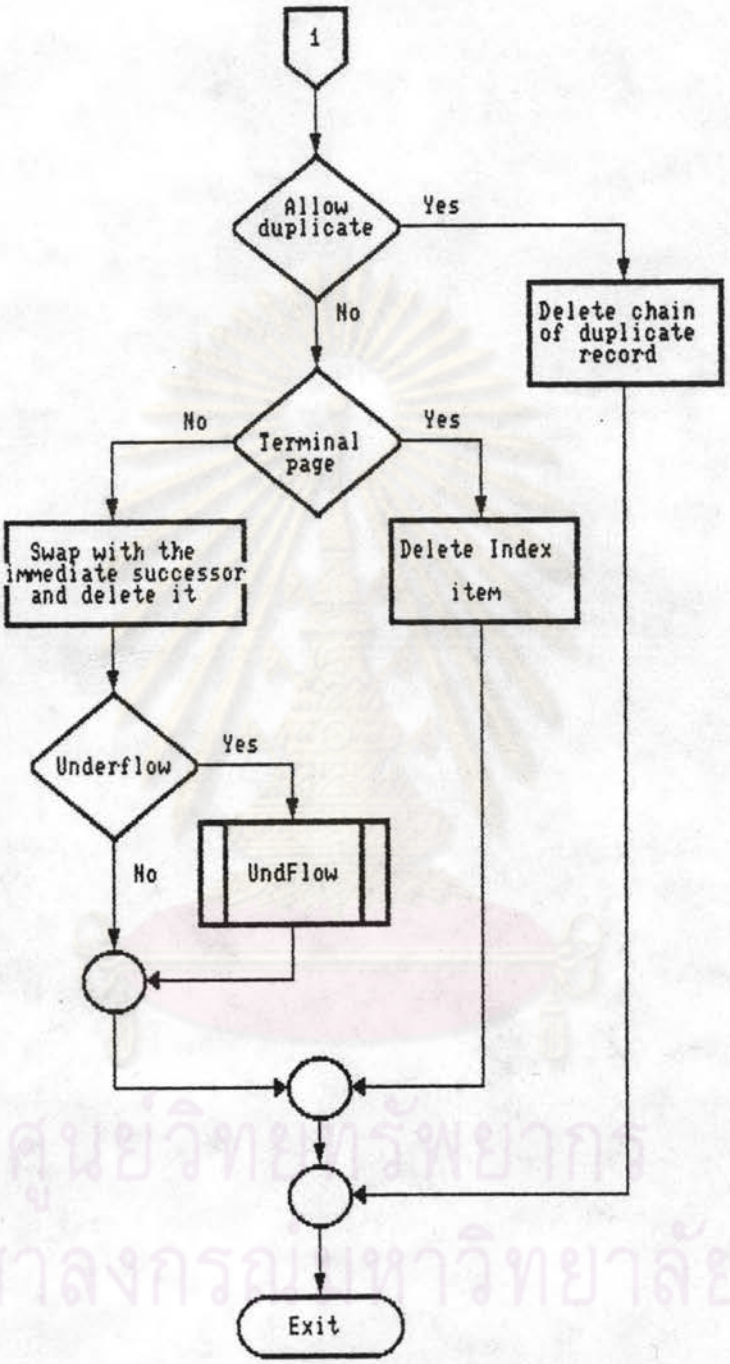
ภาพที่ ก.24 ผังงานของฟังก์ชัน DelKeySet



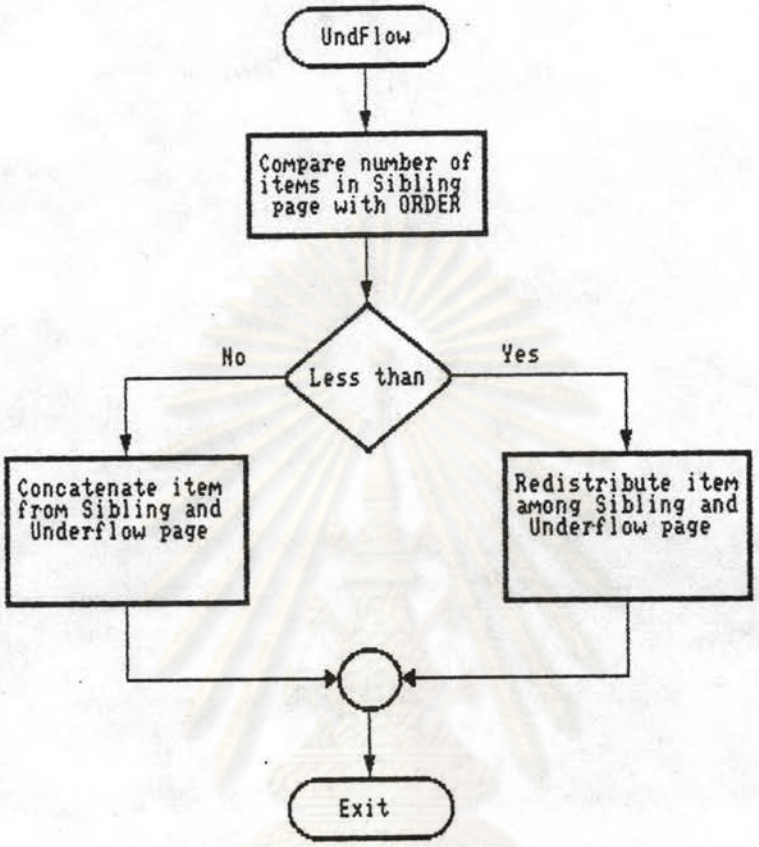
ภาพที่ ก.25 ผังงานของฟังก์ชัน DelKey



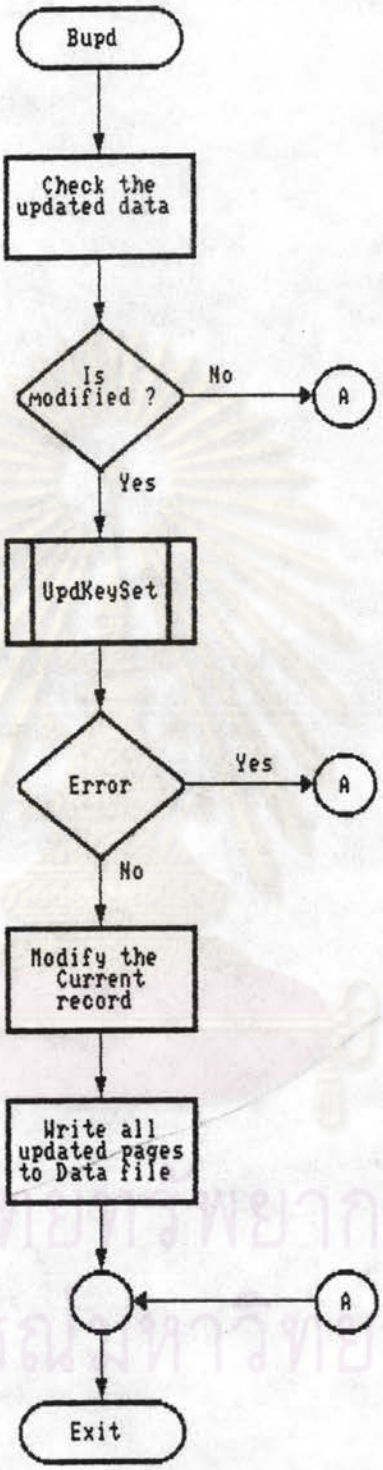
ภาพที่ ก.26 ผังงานของฟังก์ชัน DelAtR



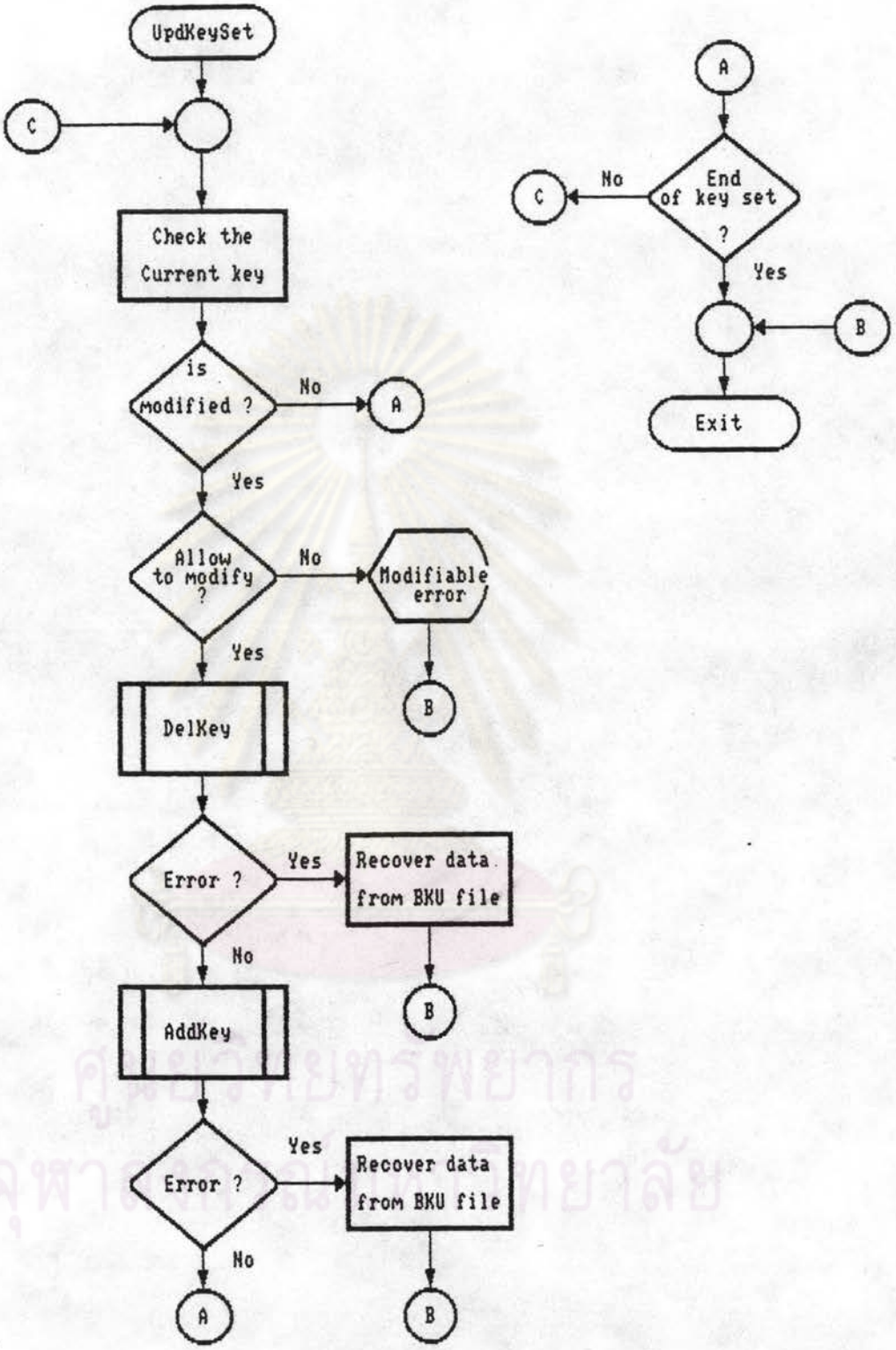
ภาพที่ ก.26 ผังงานของฟังก์ชัน DelAtR(ต่อ)



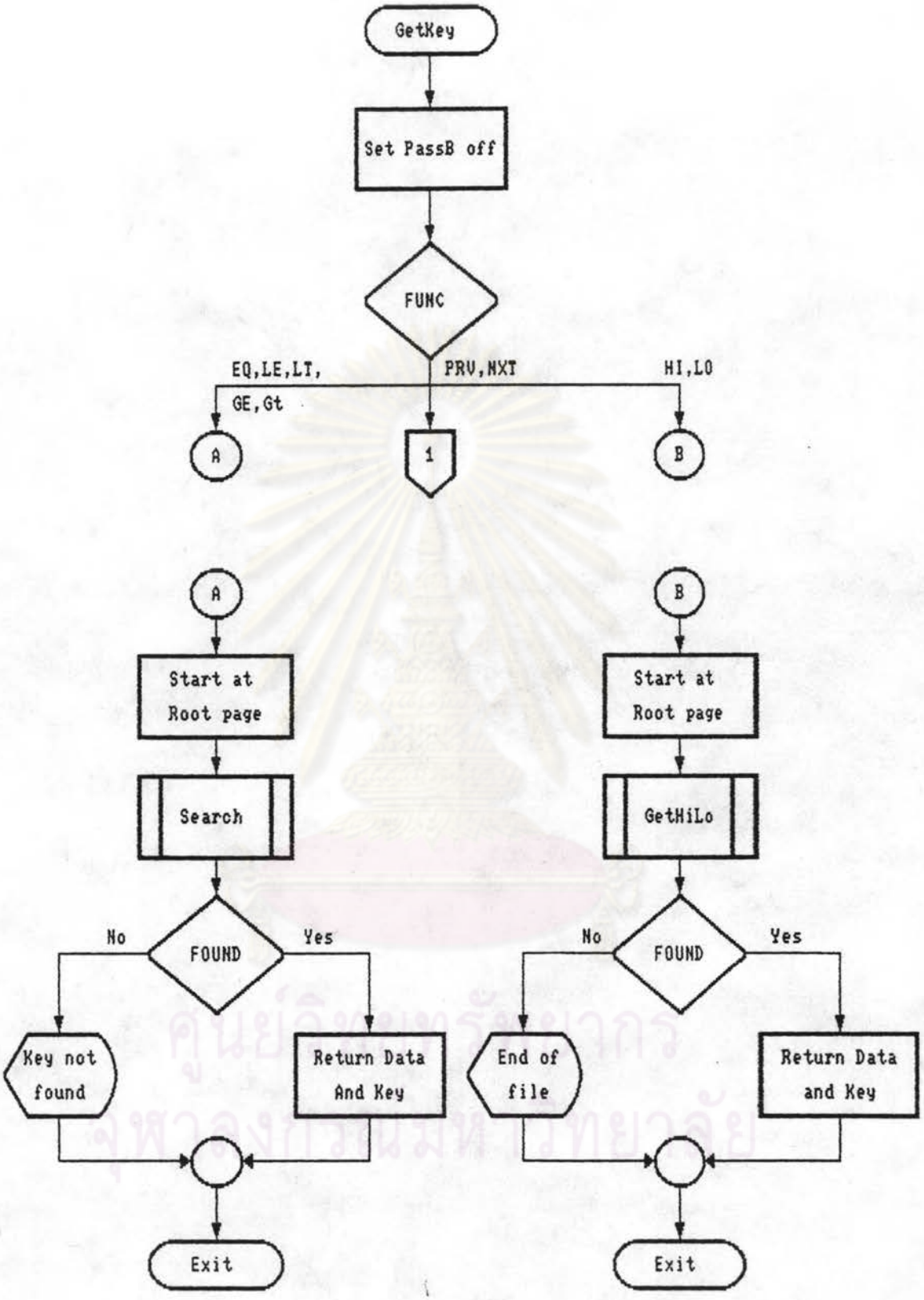
ภาพที่ ก.27 ผังงานของฟังก์ชัน UndFlow



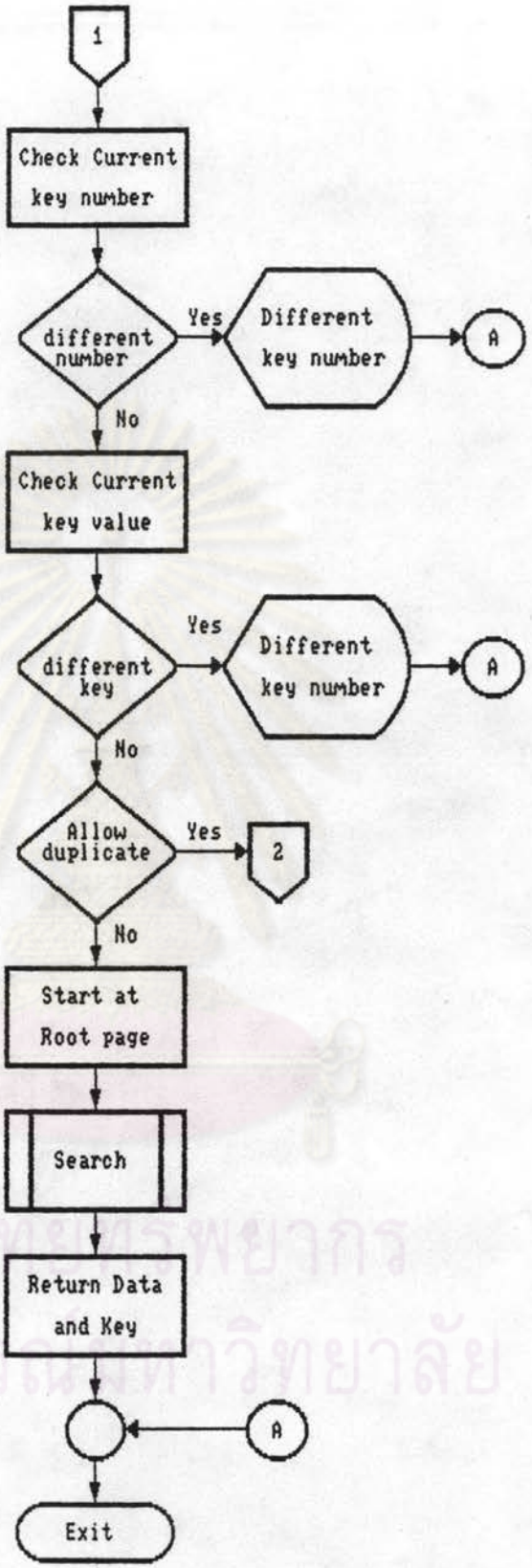
ภาพที่ ก.28 ผังงานของฟังก์ชัน BUpd



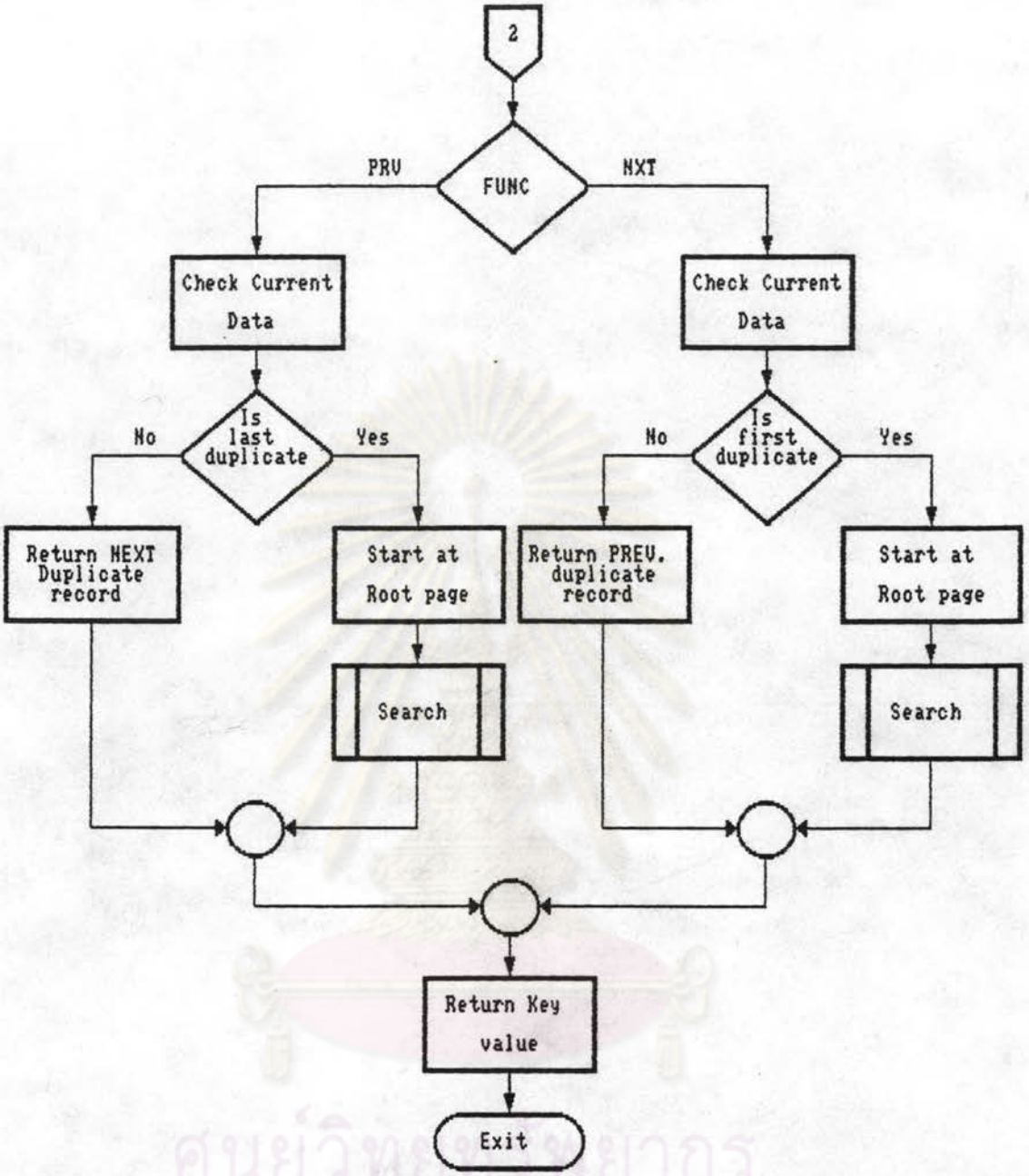
ภาพที่ ก.29 ผังงานของฟังก์ชัน UpdKeySet



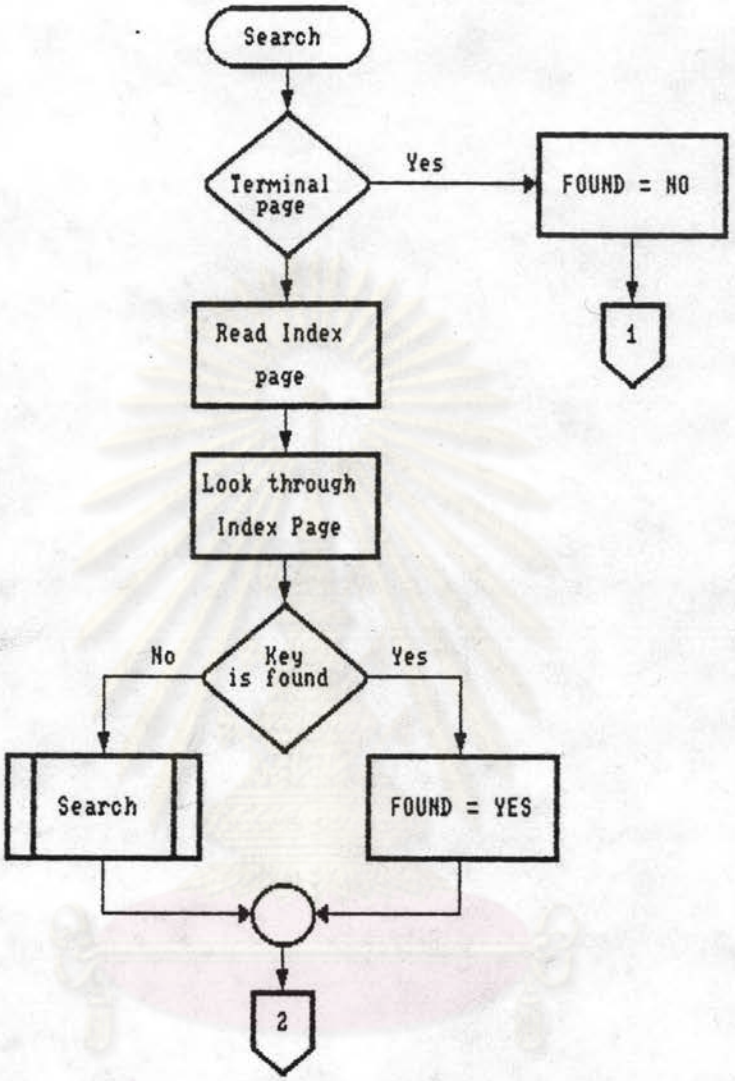
ภาพที่ ก.30 ผังงานของฟังก์ชัน GetKey



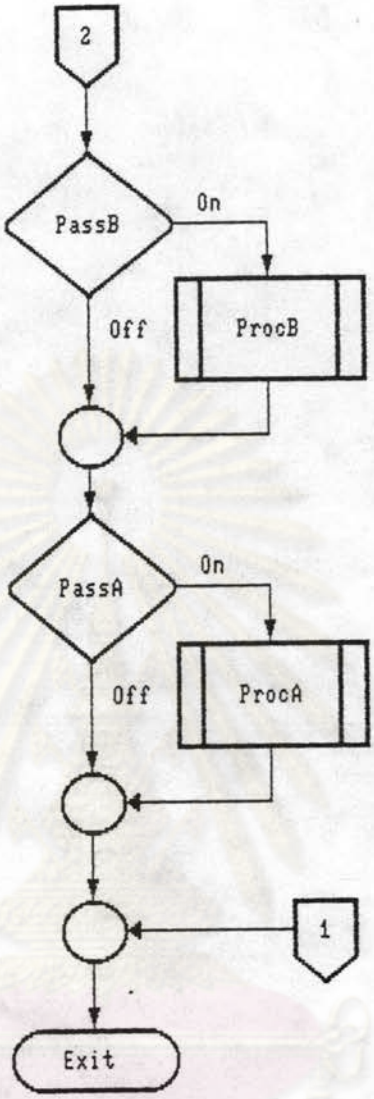
ภาพที่ ก.30 ผังงานของฟังก์ชัน GetKey (ต่อ)



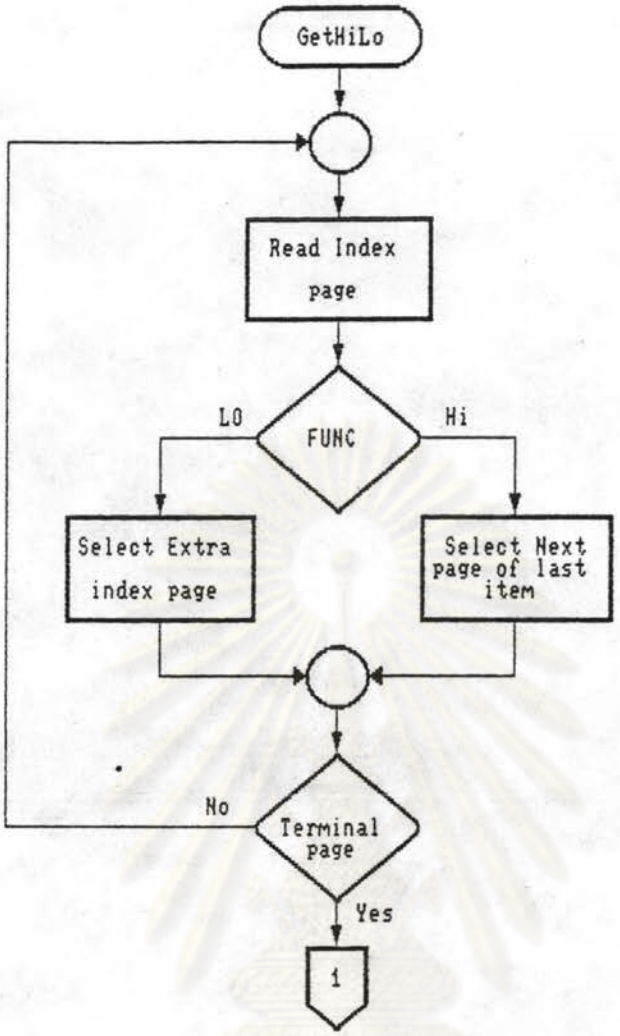
ภาพที่ ก.30 ผังงานของฟังก์ชัน GetKey (ต่อ)



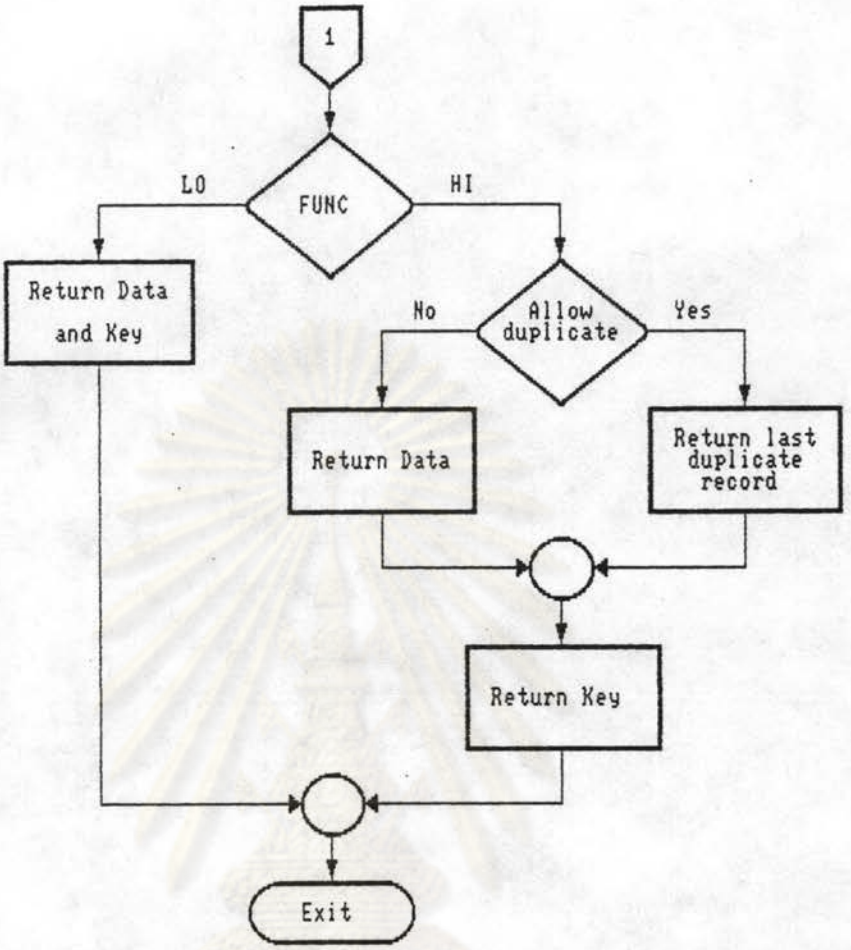
ภาพที่ ก.31 ผังงานของฟังก์ชัน Search



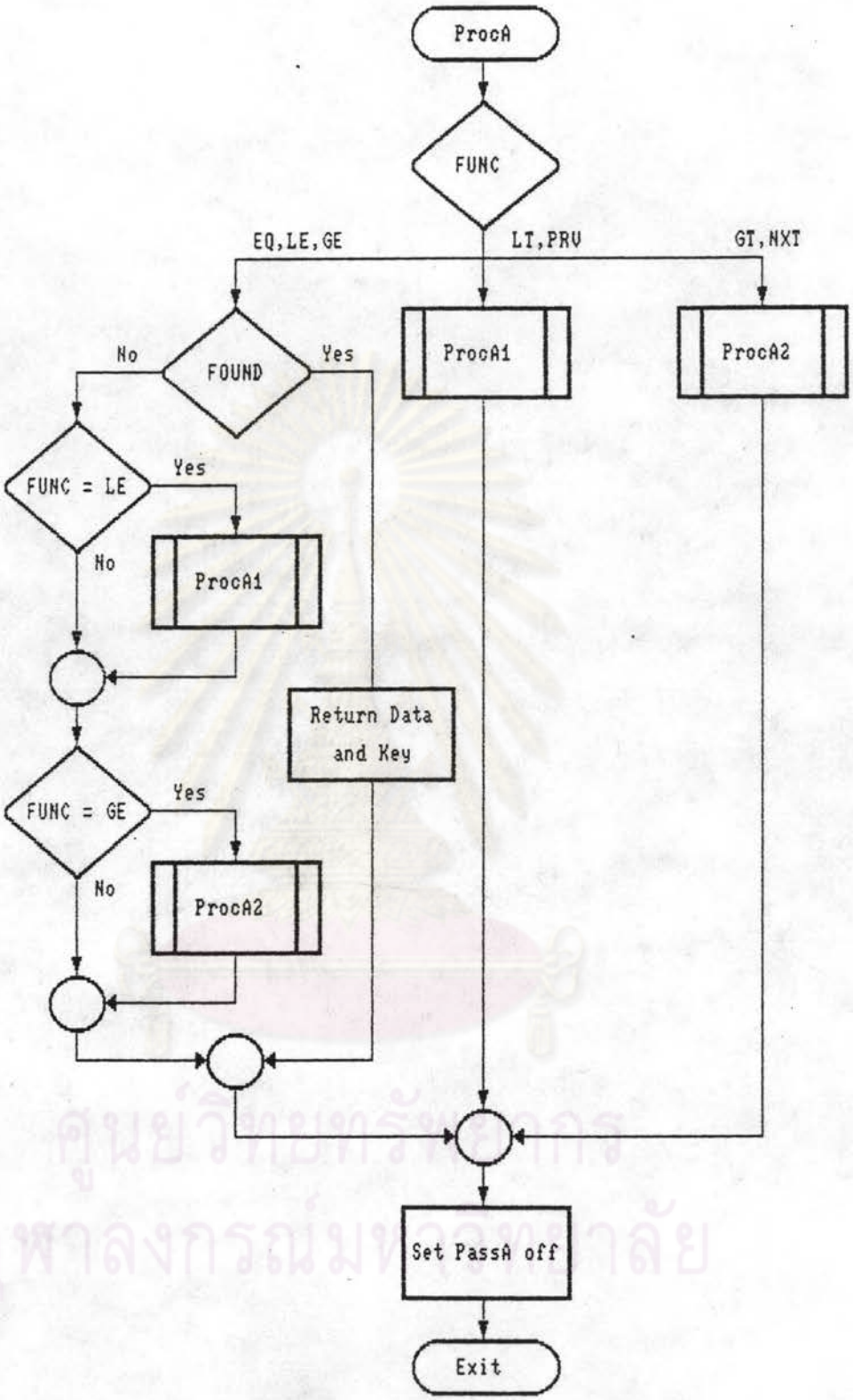
ภาพที่ ก.31 ผังงานของฟังก์ชัน Search (ต่อ)



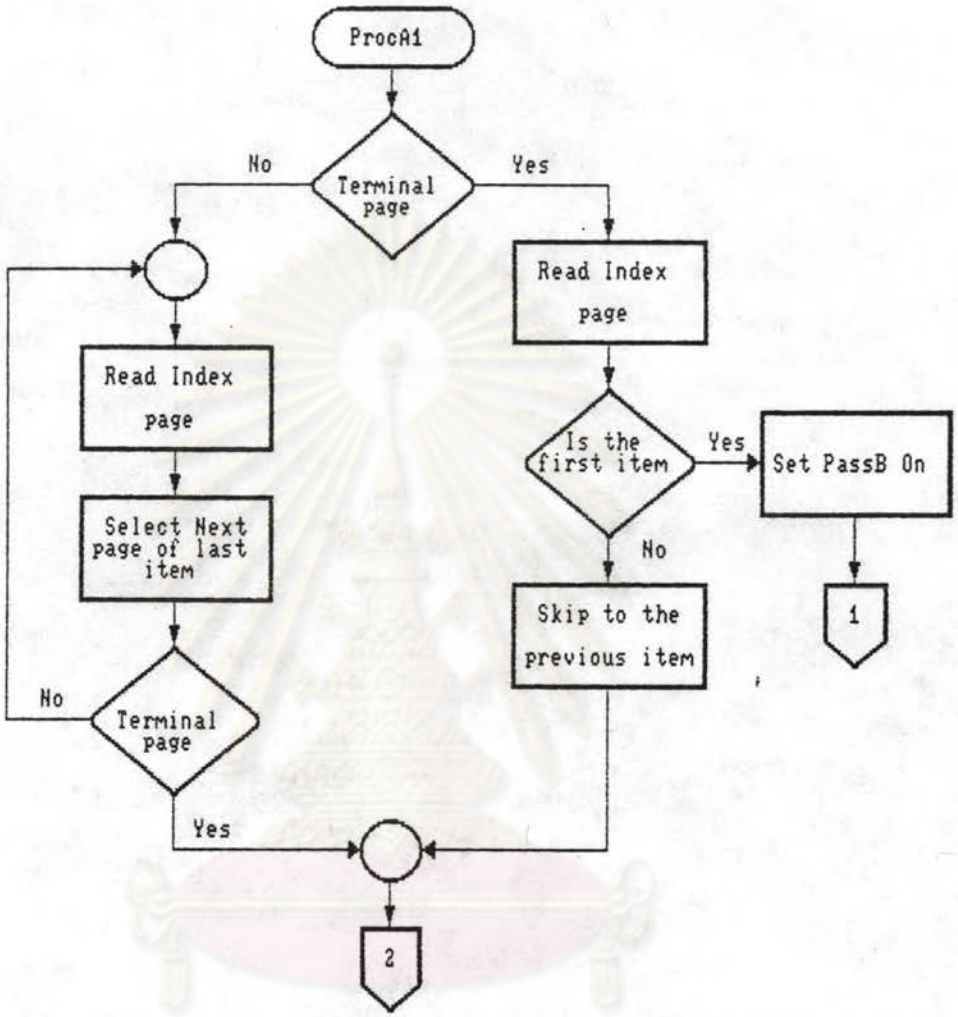
ภาพที่ ก.32 ผังงานของฟังก์ชัน GetHiLo



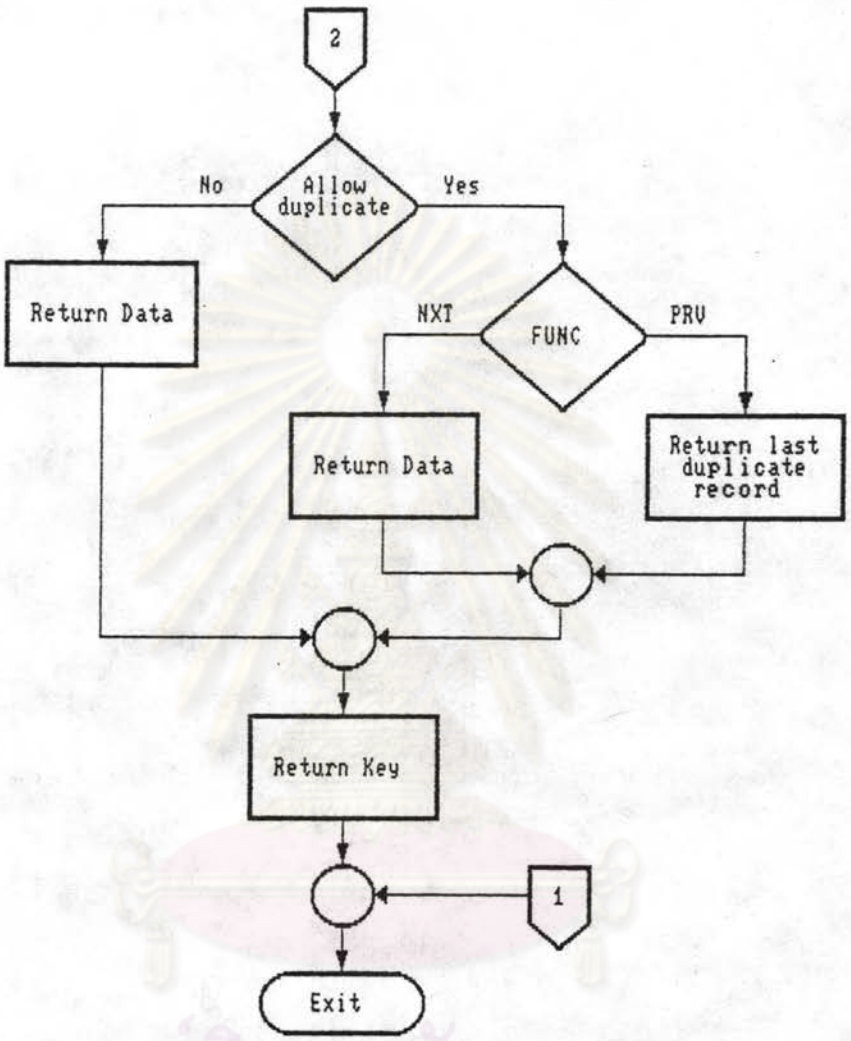
ภาพที่ ก.32 ผังงานของฟังก์ชัน GetHiLo (ต่อ)



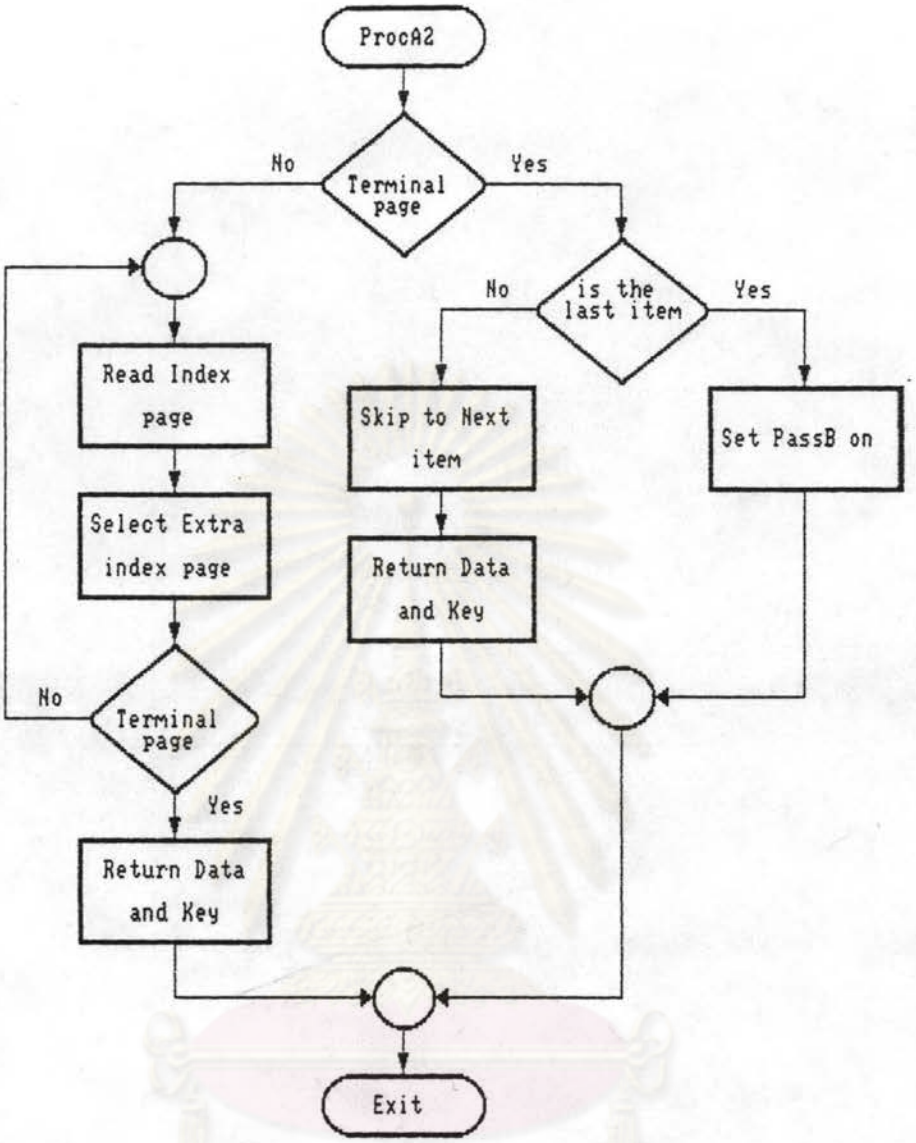
ภาพที่ ก.33 ผังงานของฟังก์ชัน ProcA



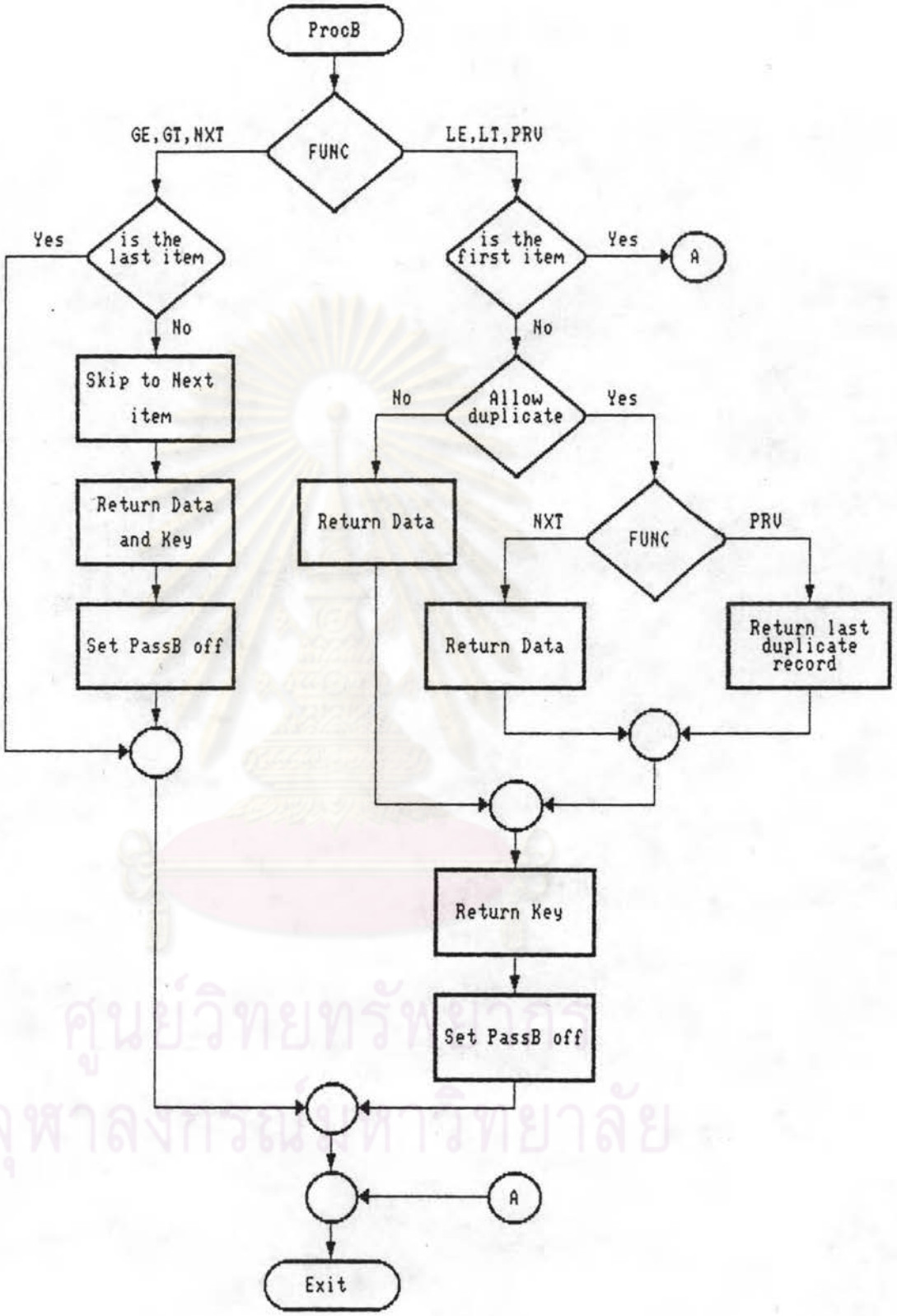
ภาพที่ ก.34 ผังงานของฟังก์ชัน ProcA1



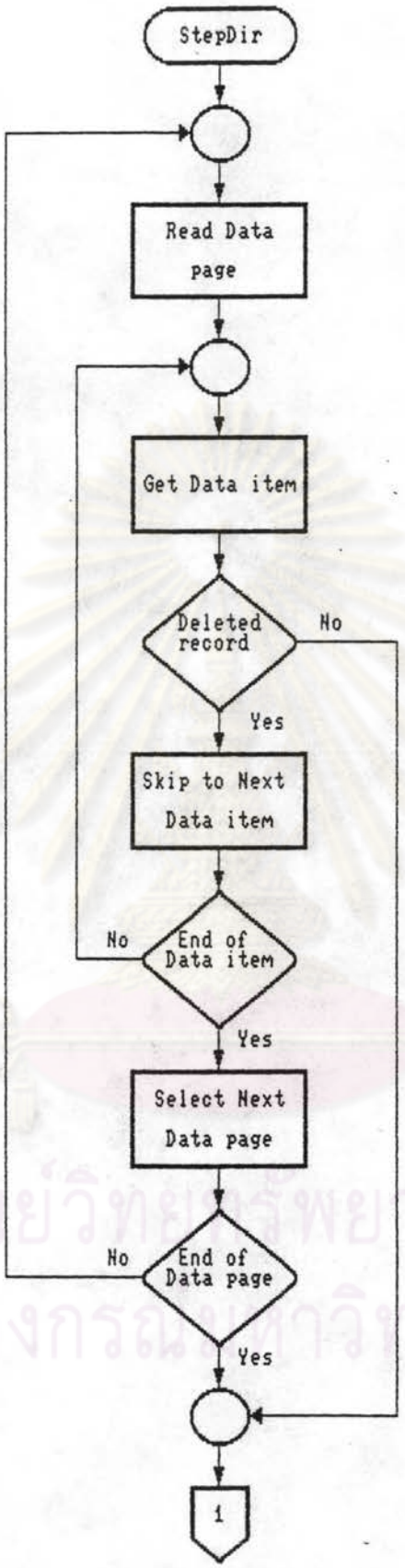
ภาพที่ ก.34 ผังงานของฟังก์ชัน ProcA1 (ต่อ)



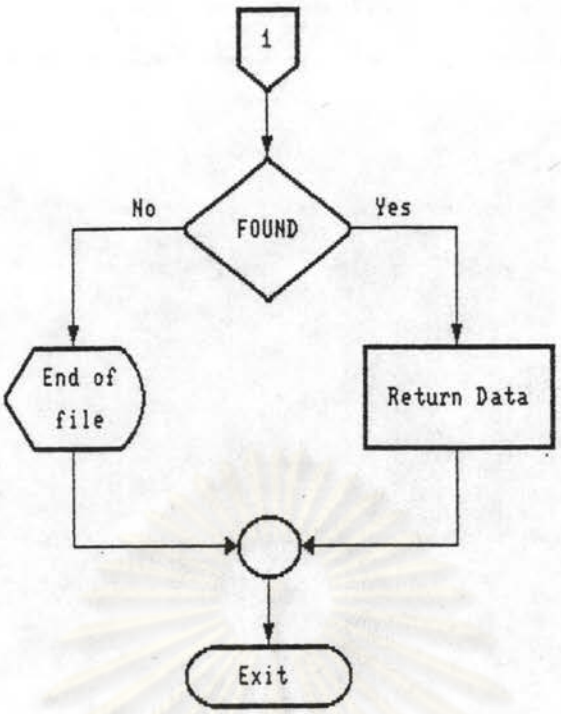
ภาพที่ ก.35 ผังงานของฟังก์ชัน ProcA2



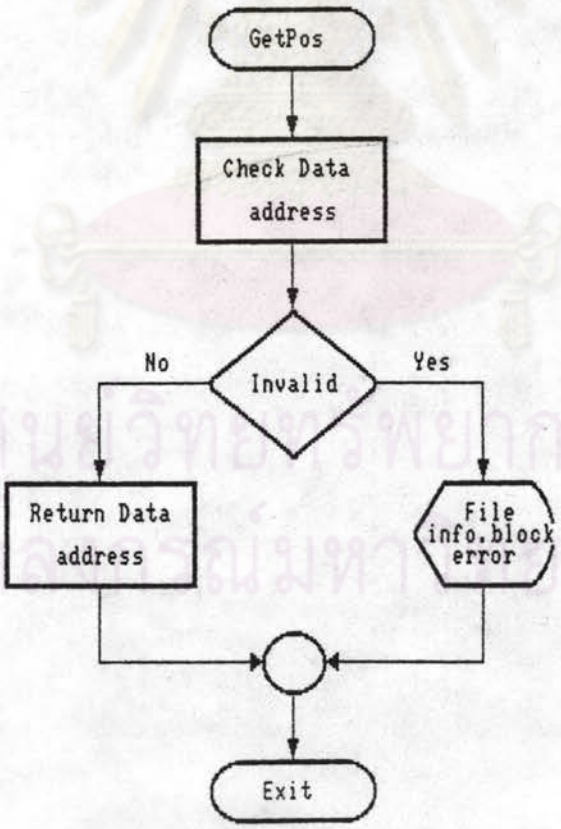
ภาพที่ ก.36 ผังงานของฟังก์ชัน ProcB



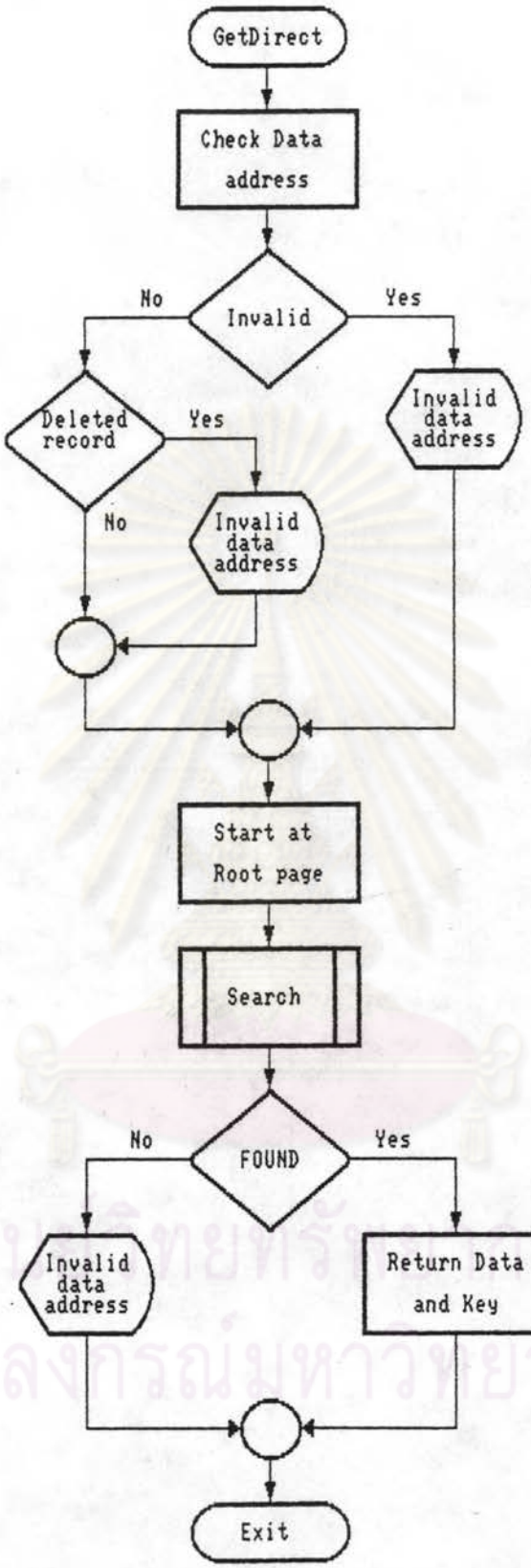
ภาพที่ ก.37 ผังงานของฟังก์ชัน StepDir



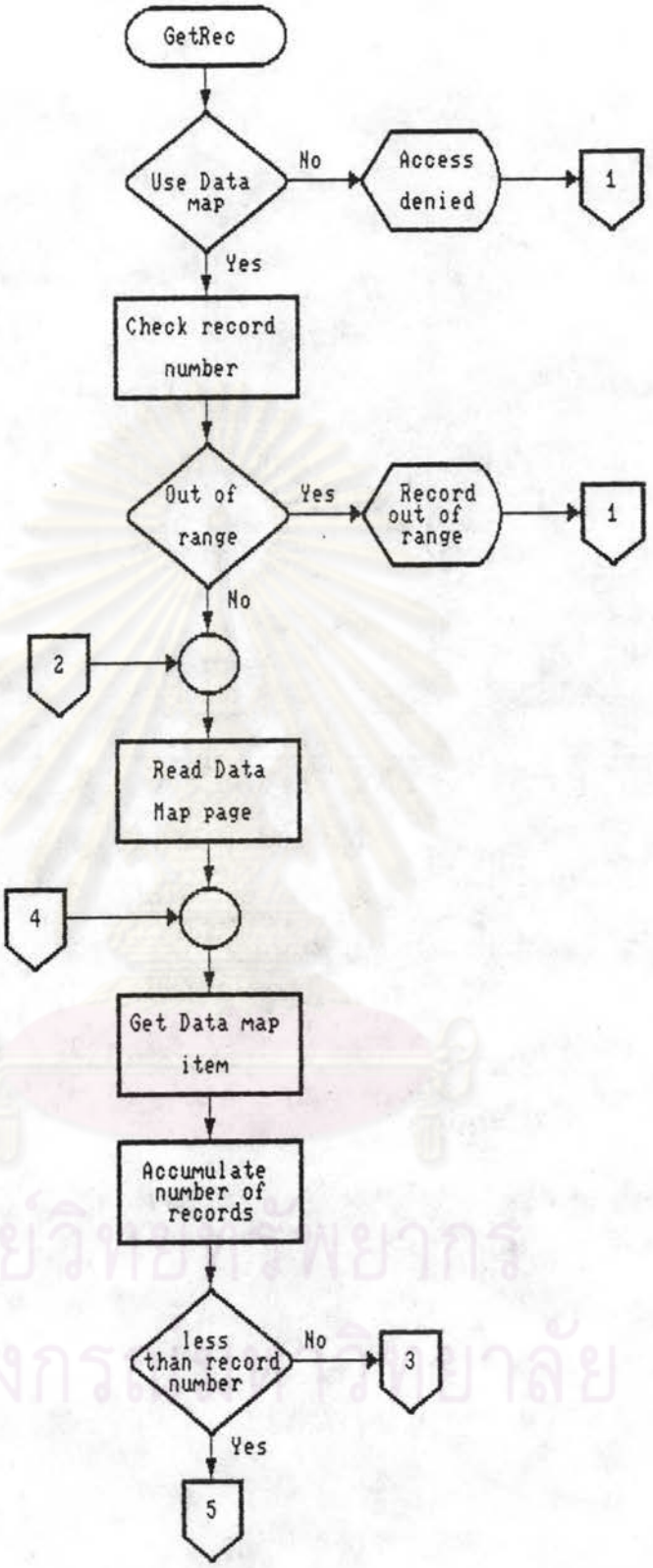
ภาพที่ ก.37 ผังงานของฟังก์ชัน StepDir (ต่อ)



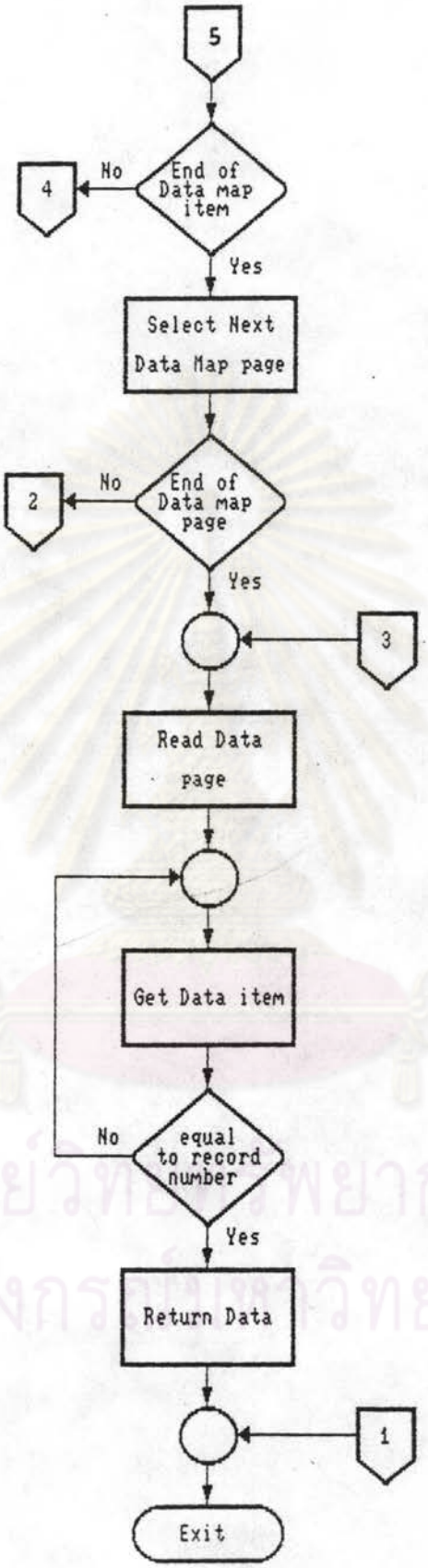
ภาพที่ ก.38 ผังงานของฟังก์ชัน GetPos



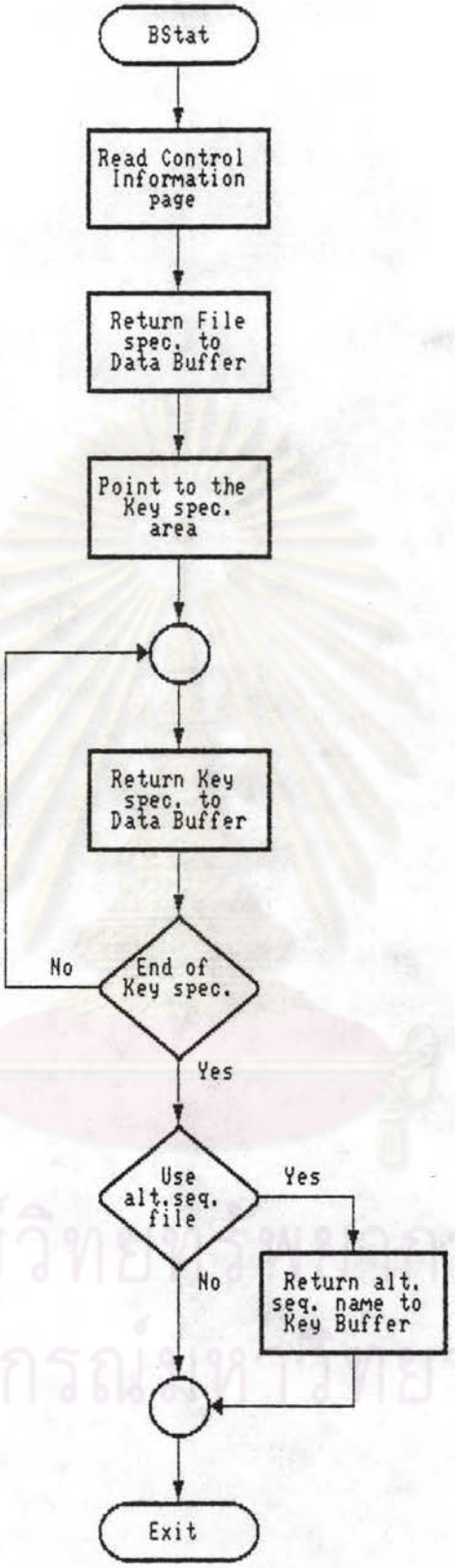
ภาพที่ ก.39 ผังงานของฟังก์ชัน GetDirect



ภาพที่ ก.40 ผังงานของฟังก์ชัน GetRec

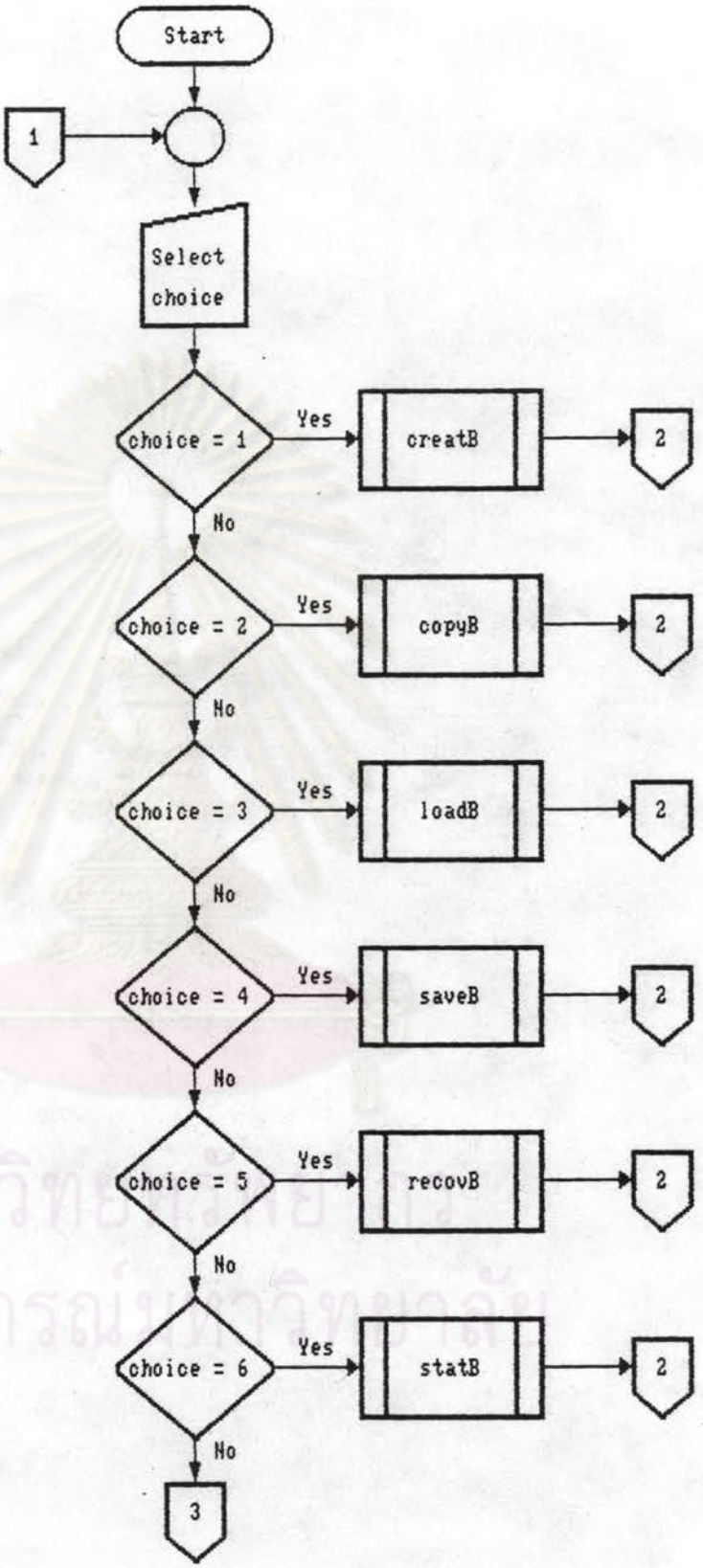


ภาพที่ ก.40 ผังงานของฟังก์ชัน GetRec (ต่อ)

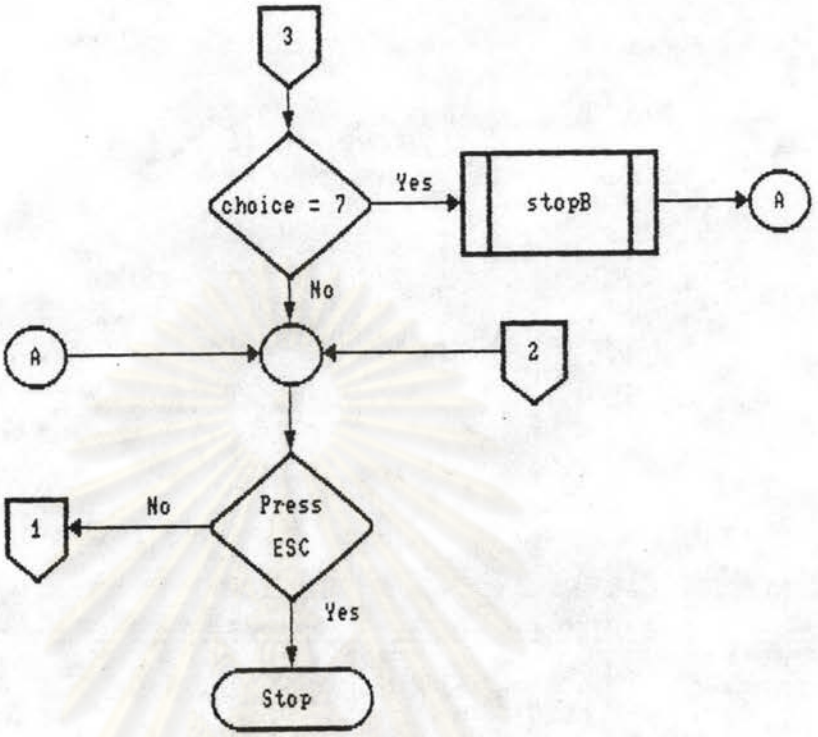


ภาพที่ ก.41 ผังงานของฟังก์ชัน BStat

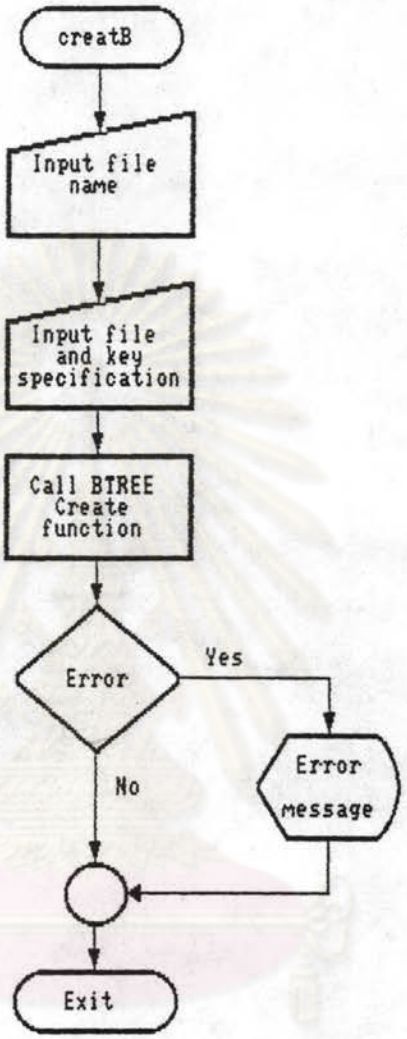
ภาพที่ ก.42 ผังงานของฟังก์ชัน BStop



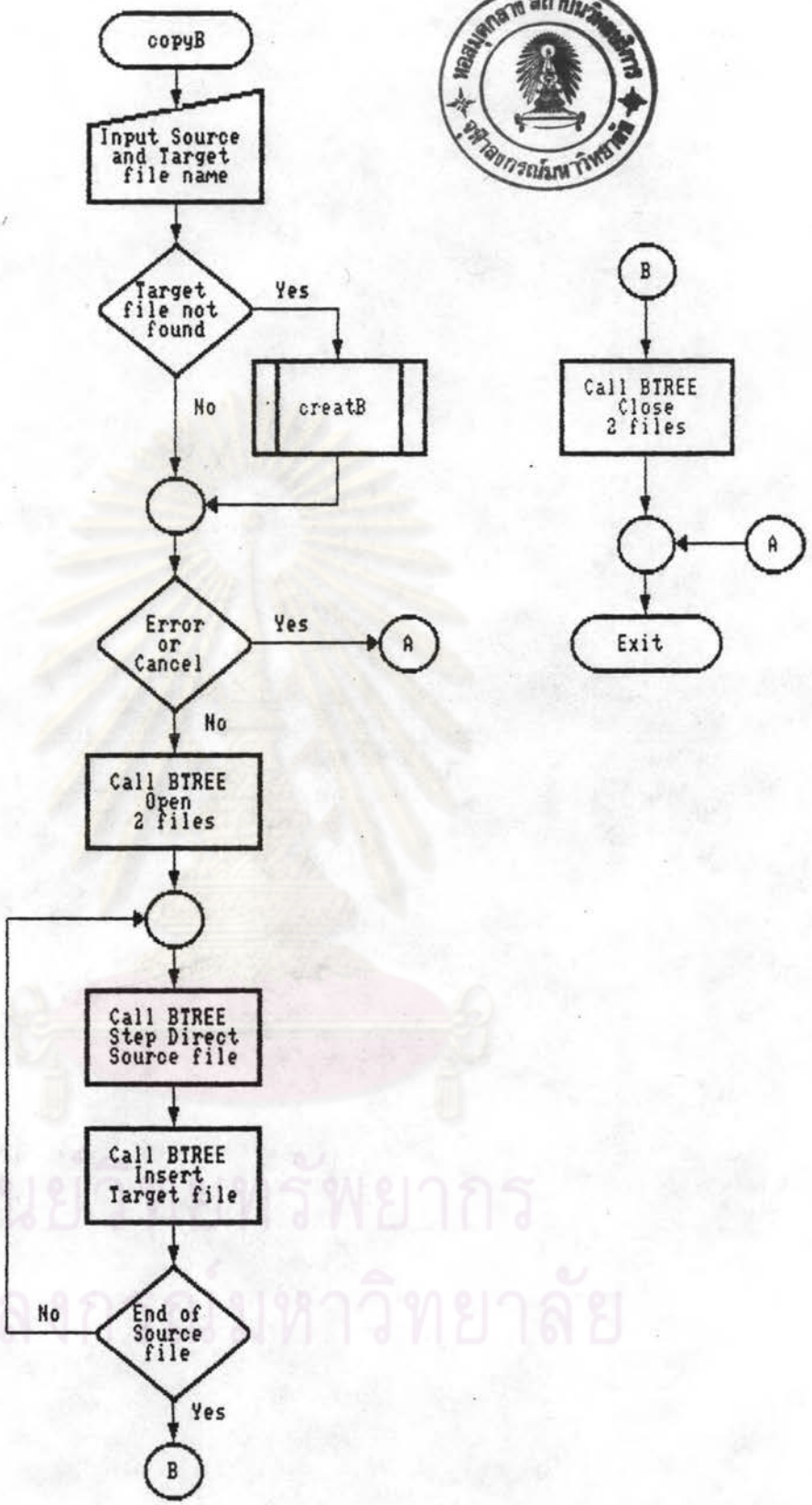
ภาพที่ ก.43 ผังงานของโปรแกรมอรรถประโยชน์



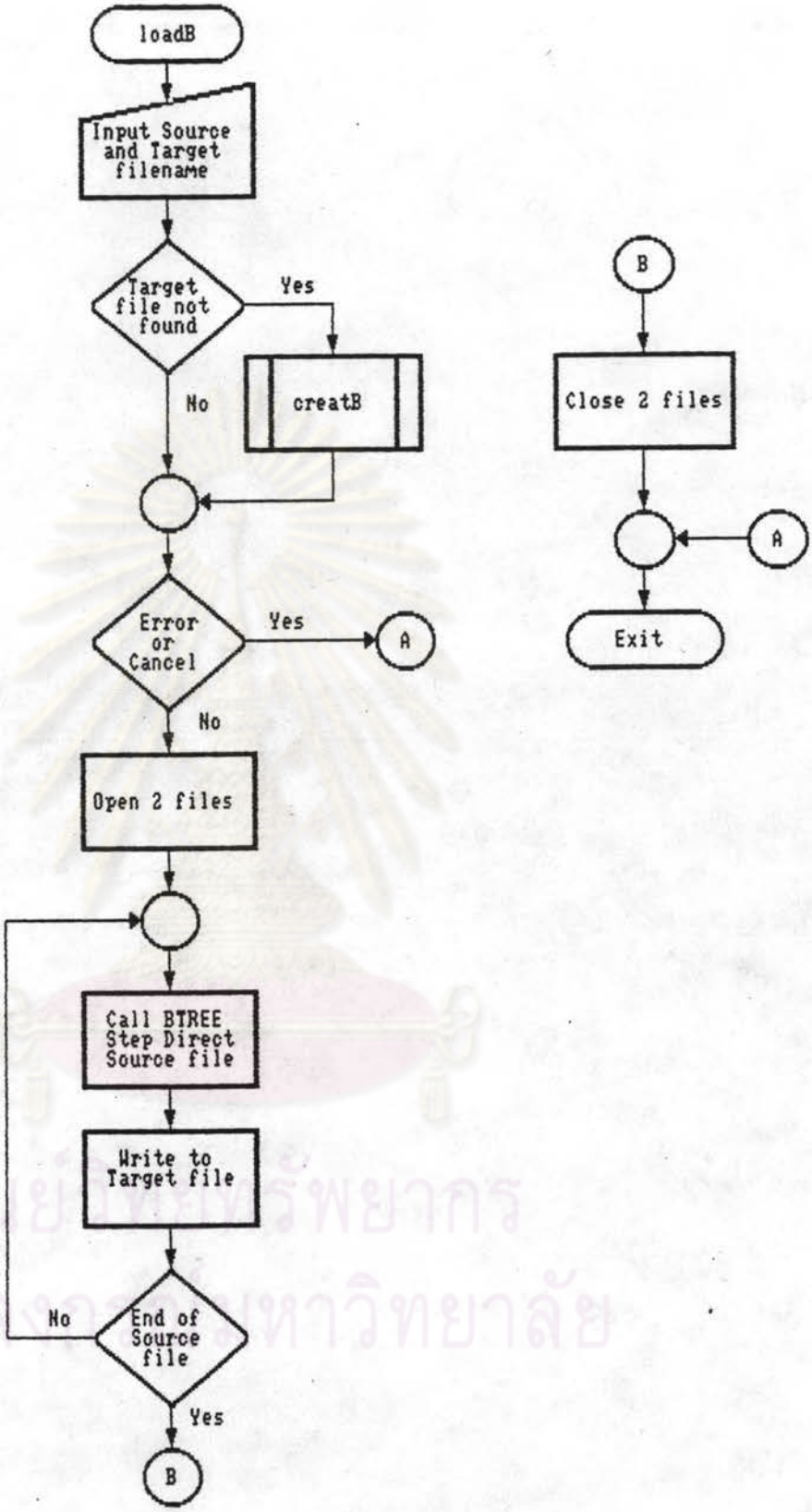
ภาพที่ ก.43 ผังงานของโปรแกรมอรรถประโยชน์ (ต่อ)



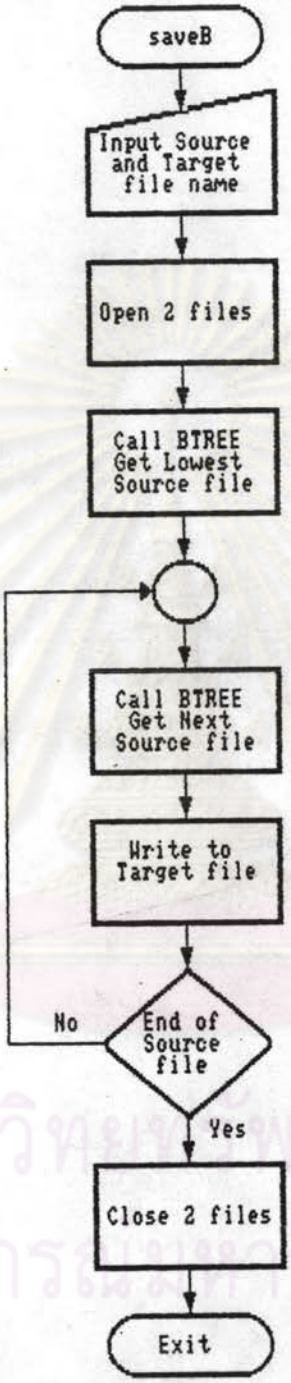
ภาพที่ ก.44 ผังงานของฟังก์ชัน creatB



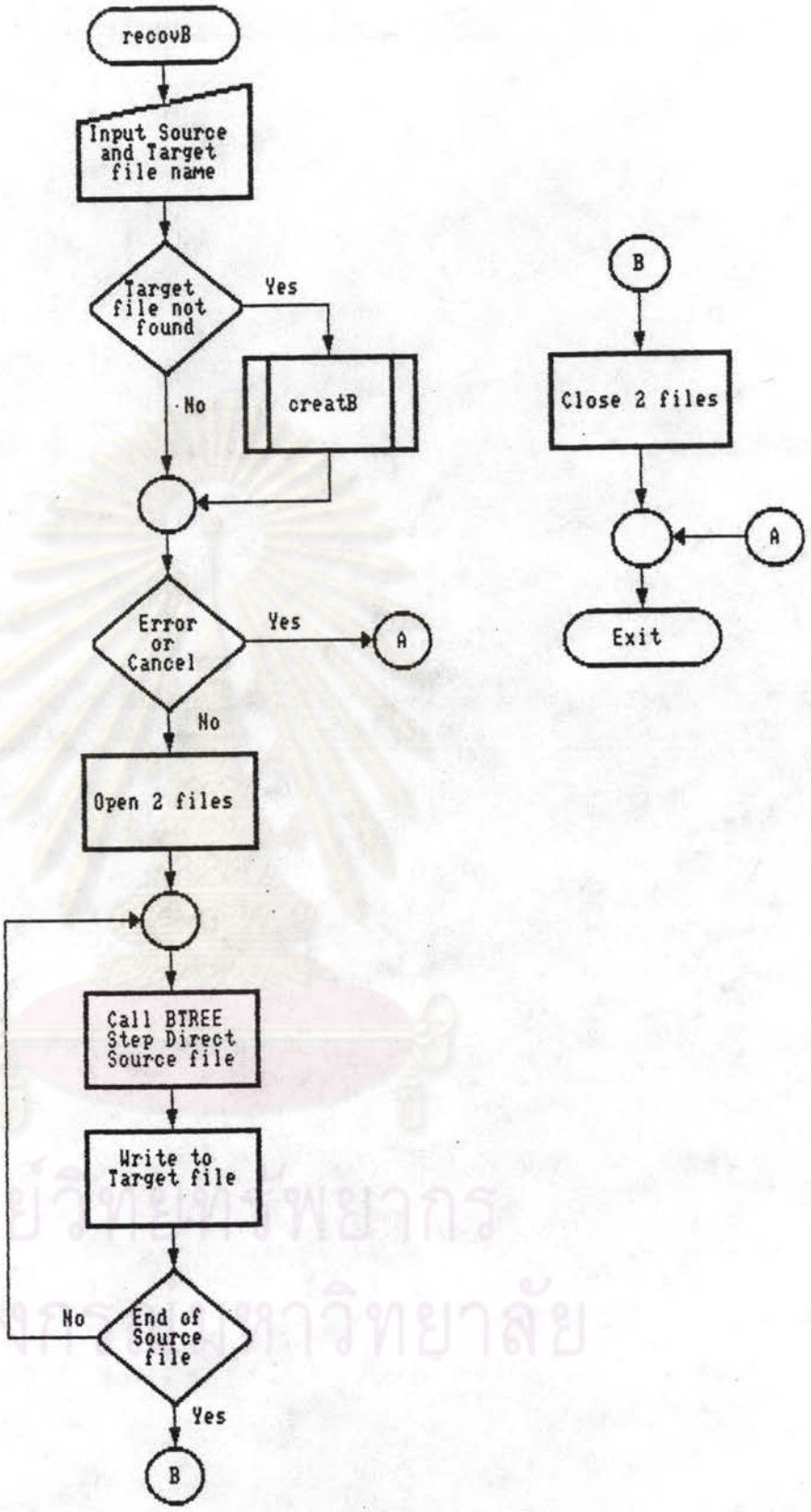
ภาพที่ ก.45 ผังงานของฟังก์ชัน copyB



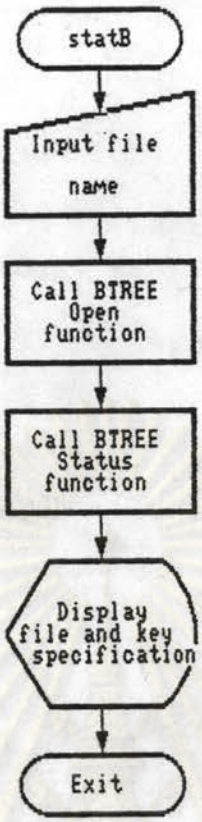
ภาพที่ ก.46 ผังงานของฟังก์ชัน loadB



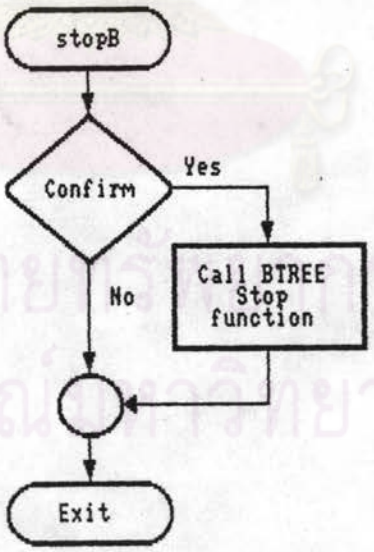
ภาพที่ ก.47 ผังงานของฟังก์ชัน saveB



ภาพที่ ก.48 ผังงานของฟังก์ชัน recovB



ภาพที่ ก.49 ผังงานของฟังก์ชัน statB



ภาพที่ ก.50 ผังงานของฟังก์ชัน stopB

ภาคผนวก ข

คำอธิบายเกี่ยวกับรหัสความผิดพลาด

คำอธิบายเกี่ยวกับรหัสความผิดพลาด

ทุกครั้งที่มีการเรียกใช้ฟังก์ชันจัดการแฟ้มข้อมูลดัชนี ซึ่งประจำในหน่วยความจำ ฟังก์ชันเหล่านั้นจะส่งรหัสความผิดพลาดกลับมาที่โปรแกรมที่เรียกใช้ทุกครั้ง หลังจากจบการทำงานของฟังก์ชัน รหัสความผิดพลาดจะเท่ากับ 0 เสมอเมื่อการทำงานของฟังก์ชันเป็นไปอย่างถูกต้อง และมีค่ามากกว่า 0 เมื่อเกิดความผิดพลาดในระหว่างการทำงาน รหัสความผิดพลาดเหล่านั้น มีความหมายดังนี้

01 Invalid operation

เกิดจากการกำหนดหมายเลขฟังก์ชันใน operation number ไม่อยู่ในช่วงของ 1 ถึง 24

02 Invalid file information block

กำหนดใช้ file information block ที่ไม่ถูกต้อง อาจเกิดจากการบันทึกข้อมูลทับลงในเนื้อที่ของ file information block โดยการกระทำของโปรแกรมหรือการใช้บัฟเฟอร์ข้อมูลที่ไม่ใช่ข้อมูลภายใน file inforamtion block ที่ใช้อยู่ในขณะนั้น

03 No open file

เกิดจากการใช้ฟังก์ชันต่าง ๆ ของฟังก์ชันจัดการแฟ้มข้อมูลดัชนี โดยที่ไม่ได้ทำการเปิดแฟ้มข้อมูลดัชนีขึ้นก่อน

04 Access denied

อาจเกิดจากสาเหตุใดสาเหตุหนึ่ง เช่น ทำการเพิ่มเติม ลบ หรือแก้ไขระเบียนข้อมูลภายในแฟ้มข้อมูลดัชนี ที่เปิดใช้ในลักษณะของการอ่านระเบียนข้อมูลเพียงอย่างเดียว หรือ การใช้ฟังก์ชันค้นหาระเบียนข้อมูลโดยการใช้คีย์ภายในแฟ้มข้อมูลที่เกิดความเสียหาย ซึ่งไม่สามารถใช้คีย์ในการค้นหาระเบียนข้อมูลได้ หรือ การใช้ฟังก์ชันค้นหาระเบียนจากหมายเลขระเบียน ในแฟ้มข้อมูลดัชนีที่ไม่กำหนดให้สร้างแผ่นแผนที่ข้อมูล

05 I/O error

เกิดความผิดพลาดในระหว่างการอ่าน หรือบันทึกข้อมูลลงในจานบันทึก ซึ่งอาจมีสาเหตุมาจาก เกิดความเสียหายในแผ่นจานบันทึกจนไม่สามารถอ่าน หรือบันทึกข้อมูลได้

06 Structure error

เกิดความเสียหายกับโครงสร้างข้อมูลที่ควบคุมการเชื่อมโยงแผ่นต่าง ๆ ภายในแฟ้มข้อมูลดัชนี ทำให้ข้อมูลที่อ่านจากแฟ้ม เป็นข้อมูลที่ไม่ถูกต้อง

07 Disk full

เกิดจากแผ่นจานบันทึกไม่มีเนื้อที่เพียงพอสำหรับการบันทึกข้อมูลอีกต่อไป

08 BKU file open error

เกิดความผิดพลาดในระหว่างการสร้าง หรือการเปิด หรือการปิดแฟ้มสำรองแผ่น ซึ่งอาจมีสาเหตุมาจาก ไม่มีเนื้อที่ว่างพอสำหรับการบันทึกข้อมูลภายในจานบันทึก หรือแผ่นจานบันทึกถูกป้องการบันทึกข้อมูล (write protect) หรือไม่มีที่ว่างภายในไดเรคทอรี (Direcotry entry) ของแผ่นจานบันทึกสำหรับบันทึกชื่อแฟ้ม

09 BKU file I/O error

เกิดจากความผิดพลาดในการอ่าน หรือบันทึกข้อมูล ภายในแฟ้มสำรองแผ่น

10 File is existed

ก่อนการสร้างแฟ้มข้อมูลดัชนี ฟังก์ชันสร้างแฟ้มข้อมูลดัชนีจะตรวจสอบ ชื่อแฟ้มที่ระบุใน key buffer ถูกกำหนดแล้วในจานบันทึกหรือไม่ ถ้าตรวจพบจะยกเลิกการสร้างแฟ้มข้อมูลดัชนีนั้น

11 Create I/O error

ในการสร้างแฟ้มข้อมูลดัชนี ฟังก์ชันจะบันทึกข้อมูลเริ่มต้นลงในจานบันทึก แต่ไม่สามารถบันทึกได้ ซึ่งอาจมีสาเหตุมาจาก ไม่มีเนื้อที่ว่างพอสำหรับการบันทึกข้อมูลภายในจานบันทึก หรือ แผ่นจานบันทึกถูกป้องการบันทึกข้อมูล (write protect) หรือไม่มีที่ว่าง

ภายในไดเรคทอรีของแผ่นจานบันทึกสำหรับบันทึกชื่อแฟ้ม หรือ อาจเกิดความเสียหายกับแผ่นจานบันทึก จนไม่สามารถบันทึกข้อมูลได้

12 Invalid file name

เกิดจากการระบุชื่อแฟ้มข้อมูลดัชนีใน key buffer ไม่ถูกต้อง ชื่อแฟ้มข้อมูลจะต้องปิดท้ายด้วยการเว้นช่องว่างอย่างน้อย 1 ช่องหรือใช้รหัสแอสกี Ø แทน รูปแบบของการกำหนดชื่อแฟ้ม คือ [drive:][\path\]file name.extension อาจกำหนด drive และ path หรือไม่กำหนดก็ได้

13 Record length

เกิดจากการกำหนดขนาดของระเบียนข้อมูลสำหรับสร้างแฟ้มข้อมูลดัชนี ขนาดของระเบียนข้อมูลมีความยาวที่สุดไม่เกิน 4090 ไบต์ และมีขนาดไม่เกินผลต่างของขนาดแผ่นที่กำหนดให้แฟ้มนั้น กับขนาดระเบียนควบคุมซึ่งมีค่าเท่ากับ 6 ไบต์ และขนาดของตำแหน่งระเบียนข้อมูลที่เชื่อมต่อ ในกรณีที่กำหนดให้ระเบียนข้อมูลมีคุณสมบัติคีย์ที่มีค่าคีย์ซ้ำกันได้

14 Page size

ขนาดของแผ่นที่กำหนดให้แฟ้มข้อมูลดัชนี มีขนาดตั้งแต่ 512 - 4096 ไบต์ และขนาดที่กำหนดจะต้องเป็นผลคูณของ 512 ไบต์ นอกจากนี้การกำหนดขนาดของแผ่นที่ใช้ในขณะนั้นจะต้องไม่มากกว่า ขนาดของช่องใส่แผ่นที่ติดตั้งในหน่วยความจำ

15 Number of keys

การกำหนดจำนวนชุดคีย์สำหรับแฟ้มข้อมูลดัชนี ขึ้นกับขนาดของแผ่นที่กำหนดให้แฟ้มในขณะนั้น จำนวนชุดคีย์กำหนดได้ไม่เกิน 12 ชุดคีย์เมื่อกำหนดให้ขนาดแผ่นเท่ากับ 512 ไบต์ และกำหนดได้มากที่สุด 24 ชุดคีย์เมื่อกำหนดให้ขนาดของแผ่นมากกว่า 512 ไบต์ขึ้นไป

16 Key position

เกิดจากความผิดพลาดในการกำหนดตำแหน่งเริ่มต้นของคีย์ในระเบียนข้อมูล การกำหนดตำแหน่งเริ่มต้นของคีย์ จะต้องคำนึงถึงขนาดของคีย์ที่ต้องกำหนดต่อมา คือ มี

ผลรวมที่ไม่เกินขนาดของระเบียนข้อมูลนั้น

17 Key length

การกำหนดความยาวของคีย์สำหรับแต่ละชุดคีย์ ผลรวมของความยาวแต่ละเซกเมนต์คีย์ภายในชุดคีย์นั้น จะต้องมีความยาวที่มากที่สุดไม่เกิน 255 ไบต์ ความยาวของคีย์จะต้องสัมพันธ์กับขนาดของแผ่น คือ เมื่อคำนวณจำนวนดัชนีจากความยาวของคีย์ภายในแผ่นดัชนี จะต้องมีจำนวนดัชนีอย่างน้อยที่สุด 4 ดัชนี ต่อ 1 แผ่นดัชนี

18 Key type

เกิดจากการกำหนดค่าแฟลกสำหรับระบุคุณสมบัติของคีย์ ไม่ตรงตามที่กำหนดให้

19 Binary key

สำหรับคีย์ที่มีคุณสมบัติเป็นข้อมูลเลขจำนวนเต็ม จะกำหนดความยาวคีย์ได้เท่ากับ 2 หรือ 4 ไบต์เท่านั้น และไม่สามารถกำหนดให้ใช้ชุดการสลับลำดับตัวอักษร

20 Inconsistent key flags

การกำหนดคุณสมบัติคีย์ให้มีค่าคีย์ที่ซ้ำกันได้ หรือ กำหนดให้สามารถเปลี่ยนแปลงค่าของคีย์ได้ ภายในชุดคีย์หนึ่ง ๆ จะต้องกำหนดคุณสมบัติดังกล่าวให้เหมือนกันทุกเซกเมนต์คีย์ของชุดคีย์นั้น

21 Invalid alternate character sequence definition

เกิดจากความผิดพลาดในการกำหนดการใช้ชุดการสลับลำดับตัวอักษร การกำหนดชื่อแฟ้มชุดการสลับลำดับตัวอักษร จะต้องกำหนดภายใน data buffer ซึ่งต่อจากข้อมูลของคุณสมบัติคีย์ ซึ่งต้องคั่นข้อมูลเหล่านั้นกับชื่อแฟ้มด้วยรหัสควบคุม ØØACh และชื่อของแฟ้มชุดการสลับลำดับตัวอักษร จะต้องปิดท้ายด้วยการเว้นช่องว่างอย่างน้อย 1 ช่อง หรือใช้รหัสแอสกี Ø แทน และข้อมูลตำแหน่งแรกภายในแฟ้มจะต้องมีรหัสควบคุม ØØACh กำกับอยู่

22 File not found

ไม่พบชื่อแฟ้มข้อมูลดัชนี ซึ่งระบุใน key buffer ภายในแผ่นจานบันทึก

23 Unrecoverable B-Tree file

เกิดจากฟังก์ชันเปิดแฟ้มข้อมูลดัชนี ตรวจพบ แฟ้มข้อมูลดัชนีนั้นมีการปิดแฟ้มที่ไม่สมบรูณ์ ซึ่งอาจเกิดจากการหยุดทำงานของคอมพิวเตอร์ ทำให้ฟังก์ชันเปิดแฟ้มต้องค้นหาแฟ้มสำรองแผ่น เพื่อนำข้อมูลเดิมบันทึกกลับลงในแฟ้มข้อมูลดัชนีนั้น แต่ค้นหาแฟ้มสำรองแผ่นนั้นไม่พบ

24 Close error

เกิดจากความผิดพลาดในการปิดแฟ้มข้อมูลดัชนี

25 Duplicated error

เกิดจากการเพิ่มเติมระเบียนข้อมูล ซึ่งมีค่าคีย์ซ้ำกับคีย์เดิมในแฟ้ม โดยที่ชุดคีย์นั้นไม่ได้กำหนดให้มีค่าคีย์ที่ซ้ำกันได้

26 Modifiable error

เกิดจากการแก้ไขระเบียนข้อมูล ซึ่งทำให้ค่าคีย์เดิมเปลี่ยนแปลงไป โดยที่ชุดคีย์นั้นไม่ได้กำหนดให้สามารถเปลี่ยนแปลงค่าคีย์ได้

27 Invalid key number

เกิดจากการกำหนดหมายเลขชุดคีย์ใน key number เกินจากจำนวนชุดคีย์ที่ในแฟ้มข้อมูลดัชนีนั้น หมายเลขชุดคีย์ที่ใช้ จะเริ่มตั้งแต่ 0 ถึง จำนวนชุดคีย์ - 1

28 Different key number

ในการกำหนดหมายเลขชุดคีย์ สำหรับฟังก์ชันการค้นหาระเบียนข้อมูลจากคีย์ลำดับก่อนหน้า หรือจากคีย์ลำดับถัดไป จะต้องเป็นหมายเลขชุดคีย์เดียวกันจากการใช้ฟังก์ชันค้นหาระเบียนข้อมูลในครั้งก่อน

29 Key not found

ค้นหาไม่พบระเบียนข้อมูล จากค่าคีย์ที่ระบุใน key buffer

30 End of file

เกิดจากการใช้ฟังก์ชันค้นหาและอ่านระเบียนข้อมูล กระทำไปจนถึงตำแหน่งปลายสุดของระเบียนข้อมูลในแฟ้ม จนไม่สามารถค้นหาต่อไปได้อีก

31 Invalid data address

เกิดจากตำแหน่งของระเบียนข้อมูลที่ระบุใน data buffer ซึ่งใช้ในฟังก์ชันการค้นหาและอ่านระเบียนข้อมูลจากตำแหน่งระเบียน เป็นตำแหน่งระเบียนที่ไม่ถูกต้อง

32 Record is out of range

เกิดจากหมายเลขระเบียนข้อมูลที่ระบุใน data buffer ซึ่งใช้ในฟังก์ชันการค้นหาและอ่านระเบียนข้อมูลจากหมายเลขระเบียน เป็นหมายเลขระเบียนที่ไม่ถูกต้อง

33 Directory error

ค้นหาไม่พบ path ที่กำหนดในชื่อแฟ้ม หรือไดเรคทอรีที่ระบุใน key buffer สำหรับฟังก์ชันการเปลี่ยนไดเรคทอรีปัจจุบันของแผ่นจานบันทึก

99 B-Tree inactive

ใช้กำหนดภายในโปรแกรมติดต่อ เมื่อตรวจไม่พบการติดตั้งฟังก์ชันในหน่วยความจำ

ภาคผนวก ค

โปรแกรมติดต่อ และ โปรแกรมตัวอย่าง

โปรแกรมติดต่อ

การพัฒนาโปรแกรมติดต่อ ได้แบ่งออกเป็นโปรแกรมติดต่อที่ใช้กับโปรแกรมประยุกต์ 3 ภาษา คือ

1. โปรแกรมติดต่อสำหรับภาษาไมโครซอฟท์ซี พัฒนาออกเป็น 2 แบบ คือ

1.1 โปรแกรม CXBTR.C พัฒนามาจากโปรแกรมภาษาซี สามารถใช้ร่วมกับโปรแกรมประยุกต์ภาษาซีได้ โดยใช้ #include <CXBTR.C> ภายในโปรแกรมนั้น ซึ่งทำให้ผ่านการแปลภาษาร่วมกัน จากตัวแปลภาษาไมโครซอฟท์ซี

1.2 โปรแกรม CXBTRA.ASM พัฒนามาจากโปรแกรมภาษาแอสเซมบลี ใช้ร่วมกับออบเจคโมดูล (Object Module) ที่มาจากการแปลภาษาของโปรแกรมประยุกต์ภาษาซี โดยการนำเชื่อมโยงกับออบเจคโมดูลที่มาจากการแปลโปรแกรม CXBTRA.ASM ในช่วงเวลาการเชื่อมโยง (Link Time)

2. โปรแกรมติดต่อสำหรับภาษาไมโครซอฟท์ฟอร์แทรน ได้แก่ โปรแกรม FORXBTR.ASM เป็นโปรแกรมที่พัฒนามาจากภาษาแอสเซมบลี และใช้เชื่อมโยงกับ ออบเจคโมดูลของโปรแกรมประยุกต์ภาษาฟอร์แทรน เช่นเดียวกับโปรแกรม CXBTRA.ASM

3. โปรแกรมติดต่อสำหรับภาษาไมโครซอฟท์ปาสคาล ได้แก่ โปรแกรม PASXBTR.ASM เป็นโปรแกรมที่พัฒนามาจากภาษาแอสเซมบลี และใช้เชื่อมโยงกับ ออบเจคโมดูลของโปรแกรมประยุกต์ภาษาปาสคาล เช่นเดียวกับโปรแกรม CXBTRA.ASM

```
/* ---------------------- CXBTR.C ------------------------- */
/*                                                         */
/*          B-Tree Interface Routine for C program         */
/*     status = XBTREE(int, char *, char *, char *, int);  */

#include <dos.h>

#undef    FP_OFF
#undef    FP_SEG
#define   FP_OFF(fp)   ((unsigned)(fp))
#define   FP_SEG(fp)   ((unsigned)((unsigned long)(fp) >> 16))
#define   MK_FP(seg,ofs) ((void far *) \
                    (((unsigned long)(seg) << 16) | (unsigned)(ofs)))

#define   FAR_PTR   0       /* ++++  use near pointer  ++++++ */

#define   B_VECT    0x7C    /* B-Tree interrupt number      */
#define   BTREE_ID  0xABCD  /* B-Tree Id.                   */
#define   B_INACT   99      /* B-Tree Inactive code         */

/*       Structure for Passing Parameters of B-Tree         */
struct B_PARMS
{  char * DAT_OFS;          /* Data Buffer offset           */
#if  !(FAR_PTR)
   unsigned int DAT_SEG;    /* Data Buffer segment          */
#endif
   char * PIB_OFS;          /* Position Info. Block offset  */
#if  !(FAR_PTR)
   unsigned int PIB_SEG;    /* Position Info. Block segment */
#endif
   char * FCB_OFS;          /* File Control Block offset    */
#if  !(FAR_PTR)
   unsigned int FCB_SEG;    /* File Control Block segment   */
#endif
   int  FUNCTION;           /* B-Tree Function Number       */
   char * KEY_OFS;          /* Key Buffer offset            */
#if  !(FAR_PTR)
   unsigned int KEY_SEG;    /* Key Buffer segment           */
#endif
   int  KEY_NUMBER;         /* Key Number                   */
   int  * STAT_OFS;         /* Status offset                */
#if  !(FAR_PTR)
   unsigned int STAT_SEG;   /* Status segment               */
#endif
   int  reverse;            /* Reservse area                */
} XARG;

XBTREE(OP, FIL_BLK, DAT_BUF, KEY_BUF, KEY_NUM)
int  OP;
char * FIL_BLK;
char * DAT_BUF;
char * KEY_BUF;
int  KEY_NUM;
```

```
{
   struct SREGS segregs;
   union  REGS  regs;
   unsigned int far *  B_ID;
   int  msgcode = 0;
   /*                                                  */
   /*      Check BTREE program has been installed      */
   /*                                                  */
   regs.h.ah = 0x35;              /*   Get Interrupt vector address   */
   regs.h.al = B_VECT;            /*   Interrupt vector number 7Ch    */
   intdosx(&regs,&regs,&segregs);
   B_ID = MK_FP(segregs.es,regs.x.bx - 2);
   if (*B_ID != BTREE_ID)         /*     check B-Tree Id.             */
      return(B_INACT);            /*     when B-Tree inactive         */

   /*                                                          */
   /*  if use near pointer, must read data segment DS for      */
   /*           default data segment of variables              */

#if  !(FAR_PTR)
   segread(&segregs);
   XARG.DAT_SEG = XARG.PIB_SEG = XARG.FCB_SEG = XARG.KEY_SEG =
       XARG.STAT_SEG = segregs.ds;
#endif
   XARG.FUNCTION = OP;
   XARG.FCB_OFS = FIL_BLK;
   XARG.PIB_OFS = FIL_BLK + 38;
   XARG.DAT_OFS = DAT_BUF;
   XARG.KEY_OFS = KEY_BUF;
   XARG.KEY_NUMBER = KEY_NUM;
   XARG.STAT_OFS = &msgcode;
   regs.x.dx = FP_OFF(&XARG);
   int86(B_VECT,&regs,&regs);     /*   Call B-Tree function           */
   return(msgcode);
}
```

```
;                   CXBTRA.ASM
;
;    B-Tree Interface Routine for  C program
;
;    status = XBTREE(OPERATION,FIL_BLK,DAT_BUF,KEY_BUF,KEY_NUM);
;
FAR_PTR      EQU     0      ; use near pointer
;
; Structure for Passing Parameters in XPARMS of B-Tree

B_PARMS      STRUC
DAT_OFS      DW      ?      ; Data Buffer offset
DAT_SEG      DW      ?      ; Data Buffer segment
PIB_OFS      DW      ?      ; Position Information Block offset
PIB_SEG      DW      ?      ; Position Information Block segment
FCB_OFS      DW      ?      ; File Control Block offset
FCB_SEG      DW      ?      ; File Control Block segment
FUNCTION     DW      ?      ; Function number
KEY_OFS      DW      ?      ; Key Buffer offset
KEY_SEG      DW      ?      ; Key Buffer segment
KEY_NUMBER   DW      ?      ; Key number
STAT_OFS     DW      ?      ; Status code offset
STAT_SEG     DW      ?      ; Status code segment
RESERVE      DW      ?      ; Reserve area
B_PARMS      ENDS

; Structure for accessing parameters on stack
PARMS        STRUC
             DW      ?      ; Pushed BP
             DW      ?      ; Short return address
op           DW      ?      ; Operation number
     IF  (FAR_PTR  EQ  0)
filblk       DW      ?      ; File information block
datbuf       DW      ?      ; Data buffer
keybuf       DW      ?      ; Key buffer
     ELSE
filblk       DD      ?      ; File information block
datbuf       DD      ?      ; Data buffer
keybuf       DD      ?      ; Key buffer
     ENDIF
keynum       DW      ?      ; Key number
PARMS        ENDS

B_ID         EQU     0ABCDh   ; B-Tree Id.

_DATA        SEGMENT WORD PUBLIC 'DATA'
XPARMS       DW      13 DUP(0)          ; define parameter block for B-Tree
_DATA        ENDS

DGROUP       GROUP   _DATA
             ASSUME  CS: _TEXT, DS: DGROUP
_TEXT        SEGMENT BYTE PUBLIC 'CODE'
             public  XBTREE
```

```
XBTREE      proc    near
            push    bp
            mov     bp,sp
            sub     sp,2            ; define local var. of status code
;
;     Check BTREE program has been installed
;
            mov     ax,357bh        ; get interrupt vector of B-Tree system
            int     21h
            cmp     word ptr es:[bx-2],B_ID ; check B-Tree resident ?
            jz      setPARMS
            mov     ax,99           ; if B-Tree Inactive
            jmp     endxbtr
;
setPARMS:                           ; setup parameter which passs to B-Tree
;
            lea     bx,XPARMS       ; load parameter block address
            mov     ax,[bp].op
            mov     [bx].FUNCTION,ax ; set Operation number
      IF  (FAR_PTR  EQ  0)
            mov     dx,[bp].keybuf
            mov     [bx].KEY_OFS,dx  ; load Key buffer offset
            mov     dx,[bp].datbuf
            mov     [bx].DAT_OFS,dx  ; load Data buffer offset
            mov     dx,[bp].filblk
            mov     [bx].FCB_OFS,dx  ; load FCB buffer address
            add     dx,38
            mov     [bx].PIB_OFS,dx  ; load PIB buffer address
            mov     ax,ds
            mov     [bx].FCB_SEG,ax
            mov     [bx].PIB_SEG,ax
            mov     [bx].DAT_SEG,ax
            mov     [bx].KEY_SEG,ax
      ELSE
            les     di,[bp].keybuf
            mov     [bx].KEY_OFS,di  ; load Key buffer offset
            mov     [bx].KEY_SEG,es  ;  and segment
            les     di,[bp].datbuf
            mov     [bx].DAT_OFS,di  ; load Data buffer offset
            mov     [bx].DAT_SEG,es  ;  and segment
            les     di,[bp].filblk
            mov     [bx].FCB_OFS,di  ; load FCB buffer address
            add     di,38
            mov     [bx].PIB_OFS,di  ;  and PIB buffer address
            mov     [bx].FCB_SEG,es
            mov     [bx].PIB_SEG,es
      ENDIF
            mov     ax,[bp].keynum
            mov     [bx].KEY_NUMBER,ax ; set Key number
            mov     ax,ss
            mov     [bx].STAT_OFS,sp  ; load status code address
            mov     [bx].STAT_SEG,ax
```

```
;-------------------------------------------------------------;
            lea     dx,XPARMS
            int     7Ch             ; Call B-Tree function
;-------------------------------------------------------------;
            mov     ax,[bp-2]       ; return status code
endxbtr:
            mov     sp,bp
            pop     bp
            ret                     ; return to caller
XBTREE      endp
_TEXT       ENDS
            END
```

```
;                   FORXBTR.ASM
;
;    B-Tree Interface routine for FORTRAN program
;
;    INTEGER *2 FUNCTION XBTREE(OP,FILBLK,DATBUF,KEYBUF,KEYNUM)
;
; Structure for Passing Parameters in XPARMS of B-Tree

B_PARMS      STRUC
DAT_OFS      DW      ?   ; Data buffer offset
DAT_SEG      DW      ?   ; Data buffer segment
PIB_OFS      DW      ?   ; Position Information Block offset
PIB_SEG      DW      ?   ; Position Information Block segment
FCB_OFS      DW      ?   ; File Control Block offset
FCB_SEG      DW      ?   ; File Control Block segment
FUNCTION     DW      ?   ; Function number
KEY_OFS      DW      ?   ; Key buffer offset
KEY_SEG      DW      ?   ; Key buffer segment
KEY_NUMBER   DW      ?   ; Key number
STAT_OFS     DW      ?   ; Status code offset
STAT_SEG     DW      ?   ; Status code segment
RESERVE      DW      ?   ; Reserve area
B_PARMS      ENDS

;    Structure for Passing Parameters on stack
PARMS        STRUC
             DW      ?   ; Pushed BP
             DD      ?   ; Long return address
keynum       DD      ?   ; Key number
keybuf       DD      ?   ; Key buffer
datbuf       DD      ?   ; Data buffer
filblk       DD      ?   ; File information block
op           DD      ?   ; Function number
PARMS        ENDS

B_ID         equ     0ABCDh   ; B-Tree Id.

DATA         SEGMENT PUBLIC 'DATA'
XPARMS       DW      13 DUP(0)    ; define parameter block for B-Tree
DATA         ENDS

DGROUP       GROUP   DATA
CODE         SEGMENT 'CODE'
             ASSUME  CS:CODE,DS:DGROUP,SS:DGROUP
             public  XBTREE
XBTREE       proc    far
             push    bp
             mov     bp,sp
             sub     sp,2              ; define local var. of status code
             mov     ax,357Ch          ; get interrupt vector of B-Tree system
             int     21h
             cmp     word ptr es:[bx-2],B_ID ; check B-Tree resident ?
             jz      setPARMS
```

```
            mov     ax,99                   ; if B-Tree Inactive
            jmp     endxbtr
setPARMS:                                   ; setup parameter which passs to B-Tree
            lea     bx,XPARMS               ; load parameter block address
            les     di,[bp].keynum
            mov     ax,es:[di]
            mov     [bx].KEY_NUMBER,ax      ; set Key number
            les     di,[bp].keybuf
            mov     [bx].KEY_OFS,di         ; load Key buffer offset
            mov     [bx].KEY_SEG,es         ;  and segment
            les     di,[bp].datbuf
            mov     [bx].DAT_OFS,di         ; load Data buffer offset
            mov     [bx].DAT_SEG,es         ;  and segment
            les     di,[bp].filblk
            mov     [bx].FCB_OFS,di         ; load FCB buffer address
            add     di,38
            mov     [bx].PIB_OFS,di         ;  and PIB buffer address
            mov     [bx].FCB_SEG,es
            mov     [bx].PIB_SEG,es
            les     di,[bp].op
            mov     ax,es:[di]
            mov     [bx].FUNCTION,ax        ; set Operation number
            mov     ax,ss
            mov     [bx].STAT_OFS,sp        ; load status code address
            mov     [bx].STAT_SEG,ax
;-----------------------------------------------------------------
            lea     dx,XPARMS
            int     7Ch                     ; Call B-Tree function
;-----------------------------------------------------------------
            mov     ax,[bp-2]               ; return status code
endxbtr:
            mov     sp,bp
            pop     bp
            ret     20      ; return to caller and pop 20 bytes of stack
XBTREE      endp
CODE        ENDS
            END
```

```
;                   PASXBTR.ASM
;
;    B-Tree Interface Routine for Pascal program
;
;    function XBTREE(      op     : integer;
;                      vars filblk : string;
;                      vars datbuf : string;
;                      vars keybuf : string;
;                           keynum : integer;) : integer; extern;
;
; Structure for Passing Parameters in XPARMS of B-TREE

B_PARMS      STRUC
DAT_OFS      DW      ?    ; Data buffer offset
DAT_SEG      DW      ?    ; Data buffer segment
PIB_OFS      DW      ?    ; Position Infomation Block offset
PIB_SEG      DW      ?    ; Position Infomation Block segment
FCB_OFS      DW      ?    ; File Control Block offset
FCB_SEG      DW      ?    ; File Control Block segment
FUNCTION     DW      ?    ; Function number
KEY_OFS      DW      ?    ; Key buffer offset
KEY_SEG      DW      ?    ; Key buffer segment
KEY_NUMBER   DW      ?    ; Key number
STAT_OFS     DW      ?    ; Status code offset
STAT_SEG     DW      ?    ; Status code segment
RESERVE      DW      ?    ; Reserve area
B_PARMS      ENDS

; Structure for accessing parameters on stack
PARMS        STRUC
             DW      ?    ; Pushed BP
             DD      ?    ; Long return address
keynum       DW      ?    ; Key number
keybuf       DD      ?    ; Key buffer
             DW      ?    ; Key buffer length
datbuf       DD      ?    ; Data buffer
             DW      ?    ; Data buffer length
filblk       DD      ?    ; File information block
             DW      ?    ; File information block length
op           DW      ?    ; Function number
PARMS        ENDS

B_ID         equ     0ABCDh   ; B-Tree Id.

DATA         SEGMENT PUBLIC 'DATA'
XPARMS       DW      13 DUP(0)    ; define parameter block for B-Tree
DATA         ENDS

DGROUP       GROUP   DATA
PASXBTR      SEGMENT 'CODE'
```

```
                ASSUME  CS:PASXBTR,DS:DGROUP,SS:DGROUP
                public  XBTREE
XBTREE          proc    far
                push    bp
                mov     bp,sp
                sub     sp,2            ; define local var. of status code
                mov     ax,357Ch        ; get interrupt vector of B-Tree system
                int     21h
                cmp     word ptr es:[bx-2],B_ID ; check B-Tree resident ?
                jz      setPARMS
                mov     ax,99                   ; if B-Tree Inactive
                jmp     endxbtr
setPARMS:                               ; setup parameter which passs to B-Tree
                lea     bx,XPARMS       ; load parameter block address
                mov     ax,[bp].keynum
                mov     [bx].KEY_NUMBER,ax ; set Key number
                les     di,[bp].keybuf
                mov     [bx].KEY_OFS,di ; load Key buffer offset
                mov     [bx].KEY_SEG,es ;  and segment
                les     di,[bp].datbuf
                mov     [bx].DAT_OFS,di ; load Data buffer offset
                mov     [bx].DAT_SEG,es ;  and segment
                les     di,[bp].filblk
                mov     [bx].FCB_OFS,di ; load FCB buffer address
                add     di,38
                mov     [bx].PIB_OFS,di ;  and PIB buffer address
                mov     [bx].FCB_SEG,es
                mov     [bx].PIB_SEG,es
                mov     ax,[bp].op
                mov     [bx].FUNCTION,ax ; set Operation number
                mov     ax,ss
                mov     [bx].STAT_OFS,sp ; load status code address
                mov     [bx].STAT_SEG,ax
;-------------------------------------------------------------$
                lea     dx,XPARMS
                int     7Ch             ; Call B-Tree function
;-------------------------------------------------------------$
                mov     ax,[bp-2]       ; return status code
endxbtr:
                mov     sp,bp
                pop     bp
                ret     22    ; return to caller and pop 22 bytes of stack
XBTREE          endp
PASXBTR         ENDS
                END
```

โปรแกรมตัวอย่าง

สำหรับโปรแกรมตัวอย่าง ประกอบด้วยโปรแกรมประยุกต์ที่พัฒนาเพื่อใช้ทดสอบการเรียกใช้ฟังก์ชันจัดการแฟ้มข้อมูลดัชนี มีทั้งหมด 7 โปรแกรม

โปรแกรมที่ 1 เป็นโปรแกรมที่พัฒนามาจากภาษาซี เพื่อทดสอบการสร้างแฟ้มข้อมูลดัชนี แฟ้มข้อมูลดัชนีที่ถูกสร้างขึ้นมีชื่อว่า "CREATE.TST" คุณสมบัติของแฟ้ม ประกอบด้วย

- ความยาวของระเบียนข้อมูล เท่ากับ 38 ไบต์
- ขนาดของแผ่นภายในแฟ้ม เท่ากับ 512 ไบต์
- จำนวนของคีย์ที่กำหนดภายในแฟ้ม เท่ากับ 2 คีย์

  คีย์ที่ 0 ประกอบด้วย
  - ตำแหน่งเริ่มต้นของคีย์ เท่ากับ 1
  - ความยาวของคีย์ เท่ากับ 4 ไบต์
  - คุณสมบัติของคีย์ที่ 0 คือ เป็น Modifiable Key

  คีย์ที่ 1 ประกอบด้วย
  - ตำแหน่งเริ่มต้นของคีย์ เท่ากับ 5
  - ความยาวของคีย์ เท่ากับ 20 ไบต์
  - คุณสมบัติของคีย์ที่ 1 คือ เป็น Duplicates Key และ เป็น Modifiable Key

โปรแกรมที่ 2 เป็นโปรแกรมที่พัฒนามาจากภาษาซี เพื่อใช้แสดงวิธีการเพิ่มเติม การลบ และแก้ไขระเบียนข้อมูลจากแฟ้มข้อมูลที่สร้างจากโปรแกรมที่ 1 โดยการลบ และการแก้ไขระเบียนข้อมูล จะต้องใช้ฟังก์ชันจัดการแฟ้มข้อมูลในการค้นหาระเบียนข้อมูลที่ต้องการ ก่อนทำการลบ หรือแก้ไขระเบียนข้อมูล

โปรแกรมที่ 3 เป็นโปรแกรมที่พัฒนามาจากภาษาฟอร์แทรน เพื่อแสดงการค้นหาระเบียนข้อมูล จากการเปรียบเทียบค่าคีย์ในลักษณะเงื่อนไขต่าง ๆ ได้แก่ การเปรียบเทียบคีย์ในลักษณะที่เท่ากัน ลักษณะที่มากกว่าหรือเท่ากับ ลักษณะที่มากกว่า ลักษณะที่น้อยกว่าหรือเท่ากับ และลักษณะที่น้อยกว่า โดยใช้แฟ้มข้อมูลที่มาจากโปรแกรมที่ 1

โปรแกรมที่ 4 เป็นโปรแกรมที่พัฒนามาจากภาษาฟอร์แทรน เพื่อแสดงระเบียนข้อมูลทั้งหมดที่มีในแฟ้มข้อมูล เรียงตามลำดับของคีย์ที่กำหนดภายในแฟ้มข้อมูล โดยการเรียกใช้ฟังก์ชันค้นหาระเบียนข้อมูลจากคีย์ที่อยู่ในลำดับถัดไป กับ การค้นหาจากคีย์ที่อยู่ในลำดับก่อนหน้า ซึ่งเริ่มต้นระเบียนข้อมูลแรกของแฟ้มข้อมูล จากการเรียกใช้การค้นหาระเบียนข้อมูลที่มีค่าคีย์น้อยที่สุด กับ การเริ่มต้นระเบียนข้อมูล จากการค้นหาระเบียนข้อมูลที่มีค่าคีย์มากที่สุดตามลำดับ

โปรแกรมที่ 5 เป็นโปรแกรมที่พัฒนามาจากภาษาปาสคาล เพื่อแสดงรายละเอียดของสถานภาพทั้งหมดที่มีอยู่ภายในแฟ้มข้อมูล ได้แก่ คุณสมบัติของแฟ้มข้อมูล คือ ความยาวของระเบียนข้อมูล ขนาดของแผ่น จำนวนของคีย์ที่กำหนดภายในแฟ้ม จำนวนของระเบียนข้อมูลทั้งหมด จำนวนของระเบียนข้อมูลที่ว่าง และจำนวนของแผ่นที่ว่าง กับ ลักษณะของคีย์ที่กำหนดทั้งหมด คือ ตำแหน่งเริ่มต้นของคีย์ ความยาวของคีย์ คุณสมบัติของคีย์ และจำนวนของดัชนีทั้งหมดของแต่ละคีย์

โปรแกรมที่ 6 เป็นโปรแกรมที่พัฒนามาจากภาษาปาสคาล เพื่อใช้ในการอ่านระเบียนข้อมูล จากตำแหน่งของระเบียนข้อมูลที่มีอยู่ในแฟ้มข้อมูลโดยตรง ทั้งนี้เพื่อใช้เป็นวิธีการกู้ข้อมูลจากแฟ้มข้อมูลดัชนีที่เสียหาย ซึ่งไม่สามารถเปิดใช้แฟ้มข้อมูลนั้นได้

โปรแกรมที่ 7 เป็นโปรแกรมที่พัฒนามาจากภาษาปาสคาล เพื่อใช้แสดงวิธีการสร้างแฟ้มดัชนีภายนอก(External Index File) และวิธีการนำแฟ้มดัชนีภายนอกมาใช้ในการค้นหาระเบียนข้อมูลจากแฟ้มข้อมูลเดิม แบ่งโปรแกรมออกเป็น 2 โปรแกรม คือ

โปรแกรมที่ 7.1 แสดงวิธีการสร้างแฟ้มดัชนีภายนอก เพื่อเพิ่มเติมการกำหนดคีย์ที่ใช้ในการค้นหาระเบียนข้อมูลที่มีในแฟ้มข้อมูลเดิม ภายในแฟ้มดัชนีภายนอก จะเป็นแฟ้มชนิดเดียวกับแฟ้มข้อมูลดัชนี ประกอบด้วย

- ความยาวของระเบียนข้อมูล เท่ากับ ความยาวของคีย์ชุดใหม่ บวกกับความยาวของตำแหน่งระเบียนข้อมูล ซึ่งเท่ากับ 4 ไบต์
- ขนาดของแผ่นภายในแฟ้ม เท่ากับ 512 ไบต์
- จำนวนของคีย์ เท่ากับ 1 คีย์

คีย์ที่ ø ประกอบด้วย

- ตำแหน่งเริ่มต้นของคีย์ ภายในระเบียนข้อมูลของแฟ้มดัชนีภายนอก ต้องเท่ากับ 5 เสมอ เพราะ ข้อมูล 4 ไบต์แรก จะเป็นตำแหน่งของระเบียนข้อมูล
- ความยาวของคีย์ เท่ากับ ความยาวของคีย์ชุดใหม่
- คุณสมบัติของคีย์ชุดใหม่ กำหนดให้คีย์เป็น Duplicates Key และ Modifiable Key

โปรแกรมที่ 7.2 แสดงวิธีการนำคีย์ชุดใหม่มาค้นหาตำแหน่งระเบียนข้อมูลจากแฟ้มดัชนีภายนอก เพื่อนำมาดึงระเบียนข้อมูลจากแฟ้มข้อมูลในภายหลัง

```
/*                    C Demo. Program #1                              */
/*   This program demonstrate how to use create B-Tree file function  */
/*                                                                    */

#define   Create    1
#define   DUP       1
#define   MOD       2

struct  key_spec
    {  int  key_pos;
       int  key_len;
       int  key_flag;
       char reserve[10];
    };

struct  file_spec
    {  int  rec_len;
       int  page_size;
       int  ndx_cnt;
       char reserve[8];
       int  dat_map;
       struct key_spec  key_buf[3];
    };

struct file_spec  spec_buf;
char fil_blk[128];
char file_name[20] = "create.tst";

main()
{  int status;

   printf("\n                    C Demo. Program #1\n");
   printf("  This Program illustrate how an application can use B-Tree\n");
   printf("  function to create B-Tree file,named CREATE.TST\n");
   printf("  Note :  This program will create B-Tree file which one record\n");
   printf("          has 4 fields. field #1 = 4 characters,\n");
   printf("                        field #2 = 20 characters,\n");
   printf("                        field #3 = 10 characters,\n");
   printf("                        field #4 = 4 characters\n");
   printf("          create 2 keys :\n");
   printf("  Key #0  Key position = 1, Key length = 4, Non-duplicated Key and\n");
   printf("          allow to modify key\n");
   printf("  Key #1  Key position = 5, Key length = 20, Duplicated Key and\n");
   printf("          allow to modify key\n");

   spec_buf.rec_len   = 38;                 /* Record Length          */
   spec_buf.page_size = 512;                /* Page Size              */
   spec_buf.ndx_cnt   = 2;                  /* Number of keys in file */
   spec_buf.key_buf[0].key_pos  = 1;        /* Key# 0 position        */
   spec_buf.key_buf[0].key_len  = 4;        /* Key# 0 length          */
```

```
   spec_buf.key_buf[0].key_flag = MOD;        /* Key# 0 flag                */
   spec_buf.key_buf[1].key_pos  = 5;          /* Key# 1 Segment 1 position */
   spec_buf.key_buf[1].key_len  = 20;         /* Key# 1 Segment 1 Length   */
   spec_buf.key_buf[1].key_flag = DUP | MOD;  /* Key# 1 Segment 1 flag     */

   status = XBTREE(Create,fil_blk,&spec_buf,file_name,0);
   if (status != 0)
      printf("Create file failed. status code = %d\n",status);
   else
      printf("Create file completed\n");
   exit(status);
}
```

```
/*                    C Demo. Program #2                              */
/*   This program demonstrate how to use insert, delete and           */
/*        update function.                                            */
/*                                                                    */

#define   Open       2     /* Open  operation number       */
#define   Close      3     /* Close operation number       */
#define   Insert     4     /* Insert operation number      */
#define   Delete     5     /* Delete operation number      */
#define   Update     6     /* Update operation number      */
#define   Get_GE    10     /* Get Greater than or Equal    */
#define   EOF_CODE   9     /* End of file error code       */

struct   DATA_FIELD
     {   char FIELD1[4];
         char FIELD2[20];
         char FIELD3[10];
         char FIELD4[4];
         char reserve[2];
     };

struct DATA_FIELD  DATA;
char  KEY_BUF[8];
int   status;
int   func;
char  file_name[20];
char  FIL_BLK[128];
char  ans;

main()
{
   printf("\n                    C Demo. Program #2\n");
   printf("  This Program illustrate how an application can use B-Tree\n");
   printf("  function to insert, delete, update data in B-Tree file.\n");
   printf("  Note :  This program will use B-Tree file which one record\n");
   printf("          has 4 fields. field #1 = 4 characters, field #2 = \n");
   printf("          20 characters, field #3 = 10 characters and \n");
   printf("          field #4 = 4 characters\n\n");
   printf(" First, Open file : ");
   gets(file_name);
   memset(&DATA,'\0',40);
   status = XBTREE(Open, FIL_BLK, &DATA, file_name, 0);
   if (status != 0)  {
      printf(" Open file failed. Status = %d\n",status);
      exit(status);
   }
   do  {
      printf("\n      1. Insert data\n");
      printf("      2. Delete data\n");
      printf("      3. Update data\n");
```

```
      printf("      4. Exit\n\n");
      printf("  choose number : ");
      func = getche(); printf("\n");
      switch (func)  {
         case '1' : INSB();
                    break;
         case '2' : DELB();
                    break;
         case '3' : UPDB();
                    break;
      }
   } while (func != '4');
   status = XBTREE(Close, FIL_BLK, &DATA, file_name, 0);
   if (status != 0)
      printf(" Close input file failed. Status = %d\n",status);
   exit(status);
}

INSB()
{
   printf("\n enter new data\n");
   printf("  field#1 [ 4 characters] : "); gets(DATA.FIELD1);
   printf("  field#2 [20 characters] : "); gets(DATA.FIELD2);
   printf("  field#3 [10 characters] : "); gets(DATA.FIELD3);
   printf("  field#4 [ 4 characters] : "); gets(DATA.FIELD4);
   status = XBTREE(Insert, FIL_BLK, &DATA, KEY_BUF, 0);
   if (status != 0)  {
      printf(" Error! Insert data record. Status = %d\n",status);
      exit(status);
   }
   else
     printf(" Insert data completed\n");
}

DELB()
{
   printf(" Before to delete data must use key to get data record.\n");
   printf(" enter key value [use field #1] : "); gets(KEY_BUF);
   status = XBTREE(Get_GE, FIL_BLK, &DATA, KEY_BUF, 0);
   if (status != 0)  {
      printf(" Error! Get Greater than or Equal key. Status = %d\n",status);
      exit(status);
   }
   printf(" Return data record that you want to delete.\n");
   printf(" data : %s  key : %s\n",&DATA,KEY_BUF);
   status = XBTREE(Delete, FIL_BLK, &DATA, KEY_BUF, 0);
   if (status != 0)  {
      printf(" Error! Delete data record. Status = %d\n",status);
      exit(status);
   }
```

```
   else
     printf(" Delete data completed\n");
}

UPDB()
{
   printf(" Before to update data must use key to get data record.\n");
   printf(" enter key value [use field #1] : "); gets(KEY_BUF);
   status = XBTREE(Get_GE, FIL_BLK, &DATA, KEY_BUF, 0);
   if (status != 0)  {
      printf(" Error! Get Greater than or Equal key. Status = %d\n",status);
      exit(status);
   }
   printf(" Return data record that you want to update.\n");
   printf(" data : %s  key : %s\n",&DATA,KEY_BUF);
   printf(" enter new field#1 [ 4 char.] : "); gets(DATA.FIELD1);
   printf(" enter new field#2 [20 char.] : "); gets(DATA.FIELD2);
   printf(" enter new field#3 [10 char.] : "); gets(DATA.FIELD3);
   printf(" enter new field#4 [ 4 char.] : "); gets(DATA.FIELD4);
   status = XBTREE(Update, FIL_BLK, &DATA, KEY_BUF, 0);
   if (status != 0)  {
      printf(" Error! Update data record. Status = %d\n",status);
      exit(status);
   }
   else  {
      printf(" Update data completed\n");
      printf(" Return the updated data : %s\n",&DATA);
   }
}
```

```
C             FORTRAN DEMO PROGRAM #3
C    This program demonstrates how to use the GET KEY function
C
C    Description :
C       1. Open B-TREE file
C       2. Enter KEY NUMBER to search
C       3. Enter KEY VALUE
C       4. Demo Call XBTREE FUNCTION to search
C          4.1 key that is EQUAL to KEY VALUE
C          4.2 key that is LESS THAN or EQUAL to KEY VALUE
C          4.3 key that is LESS THAN KEY VALUE
C          4.4 key that is GREATER THAN or EQUAL to KEY VALUE
C          4.5 key that is GREATER THAN KEY VALUE
C       5. Do loop to step 2 until NO KEY to search
C       6. Close B-TREE file
C
      PROGRAM  FGETKEY
      INTEGER *2 FILBLK(64)
      INTEGER *2 STAT, KEYNUM
      CHARACTER *20 KEY
      CHARACTER *20 KEYBUF
      CHARACTER *20 FNAME
      CHARACTER ANS
C
C  Declare function to interface to B-TREE function
C
      INTEGER *2 XBTREE
C
C  Declare COMMON memory area of data record and data buffer
C
      CHARACTER DATBUF(38)
      CHARACTER FLD1 *4, FLD2 *20, FLD3 *10, FLD4 *4
      EQUIVALENCE (FLD1,DATBUF(1)),(FLD2,DATBUF(5))
      EQUIVALENCE (FLD3,DATBUF(25)),(FLD4,DATBUF(35))
C
C  Format of OUTPUT
C
1     FORMAT(A)
2     FORMAT(I2)
5     FORMAT(20X,'FORTRAN Demo. Program #3'//
     -5X,'This Program demonstrate how to use get key function',/)
10    FORMAT(1X,'Open input file name : '\)
15    FORMAT(1X,'Enter Key# : '\)
20    FORMAT(1X,'Enter Key value : '\)
25    FORMAT(1X,'field#1 : ',A,'  field#2 : ',A,' field#3 : ',A,
     -' field#4 : ',A)
30    FORMAT(/,1X,'Search data that is EQUAL to key value')
31    FORMAT(/,1X,'Search data that LESS THAN OR EQUAL to key value')
32    FORMAT(/,1X,'Search data that LESS THAN key value')
33    FORMAT(/,1X,'Search data that GREATER THAN OR EQUAL to key value')
```

```
34    FORMAT(/,1X,'Search data that GREATER THAN key value')
40    FORMAT(1X,'Return key : ',A)
100   FORMAT(1X,'Open failed. CODE = ',I2)
105   FORMAT(5X,'Error! Get the EQUAL key. CODE = ',I2)
106   FORMAT(5X,'Error! Get the LESS THAN OR EQUAL key. CODE = ',I2)
107   FORMAT(5X,'Error! Get the LESS THAN key. CODE = ',I2)
108   FORMAT(5X,'Error! Get the GREATER THAN OR EQUAL key. CODE = ',I2)
109   FORMAT(5X,'Error! Get the GREATER THAN key. CODE = ',I2)
110   FORMAT(1X,'Error! Close file. CODE = ',I2)
C
C  Open B-TREE file using open mode = 2 (read data only)
C
      WRITE(*,5)
      WRITE(*,10)
      READ(*,1)FNAME
      STAT=XBTREE(2,FILBLK,DATBUF,FNAME,2)
      IF (STAT.NE.0) THEN
         WRITE(*,100)STAT
         STOP
      ENDIF
C  Start to search key and data
      ANS = ' '
1000  WRITE(*,15)
      READ(*,2)KEYNUM
      WRITE(*,20)
      READ(*,1)KEY
      KEYBUF = KEY
C  Perform get the data that key is EQUAL to key value
      STAT = XBTREE(7,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,105)STAT
      ELSE
         WRITE(*,30)
         WRITE(*,25)FLD1,FLD2,FLD3,FLD4
      ENDIF
      KEYBUF = KEY
C  Perform get the data that key is LESS THAN key value
      STAT = XBTREE(8,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,106)STAT
      ELSE
         WRITE(*,31)
         WRITE(*,25)FLD1,FLD2,FLD3,FLD4
         WRITE(*,40)KEYBUF
      ENDIF
      KEYBUF = KEY
C  Perform get the data that key is LESS THAN or EQUAL to key value
      STAT = XBTREE(9,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,107)STAT
```

```
      ELSE
         WRITE(*,32)
         WRITE(*,25)FLD1,FLD2,FLD3,FLD4
         WRITE(*,40)KEYBUF
      ENDIF
      KEYBUF = KEY
C  Perform get the data that key is GREATER THAN or EQUAL to key value
      STAT = XBTREE(10,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,108)STAT
      ELSE
         WRITE(*,33)
         WRITE(*,25)FLD1,FLD2,FLD3,FLD4
         WRITE(*,40)KEYBUF
      ENDIF
      KEYBUF = KEY
C  Perform get the data that key is GREATER THAN key value
      STAT = XBTREE(11,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,109)STAT
      ELSE
         WRITE(*,34)
         WRITE(*,25)FLD1,FLD2,FLD3,FLD4
         WRITE(*,40)KEYBUF
      ENDIF
C  Ask for some more search
      WRITE(*,'(A\)')' Search data again [Y/N]? '
      READ(*,'(A)')ANS
      IF (ANS.NE.'N') GOTO 1000
C  End process and close B-TREE file
      STAT=XBTREE(3,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,110)STAT
         STOP
      ELSE
         WRITE(*,'(/,A)')' END'
      ENDIF
      END
```

```
C                  FORTRAN DEMO PROGRAM #4
C  Listing data records in B-TREE file according to a given KEY NUMBER
C    - List ascending by get the LOWEST key and skip by get NEXT key
C    - List decending by get the HIGHEST key and skip by get PREVIOUS key
C
      PROGRAM  FKEYSKIP
      INTEGER *2 FILBLK(64)
      INTEGER *2 STAT, KEYNUM
      CHARACTER *20 KEYBUF
      CHARACTER *20 FNAME
      CHARACTER *2 ANS
C
C  Declare function to interface to B-TREE function
C
      INTEGER *2 XBTREE
C
C  Declare COMMON memory area of data record and data buffer
C
      CHARACTER DATBUF(38)
      CHARACTER FLD *4, FLD2 *20, FLD3 *10, FLD4 *4
      EQUIVALENCE (FLD1,DATBUF(1)),(FLD2,DATBUF(5))
      EQUIVALENCE (FLD3,DATBUF(25)),(FLD4,DATBUF(35))
C
C  FORMAT of output
C
1     FORMAT(A)
2     FORMAT(I2)
5     FORMAT(//20X,'FORTRAN Demo. Program #4'/,
     -5X,'This Program demonstrate how to use get lowest, highest,',
     -/5X,'previous, next key function'//)
6     FORMAT(1X,'Now, program will list data from Lowest to Highest')
7     FORMAT(/1X,'Now, program will list data from Highest to Lowest')
10    FORMAT(1X,'First, Open input file :- '\)
15    FORMAT(1X,'Use Key number : '\)
25    FORMAT(1X,'field#1 : ',A,' field#2 : ',A,' field#3 : ',A,
     -' field#4 : ',A)
30    FORMAT(/8X,'Press ENTER KEY to continue....'\)
100   FORMAT(1X,'Error! Open failed. Status = ',I2)
105   FORMAT(5X,'Error! Get data record from key. STATUS = ',I2)
110   FORMAT(1X,'Error! Close failed. Status = ',I2)
C
C  Open B-TREE file using open mode = 2 (read data only)
C
      WRITE(*,5)
      WRITE(*,10)
      READ(*,1)FNAME
      STAT=XBTREE(2,FILBLK,DATBUF,FNAME,2)
      IF (STAT.NE.0) THEN
         WRITE(*,100)STAT
         STOP
```

```
      ENDIF
C  Read KEY NUMBER
      WRITE(*,6)
      WRITE(*,15)
      READ(*,2)KEYNUM
C  Perform list data record from the lowest to the highest
C  Perform start at the lowest key by get the lowest key
      STAT = XBTREE(14,FILBLK,DATBUF,KEYBUF,KEYNUM)
1000  IF (STAT.EQ.30) GOTO 2000
      IF (STAT.NE.0) THEN
         WRITE(*,105)STAT
         STOP
      ENDIF
      WRITE(*,25)FLD1,FLD2,FLD3,FLD4
C  Perform skip next data record by use get NEXT key for skipping key
      STAT = XBTREE(13,FILBLK,DATBUF,KEYBUF,KEYNUM)
      GOTO 1000
2000  CONTINUE
C  Perform list data record from the highest to the lowest
      WRITE(*,30)
      READ(*,1)ANS
      WRITE(*,7)
      WRITE(*,15)
      READ(*,2)KEYNUM
C  Perform start at the highest key by get the highest key
      STAT = XBTREE(15,FILBLK,DATBUF,KEYBUF,KEYNUM)
2500  IF (STAT.EQ.30) GOTO 3000
      IF (STAT.NE.0) THEN
         WRITE(*,105)STAT
         STOP
      ENDIF
      WRITE(*,25)FLD1,FLD2,FLD3,FLD4
C  Perform skip down data record by use get PREVIOUS key for skipping key
      STAT = XBTREE(12,FILBLK,DATBUF,KEYBUF,KEYNUM)
      GOTO 2500
3000  CONTINUE
C  End process and close B-TREE file
      STAT=XBTREE(3,FILBLK,DATBUF,KEYBUF,KEYNUM)
      IF (STAT.NE.0) THEN
         WRITE(*,110)STAT
         STOP
      ELSE
         WRITE(*,'(/,A)')' END'
      ENDIF
      END
```

```
{                    Pascal Demo. Program #5                          }
{   This Program illustrate how application can use B-Tree function   }
{   to report the defined characteristics of a B-Tree file and        }
{   statistics on its content.                                        }
{                                                                     }
program  PSTAT(INPUT,OUTPUT);

const
     OPEN    = 2;       { Open operation          }
     CLOSE   = 3;       { Close operation         }
     GET_STAT= 20;      { Get status operation    }
     RDONLY  = 2;       { Read only open mode     }
     DUP_F   = 1;       { Duplicate flag          }
     MOD_F   = 2;       { Modifiable flag         }
     BIN_F   = 4;       { Binary flag             }
     SEG_F   = 8;       { Segment key flag        }
     ALT_F   = 16;      { Alternate character flag }
     THI_F   = 32;      { Thai character flag     }
type
     KEY_SPEC = record
       KEY_POS   : integer;
       KEY_LEN   : integer;
       KEY_FLAG  : integer;
       TOTAL_KEY : integer4;
       RESERVE   : string(6);
     end;

     FILE_SPEC  = record
       case integer of
        1 : (REC_LEN    : integer;
             PAGE_SIZE  : integer;
             NDX_CNT    : integer;
             TOTAL_REC  : integer4;
             FREE_REC   : integer;
             FREE_PAGE  : integer;
             DATMAP     : integer;
             KEY_SP     : array [0..24] of KEY_SPEC);
        2 : (BUFFER     : string(416));
       end;
var
     FILE_SP   : FILE_SPEC;
     ALT_NAM   : string(8);
     STATUS    : integer;
     FNAME1    : string(20);
     FILBLK    : string(128);
     i, j, numkey : integer;
     DMAP, DUPF, MODF, STRF, SEGF, ALTF, THIF : boolean;
     ans       : char;
{                                                                     }
{          External B-Tree interface function                         }
```

```
{                                                                    }
function XBTREE (       op      : integer;
                 vars   pos_blk : string;
                 vars   dat_buf : string;
                 vars   key_buf : string;
                        key_num : integer) : integer; extern;
begin
   writeln;
   writeln('              Pascal Demo. Program #5');
   writeln('This Program illustrate how application can use B-Tree function');
   writeln('to report the defined characteristics of a B-Tree file and');
   writeln('statistics on its content.');
   writeln;
   write('Enter file name :- ');
   readln(FNAME1);
   STATUS := XBTREE(OPEN, FILBLK, FILE_SP.BUFFER, FNAME1, RDONLY);
   if STATUS <> 0 then
     begin
       writeln('Open file failed. Status = ',STATUS:2);
       return;
     end;
   ALT_NAM := '        ';
   STATUS := XBTREE(GET_STAT, FILBLK, FILE_SP.BUFFER, ALT_NAM, 0);
   if STATUS <> 0 then
     begin
       writeln('Get file status failed. Status = ',STATUS:2);
       return;
     end;
   if FILE_SP.DATMAP = -1 then
      DMAP := TRUE;
   with FILE_SP do
    begin
      writeln('              Display file specification');
      writeln('Record length : ',REC_LEN:4,'    Page size : ',PAGE_SIZE:4,
         '    Number of key : ',NDX_CNT:2);
      writeln('Total records : ',TOTAL_REC:6);
      writeln('Number of free record : ',FREE_REC:4,
         '      Number of free page : ',FREE_PAGE:4);
    end;
   writeln('Use data map option : ',DMAP);
   writeln('Alternate character set name : ',ALT_NAM);
   writeln;
   writeln('                  Display key specification');
   writeln(' #  Position  Length  String  Duplicate  Modifiable',
      '  Alternate  Thai    Total');
   i := 0;
   j := 0;
   numkey := FILE_SP.NDX_CNT;
   while j < numkey do
    begin
```

```
      with FILE_SP do
      begin
       with KEY_SP[i] do
        begin
          DUPF := ((KEY_FLAG AND DUP_F) <> 0);
          MODF := ((KEY_FLAG AND MOD_F) <> 0);
          STRF := ((KEY_FLAG AND BIN_F) = 0);
          SEGF := ((KEY_FLAG AND SEG_F) <> 0);
          ALTF := ((KEY_FLAG AND ALT_F) <> 0);
          THIF := ((KEY_FLAG AND THI_F) <> 0);
          writeln(j:2,'    ',KEY_POS:4,'     ',KEY_LEN:3,'     ',STRF:5,
            '    ',DUPF:5,'       ',MODF:5,'      ',ALTF:5,'    ',THIF:5,
            '  ',TOTAL_KEY:6);
        end;
       end;
       if not SEGF then
          j := j + 1;
       i := i + 1;
     end;
    STATUS := XBTREE(CLOSE, FILBLK, FILE_SP.BUFFER, FNAME1, 0);
    if STATUS <> 0 then
      begin
       writeln('Close file failed. Status = ',STATUS:2);
       return;
      end;
end.
```

```
{                    Pascal Demo. Program #6                              }
{   This Program illustrates how step direct can retrieve data record in  }
{   the location physically folllowing the current record in the B-Tree    }
{   file.                                                                 }
{                                                                         }
program  PDMPREC(INPUT,OUTPUT);

const
     OPEN      = 2;     { Open operation              }
     CLOSE     = 3;     { Close operation             }
     STEPDIR   = 18;    { Step direct operation       }
     RDONLY    = 2;
     EOF_CODE  = 30;
type
     DATA_FIELD = record
       case integer of
        1 : (Field1   : string(4);
             Field2   : string(20);
             Field3   : string(10);
             Field4   : string(4));
        2 : (BUFFER   : string(38));
       end;
var
     DATA    : DATA_FIELD;
     KEY_BUF : string(8);
     i       : integer4;
     STATUS  : integer;
     FNAME1  : string(20);
     DATAPOS : string(128);
     ans     : char;

function XBTREE (       op      : integer;
                 vars   pos_blk : string;
                 vars   dat_buf : string;
                 vars   key_buf : string;
                        key_num : integer) : integer; extern;
begin
   writeln;
   writeln('                    Pascal Demo. Program #7');
   writeln('  This Program demonstrates how to use step direct function.');
   writeln('  Note :  This program must use B-Tree file that created with');
   writeln('          the use data map option.');
   writeln;
   write('First, Open input file :- ');
   readln(FNAME1);
   STATUS := XBTREE(OPEN, DATAPOS, DATA.BUFFER, FNAME1, RDONLY);
  if STATUS <> 0 then
     begin
       writeln('Open input file failed. Status = ',STATUS:2);
       return;
```

```
    end;
  writeln('Now, program will list data record that are physically ordered');
  write('   in B-Tree file.   Press ENTER Key to continue ......');
  read(ans);
  writeln;
  STATUS := XBTREE(STEPDIR, DATAPOS, DATA.BUFFER, KEY_BUF, 0);
  i := 1;
  while (STATUS <> EOF_CODE) do
    begin
      if STATUS <> 0 then
        begin
          writeln('Error! dumping file. Status = ',STATUS:2);
          return;
        end;
      with DATA do
         writeln('record# ',i:3,' : ',FIELD1,FIELD2,FIELD3,FIELD4);
      i := i + 1;
      STATUS := XBTREE(STEPDIR, DATAPOS, DATA.BUFFER, KEY_BUF, 0);
    end;
  writeln;
  writeln('              Process complete successfully');
  STATUS := XBTREE(CLOSE, DATAPOS, DATA.BUFFER, FNAME1, 0);
  if STATUS <> 0 then
    begin
      writeln('Close input file failed. Status = ',STATUS:2);
      return;
    end;
end.
```

```
{                       Pascal Demo. Program #7.1                          }
{   This Program illustrate how an application can use function 'get       }
{   position' to construct an external index file for an existing B-Tree   }
{   file. Once an external index file is created, the application can read }
{   the external index from lowest to highest and can use get direct to    }
{   sort the record in a B-Tree file by some field that was not originally }
{   defined as a key field                                                 }
{                                                                          }
program  PEXNDX(INPUT,OUTPUT);

const
     CREATE    = 1;     { Create operation                                 }
     OPEN      = 2;     { Open operation                                   }
     CLOSE     = 3;     { Close operation                                  }
     INSERT    = 4;     { Insert record operation                          }
     GET_LOW   = 14;    { Get the lowest key operation                     }
     GET_NXT   = 13;    { Get the next key operation                       }
     GET_POS   = 17;    { Get the physically location of data record       }
     RDWR_BAK  = 0;     { Open mode for read and write with backup file    }
     RDONLY    = 2;     { Open mode for read data only                     }
     EOF_CODE  = 30;    { End of file error code                           }
     DUP_F     = 1;
     MOD_F     = 2;
type
     KEY_SPEC = record
       KEY_POS  : integer;
       KEY_LEN  : integer;
       KEY_FLAG : integer;
       RESERVE  : string(10);
     end;

     FILE_SPEC  = record
       case integer of
        1 : (REC_LEN    : integer;
             PAGE_SIZE  : integer;
             NDX_CNT    : integer;
             RESERVE    : string(10);
             KEY_SP     : KEY_SPEC );
        2 : (BUFFER     : string(32));
       end;

     DATA_FIELD = record
       case integer of
        1 : (Field1   : string(4);
             Field2   : string(20);
             Field3   : string(10);
             Field4   : string(4));
        2 : (DAT_POS  : string(4));
        3 : (BUFFER   : string(38));
       end;
```

```
     NDX_FIELD  = record
       case integer of
        1 : (NDX_POS : string(4);
             NDX_KEY : string(4));
        2 : (BUFFER  : string(8));
       end;
var
     NDX_SP  : FILE_SPEC;
     DATA    : DATA_FIELD;
     NDX     : NDX_FIELD;
     KEY_BUF : string(8);
     KEY_BUF2 : string(8);
     STATUS  : integer;
     FNAME1, FNAME2  : string(20);
     INDEXPOS, DATAPOS : string(128);
     ans     : char;
{                                                              }
{         External B-Tree interface function                   }
{                                                              }
function XBTREE (       op      : integer;
                 vars  pos_blk : string;
                 vars  dat_buf : string;
                 vars  key_buf : string;
                       key_num : integer) : integer; extern;
begin
   writeln;
   writeln('                Pascal Demo. Program #6.1');
   writeln('  This program illustrate how an application can apply get');
   writeln('  position and get direct to manipulate data of original');
   writeln('  B-Tree file for constructing the external index file.');
   writeln;
   write('First, create external index file :- ');
   readln(FNAME1);
   NDX_SP.REC_LEN   := 8;
   NDX_SP.PAGE_SIZE := 512;
   NDX_SP.NDX_CNT   := 1;
   with NDX_SP do
     begin
       KEY_SP.KEY_POS  := 5;
       KEY_SP.KEY_LEN  := 4;
       KEY_SP.KEY_FLAG := DUP_F + MOD_F;
     end;
   STATUS := XBTREE(CREATE, INDEXPOS, NDX_SP.BUFFER, FNAME1, 0);
   if STATUS <> 0 then
     begin
       writeln('Create external index file failed. Status = ',STATUS:2);
       return;
     end;
   STATUS := XBTREE(OPEN, INDEXPOS, NDX.BUFFER, FNAME1, RDWR_BAK);
```

```
if STATUS <> 0 then
  begin
    writeln('Open external index file failed. Status = ',STATUS:2);
    return;
  end;
writeln('Create and open external index file complete');
writeln; write('Next, open the original B-Tree file :- ');
readln(FNAME2);
STATUS := XBTREE(OPEN, DATAPOS, DATA.BUFFER, FNAME2, RDONLY);
if STATUS <> 0 then
  begin
    writeln('Open input file failed. Status = ',STATUS:2);
    return;
  end;
writeln('Open original B-Tree file complete');
writeln;
writeln('Now, program will read and get position of data from the');
writeln('  existiong B-Tree file and insert the new index to the');
writeln('  external index file.');
writeln;
writeln('       Please wait .......');
STATUS := XBTREE(GET_LOW, DATAPOS, DATA.BUFFER, KEY_BUF, 0);
while (STATUS <> EOF_CODE) do
  begin
     if STATUS <> 0 then
       begin
          writeln('Error! reading file. Status = ',STATUS:2);
          return;
       end;
     NDX.NDX_KEY := DATA.FIELD4;
     STATUS := XBTREE(GET_POS, DATAPOS, DATA.BUFFER, KEY_BUF, 0);
     NDX.NDX_POS := DATA.DAT_POS;
     STATUS := XBTREE(INSERT, INDEXPOS, NDX.BUFFER, KEY_BUF2, 0);
     if STATUS <> 0 then
       begin
          writeln('Error! inserting record. Status = ',STATUS:2);
          return;
       end;
     STATUS := XBTREE(GET_NXT, DATAPOS, DATA.BUFFER, KEY_BUF, 0);
  end;
writeln; writeln('       Process complete successfully');
STATUS := XBTREE(CLOSE, INDEXPOS, NDX.BUFFER, FNAME1, 0);
if STATUS <> 0 then
  begin
    writeln('Close external index file failed. Status = ',STATUS:2);
    return;
  end;
STATUS := XBTREE(CLOSE, DATAPOS, DATA.BUFFER, FNAME2, 0);
if STATUS <> 0 then
  begin
```

```
        writeln('Close input file failed. Status = ',STATUS:2);
        return;
      end;
end.
```

```
{                       Pascal Demo. Program #7.2                          }
{   This Program illustrate how an application can read the external index }
{   from lowest to highest and can use get direct to sort the record in a  }
{   B-Tree file by some field that was not originally defined as a key     }
{   field .                                                                }
{                                                                          }
program  PEXNDX(INPUT,OUTPUT);

const
     OPEN      = 2;     { Open operation                                   }
     CLOSE     = 3;     { Close operation                                  }
     GET_LOW   = 14;    { Get the lowest key operation                     }
     GET_NXT   = 13;    { Get the next key operation                       }
     GET_DIRECT= 16;    { Get direct to the record by data position        }
     GET_POS   = 17;    { Get the physically location of data record       }
     RDWR_BAK  = 0;     { Open mode for read and write with backup file    }
     RDONLY    = 2;     { Open mode for read data only                     }
     EOF_CODE  = 30;    { End of file error code                           }
type
     DATA_FIELD = record
       case integer of
        1 : (Field1   : string(4);
             Field2   : string(20);
             Field3   : string(10);
             Field4   : string(4));
        2 : (DAT_POS  : string(4));
        3 : (BUFFER   : string(38));
       end;

     NDX_FIELD  = record
       case integer of
        1 : (NDX_POS : string(4);
             NDX_KEY : string(4));
        2 : (BUFFER  : string(8));
       end;
var
     DATA    : DATA_FIELD;
     NDX     : NDX_FIELD;
     KEY_BUF : string(8);
     KEY_BUF2 : string(8);
     STATUS  : integer;
     FNAME1, FNAME2  : string(20);
     INDEXPOS, DATAPOS : string(128);
     ans     : char;
{                                                                          }
{        External B-Tree interface function                               }
{                                                                          }
function XBTREE (       op      : integer;
                 vars   pos_blk : string;
                 vars   dat_buf : string;
```

```
                vars  key_buf : string;
                      key_num : integer) : integer; extern;
begin
   writeln;
   writeln('                  Pascal Demo. Program #6.2');
   writeln('  This program illustrate how an application can apply get');
   writeln('  position and get direct to manipulate data of original');
   writeln('  B-Tree file for constructing the external index file.');
   writeln;
   write('First, open external index file :- ');
   readln(FNAME1);
   STATUS := XBTREE(OPEN, INDEXPOS, NDX.BUFFER, FNAME1, RDWR_BAK);
   if STATUS <> 0 then
     begin
       writeln('Open external index file failed. Status = ',STATUS:2);
       return;
     end;
   writeln('Open external index file complete');
   writeln; write('Next, open the original B-Tree file :- ');
   readln(FNAME2);
   STATUS := XBTREE(OPEN, DATAPOS, DATA.BUFFER, FNAME2, RDONLY);
   if STATUS <> 0 then
     begin
       writeln('Open input file failed. Status = ',STATUS:2);
       return;
     end;
   writeln('Open original B-Tree file complete');
   writeln;
   writeln('Now, program will read index from external index file and');
   writeln('  use get direct to retrieve data of the original B-Tree');
   writeln('  file.  Press ENTER Key to continue.....');
   read(ans);
   writeln;
   STATUS := XBTREE(GET_LOW, INDEXPOS, NDX.BUFFER, KEY_BUF, 0);
   while (STATUS <> EOF_CODE) do
     begin
       if STATUS <> 0 then
         begin
           writeln('Error! reading index file. Status = ',STATUS:2);
           return;
         end;
       DATA.DAT_POS := NDX.NDX_POS;
       STATUS := XBTREE(GET_DIRECT, DATAPOS, DATA.BUFFER, KEY_BUF2, 0);
       if STATUS <> 0 then
         begin
           writeln('Error! reading data record. Status = ',STATUS:2);
           return;
         end;
       with DATA do
          writeln('Data record : ',FIELD1,FIELD2,FIELD3,FIELD4);
```

```
        STATUS := XBTREE(GET_NXT, INDEXPOS, NDX.BUFFER, KEY_BUF, 0);
      end;
    STATUS := XBTREE(CLOSE, INDEXPOS, NDX.BUFFER, FNAME1, 0);
    if STATUS <> 0 then
      begin
        writeln('Close external index file failed. Status = ',STATUS:2);
        return;
      end;
    STATUS := XBTREE(CLOSE, DATAPOS, DATA.BUFFER, FNAME2, 0);
    if STATUS <> 0 then
      begin
        writeln('Close input file failed. Status = ',STATUS:2);
        return;
      end;
end.
```

ประวัติผู้เขียนวิทยานิพนธ์

นายพงศ์เอก เลิศทรัพย์เจริญ เกิดเมื่อวันที่ 26 พฤศจิกายน พ.ศ. 2504 ที่กรุงเทพมหานคร สำเร็จการศึกษาวิทยาศาสตร์บัณฑิต (วิทยาศาสตร์ทางทะเล) จากคณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2526 และในปีการศึกษา 2528 เข้าศึกษาระดับปริญญามหาบัณฑิต ในสาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย