ปีการศึกษา 2533

โปรแกรมเขียนแบบอย่างง่าย

โดย

นาย ฐิตะพล หุยะนันท์

อาจารย์ที่ปรึกษา

อาจารย์ วราคม เนิดน้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

โปรแกรมเขียนแบบอย่างง่าย

โดย

นาย ฐิตะพล หุยะนันท์ รหัสประจำตัว 30.1067

อาจารย์ที่ปรึกษา

อาจารย์ วราคม เนิดน้อย

เลขหมู่ T. 33115 ฐ 3
เลขทะเบียน 027948
วัน, ด. น ปี 18 ก.ค. 2534

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมศาสตร์เครื่องกล
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
ปีการศึกษา 2533

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
027948

สารบัญ

บทที่ 1 บทนำ 1

บทที่ 2 แนวทางการออกแบบโปรแกรมเชิงแบบอย่างง่าย 3

บทที่ 3 Computer User Interface Routine 5

3.1) การรับข้อมูลอักขระและตัวเลขโดยตรง 5

3.2) การรับข้อมูลของตำแหน่งโดยใช้วิธี Cursor Feedback 7

3.3) การควบคุมขั้นตอนในการรับข้อมูล 9

บทที่ 4 Drawing Routine 12

4.1) การวาดเส้นตรง 12

4.2) การวาดเส้นโค้ง 12

4.3) การวาดวงกลม 14

บทที่ 5 Instruction Processing Routine 15

5.1) หลักการทำงานทั่วไปของคำสั่ง 15

5.2) การทำงานของคำสั่งในโปรแกรมเชิงแบบอย่างง่าย 16

บทที่ 6 Data Processing Routine 25

6.1) โครงสร้างข้อมูล 25

6.2) หลักการทำงานของการจัดการข้อมูล 28

6.3) การทำงานของการจัดการเบื้องต้น 28

บทที่ 7 Main Menu Program 33

บทที่ 8 บทสรุปและวิจารณ์ 35

กิตติกรรมประกาศ

เอกสารอ้างอิง

ภาคผนวก

- ภาคผนวก ก: *Link List*
- ภาคผนวก ข: *โปรแกรมตัวอย่าง*
- ภาคผนวก ค: *Scan Code และ รหัส ASCII*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมเขียนแบบอย่างง่าย

ฐิตะพล หุยะนันท์
วราคม เนิดน้อย อาจารย์ที่ปรึกษา
ปีการศึกษา 2533

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เรียบเรียงขึ้นจากโปรแกรมที่ได้พัฒนาขึ้นมาใช้สำหรับการเขียนแบบอย่างง่าย โดยการรับข้อมูลเข้าจากทางแป้นพิมพ์(Keyboard),แสดงผลและสิ่งที่ต้องการผ่านทางจอภาพ(Display) ซึ่งทั้งสองส่วนนี้สามารถติดต่อกับผู้ใช้ภายใต้การควบคุมของ โปรแกรมสื่อกลางติดต่อระหว่างคอมพิวเตอร์กับผู้ใช้ (Computer - User Interface Routine) และส่วนแสดงผลที่เป็นรูปภาพนั้นจะอยู่ภายใต้การทำงานของ โปรแกรมวาดภาพ(Drawing Routine) โดยที่ โปรแกรมประเมินผลคำสั่ง(Instruction Processing Routine) จะทำหน้าที่ในการประเมินหา ข้อมูลรูปภาพพื้นฐาน(Basic Graphics Data) ตามชนิดของภาพ ซึ่งจะมี แบบกำหนดของภาพ(Graphics Format) ที่ต่างกันออกไป และเพื่อความสะดวกรวดเร็วในการกู้ข้อมูลเหล่านี้ จึงจำเป็นที่ต้องจัดการภายใน หน่วยความจำ(Memory) ในรูปของ แฟ้มข้อมูลรูปภาพชั่วคราว (Temperary Drawing File) เสียก่อนที่จะนำลงไปเก็บไว้ใน จานบันทึกข้อมูล(Diskette) ต่อไป ซึ่งขั้นตอนและกระบวนการในการจัดการข้อมูลเหล่านี้จะอยู่ภายใต้การทำงานของ โปรแกรมประเมินผลข้อมูล(Data Processing Routine) โดยการทำงานของทุกโปรแกรมที่ได้กล่าวมาแล้วนั้นอยู่ภายใต้การจัดการและควบคุมขั้นตอนการทำงานของ โปรแกรมเมนูหลัก(Main Menu Program)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Simple CAD Program

Thitaphol Huyanan

Warakhom Nerdnoi Advisor

Abstract

This thesis concerns a study and design of a two - dimensional picture drawing computer program which is used as a basis for a research concerning CAD (Computer - Aided Design or Computer - Aided Drifting). The program is devided into five subroutines. The first, the Drawing routine, is used to draw a picture on the computer screen. The second, the Computer - User Interface routine, is used to manage the information inputted through the keyboard. The third, the Instruction Processing routine, is used to translation and carry out commands. The fourth, the Data Processing routine, is used to store data on a diskette or temporarily in the memory. Finally, the Main Menu routine is used to control all prior subroutines. All programs are written in BASIC utilizing the program Turbo Basic of Borland International, Inc. . It is found that the efficiency of the program depands on the detail structure and language used to manage data. The problems of picture drawing and data storage are considered in detail.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในปัจจุบันได้มีการนำโปแกรมที่ช่วยในการเขียนแบบประเภท CAD ที่ย่อมาจาก Computer Aided Design และ/หรือ Computer Aided Drifting เข้ามาใช้อย่างกว้างขวางในสาขาต่างๆ เช่น วิศวกรรมศาสตร์, สถาปัตยกรรมศาสตร์และ การโฆษณาเป็นต้น โดยโปรแกรมสำเร็จรูปเหล่านี้ได้ถูกค้นคิดและพัฒนามาจากต่างประเทศ จึงเป็นการยากที่นำมาพัฒนาต่อเพื่อที่จะขยายขอบเขตและลักษณะการทำงานในด้านอื่นต่อไป ทั้งยังต้องเสียเงินตราไปเป็นจำนวนมากเพื่อจะได้มาซึ่งโปรแกรมประเภทนี้

ด้วยเหตุผลดังที่กล่าวมาแล้วนั้น ได้เป็นสิ่งจูงใจที่จะทำการศึกษาและพัฒนาโปรแกรมประเภทขึ้นมาเอง และจากคำแนะนำของอาจารย์ที่ปรึกษา ทำให้ผู้จัดทำได้ค้นคิดโปรแกรมเขียนแบบอย่างง่ายขึ้น เพื่อที่จะใช้เป็นต้นแบบในการพัฒนาขั้นต่อๆไปของโปรแกรมประเภทนี้ ซึ่งจะก่อให้เกิดการพึ่งตนเองทางเทคโนโลยีด้านนี้ต่อไปในอนาคต รวมทั้งการนำข้อมูลรูปภาพไปใช้ประโยชน์ในด้านอื่นๆ เช่น นำไปใช้เป็นข้อมูลในการควบคุม CNC เป็นต้น

ดังนั้นเนื้อหาในปริญญานิพนธ์ฉบับนี้ จะเป็นปูพื้นความรู้เพื่อทำความเข้าใจในเรื่องโปรแกรมเขียนแบบอย่างง่าย ตั้งแต่หลักการเบื้องต้น ไปจนถึงสิ่งที่ควรจะพัฒนาแก้ไขเพิ่มเติม ดังลำดับที่แสดงไว้ข้างล่างนี้ะ

บทที่ 2: แนวทางการออกแบบโปรแกรมเขียนแบบอย่างง่าย. เป็นการอธิบายถึงลักษณะโครงสร้างโปรแกรมที่ได้กำหนดขึ้น รวมทั้งลักษณะทิศทางการไหลของข้อมูลระหว่างส่วนต่างๆของโปรแกรม

บทที่ 3: Computer User Interface Routine. จะอธิบายการรับข้อมูลต่างๆ จากแป้นพิมพ์ รวมทั้งขั้นตอนและวิธีการควบคุมการรับข้อมูลที่ได้จากการติดต่อกับผู้ใช้ อีกทั้งยังจะกล่าวถึงโปรแกรมอรรถประโยชน์ต่างๆ ที่นำมาใช้ร่วมด้วย

บทที่ 4: Drawing Routine. เป็นการแสดงหลักการและขั้นตอนในการวาดเส้นตรง, ส่วนโค้งของวงกลม และ การวาดวงกลม

บทที่ 5: Instruction Processing Routine. อธิบายถึงหลักการทำงานทั่วไปของคำสั่ง และการทำงานของคำสั่งในโปรแกรมเขียนแบบอย่างง่าย

บทที่ 6: Data Processing Routine. จะให้นิยามของประเภทของข้อมูล และอธิบายโครงสร้างข้อมูล รวมทั้งหลักการทำงานของการจัดการข้อมูล ทั้งยังพูดถึงการทำงานของการจัดการเบื้องต้นที่ได้เขียนขึ้นมาเพื่อใช้กับโปรแกรมเขียนแบบอย่างง่ายนี้

บทที่ 7: Main Menu Program. จะอธิบายหลักการและขั้นตอนการควบคุมการทำงานของโปรแกรมทั้งหมด

บทที่ 8: บทสรุปและวิจารณ์. จะเป็นการกล่าวถึงผลของการเขียนโปรแกรมทั้งหมดว่าเป็นเช่นไร มีปัญหาและอุปสรรคใดเกิดขึ้นมาบ้าง รวมทั้งวิธีแก้ไขที่ได้ทำไปแล้ว

บทที่ 9: บทส่งท้าย. จะเป็นการแนะนำสิ่งที่ควรทำเพิ่มเติมหรือแก้ไข เพื่อประโยชน์ในการพัฒนาต่อไปในอนาคตข้างหน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ในการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และเพื่อความเข้าที่ถ่องแท้ จึงได้มีโปรแกรมประกอบดังที่แสดงไว้ในภาคผนวก ข. ซึ่งโปรแกรมเหล่า
ได้สร้างและพัฒนาขึ้นด้วย BASIC Compiler ที่ชื่อ Turbo Basic 1.0 ของ Borland International
ายใต้การทำงานของ MS.DOS version 4.01 บนเครื่อง IBM PC AT และ Compatible ที่มีระบบการ
สดงผลเป็น EGA(Enhanced Graphics Adapter) และ VGA(Veido Graphics Array)เท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

แนวทางการออกแบบโปรแกรมเขียนแบบอย่างง่าย

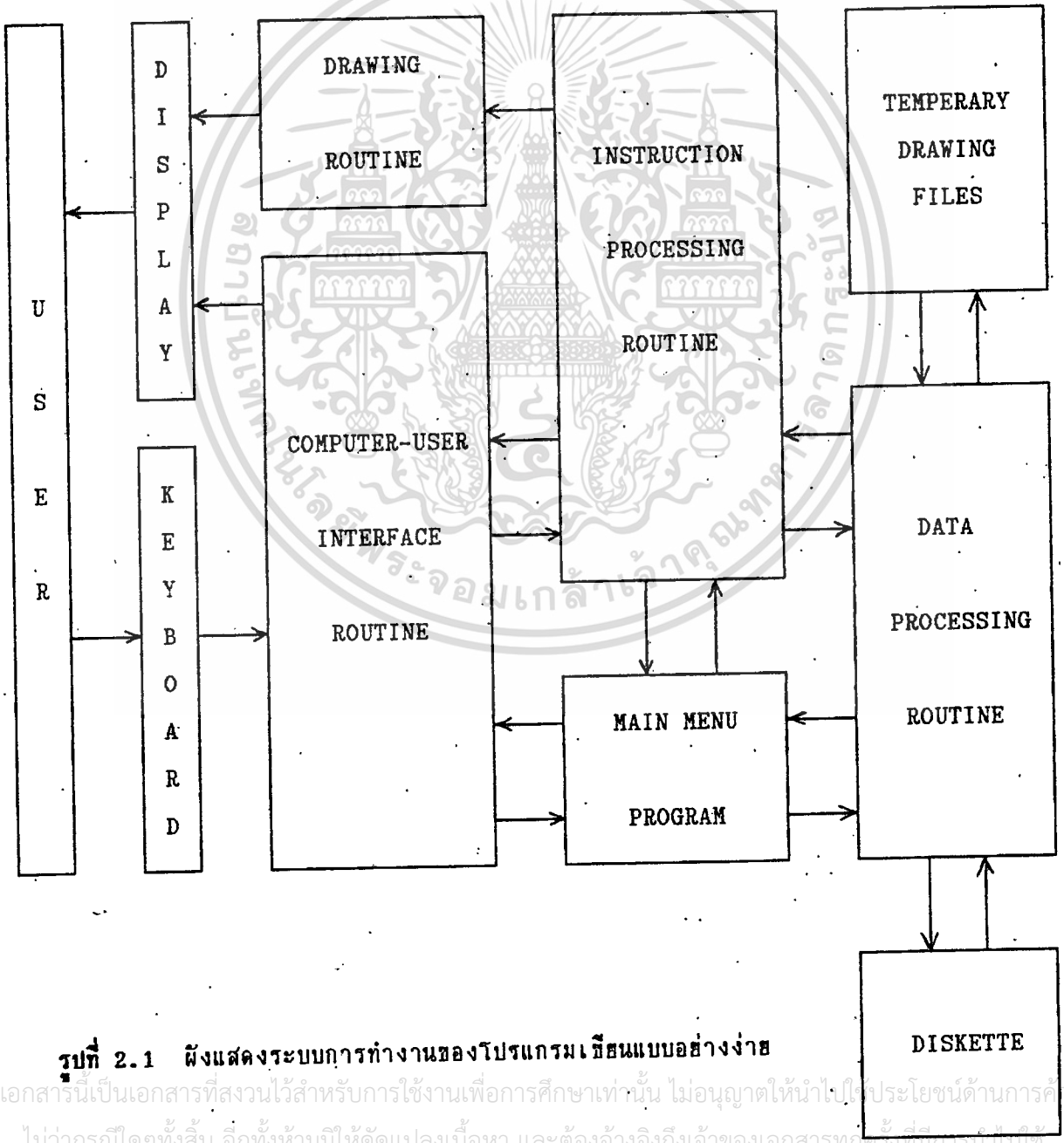
เพื่อความเข้าใจที่ถูกต้องโปรดพิจารณารูปที่ 2.1 จะเห็นว่า ระบบการทำงานของโปรแกรมเขียนแบบอย่างง่ายนั้น ประกอบด้วย โมดูล(module) ต่างๆ รวมกันทั้งหมด 10 โมดูล ซึ่งสามารถจำแนกออกเป็นกลุ่มใหญ่ๆ ตามลักษณะของโมดูลนั้นๆ ได้เป็น 3 กลุ่ม ดังนี้

กลุ่มที่ 1. กลุ่มผู้ใช้: User Module

กลุ่มที่ 2. กลุ่ม Hardware: ในกลุ่มนี้ประกอบด้วย

-ส่วนอุปกรณ์รับส่งข้อมูล: Keyboard Module, Display Module และ Diskette Module

-ส่วนหน่วยความจำ: Temperary Drawing Files Module



รูปที่ 2.1 ผังแสดงระบบการทำงานของโปรแกรมเขียนแบบอย่างง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กลุ่มที่ 3. กลุ่ม Software: ในกลุ่มนี้ประกอบด้วย

-ส่วนติดต่อกับผู้ใช้: Computer-User Interface Routine Module

-ส่วนแสดงผล: Drawing Routine Module

-ส่วนประเมินผล: Instruction Processing Routine Module

-ส่วนจัดการข้อมูล: Data Processing Routine Module

-ส่วนควบคุมโปรแกรม: Main Menu Program Module

โดยในกลุ่มที่ 3 นี้เองที่จะเป็นส่วนของ โปรแกรมเชียนแบบอย่างง่าย และสามารถกำหนดการทำงานในแต่ละโมดูลได้จาก"ทิศทางการไหลของข้อมูล" ที่กำหนดไว้ในรูปที่ 2.1 เช่นกัน โดยสามารถอธิบายภาครวมของการทำงานของโมดูลในกลุ่มที่ 3 ร่วมกับกลุ่มอื่น ได้ดังนี้:

เมื่อ User พิมพ์คำสั่งด้วย Keyboard ผ่าน Command Prompt ที่ Main Menu Program ได้สั่งให้ Computer - User Interface Routine แจ้ง User ผ่าน Display ให้ทราบว่า โปรแกรมพร้อมจะรับคำสั่งแล้ว และเมื่อ Main Menu Program ได้รับคำสั่งจาก Computer - User Interface Routine ก็จะส่งการทำงานไปยัง Instruction Processing Routine เพื่อทำงานตามขั้นตอนของคำสั่งนั้นๆ โดยจะทำการแจ้งและรับข้อมูลที่ต้องการจาก User ผ่านทาง Computer - User Interface Routine หรือรับข้อมูลจาก Data Processing Routine และเมื่อประเมินผลเรียบร้อยแล้ว ก็จะส่งผลข้อมูลไปยังที่ Drawing Routine เพื่อแสดงภาพบน Display และก็จะส่งข้อมูลไปที่ Data Processing Routine ด้วย เพื่อที่จัดการข้อมูลเหล่านี้ใน Temperary Drawing Files แล้วจะส่งการทำงานกลับไปยัง Main Menu Program และเมื่อ Main Menu Program สั่งการทำงานเกี่ยวกับการจัดการข้อมูลใน Diskette ให้แก่ Data Processing Routine แล้วก็จะเริ่มทำการถ่ายเทข้อมูลระหว่าง Temperary Drawing Files กับ Diskette และเมื่อการแลกเปลี่ยนเสร็จสิ้น ก็จะส่งการทำงานกลับไปยัง Main Menu Program

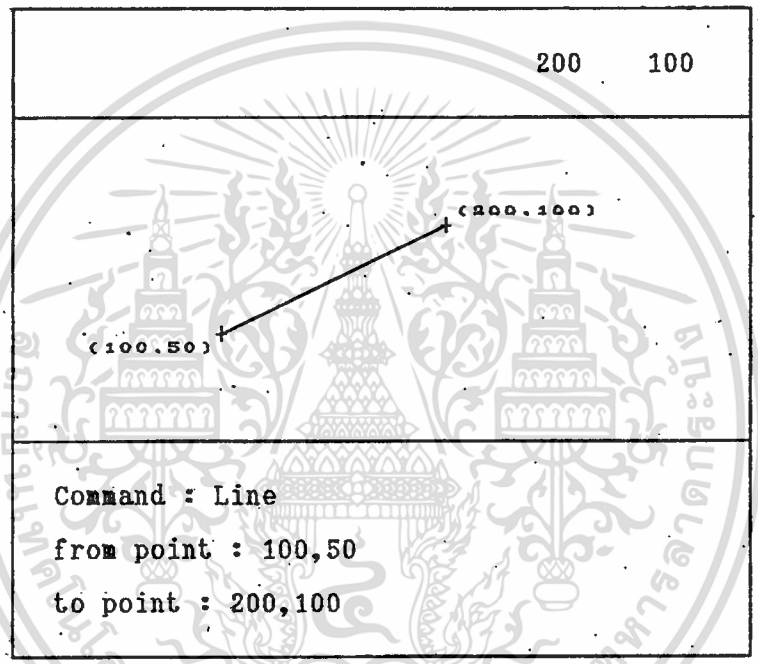
ฉนั้นเมื่อนำเอาหลักการทำงานของแต่ละโมดูลไปเขียนโปรแกรม ก็จะได้โปรแกรมเชียนแบบอย่างง่ายตามโครงสร้างที่ได้กำหนดเอาไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

Computer - User Interface Routine

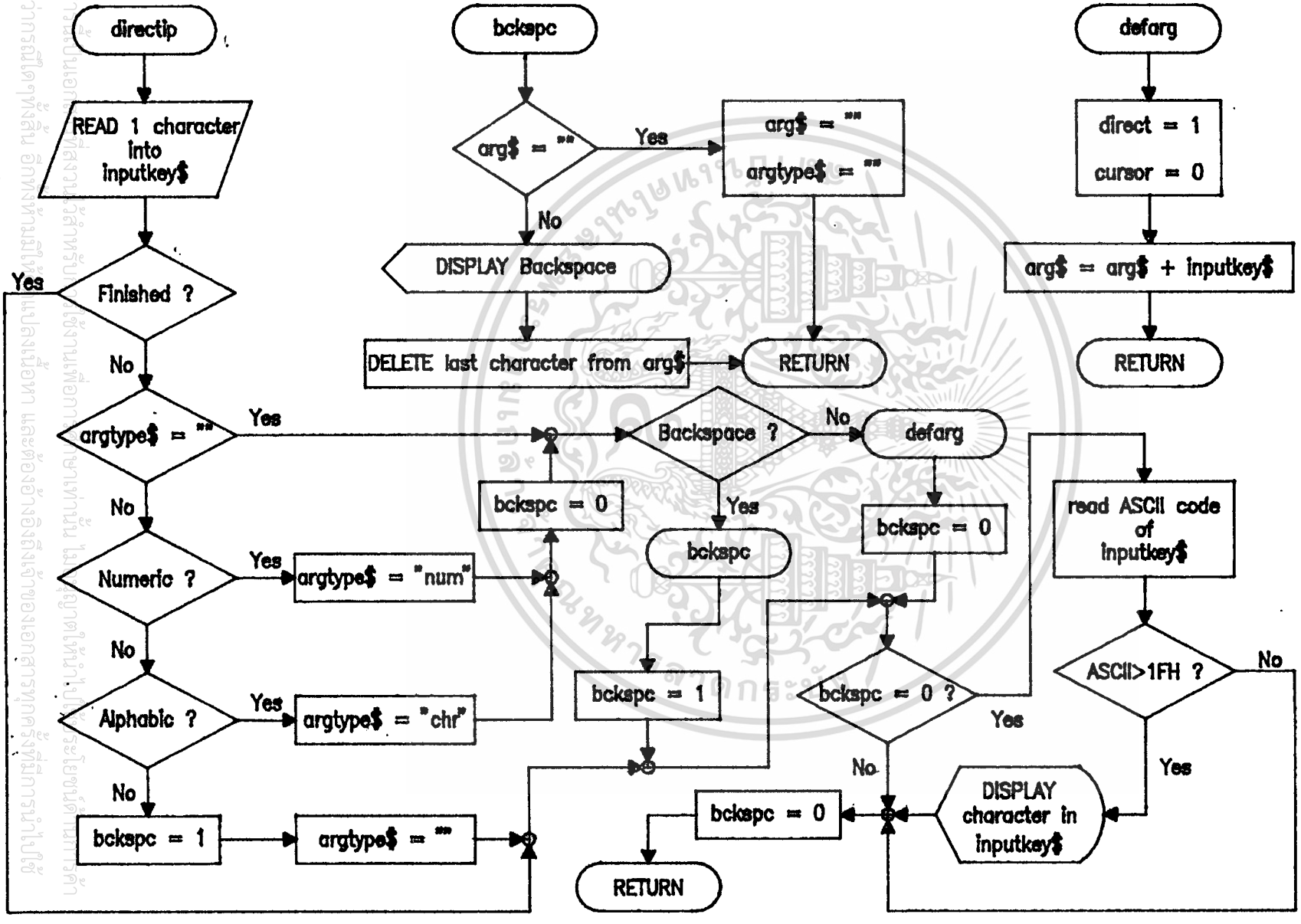
Routine นี้เป็นการจัดการกระบวนการติดต่อระหว่างผู้ใช้กับโปรแกรมเขียนแบบอย่างง่าย โดยได้กำหนดส่วนของจอภาพที่ใช้แจ้งข้อมูลที่ต้องการ และผลตอบสนองของการป้อนข้อมูลเข้าจากผู้ใช้ที่ส่งผ่านแป้นพิมพ์ ณ.บริเวณ 3 บรรทัดสุดท้ายของจอภาพ ทั้งยังจัดการแสดงตำแหน่งพิกัดปัจจุบันไว้ที่บรรทัดบนสุดของจอภาพ ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 การแบ่งจอภาพ

ซึ่งจะเห็นว่าส่วนแสดงผลภาพนั้นจะเป็นพื้นที่ที่ใช้ในการแสดงผลรูปภาพ รวมทั้งกระบวนการของรูปภาพ (image processing) ทั้งหมดก็จะแสดงผลในพื้นที่นี้ด้วย โดยที่ขั้นตอนในการรับข้อมูลจากแป้นพิมพ์มายัง routine นี้ รวมทั้งการแสดงสิ่งที่ต้องการและผลตอบสนองของการรับข้อมูลนั้น จะมีเทคนิคที่นำมาใช้อยู่ 2 ประเภทคือ การรับข้อมูลโดยตรง(direct input) และ การรับข้อมูลด้วยวิธี Cursor Feedback ซึ่งทั้งเทคนิคทั้งสองนี้อยู่ภายใต้การควบคุมขั้นตอนต่าง ๆ ด้วยส่วนควบคุมตัวเดียวกัน ดังจะอธิบายในเนื้อหาต่อไปนี้

3.1 การรับข้อมูลอักขระและตัวเลขโดยตรง: เมื่อให้ arg$ เป็นพื้นที่ใช้เก็บ อักขระหรือตัวเลข ที่รับจากแป้นพิมพ์ครั้งละ 1 ตัวอักษรด้วยการส่งผ่านตัวแปร inputkey$ เข้ามา ถ้ากำหนดให้ argtype$ เป็นตัวแปรที่ใช้ระบุชนิดของข้อมูลใน arg$ และการกำหนดให้ตัวแปร direct มีค่าเท่ากับ 1 เมื่อมีการใช้วิธีการนี้แล้ว ดังนั้นจึงสามารถแสดงขั้นตอนการทำงานของเทคนิคนี้ได้ ดังแสดงในรูปที่ 3.2 และสามารถนำไปเขียนเป็นโปรแกรมการทำงานที่ชื่อ SUBDRT.INC ซึ่งแสดงไว้ใน ภาคผนวก ข.3 โดยเมื่อสิ้นสุดการ


directip
READ 1 character into inputkey$
Finished ?
Yes
No
argtype$ = ""
Yes
No
Numeric ?
Yes
argtype$ = "num"
No
Alphabic ?
Yes
argtype$ = "chr"
No
bckspc = 1
argtype$ = ""
bckspc
arg$ = ""
Yes
No
DISPLAY Backspace
DELETE last character from arg$
RETURN
Backspace ?
No
Yes
bckspc = 0
bckspc
bckspc = 1
bckspc = 0 ?
Yes
No
bckspc = 0
RETURN
defarg
arg$ = ""
argtype$ = ""
RETURN
defarg
direct = 1
cursor = 0
arg$ = arg$ + inputkey$
RETURN
defarg
bckspc = 0
read ASCII code of inputkey$
ASCII>1FH ?
No
Yes
DISPLAY character in inputkey$

รูปที่ 3.2

ทำงานของ subroutine นี้แล้ว ข้อมูลที่นำเข้าจะถูกเก็บไว้ในตัวแปร arg$ และชนิดของข้อมูลจะถูกเก็บไว้ในตัวแปร argtype$ โดย subroutine จะทำการกำหนดค่าตัวแปร direct = 1 และตัวแปร cursor = 0 โดยจะแสดงผลตอบสนองการรับข้อมูลที่ 3 บรรทัดล่างสุดของจอภาพ โดยการกำหนดตัวแปร bckspc ให้เท่ากับ 1 ถ้าต้องการลบตัวอักษรข้างหน้าออกจากข้อมูลที่รับเข้าไป 1 ตัวอักษร ที่เรียกว่า backspace ฉะนั้นข้อมูลใน arg$ จะเป็นข้อมูลที่ได้จากการรับเข้าโดยตรง

3.2 การรับข้อมูลของตำแหน่งโดยใช้วิธี Cursor Feedback: เป็นการรับข้อมูลของค่าตำแหน่งบนพิกัด โดยที่ค่าในแนวนอนจะเก็บไว้ในตัวแปร x และค่าในแนวดิ่งจะเก็บไว้ในตัวแปร y ซึ่งตำแหน่ง x, y นี้จะตรงกับจุดอ้างอิงของ Cursor Image ที่ชี้ไปยังจุดต่าง ๆ บนพิกัด และในส่วนหลักการของ Cursor Feedback นั่นคือ การจะทำอย่างไรเมื่อ Cursor Image ณ.ตำแหน่งเดิมถูกย้ายไปตำแหน่งใหม่แล้วไม่ทำให้ ภาพใน Original Image ณ.ตำแหน่งเก่าไม่เปลี่ยนแปลง ซึ่งสามารถทำได้ดังนี้ :

1) จะต้องสร้าง Cursor Image ที่กว้าง xm pixel และสูง ym pixel โดยมีจุดอ้างอิง ที่มุมล่างซ้ายของกรอบ Image คือที่ (o,p) และเก็บไว้ในตัวแปร cur%

2) จัดการทำสำเนา Original Image ที่มีขนาดเท่ากับบ Cursor Image โดยมีจุดอ้างอิงที่ (x_1,y_1)ไปเก็บไว้ใน Temperary Array ณ.ที่นี้จะใช้ตัวแปร rep%

3) แสดง Cursor Image ใน cur% ณ.ตำแหน่ง x_1,y_1

4) คำนวนตำแหน่งอ้างอิงใหม่ตามทิศทางการเคลื่อนที่ของ Cursor สมมติว่าได้เป็นที่ (x_2,y_2)

5) นำสำเนา Original Image ใน rep% ไปแก้ไข Image ณ.จุดอ้างอิง (x_1,y_1)

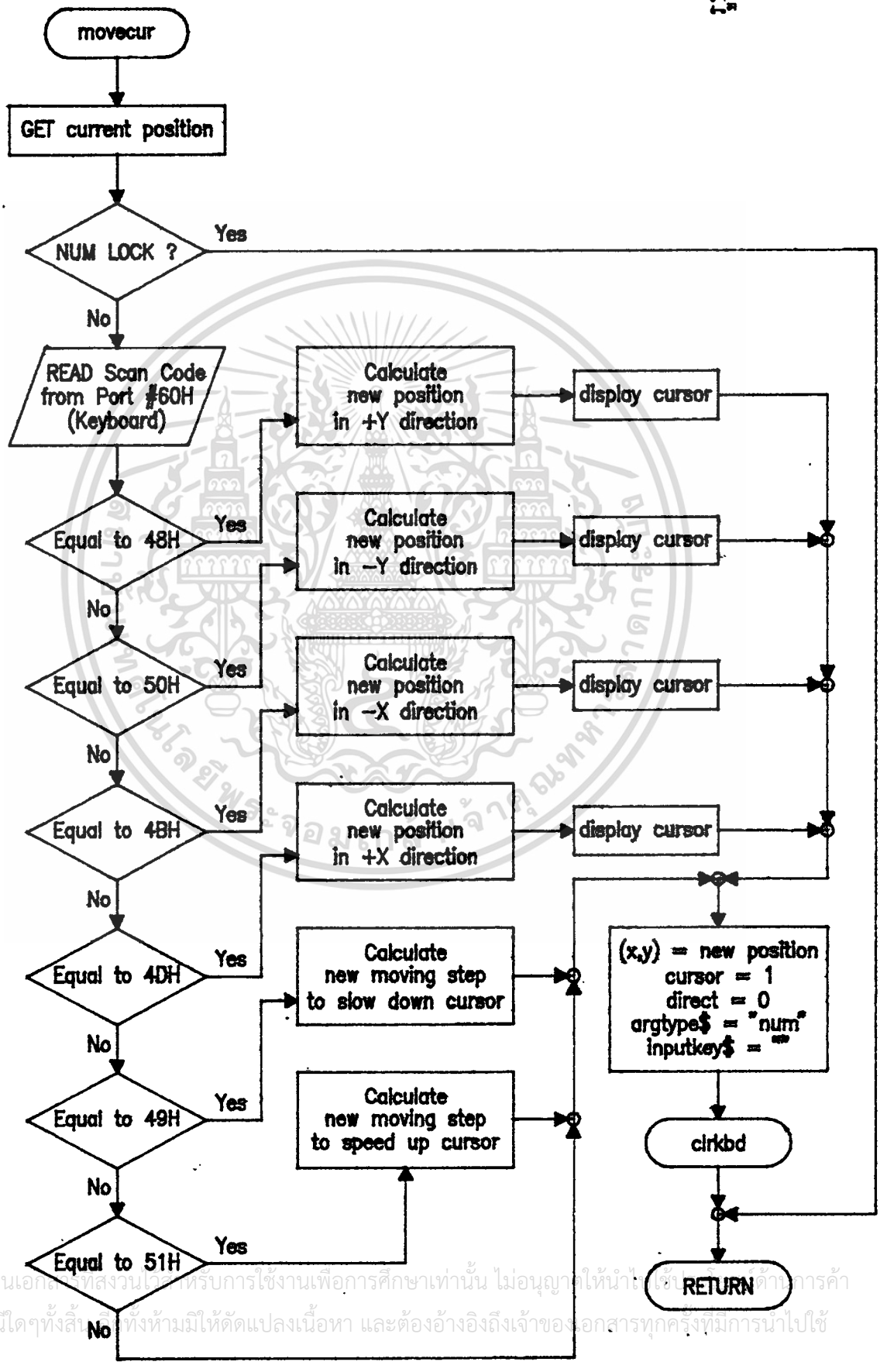
6) ทำตามข้อ 1-5 โดยเปลี่ยนจุดอ้างอิงจาก (x_1,y_1)เป็น (x_2,y_2)

โดยที่ในการเคลื่อนที่ทุกครั้งของ Cursor Image ตำแหน่ง (o,p) จะเป็นตำแหน่งปัจจุบันของพิกัดเสมอ ดังนั้นค่าของ o จึงจะถูกเก็บไว้ในตัวแปร x และค่าของ p จะถูกเก็บไว้ในตัวแปร y ด้วยหลักการที่กล่าวมาสามารถนำมาเขียนเป็นโปรแกรมที่มีชื่อ SUBCUR.INC ที่แสดงไว้ในภาคผนวก ข.3 ซึ่งสามารถนำไปเขียนเป็นผังการทำงานได้ดังรูปที่ 3.3 โดยที่ขั้นตอนในการแสดงภาพ (display) จะเป็นไปตามวิธี Cursor Feedback ที่ได้อธิบายไว้แล้ว และเมื่อสิ้นสุดการทำงานของ subroutine นี้จะให้ผลดังนี้ ค่าของตัวแปร cursor = 1 และ direct = 0 และจะนำตำแหน่งปัจจุบันไปเก็บในตัวแปร x และ y โดยที่กำหนดให้ตัวแปร argtype$ = "num"

ส่วนการคำนวนตำแหน่งใหม่นั้นได้จากการบวก/ลบ ค่าความเปลี่ยนแปลงในแนวตั้ง-นอน(ตามที่กำหนดด้วยการกำหนดความเร็ว Cursor) ไปตามทิศทางการเคลื่อนที่ของ Cursor ที่ได้รับจากการกด pad keys บนแป้นพิมพ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.3



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น ทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การควบคุมขั้นตอนในการรับข้อมูล: เป็นส่วนที่ใช้ในการตรวจสอบ Syntax ของข้อมูล และตรวจว่าการรับข้อมูลนั้นได้สิ้นสุดแล้วหรือไม่อย่างไร โดยการกำหนดค่าต่าง ๆ ลงในตัวแปร flag% ที่ใช้เป็นค่าในการควบคุมการรับข้อมูล ซึ่งความหมายของค่าต่าง ๆ เป็นไปดังตารางที่ 3.1 และเมื่อการรับข้อมูลสิ้นสุดลงส่วนควบคุมนี้จะทำการเปลี่ยนข้อมูลในตัวแปร arg$ ให้เป็นไปตามชนิดของข้อมูลที่กำหนดใน argtype$และลักษณะของข้อมูลในตัวแปร arg$ ซึ่งสามารถสรุปได้ดังตารางที่ 3.2 โดยจากหลักการเหล่านี้สามารถนำไปเขียนเป็นโปรแกรม "SUBCTRL.INC" ที่แสดงไว้ในภาคผนวก ข.3 และสรุปเป็นผังการทำงานได้ดังรูปที่ 3.4

data format	argtype$	arg$	direct	cursor
coordinate x, y	num	±xxy,±yyy	1 0	0 1
magnitude+direction argt#<aval#	num	±mmm<±ddd	1	0
general numeric	num	±nnnn	1	0
general alphabic	chr	$$$$$	1	0

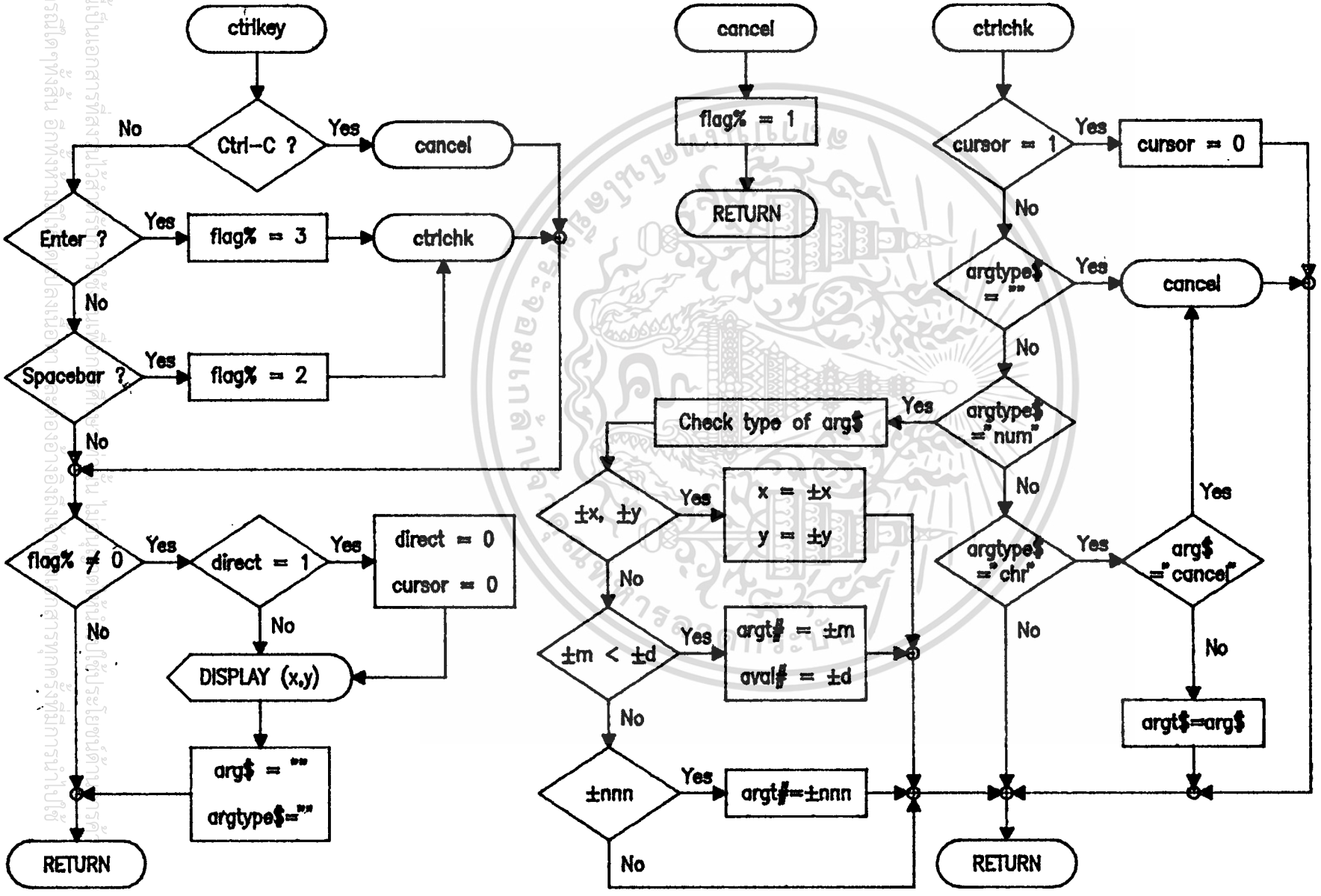
ตารางที่ 3.1 data format

flag%	ความหมาย
0	ไม่มีการไม่มีการประเมินผล / ยังรับข้อมูลเข้าไม่เสร็จ
1	ยกเลิกกระบวนการทั้งหมด
2	เสร็จสิ้นการรับข้อมูลด้วยการกด Space Bar
3	เสร็จสิ้นการรับข้อมูลด้วยการกด Enter
4	Invalid Input

ตารางที่ 3.2 ค่าของ flag%

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

027948


ctrlkey
Ctrl-C ?
No
Yes
cancel
Enter ?
Yes
flag% = 3
ctrlchk
No
Spacebar ?
Yes
flag% = 2
No
flag% ≠ 0
Yes
direct = 1
Yes
direct = 0
cursor = 0
No
DISPLAY (x,y)
No
arg$ = ""
argtype$=""
RETURN
cancel
flag% = 1
RETURN
Check type of arg$
±x, ±y
Yes
x = ±x
y = ±y
No
±m < ±d
Yes
argt# = ±m
aval# = ±d
No
±nnn
Yes
argt#=±nnn
No
ctrlchk
cursor = 1
Yes
cursor = 0
No
argtype$ = ""
Yes
cancel
No
argtype$ = "num"
Yes
No
argtype$ = "chr"
Yes
arg$ = "cancel"
Yes
No
No
argt$=arg$
RETURN

รูปที่ 3.4

นอกจากนี้เพิ่มความสมบูรณ์และถูกต้องในการติดต่อกับผู้ใช้ จึงต้องอาศัย subroutine ต่อไปนี้ที่อยู่ใน โปรแกรม "UTILITY.INC" ที่แสดงไว้ใน ภาคผนวก ข.3 ช่วยทำงาน

3.3.1) *SCRSETUP routine:* ใช้ในการกำหนดส่วนต่าง ๆ ของจอภาพตามรูปที่ 3.1

3.3.2) *SCROLLUP routine:* ใช้ในการเลื่อน 3 บรรทัดล่างสุดขึ้นไป 1 บรรทัด โดยไม่ทำให้ส่วนอื่น ของจอภาพเลื่อนตามขึ้นไป

3.3.3) *CLRKBD routine* : ใช้ในการ clear keyboard buffer ให้ว่าง เพื่อที่จะสามารถรับ การป้อนข้อมูลเข้าจากแป้นพิมพ์ได้ โดยไม่เกิดข้อผิดพลาดขึ้น

3.3.4) *MAKECUR routine* : ใช้สำหรับสร้างและเก็บ Cursor Image ขนาด xm x ym

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

Drawing Routine

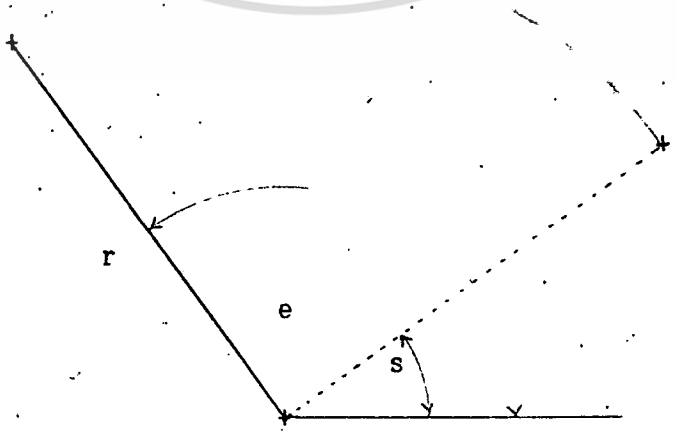
Routine นี้เป็นส่วนที่ใช้ในการแสดงผลรูปภาพ ก่อนอื่นต้องทำความเข้าใจให้เหมือนกันว่า รูปภาพใด ๆ นั้นจะประกอบขึ้นด้วย 2 drawing elements ที่สำคัญ นั่นก็คือ *เส้นตรง*(line element) และ *เส้นโค้ง*(arc element) ดังนั้นในการแสดงผลภาพใด ๆ ก็คือ การวาดเส้นตรงและเส้นโค้งที่ประกอบที่กันขึ้นเป็นภาพนั้น ๆ นั่นเอง

4.1 การวาดเส้นตรง: ปัจจุบันมี algorithm มากมายที่ใช้ใน การสร้างเส้นตรง(line generation) บนจอภาพ ซึ่ง ฉ. ที่จะเลือกใช้คำสั่งที่ใช้ในการสร้างเส้นตรงที่มีอยู่ใน Turbo Basic Compiler มาใช้ในการสร้างเส้นตรงของโปรแกรมเขียนแบบอย่างง่าย ตาม procedure ที่แสดงไว้ในโปรแกรม SUBLINE.INC ดังที่แสดงไว้ใน ภาคผนวก ข.2 และมีรูปแบบการใช้งานดังนี้

```
CALL linef(x_start, y_start, x_end, y_end)
```

จะเห็นว่าข้อมูลพื้นฐานในการสร้างเส้นตรงก็คือ *จุดเริ่มต้น*(x_start, y_start) และ *จุดสุดท้าย* (x_end, y_end) นั่นเอง

4.2 การวาดเส้นโค้ง: เส้นก็โค้งคือ การลากเส้นต่อเนื่องด้วยรัศมีค่าหนึ่ง รอบ ๆ จุดศูนย์กลางหนึ่ง ๆ โดยจะเริ่มทำการวาดตั้งแต่มุมเริ่มต้นที่กำหนด จนถึงมุมสุดท้ายที่กำหนด ดังแสดงในรูปที่ 4.1



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.1

เมื่อกำหนดให้ (x_o, y_o) เป็นจุดศูนย์กลาง, r เป็นรัศมีความโค้ง, s เป็นมุมเริ่มต้น และ e เป็นมุมสุดท้ายแล้ว ดังนั้นจะสามารถอธิบายหลักการสร้างเส้นโค้งได้ดังนี้ะ

1) เพื่อความเรียบของเส้นโค้ง(smoothing of arc) ดังนั้นในการลากเส้นต่อเนื่องจะต้องแบ่งเส้นโค้งออกเป็นส่วนที่เท่า ๆ กัน แล้วลากเส้นตรงในแต่ละส่วนเข้าด้วยกัน โดยจำนวนที่แบ่งสามารถหาได้จาก

$$n = \text{INTEGER}(\text{ABSOLUTE}((20 + 2r)(s - e) / 180))$$

2) จากข้อ(1) ดังนั้นในการวาดเส้นตรงแต่ละส่วน จะเป็นการวาดส่วนของเส้นโค้งที่ละ dt องศา โดยสามารถหาค่าของ dt ได้จาก

$$dt = (s - e) / n$$

3) ดังนั้นในการสร้างเส้นโค้งจากมุมเริ่มต้น s จนถึงมุมสุดท้าย e สามารถทำได้โดยการสร้างเส้นตรงเชื่อมระหว่าง จุด(x_1, y_1)และ จุด(x_2, y_2) เมื่อให้ i = 0 ถึง n - 1 ฉะนั้นจะได้

$$x_1 = x_o + r \cos(s + i \times dt)$$
$$y_1 = y_o + r \sin(s + i \times dt)$$
$$x_2 = x_o + r \cos(s + (i + 1) \times dt)$$
$$y_2 = y_o + r \sin(s + (i + 1) \times dt)$$

4) และเพื่อให้เส้นโค้งสร้างจนถึงมุมสุดท้ายจริง ก็สามารถทำได้โดยการสร้างเส้นตรงเชื่อมระหว่าง จุด(x_1, y_1) และ จุด(x_2, y_2) โดยจะกำหนดให้

$$x_1 = x_o + r \cos(s + n \times dt)$$
$$y_1 = y_o + r \sin(s + n \times dt)$$
$$x_2 = x_o + r \cos(e)$$
$$y_2 = y_o + r \sin(e)$$

จากหลักการที่กล่าวมาแล้วทั้ง 4 ข้อนั้น สามารถนำมาเขียนเป็น procedure ที่แสดงในโปรแกรม SUBARC.INC ดังแสดงไว้ใน ภาคผนวก ข.2 ซึ่งมีรูปแบบการใช้งานดังนี้

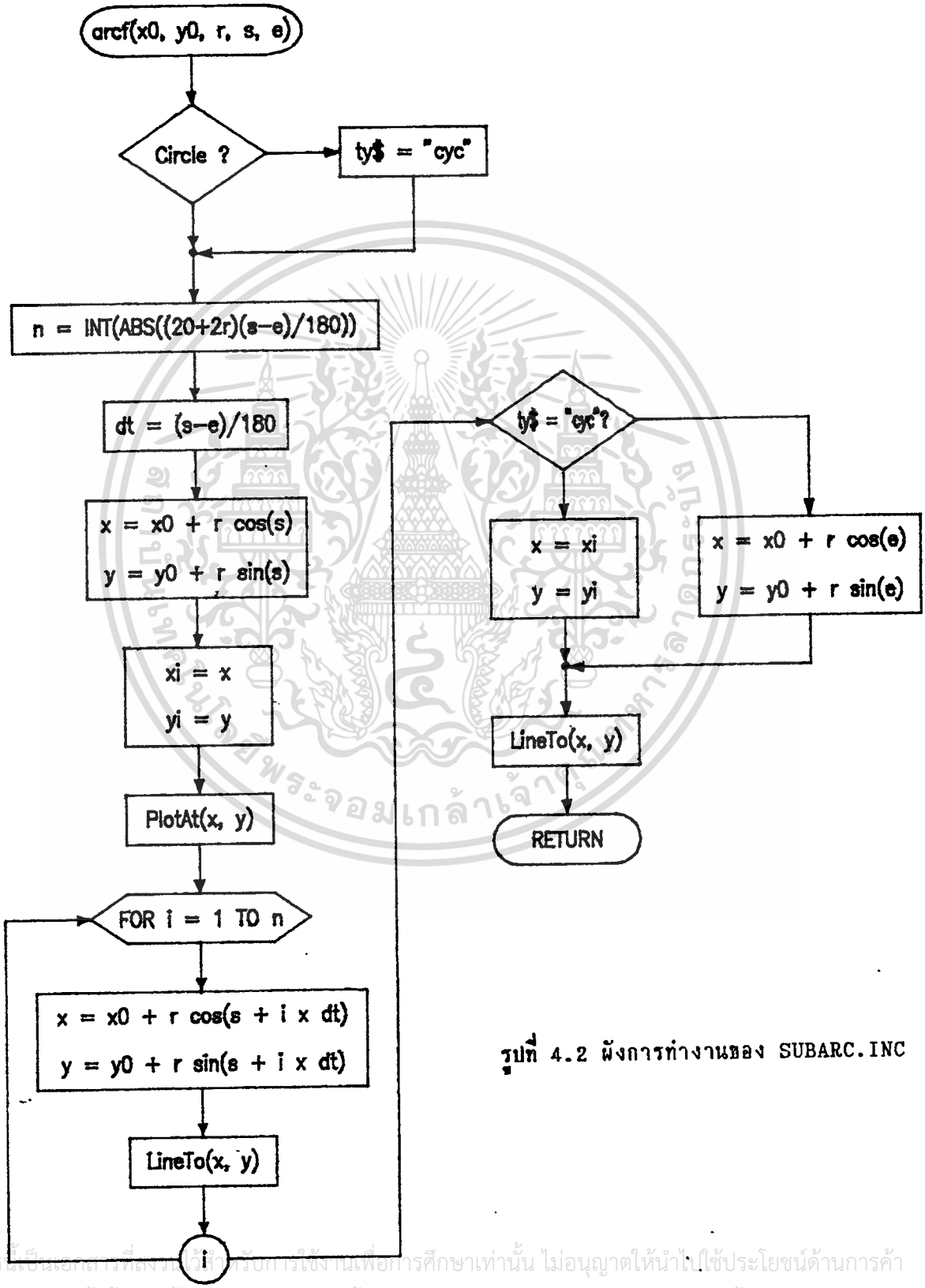
```
CALL arcf(x_center, y_center, radius, start_angle, end_angle)
```

และให้ง่ายต่อการทำความเข้าใจ จึงสรุปขั้นตอนการทำงานต่าง ๆ เป็น ผังแสดงการสร้างเส้นโค้ง ดังจะสามารถแสดงได้ในรูปที่ 4.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การวาดวงกลม: คือการสร้างเส้นโค้งที่มี มุมเริ่มต้น s เท่ากับ 0 และมุมสุดท้าย e เป็น 360 องศา โดยอาศัย procedure ที่ใช้สร้างเส้นโค้ง ฉะนั้นจะมีรูปแบบการสร้างวงกลมดังนี้

```
CALL arcf(x_center, y_center, radius, 0, 360)
```



รูปที่ 4.2 ผังการทำงานของ SUBARC.INC

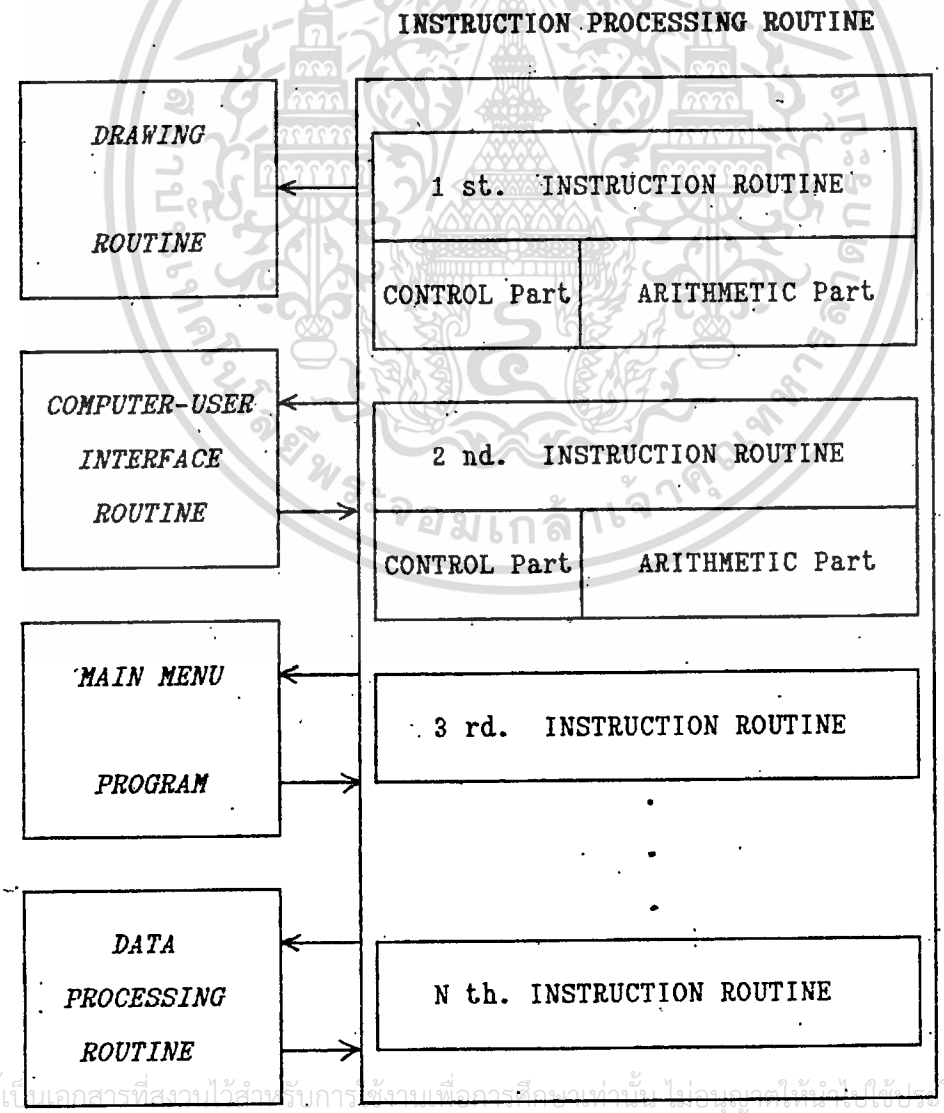
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

Instruction Processing Routine

Instruction Processing Routine นี้เป็นโปรแกรมที่รวบรวม routine คำสั่งต่าง ๆ ที่ใช้ในโปรแกรมเขียนแบบอย่างง่าย โดยแต่ละ routine ทำงานอิสระต่อกัน และเรียกใช้ routine อื่นที่อยู่ในโปรแกรมซึ่งเป็น function และ procedure ได้ โดยก่อนที่จะกล่าวถึงหลักการทำงานของคำสั่งในโปรแกรมเขียนแบบอย่างง่าย ขอให้ทำความเข้าใจเกี่ยวกับ หลักการทำงานของคำสั่งทั่วไปเสียก่อน

5.1 **หลักการทำงานทั่วไปของคำสั่ง:** คำสั่งเกี่ยวกับการสร้างภาพโดยทั่วไปแล้วประกอบด้วย *ส่วนการควบคุมการทำงาน* และ *ส่วนคำนวณข้อมูลพื้นฐาน* ซึ่งในการทำงานในแต่ละส่วนจะมีการติดต่อ และเรียกใช้บริการจาก routine ต่าง ๆ ที่จำเป็นสำหรับคำสั่งนั้น ๆ โดยข้อมูลพื้นฐานที่ใช้ในการสร้างภาพนั้นจะได้จากการคำนวณและการประมวลผลข้อมูลที่ได้รับมาจากการใช้คำสั่งนั้น ๆ ดังแสดงในรูปที่ 5.1



รูปที่ 5.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.1) *ส่วนการควบคุมการทำงาน:* เป็นส่วนที่ใช้สำหรับควบคุมขั้นตอนการทำงาน และการรับส่งข้อมูลของคำสั่งนั้น ๆ รวมทั้งการควบคุมการทำงานร่วมกับโมดูลอื่น ๆ ที่เกี่ยวข้อง ดังหลักการที่จะอธิบายต่อไปนี้:

1) เมื่อ Main Menu Program ส่งการทำงานตามคำสั่งที่ระบุใน command prompt มาที่ routine ของคำสั่งนั้น ๆ ที่อยู่ในโปรแกรม INSTRUCT.INC แล้ว ส่วนการควบคุมการทำงานนี้ก็จะทำการติดต่อกับผู้ใช้ เพื่อที่จะแจ้งและรับข้อมูลที่จำเป็นในการทำงานของส่วนคำนวณข้อมูลพื้นฐาน โดยผ่านทางโมดูล computer-user interface routine ของโปรแกรม

2) และเมื่อได้ข้อมูลพื้นฐานในการสร้างรูปจากส่วนคำนวณข้อมูลพื้นฐานแล้ว ก็จะทำการส่งข้อมูลพื้นฐานที่ได้ไปยัง routine สร้างภาพตามลักษณะของข้อมูลพื้นฐานนั้น ๆ ต่อไป

3) จนกระทั่งการสร้างภาพเสร็จสิ้นลงแล้ว ส่วนการควบคุมการทำงานของคำสั่งนี้จะส่งการทำงานกลับไปที่ command prompt ของ Main Menu Program ดังเดิม

5.1.2) *ส่วนคำนวณข้อมูลพื้นฐาน:* เป็นส่วนที่ทำหน้าที่ในการคำนวณหาข้อมูลพื้นฐานตามลักษณะของข้อมูลที่ได้รับมาจากในส่วนการควบคุมการทำงาน ในการคำนวณหาข้อมูลพื้นฐานหนึ่ง ๆ อาจได้มาจากข้อมูลที่ได้รับซึ่งมีหลาย ๆ ลักษณะก็ได้ ซึ่งแต่ละลักษณะของข้อมูลที่ได้รับจะเป็นสิ่งกำหนดลักษณะของการคำนวณเพื่อที่จะได้มาซึ่งข้อมูลพื้นฐานสำหรับการสร้างภาพนั้น ๆ

5.2 การทำงานของคำสั่งในโปรแกรมเชิงนแบบอย่างง่าย: จากหัวข้อที่ผ่านมาจะเห็นได้ว่า การทำงานของคำสั่งใด ๆ ในโปรแกรมเชิงนแบบอย่างง่ายนั้น จะขึ้นอยู่กับการทำงานร่วมกันของส่วนการควบคุมการทำงานและส่วนคำนวณข้อมูลพื้นฐาน โดยในโปรแกรมเชิงนแบบอย่างง่ายนี้จะมีคำสั่งที่ได้เชิงนขึ้นและทำงานได้จริงดังนี้:

- คำสั่ง LINE
- คำสั่ง ARC
- คำสั่ง CIRCLE
- คำสั่ง COLOUR
- คำสั่ง LINETYPE
- คำสั่ง STATUS
- คำสั่ง QUIT

ซึ่งในส่วนคำนวณของแต่ละโปรแกรมจะอาศัย function และ procedure ต้องต่อไปนี้ร่วมกัน ในการคำนวณหาข้อมูลพื้นฐานสำหรับการสร้างภาพ อันประกอบด้วย routine ต่าง ๆ ดังนี้

1) *Function FNangle#:* เป็น function routine ซึ่งใช้สำหรับการคำนวณหา ค่าของมุมในหน่วยองศาที่วัดจากแกน x ถึงเส้นตรงที่ได้กำหนด จุดเริ่มต้น(x_s, y_s) และจุดสุดท้าย(x_E, y_E)มาให้ โดยอาศัยหลักการคำนวณดังจะอธิบายต่อไปนี้ และเพื่อความเข้าใจจงพิจารณาโปรแกรม SUBANG.INC ที่แสดงอยู่ในภาคผนวก ข.2 ประกอบไปด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อกำหนดให้ $\Delta x = x_E - x_S$

และ $\Delta y = y_E - y_S$

กรณีที่ $\Delta x = 0$, $\theta = 90^\circ$ เมื่อ $\Delta y > 0$

$\theta = -90^\circ$ เมื่อ $\Delta y < 0$

กรณีที่ $\Delta y = 0$, $\theta = 0^\circ$ เมื่อ $\Delta x > 0$ และ $\Delta x = 0$

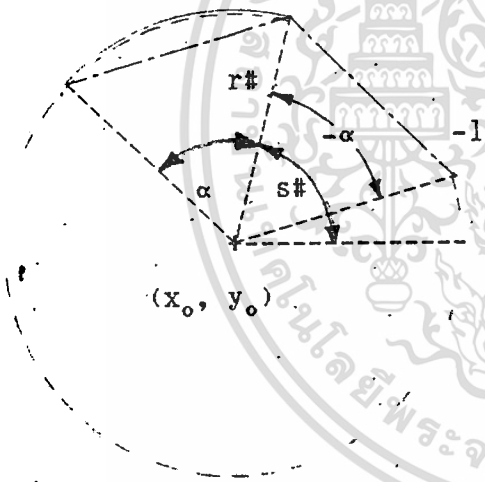
$\theta = 180^\circ$ เมื่อ $\Delta x < 0$

กรณีที่ $x = 0$ และ $y = 0$

$$\theta = \frac{180}{\pi} \tan^{-1}(\Delta y / \Delta x)$$

โดยที่ $0^\circ \leq \theta \leq 360^\circ$

2) *Procedure sclarc:* เป็น procedure routine ที่ใช้สำหรับการคำนวณหา ค่าของมุมสุดท้ายของเส้นโค้ง e# โดยอาศัยข้อมูลของ มุมเริ่มต้นเส้นโค้ง s#, รัศมีความโค้ง r# และ ความยาว chord lc# ที่กำหนดมาให้ ซึ่งสามารถคำนวณหาได้จากหลักการต่อไปนี้



รูปที่ 5.2

เมื่อพิจารณารูปที่ 5.2 และ รูปที่ 5.3 จะพบว่าสามารถหา e# ได้จากการอาศัย ทฤษฎีของพิธากอรัส ที่ว่าด้วยสามเหลี่ยมมุมฉาก ถ้ากำหนดให้มุม α เป็นมุมตรงข้าม chord ที่ยาว $\pm$lc# และทำให้หามุมสุดท้าย e# ได้จากสมการ

$$e\# = s\# + \alpha$$

โดยที่สามารถหามุม α ได้จากสามเหลี่ยมมุมฉากที่มีมุมยอด $\alpha/2$, มีฐานยาว lc#/2 และ มีด้านตรงข้ามมุมฉากยาว r# ฉะนั้นหามุม α ได้จาก

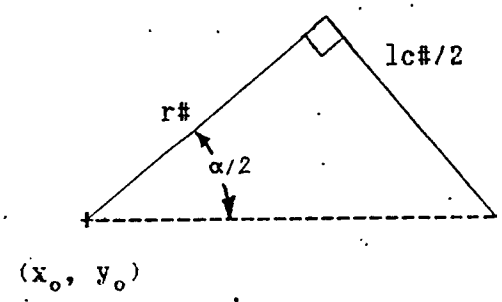
$$\alpha = (180/\pi)(2\sin^{-1}(lc\#/2r\#))$$

แต่เนื่องจาก

$$\sin^{-1}(B/A) = \tan^{-1}(B/(A^2 - B^2)^{1/2})$$

ดังนั้นจะได้

$$\alpha = \frac{360}{\pi} \tan^{-1}\left[\frac{lc\#/2}{r^2 - (lc\#/2)^2}\right]$$



รูปที่ 5.3

และเพื่อความเข้าใจที่ถูกต้อง จงพิจารณาจาก sclarc procedure routine ที่อยู่ในโปรแกรม ARCUTY.INC ที่แสดงไว้ใน ภาคผนวก ข.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) *Procedure threepoint:* เป็น procedure routine มาที่ใช้สำหรับการคำนวณหา จุดศูนย์กลาง (x_o, y_o), รัศมีความโค้ง r, มุมเริ่มต้นของเส้นโค้ง s และ มุมสุดท้ายของเส้นโค้ง e โดยอาศัยข้อมูลของจุด 3 จุดที่กำหนดคือ (x_I, y_I), (x_J, y_J) และ (x_K, y_K) ซึ่งทั้ง 3 จุดนี้อยู่บนเส้นโค้งเดียวกัน และจะทำให้สมการต่อไปนี้เป็นจริง

$$(x - x_o)^2 + (y - y_o)^2 = r^2$$

โดยสามารถกระจายเทอมได้เป็น

$$x^2 - 2.xx_o + x_o^2 + y^2 - 2.yy_o + y_o^2 = r^2 \qquad (3.1)$$

เมื่อแทนค่า (x_I, y_I) ลงในสมการ (3.1) แล้วจะได้

$$x_I^2 - 2.x_Ix_o + x_o^2 + y_I^2 - 2.y_Iy_o + y_o^2 = r^2 \qquad (3.2)$$

เมื่อแทนค่า (x_J, y_J) ลงในสมการ (3.1) แล้วจะได้

$$x_J^2 - 2.x_Jx_o + x_o^2 + y_J^2 - 2.y_Jy_o + y_o^2 = r^2 \qquad (3.3)$$

เมื่อแทนค่า (x_K, y_K) ลงในสมการ (3.1) แล้วจะได้

$$x_K^2 - 2.x_Kx_o + x_o^2 + y_K^2 - 2.y_Ky_o + y_o^2 = r^2 \qquad (3.4)$$

เมื่อให้ สมการ (3.3) - สมการ (3.2) จะได้

$$(x_J^2 - x_I^2) + 2.x_o(x_I - x_J) + (y_J^2 - y_I^2) + 2.y_o(y_I - y_J) = 0$$

สามารถจัดเทอมใหม่ได้เป็น

$$2(x_I - x_J).x_o + 2(y_I - y_J).y_o = (x_I^2 - x_J^2) + (y_I^2 - y_J^2) \qquad (3.5)$$

ในทำนองเดียวกันเมื่อให้ สมการ (3.4) - สมการ (3.2) ก็จะได้

$$2(x_I - x_K).x_o + 2(y_I - y_K).y_o = (x_I^2 - x_K^2) + (y_I^2 - y_K^2) \qquad (3.6)$$

โดยสามารถนำสมการ (3.5) และ สมการ (3.6) มาเขียนในรูปเมตริกได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\begin{bmatrix} 2(x_1 - x_J) & 2(y_1 - y_J) \\ 2(x_1 - x_K) & 2(y_1 - y_K) \end{bmatrix} \begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} (x_1^2 - x_J^2) + (y_1^2 - y_J^2) \\ (x_1^2 - x_K^2) + (y_1^2 - y_K^2) \end{bmatrix}$$

และสามารถเขียนลดรูปได้ดังนี้

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

โดยที่

$$a_{11} = 2(x_1 - x_J)$$
$$a_{12} = 2(y_1 - y_J)$$
$$a_{21} = 2(x_1 - x_K)$$
$$a_{22} = 2(y_1 - y_K)$$
$$b_1 = (x_1^2 - x_J^2) + (y_1^2 - y_J^2)$$
$$b_2 = (x_1^2 - x_K^2) + (y_1^2 - y_K^2)$$

และสามารถหาค่าของมุม s ได้จาก

$$s = FNangle\#(x_o, y_o, x_1, y_1)$$

และสามารถหาค่าของมุม e ได้จาก

$$e = FNangle\#(x_o, y_o, x_J, y_J)$$

และสามารถหาค่าของ r ได้จาก

$$r = ((x_1 - x_o)^2 + (y_1 - y_o)^2)^{1/2}$$

และจากหลักการดังที่กล่าวมาแล้วนั้น สามารถนำไปเขียนเป็น threepoint procedure routine ที่อยู่ในโปรแกรม ARCUTY.INC ซึ่งแสดงไว้ใน ภาคผนวก ข.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4) *Function FNltype%:* เป็น function routine ซึ่งสร้างขึ้นมาใช้ในการกำหนด รูปแบบของเส้นตรง(line pattern) โดยจะขึ้นอยู่กับการเลือก ชนิดของเส้นตรง(linetype) ซึ่งแต่ละชนิดจะมีรูปแบบแตกต่างกัน ดังต่อไปนี้ เมื่อให้ FNltype% เป็น parameter ที่ใช้ส่ง รูปแบบของเส้นตรง กลับ

กรณีที่ชนิดของเส้นตรงเป็น "CONTINUE" จะกำหนดให้ FNltype% = &B1111111111111111
กรณีที่ชนิดของเส้นตรงเป็น "DOTTED" จะกำหนดให้ FNltype% = &B1000000010000000
กรณีที่ชนิดของเส้นตรงเป็น "HIDDEN" จะกำหนดให้ FNltype% = &B1111000011110000
กรณีที่ชนิดของเส้นตรงเป็น "DASHED" จะกำหนดให้ FNltype% = &B1111111111110000
กรณีที่ชนิดของเส้นตรงเป็น "DASHDOT" จะกำหนดให้ FNltype% = &B1111111110001000
กรณีที่ชนิดของเส้นตรงเป็น "DEVIDED" จะกำหนดให้ FNltype% = &B1111111100100100

ดังข้อกำหนดที่กล่าวมาแล้วสามารถนำไปเขียนเป็นโปรแกรม LTYPE.INC ที่แสดงใน ภาคผนวก ข.2

5) *Function FNcolor:* เป็น function routine ที่ใช้ในการกำหนด รหัสสี(colour code) ของ element ทั้งหลาย โดยจะขึ้นอยู่กับการเลือก ชนิดของสี ซึ่งแต่ละชนิดจะมีรหัสที่แตกต่างกัน ดังต่อไปนี้ เมื่อให้ FNcolor เป็น parameter ที่ใช้ส่ง รหัสสี กลับ

กรณีที่เลือกใช้ "BLACK" ดังนั้นจะกำหนดให้ FNcolor = 0
กรณีที่เลือกใช้ "BLUE" ดังนั้นจะกำหนดให้ FNcolor = 1
กรณีที่เลือกใช้ "GREEN" ดังนั้นจะกำหนดให้ FNcolor = 2
กรณีที่เลือกใช้ "CYAN" ดังนั้นจะกำหนดให้ FNcolor = 3
กรณีที่เลือกใช้ "RED" ดังนั้นจะกำหนดให้ FNcolor = 4
กรณีที่เลือกใช้ "MAGENTA" ดังนั้นจะกำหนดให้ FNcolor = 5
กรณีที่เลือกใช้ "BROWN" ดังนั้นจะกำหนดให้ FNcolor = 6
กรณีที่เลือกใช้ "WHITE" ดังนั้นจะกำหนดให้ FNcolor = 7
กรณีที่เลือกใช้ "GRAY" ดังนั้นจะกำหนดให้ FNcolor = 8
กรณีที่เลือกใช้ "LIGHTBLUE" ดังนั้นจะกำหนดให้ FNcolor = 9
กรณีที่เลือกใช้ "LIGHTGREEN" ดังนั้นจะกำหนดให้ FNcolor = 10
กรณีที่เลือกใช้ "LIGHTCYAN" ดังนั้นจะกำหนดให้ FNcolor = 11
กรณีที่เลือกใช้ "LIGHTRED" ดังนั้นจะกำหนดให้ FNcolor = 12
กรณีที่เลือกใช้ "LIGHTMAGENTA" ดังนั้นจะกำหนดให้ FNcolor = 13
กรณีที่เลือกใช้ "YELLOW" ดังนั้นจะกำหนดให้ FNcolor = 14
กรณีที่เลือกใช้ "BRIGHTWHITE" ดังนั้นจะกำหนดให้ FNcolor = 15

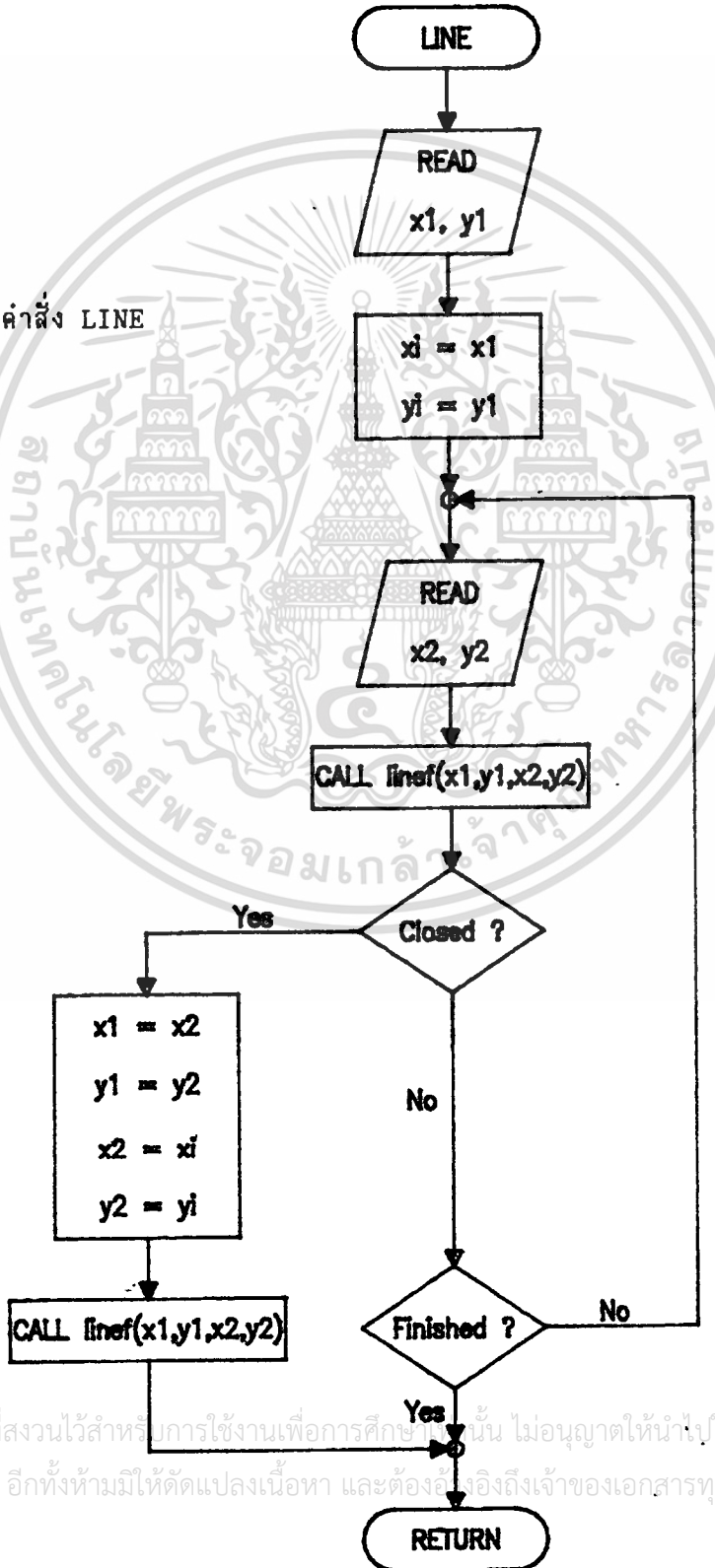
ดังข้อกำหนดที่กล่าวมาแล้วสามารถนำไปเขียนเป็นโปรแกรม COLOUR.INC ที่แสดงใน ภาคผนวก ข.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ ดัดแปลง เผยแพร่ หรือแสวงหาประโยชน์ทางการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเห็นว่า function และ procedure ที่กล่าวมาแล้วทั้งหมดนั้นเป็นส่วนของ ส่วนคำนวณข้อมูลพื้นฐาน ของโปรแกรมเขียนแบบอย่างง่าย ฉะนั้นในส่วนที่กล่าวถึงต่อไปนี้ก็คือ ส่วนการควบคุมการทำงาน ของคำสั่งในโปรแกรมเขียนแบบอย่างง่ายนี้นั่นเอง ซึ่งจะได้อธิบายถึงกระบวนการทำงานตามชนิดของคำสั่งดังต่อไปนี้

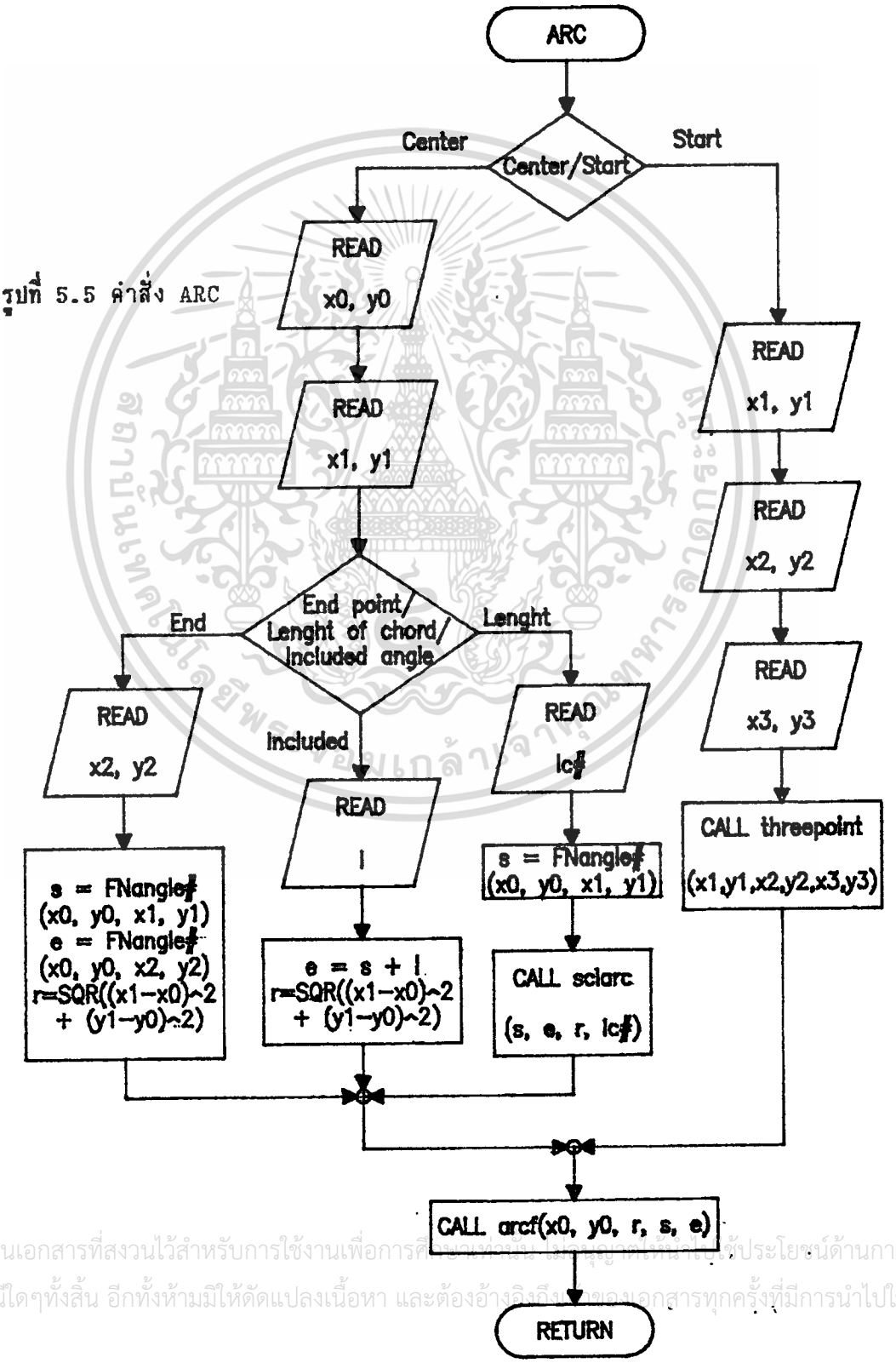
1) *คำสั่ง LINE:* เป็นคำสั่งที่ใช้ในการวาดเส้นตรง ซึ่งมีขั้นตอนการทำงานตาม ผังการทำงาน ดังที่แสดงไว้ในรูปที่ 5.4 โดยสามารถเขียนเป็น linint subroutine ที่อยู่ในโปรแกรม INSTRUCT.INC ซึ่งได้แสดงไว้ใน ภาคผนวก ข.3

รูปที่ 5.4 คำสั่ง LINE



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

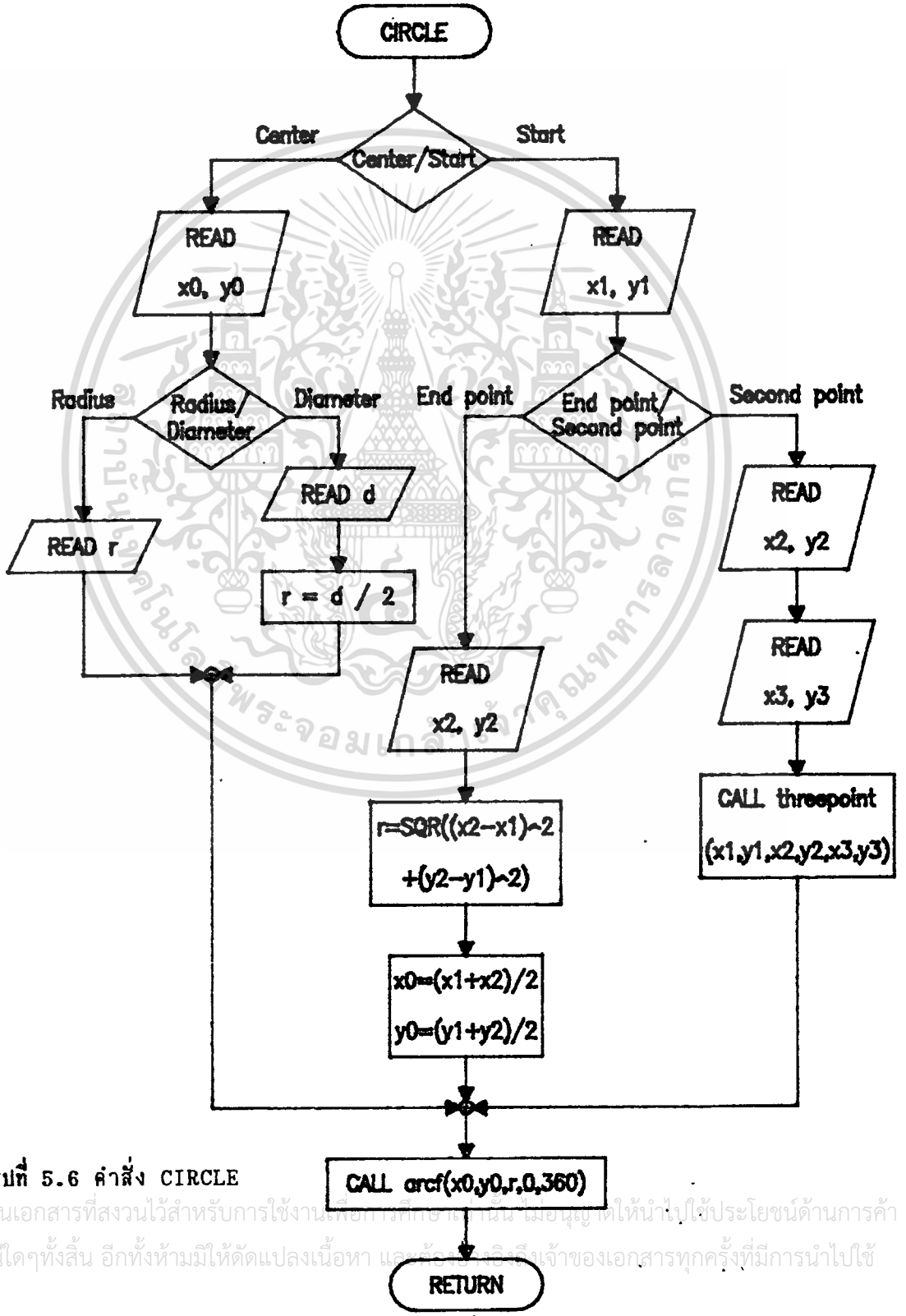
2) *คำสั่ง ARC:* เป็นคำสั่งที่ใช้ในการวาดเส้นโค้ง ซึ่งมีขั้นตอนการทำงานตาม ผังการทำงาน ดังที่แสดงไว้ในรูปที่ 5.5 โดยสามารถเขียนเป็น arcint subroutine ที่อยู่ในโปรแกรม INSTRUCT.INC ซึ่งได้แสดงไว้ใน ภาคผนวก ข.3



รูปที่ 5.5 คำสั่ง ARC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) *คำสั่ง CIRCLE:* เป็นคำสั่งที่ใช้ในการวาดวงกลม ซึ่งมีขั้นตอนการทำงานเป็นไปตาม ผังการทำงานดังที่แสดงไว้ในรูปที่ 5.6 โดยสามารถนำไปเขียนเป็น circint subroutine ซึ่งแสดงไว้ในโปรแกรม INSTRUCT.INC ดังที่แสดงไว้ใน ภาคผนวก ข.3



รูปที่ 5.6 คำสั่ง CIRCLE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) *คำสั่ง COLOUR:* เป็นคำสั่งที่ใช้ในการ กำหนดสีของ element ตามรหัสสีดังที่ได้กล่าวไว้แล้ว โดยที่ขั้นตอนการทำงานของคำสั่งนี้ สามารถนำไปเขียนเป็น clrint subroutine ซึ่งได้แสดงไว้ในโปรแกรม INSTRUCT.INC ดังที่แสดงไว้ใน ภาคผนวก ข.3

4) *คำสั่ง LINETYPE:* เป็นคำสั่งที่ใช้ในการ กำหนดรูปแบบของเส้นตรง ตามชนิดของเส้นตรงดังที่ได้กล่าวไว้แล้ว โดยขั้นตอนการทำงานของคำสั่งนี้ สามารถนำไปเขียนเป็น ltpint subroutine ซึ่งได้แสดงไว้ในโปรแกรม INSTRUCT.INC ดังที่แสดงไว้ใน ภาคผนวก ข.3

5) *คำสั่ง STATUS:* เป็นคำสั่งที่ใช้แจ้ง ชื่อ/รหัสของสีปัจจุบัน และ ชนิดของเส้นตรง โดยที่ขั้นตอนการทำงานของคำสั่งนี้ สามารถนำไปเขียนเป็น stsint subroutine ที่อยู่ปรแกรม INSTRUCT.INC ดังที่แสดงไว้ใน ภาคผนวก ข.3

6) *คำสั่ง QUIT:* เป็นคำสั่งที่ใช้ ยกเลิกการใช้โปรแกรมเขียนแบบอย่างง่าย และส่งการทำงานไปยัง DOS's prompt โดยที่ขั้นตอนการทำงานของคำสั่งนี้ สามารถนำไปเขียนเป็น break subroutine ที่อยู่ปรแกรม INSTRUCT.INC ดังที่แสดงไว้ใน ภาคผนวก ข.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

Data Processing Routine

6.1 โครงสร้างข้อมูล: พิจารณาจากรูปที่ 6.1 จะเห็นว่าเป็นการใช้โครงสร้างข้อมูลแบบ link list โดย Node แรกของlist จะเป็น Head Node เสมอ ซึ่งใน Head Node จะบรรจุข้อมูลใน field ต่าง ๆ ดังต่อไปนี้ โดยแต่ละ field จะใช้เนื้อที่ 4 byte

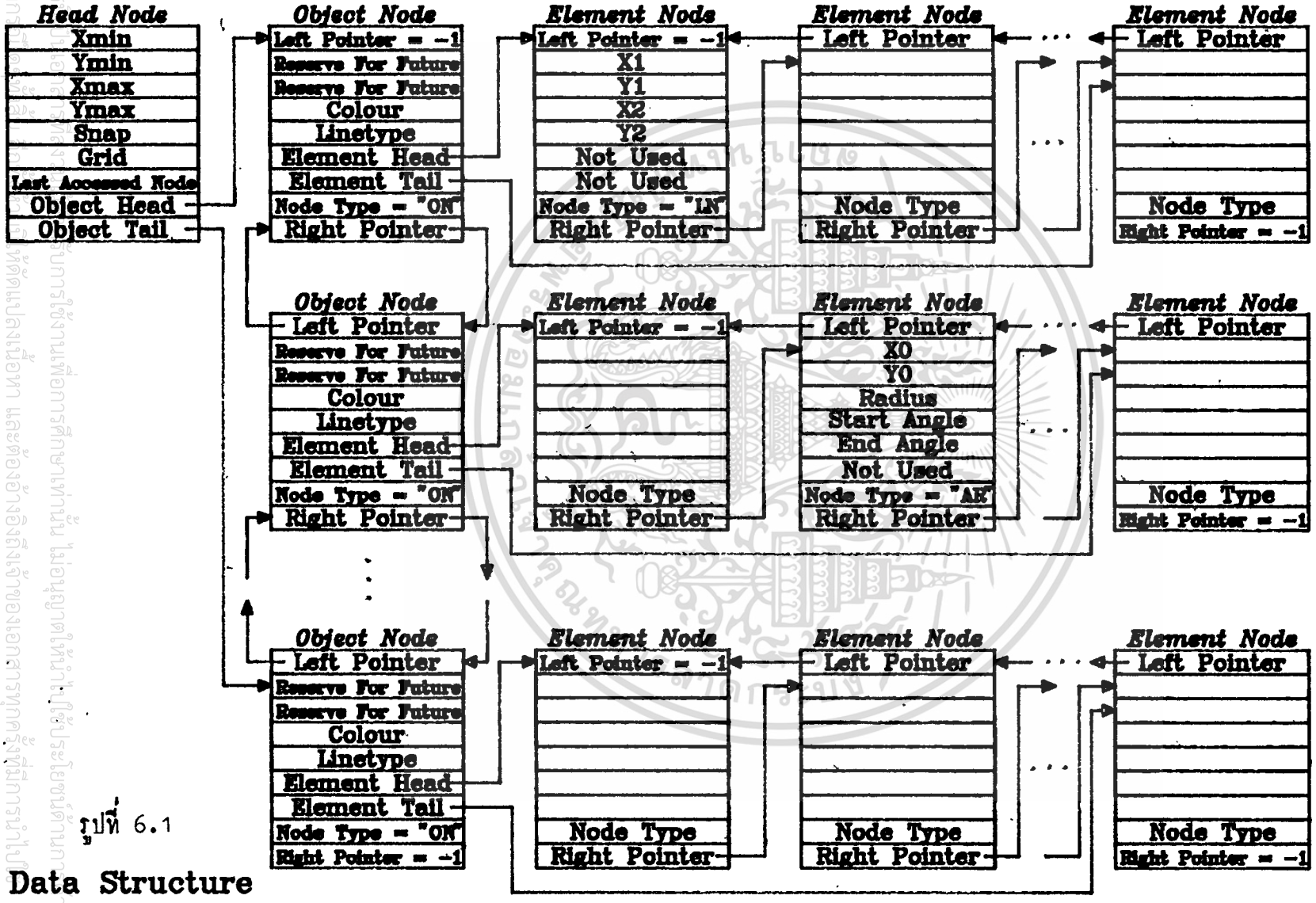
field ที่ 1,2 : ค่าตำแหน่งมุมล่างซ้ายของส่วนแสดงผลภาพ , xmin และ ymin
field ที่ 3,4 : ค่าตำแหน่งมุมบนขวาของส่วนแสดงผล , xmin , ymin
field ที่ 5 : ความแตกต่างในการเคลื่อนที่ของ Cursor , snap
field ที่ 6 : หมายเลขของ Node สุดท้ายของ list,Last Acessed Node
field ที่ 7 : หมายเลขของ Node แรกของ Object Node
field ที่ 8 : หมายเลขของ Node สุดท้ายของ Object Node

และตั้งแต่ Node ที่ 2 ลงมาจนถึง Node สุดท้ายของ list จะประกอบด้วย Node ต่าง ๆ ประกอบกัน และ field ของแต่ละ Node ก็จะใช้เนื้อที่ 4 byte โดยจะมีแบบฟอร์มที่เหมือนกันคือ

field ที่ 1 : หมายเลข Node ก่อนจะถึง Node ปัจจุบัน แต่ถ้า Node ปัจจุบันเป็น Node แรกของ Object/Element จะกำหนดให้ Field นี้มีค่าเท่ากับ -1
field ที่ 2 - 7: เป็นส่วนที่เก็บข้อมูลของ Node ซึ่งลักษณะการเก็บข้อมูลจะเป็นไปตามชนิดของ Node ที่กำหนดใน Node Type
field ที่ 8 : แสดงชนิดของ Node ปัจจุบัน
field ที่ 9 : หมายเลข Node ต่อไปจาก Node ปัจจุบัน แต่ถ้า Node ปัจจุบันเป็น Node สุดท้ายของ Object/Element จะกำหนดให้ field นี้มีค่าเท่ากับ -1

จะเห็นว่า field ที่ 2-7 จะเก็บข้อมูลตามชนิดของ Node ที่ระบุใน field ที่ 8 ซึ่งสามารถสรุปชนิดต่าง ๆ ของ Node ได้ดังตารางที่ 6.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.1

Data Structure

ชนิดของ Node	รหัสใน Node Type
Object Node	ON
Line Element Node	LN
Arc Element Node	AR
Erased Object Node	EO
Erased Line Element	EL
Erased Arc Element	EA

ตารางที่ 6.1 ตารางแสดงรหัสของ Node Type

และจากตาราง 6.1 จะได้แบบฟอร์มข้อมูลที่เก็บใน field ที่ 2-7 ดังจะแสดงในตารางที่ 6.2 สำหรับ Element Node นั้นจะมี 2 ชนิดคือ Line Element Node และ Arc Element Node ซึ่งจะถือว่าทั้งคู่เป็น Element Node เหมือนกัน

Node Type	filed ที่ 2	field ที่ 3	field ที่ 4	field ที่ 5	field ที่ 6	field ที่ 7
ON	Reverse for Future	Reverse for Future	Colour Code	Linetype Code	Element Head	Element Tail
LN	X1	Y1	X2	Y2	Not Used	Not Used
AR	X0	Y0	radius	start angle	end angle	Not Used
EO/EL/EA	Don't Change	Don't Change	Don't Change	Don't Change	Don't Change	Don't Change

ตารางที่ 6.2 ตารางแสดงข้อมูลที่จะแสดงในแต่ละ field

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยรหัสใน Node Type ของ Erased Node ในตารางทั้ง 2 ย่อมาจาก

EO : Erased Object Node

EL : Erased Line Element Node

EA : Erased Arc Element Node

ดังนั้นสรุปได้ว่าในแต่ละ Node จะแบ่งเป็น 2 ส่วนที่สำคัญคือ Pointer และ Information โดยจะกล่าวถึงความหมายของง Pointer แต่ละตัวดังนี้

Object Head : หมายเลข Node ของ Object Node แรก

Object Tail : หมายเลข Node ของ Object Node สุดท้าย

Element Head : หมายเลข Node ของ Element Node ของ Object หนึ่ง ๆ

Element Tail : หมายเลข Node ของ Element Node ของ Object หนึ่ง ๆ

Left Position : หมายเลข Node ก่อนหน้า Node ปัจจุบัน และจะเท่ากับ -1 เมื่อเป็น Node แรก

Right Position: หมายเลข Node ถัดจาก Node ปัจจุบัน และจะเท่ากับ -1 เมื่อเป็น Node สุดท้าย

6.2 หลักการทำงานของการจัดการข้อมูล: ในการจัดการข้อมูลจะแบ่งการทำงานออกเป็นคำสั่งต่าง ๆ ดังนี้

6.2.1) NEW: เป็นตัวแปรที่ใส่ค่าตำแหน่งของ Node ใหม่

6.2.2) ALLOCATE(pointer): เป็นการดึง Node ที่ pointer ชี้ใน list ขึ้นมาจัดการ

6.2.3) FREE(pointer): เป็นการปล่อย Node ที่ pointer ชี้คืนสู่ list ซึ่ง Node นี้ถูกดึงขึ้นมาด้วยคำสั่ง ALLOCATE(pointer)

6.2.4) INSERT(pointer): เป็นการเพิ่ม/แทรก Node ที่ pointer ชี้ลงไปใน list

6.2.5) DELETE(pointer): เป็นการลบ Node ที่ pointer ชี้ออกจาก list

6.2.6) INFO(pointer, field): เป็นการชี้ไปยัง field ที่ระบุไว้ของ Node ที่ pointer ชี้อยู่เพื่อทำการอ่าน/บันทึกค่าลงใน field ของ Node นั้น

6.2.7) LPTR(pointer) : เป็นการอ่าน/บันทึก Lift Pointer ของ Node ที่ pointer ชี้

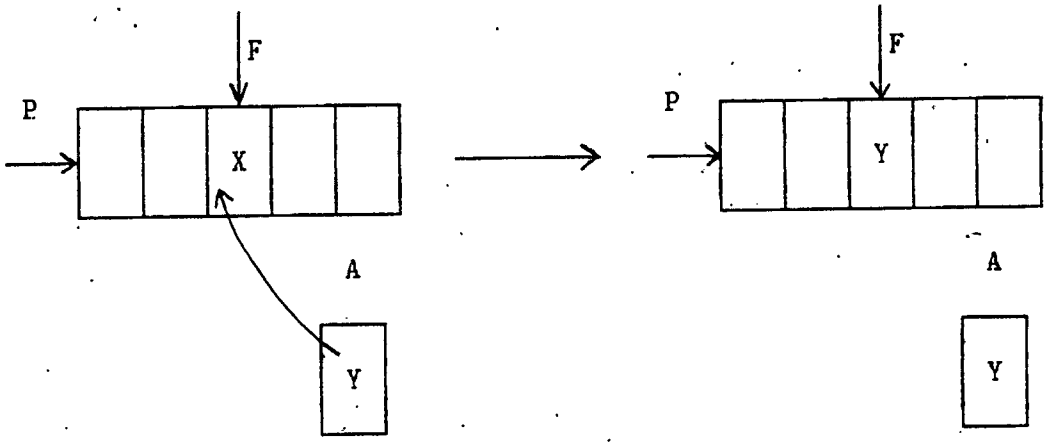
6.2.8) RPTR(pointer) : เป็นการอ่าน/บันทึก Right Pointer ของ Node ที่ pointer ชี้

6.3 การทำงานเบื้องต้นของการจัดการข้อมูล: ในหัวข้อนี้จะอธิบายการทำงานของ คำสั่งในการจัดการข้อมูลต่าง ๆ ที่ได้เขียนไว้ใช้ในโปรแกรมเขียนแบบอย่างนี้ ดังจะกล่าวถึงดังต่อไปนี้(ไม่ครบตามที่มีในหัวข้อที่แล้ว)

6.3.1) NEW: สามารถหาค่าตำแหน่งใหม่ของ Node ได้ โดยการอ่านค่าใน Last Accessed Node ที่บรรจุอยู่ใน field ที่ 7 ของ Head Node(Node ที่ 1 ของ list)

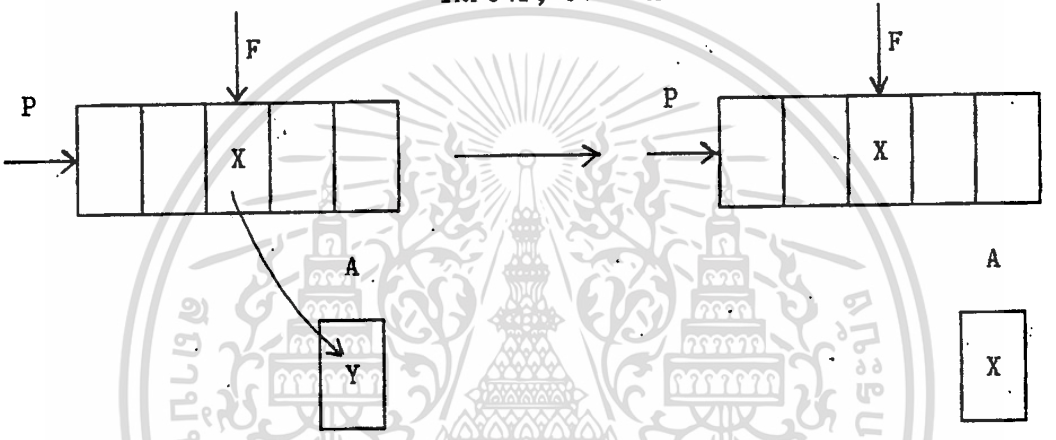
6.3.2) INFO(pointer, field): สามารถแสดงการทำงานได้ด้วยรูปที่ 6.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก) การเก็บค่าของตัวแปร A ลงใน field F ของ Node P

INFO(P, F) = A

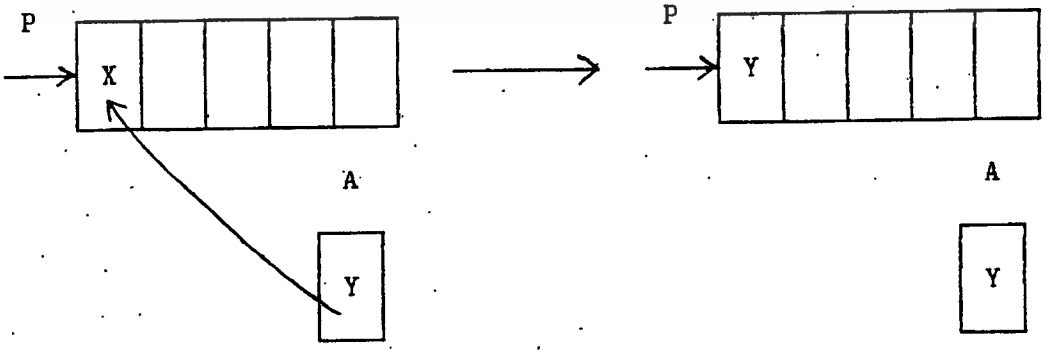


(ข)การเก็บค่าใน field F ของ Node P ลงในตัวแปร A

A = INFO(P, F)

รูปที่ 6.2 การทำงานของ INFO(pointer, field)

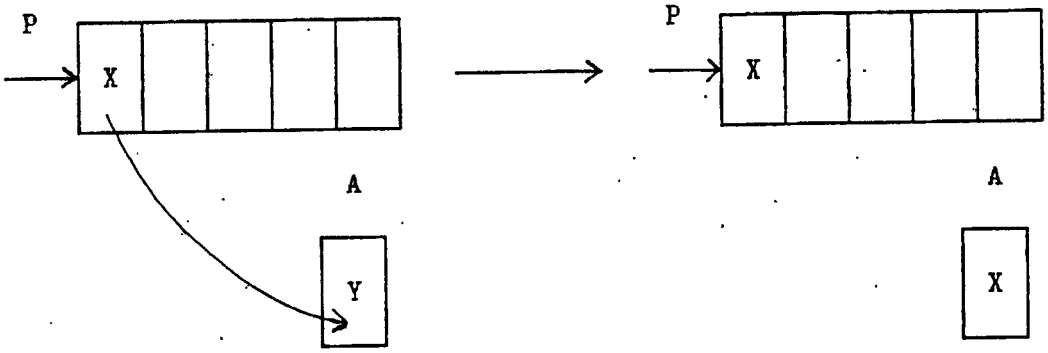
6.3.3) LPTR(pointer): สามารถอธิบายการได้ดังรูปที่ 6.3



(ก) การเก็บค่าจากตัวแปร A ลงใน Left Pointer

LPTR(P) = A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

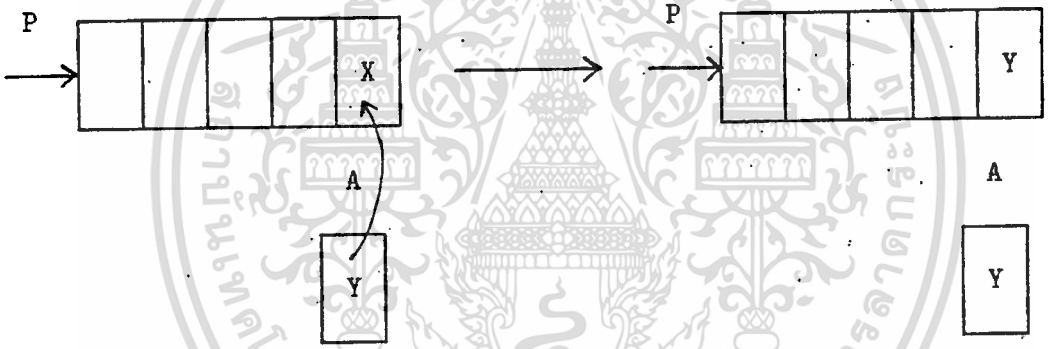


(ก) การเก็บค่าจาก Left Pointer ลงในตัวแปร A

A = LPTR(P)

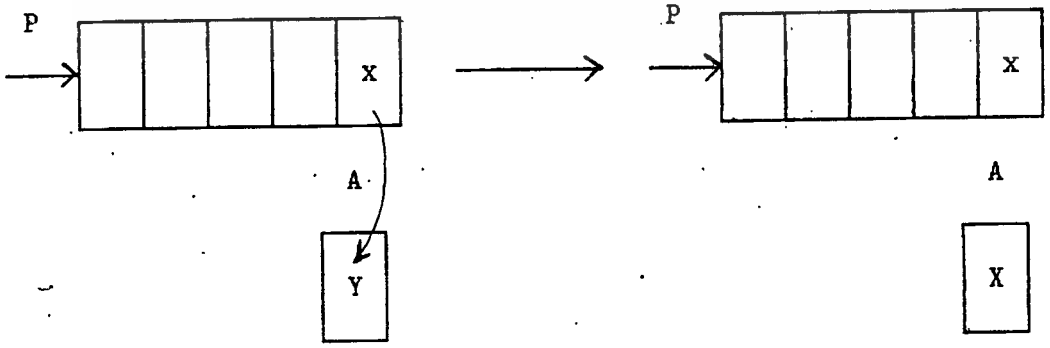
รูปที่ 6.3 การทำงานของ LPTR(pointer)

6.3.4) RPTR(pointer): สามารถอธิบายการทำงานได้ดังรูปที่ 6.4



(ก) การเก็บค่าจากตัวแปร A ลงใน Right Pointer

RPTR(P) = A



(ก) การเก็บค่าจาก Right Pointer ลงในตัวแปร A

A = RPTR(P)

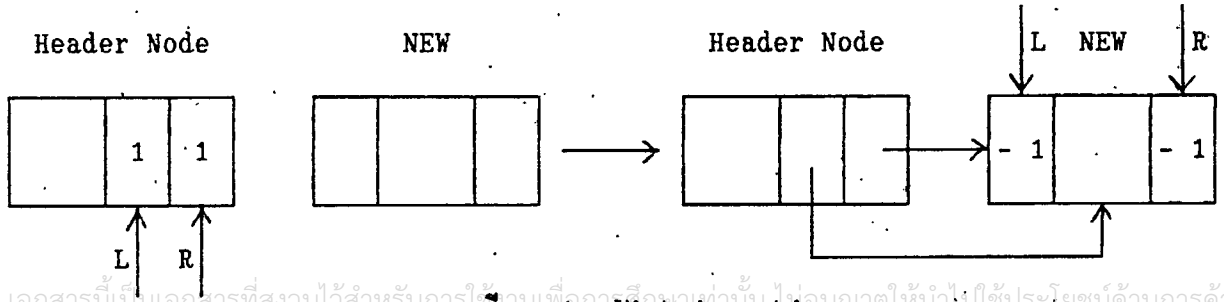
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.4 การทำงานของ RPTR(pointer)

6.3.5) INSERT(pointer): คำสั่งนี้จะทำงานได้ก็ต่อเมื่อ ได้มีการบันทึกค่าของ Left Pointer, Right Pointer และ ข้อมูลต่าง ๆ ลงไปเรียบร้อยแล้ว จากนั้นจะทำงานตาม algorithm ต่อไปนี้ โดยถ้าให้ L = Left-most Node Pointer และ R = Right-most Node Pointer จะได้

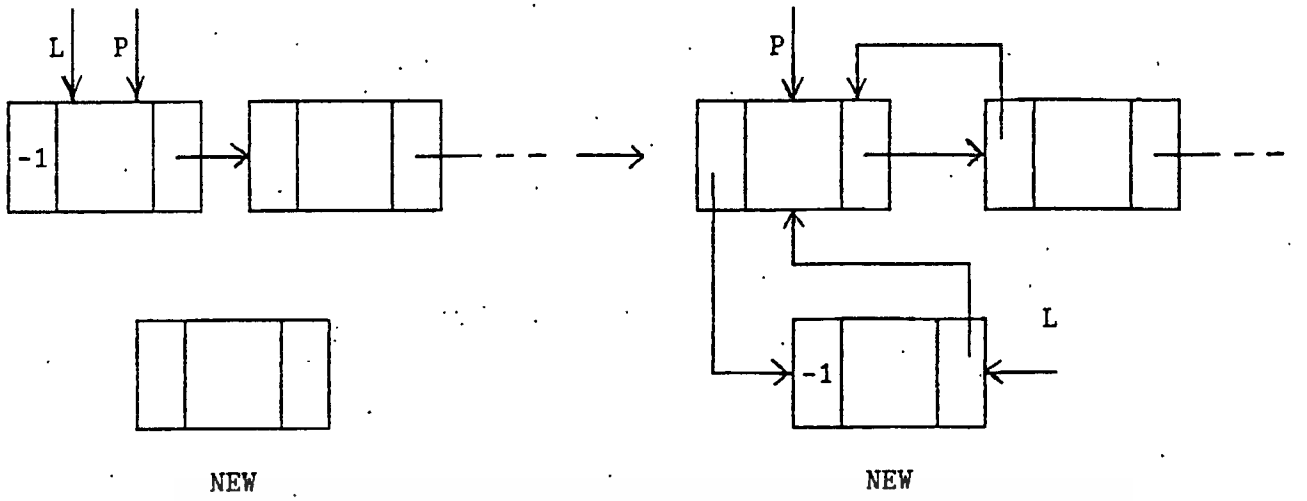
```
1. [Obtain new Node from Last Accessed Node field]
     NEW = New Node Number
2. [Copy information field]
     INFO(NEW, field) = information
3. [Insert into an empty list]
     IF R = null THEN
      LPTR(NEW) = RPTR(NEW) = null
      RETURN
4. [Left-most insertion ?]
     IF pointer = L THEN
      LPTR(NEW) = null
      RPTR(NEW) = pointer
      LPTR(pointer) = NEW
      L = NEW
      RETURN
5. [Insert in middle]
     LPTR(NEW) = LPTR(pointer)
     RPTR(NEW) = pointer
     LPTR(pointer) = NEW
     RPTR(LPTR(NEW)) = NEW
     RETURN
```

เพื่อความสะดวกในการทำความเข้าใจ ฉะนั้นจะอธิบายขั้นตอนการ insert ดังในรูปที่ 6.5 โดยแยกอธิบายเป็นกรณีต่าง ๆ ที่กำหนดใน algorithm ดังนี้

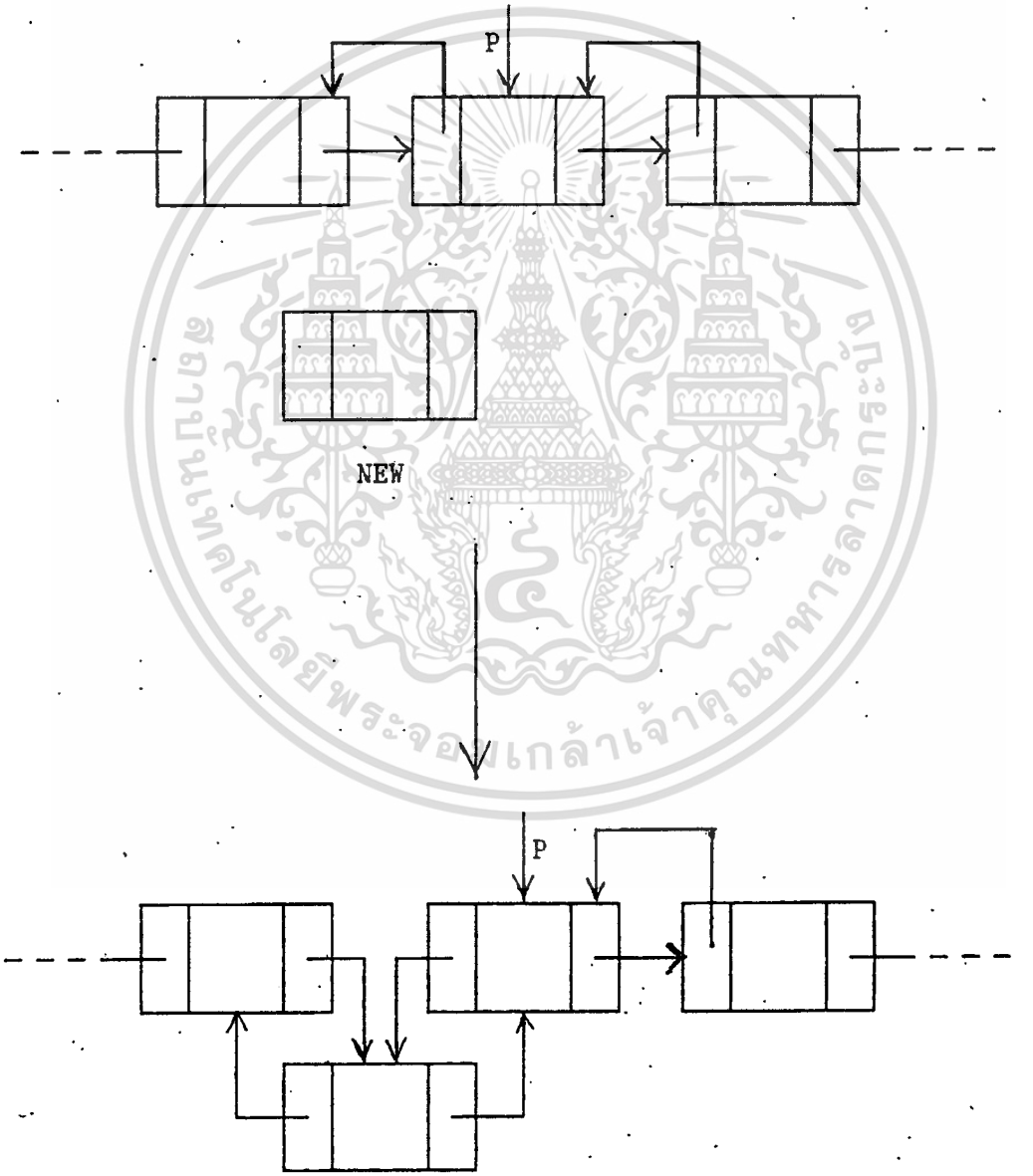


(ก) กรณี empty list insertion

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก) กรณี left-most insertion



(ค) กรณี middle insertion

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.5 การทำงานของ INSERT(pointer, field)

บทที่ 7

Main Menu Program

ดังที่ได้กล่าวมาแล้วในบทต้น ๆ ว่า โปรแกรมที่ใช้ในการจัดการควบคุมขั้นตอนการทำงานของ โมดูล โปรแกรมต่าง ๆ ของโปรแกรมเชียนแบบอย่างง่ายนี้ก็คือ Main Menu Program นั่นเอง ซึ่งในบทนี้จะเป็น การอธิบายขั้นตอนและหลักการทำงานของ Main Menu Program และเพื่อความเข้าใจที่ถูกต้องจงพิจารณา โปรแกรม MAINMENU.BAS ในภาคผนวก ข.1 ประกอบไปด้วย จะเห็นว่าขั้นตอนและหลักการทำงานต่าง ๆ จะสามารถอธิบายได้ดังนี้

1) เริ่มต้นด้วยการ include โปรแกรมที่เป็น function และ procedure เข้ามาใช้ในโปรแกรม รวมทั้งโปรแกรมที่เป็น subroutine ที่ระบุในท้ายโปรแกรมเข้ามาใช้งานด้วย

2) กำหนดชนิดของตัวแปร, ค่าของตัวแปร และค่าคงที่ต่าง ๆ ที่จะใช้ในโปรแกรม

3) ทำการ set up จอภาพดังที่แสดงไว้ในบทที่ 3 (รูปที่ 3.1)

4) สร้าง cursor image ขึ้นและนำไปเก็บไว้ที่ temperary array เพื่อนำไปใช้งานต่อไป

5) รับคำสั่งผ่าน command prompt จากแป้นพิมพ์

6) ตรวจสอบชนิดของคำสั่ง และส่งการทำงานตามคำสั่งนั้น ๆ ไปยังโปรแกรม INSTRUCT.INC

7) กรณีที่ชนิดของคำสั่งไม่ถูกต้อง ก็จะแจ้งข้อผิดพลาดด้วยข้อความ "??? Unknown Command ???"

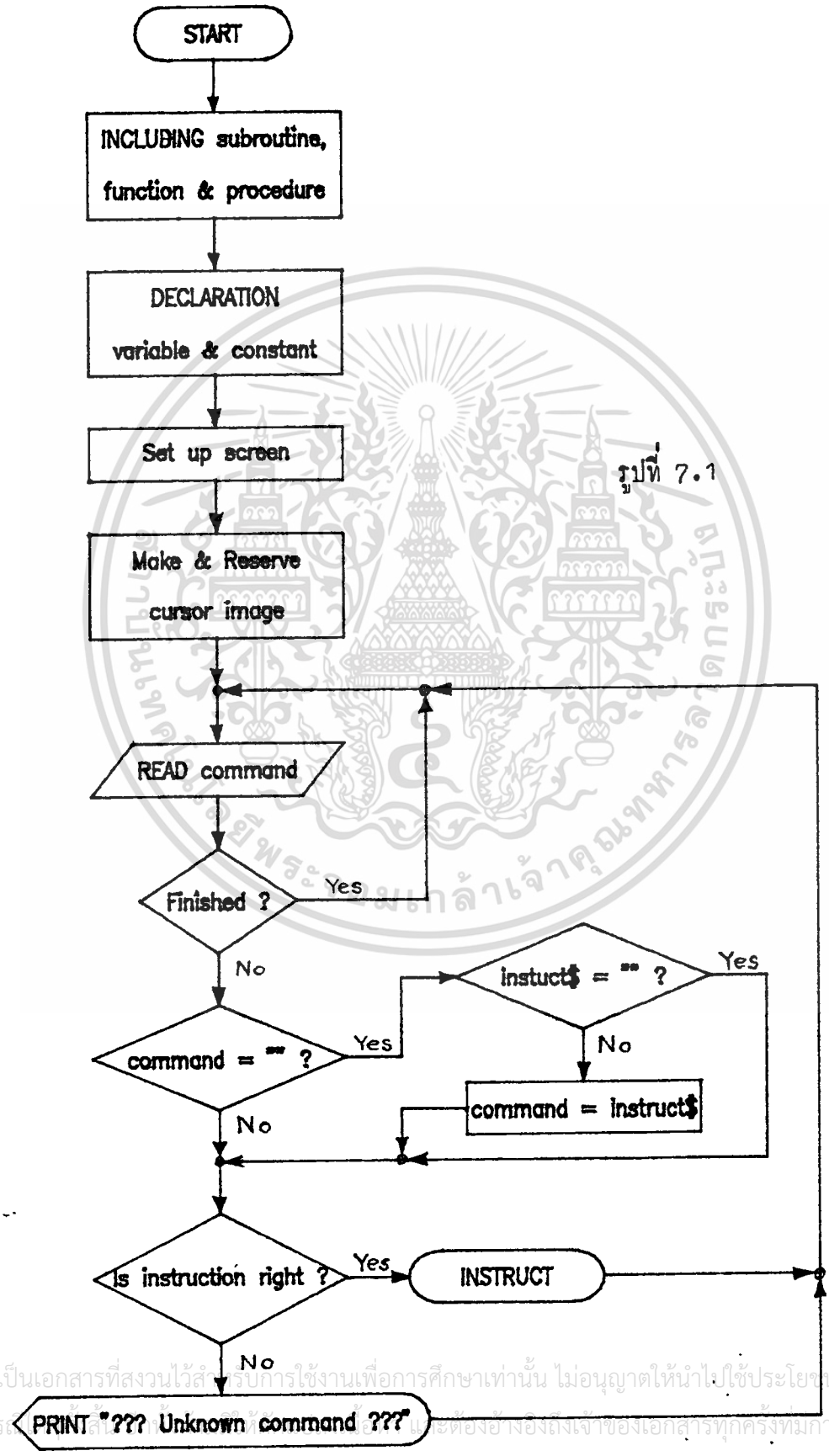
8) ส่วนกรณีที่มีการกด spacebar หรือ enter ที่ command prompt โดยไม่ได้ระบุคำสั่งใด ๆ ลง ไปเลยนั้น Main Menu Program จะถือว่า เป็นการเรียกใช้คำสั่งสุดท้ายที่เรียกใช้ไปแล้วขึ้นมาใช้อีกครั้ง

9) และขณะที่มีการทำงานในคำสั่งใด ๆ ถ้ามีความผิดพลาดจากการรับข้อมูลเกิดขึ้น ก็จะแจ้งข้อความ "<* Invalid *>" ให้ผู้ใช้ทราบ

10) โปรแกรมเชียนแบบอย่างง่ายจะทำงานจนกว่า QUIT-Instruction จะถูกเรียกใช้งาน

จากหลักการและขั้นตอนการทำงานทั้ง 10 ข้อสามารถนำมาเขียนเป็น ผังการทำงาน ดังจะแสดงได้ใน รูปที่ 7.1 ซึ่งเป็นผลให้การทำงานของโปรแกรมเชียนแบบอย่างง่ายเป็นจริงตาม ระบบที่กำหนดไว้ในรูปที่ 2.1 โดยที่ผู้ใช้จะติดต่อกับโปรแกรมให้ทำงานใด ๆ ผ่าน command prompt เท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
บทสรุปและวิจารณ์

จากการดำเนินงานที่ผ่านมาทั้งหมดบนพื้นฐานของหลักการที่ได้กล่าวมาแล้วนั้น ทำให้ทราบว่าในการพัฒนาโปรแกรมเขียนแบบอย่างง่ายบน BASIC Compiler นั้นที่เรื่องที่ลำบากที่เดียว อันเนื่องมาจากขอบเขต, ข้อจำกัด และประสิทธิภาพของ compiler ภาษานี้ที่เกี่ยวข้องกับ การจัดการข้อมูล, การสงวนเนื้อที่สำหรับตัวแปรประเภทต่าง ๆ , ความยุ่งยากในการกำหนดชนิดของตัวแปร รวมทั้งการส่งผ่านค่าของข้อมูลต่าง ๆ ผ่านตัวแปร ซึ่งต้องใช้เวลาพอจะแก้ไขปัญหาเหล่านี้ แต่ก็ไม่สามารถที่แก้ไขปัญหาได้ทั้งหมด โดยที่ยังคงเหลือปัญหาเกี่ยวกับการจัดการข้อมูลของ Data Processing routine ที่ยังไม่สามารถพัฒนาให้ใช้งานได้ ซึ่งในเรื่องมีปัจจัยอื่นที่เกี่ยวของด้วยอันได้แก่ การที่ผู้จัดทำขาดประสบการณ์และความเข้าใจอย่างลึกซึ้งในเรื่องการจัดการข้อมูล จึงทำให้ไม่สามารถพัฒนาโปรแกรมเขียนแบบอย่างง่ายได้สมบูรณ์ตามที่ตั้งใจไว้ได้

นอกจากนี้สิ่งสำคัญที่ทำให้เกิดความล่าช้าในการพัฒนาการจัดการข้อมูลก็คือ การกำหนดมาตราฐานที่จะใช้สำหรับการกำหนด โครงสร้างข้อมูล สำหรับเรื่องนี้ในขั้นต้นนั้นได้มีการขอข้อมูลจากผู้เชี่ยวชาญต่างประเทศโดยผ่านทางอาจารย์ที่ปรึกษา ซึ่งทำให้ต้องรอเป็นเวลานานพอสมควรแล้ว แต่ก็ไม่ได้รับข้อมูลใด ๆ ที่ขอไป จึงต้องทำการกำหนด โครงสร้างข้อมูล ขึ้นมาเองโดยที่ไม่สามารถจะอาศัยมาตราฐานใดมาอ้างอิงได้

จะเห็นได้ว่าปัญหาทั้งหมดนั้นจะมีแต่เฉพาะในส่วนเกี่ยวกับ การจัดการข้อมูล เท่านั้น โดยในส่วนของการประเมินผลและแสดงภาพนั้นเป็นไปตามเป้าหมายที่กำหนดไว้ในขั้นนี้ซึ่งก็ยังขาดความสมบูรณ์อยู่ ดังนั้นในส่วนของผู้จัดทำจึงหวังว่า เมื่อจะมีการพัฒนาโปรแกรมเขียนแบบอย่างง่ายต่อไปให้มีประสิทธิภาพที่ดีขึ้นควรจะต้องจัดการในสิ่งต่าง ๆ ต่อไปนี้

1. ศึกษาหลักการเกี่ยวกับการสร้างภาพให้ถ่องแท้
2. จัดหา compiler ที่มีประสิทธิภาพและคล่องตัวสำหรับพัฒนาโปรแกรม
3. ควรมีความรู้และความเข้าใจเกี่ยวกับการจัดการข้อมูล
4. ควรหามาตราฐานสากลในการกำหนดโครงสร้างข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้ ถ้าขาดซึ่งความกรุณาของ พี่มณฑา(อาจารย์ มณฑา เทียมเมือง) และ อาจารย์ จำลอง ปราบแก้ว จึงขอขอบพระคุณมา ณ. ที่นี้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3-UB / 4-LB	0	1	2	3	4	5	6	7
0	NUL	DEL	SPC	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

ตาราง ก.1 รหัส ASCII

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4-LB \ 4-UB	0	1	2	3	4	5
0	Reserve	Q	D	B	F6	2 DnArrow
1	Esc	W	F	N	F7	3 PgDn
2	! 1	E	G	M	F8	0 Ins
3	@ 2	R	H	< ,	F9	. Del
4	# 3	T	J	> .	F10	Reserve
5	$ 4	Y	K	? /	Num Lock	Reserve
6	% 5	U	L	R Shift	Scroll Lk	Reserve
7	^ 6	I	: ;	PrtScn	7 Home	F11
8	& 7	O	" '	Alt	8 UpArrow	F12
9	* 8	P	~ `	Spacebar	9 PgUp	Reserve
A	(9	{ [	L Shift	Caps Lock	-	Reserve
B	) 0	}]	| \	F1	4 LtArrow	Reserve
C	_ -	Enter	Z	F2	5	Reserve
D	+ =	Ctrl	X	F3	6 RtArrow	Reserve
E	Backspace	A	C	F4	+	Reserve
F	L/R Arrow	S	V	F5	1 End	Reserve

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตาราง ก.2 Scan Code

ภาคผนวก ข.

ประกอบด้วย:

ข.1) Main Menu (Main Program)

ข.2) Function and Procedure of Main Program

ข.3) Subroutine of Main Program

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก.1) Main Menu (Main Program)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                              '
' This routine is used for managing program instruction and being MAIN MENU.  '
'                                                                              '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'---------------------- included function & procedure ------------------------'
$include "a:\tbas\project\new\subang.inc"
$include "a:\tbas\project\new\arcuty.inc"
$include "a:\tbas\project\new\ltype.inc"
$include "a:\tbas\project\new\colour.inc"
$include "a:\tbas\project\new\subline.inc"
$include "a:\tbas\project\new\subarc.inc"
'-----------------------------------------------------------------------------'
 DEFDBL r, s, e
 %xmax = 639                : %ymax = 359
 %xmin = 0                  : %ymin = 0
 deltx = 1                  : delty = 1
 style% = &B1111111111111111 : style$ = "CONTINUE"
 colour  = 7                : colour$ = "WHITE"
 instruct = 0               :instruct$ = ""
'-----------------------------------------------------------------------------'
gosub scrsetup
gosub makecur
'------------------------- first state user interface ------------------------'
locate 25, 1: print "command: ";
startUI:
 inputkey$ = inkey$
 gosub directip
 row = csrlin
 column = pos
 gosub movecur
 gosub ctrlkey
 if argt$ < > "" then
   instruct = 1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
select case ucase$(argt$)
  case "LINE"
    instruct$ = "LINE"
    gosub scrollup
    gosub linint
    exit select
  case "CIRCLE"
    instruct$ = "CIRCLE"
    gosub scrollup
    gosub cirint
    exit select
  case "ARC"
    instruct$ = "ARC"
    gosub scrollup
    gosub arcint
    exit select
  case "LINETYPE", "LTYPE"
    instruct$ = "LINETYPE"
    gosub scrollup
    gosub ltpint
    exit select
  case "COLOUR"
    instruct$ = "COLOUR"
    gosub scrollup
    gosub clrint
    exit select
  case "STATUS"
    instruct$ = "STATUS"
    gosub scrollup
    gosub stsint
    exit select
  case "QUIT"
    instruct$ = "QUIT"
    gosub scrollup
    gosub break
    exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    case else
      gosub scrollup
      locate 25, 1: print "??? Unknown command ???";
      argt$ = ""
  end select
 end if
'------------------------------ input routine -------------------------------'
startIP:
 select case flag%
  case = 1
    if instruct = 1 then
     gosub scrollup
     locate 25, 1: print "(* cancel *)";
     gosub scrollup
     locate 25, 1: print "command: ";
     exit select
    end if
    if instruct$ < > "" then
     argt$ = instruct$
     locate row, column: print instruct$;
     exit select
    end if
  case 2, 3
    if instruct = 0 then
     locate row, column
     exit select
    end if
    gosub scrollup
    locate 25, 1: print "command: ";
    exit select
  case = 4
    gosub scrollup
    locate 25, 1: print "(* Invalid *)";
    gosub scrollup
    locate 25, 1: print "command: ";
    exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  end select
 instruct = 0
 flag% = 0
 goto startUI
end
'------------------------------------------------------------------------------'
'---------------------------- included subrotine ------------------------------'
$include "a:\tbas\project\utility.inc"
$include "a:\tbas\project\subdrt.inc"
$include "a:\tbas\project\subcur.inc"
$include "a:\tbas\project\subctrl.inc"
$include "a:\tbas\project\instruct.inc"
'------------------------------------------------------------------------------'
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก.2) Function and Procedure of Main Program

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DEF FNangle#(xs, ys, xe, ye)

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
'    This routine is used for calculating an angle in unit of degree.          '
'    To calculate the angle it require for: (1) started point [xs,ys]          '
'                                           (2) ended point   [xe,ye]          '
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
 pi# = 4 * atn(1)
 deltax# = xe - xs
 deltay# = ye - ys
 if deltax# = 0 then theta# = sgn(deltay#) * 90: goto ang
 if deltay# = 0 and deltax# < 0 then theta# =180: goto ang
 if deltay# = 0 and deltax# > 0 then theta# = 0: goto ang
 if deltax# = 0 and deltay# = 0 then theta# = 0: goto ang
 theta# = (180 / pi#) * atn(deltay# / deltax#)
 if deltax# < 0 and deltay# > 0 then theta# = 180 + theta#
 if deltax# < 0 and deltay# < 0 then theta# = theta# - 180
'------------------------------------------------------------------------------'
 ang:
  IF theta# < 0 THEN theta# = theta# + 360
  FNangle# = theta#
END DEF
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SUB sclarc(s#, e#, r#, lc#)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    This routine is used for calculating end angle of arc following given   '
'    data  start point, center point and lenght of chord.                    '
'                                                                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'---------------------------- calculate ended angle -------------------------'
alpha# = (r# ^ 2) - ((lc# ^ 2) / 4)
IF alpha# = 0 then
 e# = s# + 360
 GOTO sclend
END IF
th# = 2 * ATN((lc# / 2) / SQR(alpha#))
th# = (180 * th#) / (4 * ATN(1))
e# = s# + th#
s# = s# - (s# \ 360)
e# = e# - (e# \ 360)
IF s# > e# THEN
 buff# = e#
 e# = s#
 s# = buff#
END IF
sclend:
'----------------------------------------------------------------------------'
END SUB
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SUB threepoint(xi, yi, xj, yj, xk, yk):

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
'                                                                           '
'    This routine is used for calculating center point, radius, started    '
'    and ended angle of curve, it require for: (1) started point [xi,yi]    '
'                                              (2) second point  [xj,yj]    '
'                                              (3) ended point   [xk,yk]    '
'                                                                           '
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
SHARED x0, y0, r, s, e
'-------------------------- calculate center point -------------------------'
 pi = 4 * atn(1)
 a11 = 2 * (xj - xi): a12 = 2 * (yj - yi): b1 = yj^2 + xj^2 - yi^2 - xi^2
 a21 = 2 * (xk - xi): a22 = 2 * (yk - yi): b2 = yk^2 + xk^2 - yi^2 - xi^2
 x0 = (b1 * a22 - b2 * a12) / (a11 * a22 - a21 * a12)
 y0 = (b2 * a11 - b1 * a21) / (a11 * a22 - a21 * a12)
'------------------------------ calculate radius ----------------------------'
 r = (((xi - x0) ^ 2) + ((yi - y0) ^ 2)) ^ 0.5
'--------------------------- calculate started angle -----------------------'
 s = FNangle#(x0, y0, xi, yi)
'---------------------------- calculate ended angle -------------------------'
 e = FNangle#(x0, y0, xf, yf)
'------------------------------- sequancy angle -----------------------------'
 cent = FNangle#(x0, y0, xj, yj)
 if cent < s then
   buffer = s
   s = e
   e = buffer
 end if
'---------------------------------------------------------------------------'
END SUB
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DEF FNltype%(prestyle$)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                            '
'   This routine is used for converting style$ to code of style line        '
'                                                                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SHARED style%, flag%
'----------------------------- line style defining --------------------------'
 oldstyle% = style%
 select case ucase$(prestyle$)
   case ""
     FNltype% = oldstyle%
   case "CONTINUE"
     FNltype% = &B1111111111111111
   case "DOTTED"
     FNltype% = &B1000000010000000
   case "HIDDEN"
     FNltype% = &B1111000011110000
   case "DASHED"
     FNltype% = &B1111111111110000
   case "DASHDOT"
     FNltype% = &B1111111110001000
   case "DEVIDED"
     FNltype% = &B1111111100100100
   case else
     FNltype% = oldstyle%
     flag% = 4
 end select
'----------------------------------------------------------------------------'
END DEF
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DEF FNcolor(ncolour$)

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
'                                                                            '
'   This routine is used for convertig colour$ to colour code.               '
'                                                                            '
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
SHARED newcolour$, colour, flag%
'---------------------------- colour code defining --------------------------'
 oldcolor = colour
 select case ucase$(ncolour$)
   case ""
     FNcolor = oldcolor
   case "0",  "BLACK"
     FNcolor = 0
     newcolour$ = "BLACK"
   case "1",  "BLUE"
     FNcolor = 1
     newcolour$ = "BLUE"
   case "2",  "GREEN"
     FNcolor = 2
     newcolour$ = "GREEN"
   case "3",  "CYAN"
     FNcolor = 3
     newcolour$ = "CYAN"
   case "4",  "RED"
     FNcolor = 4
     newcolour$ = "RED"
   case "5",  "MAGENTA"
     FNcolor = 5
     newcolour$ = "MAGENTA"
   case "6",  "BROWN"
     FNcolor = 6
     newcolour$ = "BROWN"
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  case "7",  "WHITE"
    FNcolor = 7
    newcolour$ = "WHITE"
  case "8",  "GRAY"
    FNcolor = 8
    newcolour$ = "GRAY"
  case "9",  "LIGHTBLUE"
    FNcolor = 9
    newcolour$ = "LIGHTBLUE"
  case "10", "LIGHTGREEN"
    FNcolor = 10
    newcolour$ = "LIGHTGREEN"
  case "11", "LIGHTCYAN"
    FNcolor = 11
    newcolour$ = "LIGHTCYAN"
  case "12", "LIGHTRED"
    FNcolor = 12
    newcolour$ = "LIGHTRED"
  case "13", "LIGHTMAGENTA"
    FNcolor = 13
    newcolour$ = "LIGHTMAGENTA"
  case "14", "YELLOW"
    FNcolor = 14
    newcolour$ = "YELLOW"
  case "15", "BRIGHTWHITE"
    FNcolor = 15
    newcolour$ = "BRIGHTWHITE"
  case else
    FNcolor = oldcolor
    flag% = 4
 end select
'-------------------------------------------------------------------------'
END DEF
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SUB linef(xinl, yinl, xfnl, yfnl)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                        '
'   This routine is used for drawing line with required parameters:     '
'                          (1) x1, y1: initial position(started point)   '
'                          (2) x2, y2: final position(ended point)       '
'                          (3) c     : colour code                       '
'                          (4) style%: line style type code              '
'                                                                        '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SHARED colour, style%
'------------------------------- line drawing ----------------------------'
 line (xinl, yinl) - (xfnl, yfnl), colour,, style%
'-------------------------------------------------------------------------'
END SUB
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SUB arcf(xcen, ycen, radius#, stang#, edang#)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                              '
'    This program need data of: (1) center of curve [xcen,ycen : integer]    '
'                               (2) radius# of curve [radius# : real]         '
'                               (3) start angle (in degree) [stang# : real]   '
'                               (4) final angle (in degree) [edang# : real]   '
'                                                                              '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SHARED colour, style%
pi# = 4 * ATN(1)
'------------------------------- plot  curve ----------------------------------'
ty$ = "arc"
 IF stang# < 0 THEN stang# = stang# MOD 360
 IF stang# > 360 THEN stang# = stang# MOD 360
 IF edang# < 0 THEN edang# = edang# MOD 360
 IF edang# > 360 THEN edang# = stang# MOD 360
 IF stang# > edang# THEN edang# = edang# + 360
 IF stang# <> edang# OR stang# - edang# = 360 THEN plot
 edang# = stang# + 360: ty$ = "cyc"
plot:
 IF stang# > edang# THEN stang# = stang# - 360
 n = INT(ABS(stang# - edang#) * (20 + 2 * radius#) / 180)
 dt# = (edang# - stang#) / n
 stang# = pi# * stang# / 180
 edang# = pi# * edang# / 180
 dt# = pi# * dt# / 180
 x = xcen + radius# * COS(stang#)
 y = ycen + radius# * SIN(stang#)
 xbuf = x: ybuf = y
 x1 = x: y1 = y
 PSET (x1,y1), colour
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 FOR k = 1 TO n
  x = xcen + radius# * COS(stang# + k * dt#)
  y = ycen + radius# * SIN(stang# + k * dt#)
  CALL linef(x1, y1, x, y)
  x1 = x: y1 = y
 NEXT k
 IF ty$ = "arc" THEN
    xf = xcen + radius# * COS(edang#)
    yf = ycen + radius# * SIN(edang#)
  ELSE
    xf = xbuf
    yf = ybuf
 END IF
 CALL linef(x1, y1, xf, yf)
'-------------------------------------------------------------------------'
END SUB
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

M.3) Subroutine of Main Program

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
scrsetup:

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                            '
'      This routine is used for setting up the following values:             '
'             (1) screen mode 9: resolution 640 x 350 pixels / 16 colours    '
'             (2) viewport     : - define graphics screen                    '
'                                - book 1-line text at top of screen         '
'                                - book 3-lines text at bottom of screen     '
'             (3) limitation   : define coordinate system with 640 x 360     '
'                                which has origin(0,0) at upper left-hand    '
'             (4) Position     : Integer value of present coordinate which   '
'                                is used for display on upper right-hand     '
'                                                                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SCREEN 9
VIEW (0, 14)-(639, 307)
WINDOW (0, -1)-(640, 359)
'----------------------------------------------------------------------------'
RETURN


clrkbd:

'--------------------------- clear keyboard buffer --------------------------'
 DEF SEG = 0
 POKE &H041A, PEEK(&H041C)
 DEF SEG
 POKE &H006A, 0
'----------------------------------------------------------------------------'
RETURN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
scrollup:

'------------------------------ register name -------------------------------'
%AX = 1:                        %BX = 2
%CX = 3:                        %DX = 4
'---------------------------- scroll up 1 line ------------------------------'
 REG %AX,&H0601          :' mov ah,06h           ;scroll up function
:' mov al,01h            ;one line scroll up
 REG %CX,&H1600          :' mov ch,17h           ;upper left row
:' mov cl,00h            ;upper left column
 REG %DX,&H194F          :' mov dh,19h           ;lower right row
:' mov dl,4Fh            ;lower right column
 REG %BX,&HF000          :' mov bh,F0h           ;normal attribute
 CALL INTERRUPT &H10     :' int 10h              ;veido ROM call
'--------------------------------------------------------------------------'
RETURN



makecur:

'---------------------------- make arrow cursor -----------------------------'
sz = 25: xm = 9: ym = 9
DIM cur%(sz): DIM rep%(sz)
GET (x, y + ym) - (x + xm, y), rep%
 LINE (x, y) - (x + 5, y + 2), 9
 LINE - (x + 2, y + 5), 9
 LINE - (x, y), 9
 PAINT (x + 3, y + 3), 9, 9
 LINE (x + 4, y + 4)-(x + 9, y + 9), 9
GET (x, y + ym)-(x + xm, y), cur%
PUT (x, y + ym), rep%, PSET
CLS
'--------------------------------------------------------------------------'
RETURN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
directip:

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,                                                                            ,
'      This routine is used for checking directly input with take value     '
'      by pressing Numeric & Alphabic Keys. And its' result is in:          '
'              (1) arg$ : passing parameter which defined type by argtype$  '
'              (2) argtype$ = "num" if arg$ is numeric parameter.           '
'                           = "chr" if arg$ is alphabic parameter.          '
,                                                                            ,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
'----------------------------- input character -------------------------------'
inchr$ = inputkey$
IF LEN(inchr$) <> 1 THEN endkbdchk
FOR codeii = 0 TO 255
 IF inchr$ = CHR$(codeii) THEN
   inasc% = codeii
   EXIT FOR
 END IF
NEXT codeii
'--------------------------- first character checking ------------------------'
IF argtype$ < > "" THEN syntaxchk
SELECT CASE inasc%
  CASE &H30 TO &H39, &H2E, &H2D, &H2B
    argtype$ = "num"
    IF argt$ < > "" THEN argt$ = ""
    EXIT SELECT
  CASE &H21 TO &H29, &H40 TO &H7F
    argtype$ = "chr"
    EXIT SELECT
  CASE ELSE
    bckspc = 1
    argtype$ = ""
    GOTO endkbdchk
END SELECT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'------------------------------ syntax checking -----------------------------'
syntaxchk:
 SELECT CASE inasc%
   CASE = &H08:                    ' in case of backspace
     GOSUB backspc
     EXIT SELECT
   CASE ELSE
     GOSUB defarg
  END SELECT
 GOTO endkbdchk
'------------------------------ backspace process ---------------------------'
backspc:
 arg$ = LEFT$(arg$, LEN(arg$) - 1)
 IF LEN(arg$) = 0 THEN
  arg$ = "": argtype$ = ""
 END IF
 posmin = POS - 1
 LOCATE 25, posmin: PRINT CHR$(&H20);
 LOCATE 25, posmin
back1:
 bckspc = 1
RETURN
'------------------------------ argument defination -------------------------'
defarg:
 direct = 1
 arg$ = arg$ + inchr$
RETURN
'--------------------------------------------------------------------------'
endkbdchk:
 IF bckspc = 0 THEN
  IF inchr$ > CHR$(&H1F) THEN
   LOCATE 25, POS: PRINT inchr$;
  END IF
 END IF
 bckspc = 0
RETURN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
movecur:

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
'    This routine is used for changing position by inputting from keybroad   '
'    and displaying new position with the value on top right of screen and    '
'    the moving cursor on viewport of screen.                                 '
'                                                                             '
'    Initial parameter: speed = 0 ---> PgUp/PgDn has never pressed before     '
'    Initial constant depend to screen mode: (o,p) ---> real screen point     '
'                                            deltx ---> change in X-axis      '
'                                            delty ---> change in Y-axis      '
'                                                                             '
'    REMARK: 1) deltx and delty must be 'zero' in the head of main program    '
'            2) before call this routine it had to pass statement:            '
'                                                                             '
'                    PUT (x, y + ym), cur%, OR                                '
'                                                                             '
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
o = x: p = y: oldO = x: oldP = y
'---------------------------- check NUM LOCK key -----------------------------'
DEF SEG = &H0000
numlock = PEEK(&H0417)
DEF SEG
numflag = numlock AND &B00100000
IF numflag < > 0 then exitip
'------------------------------- input  loop ---------------------------------'
IF cursor = 0 THEN
  rowc = CSRLIN
  columnc = POS
END IF
SELECT CASE x
  CASE < %xmin
    x = %xmin
  CASE > %xmax - xm
    x = %xmax - xm
END SELECT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SELECT CASE y
  CASE < %ymin
    y = %ymin
  CASE > %ymax - ym
    y = %ymax - ym
END SELECT
 IF INP(&H60) = &H48 THEN up
 IF INP(&H60) = &H50 THEN down
 IF INP(&H60) = &H4B THEN left
 IF INP(&H60) = &H4D THEN right
 IF INP(&H60) = &H49 THEN fast
 IF INP(&H60) = &H51 THEN slow
GOTO exitip
'------------------------------- cursor up --------------------------------'
up:
 DELAY (.075)
 eup = p + delty
 IF eup > %ymax - ym THEN up1
 p = eup
 GOSUB display
up1:
 GOTO exitip1
'------------------------------ cursor down -------------------------------'
down:
 DELAY (.075)
 edn = p - delty
 IF edn < %ymin THEN down1
 p = edn
 GOSUB display
down1:
 GOTO exitip1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'------------------------------ cursor left --------------------------------'
left:
 DELAY (.075)
 elt = o - deltx
 IF elt < %xmin THEN left1
 o = elt
 GOSUB display
left1:
 GOTO exitip1
'------------------------------ cursor right -------------------------------'
right:
 DELAY (.075)
 ert = o + deltx
 IF ert > %xmax - xm THEN right1
 o = ert
 GOSUB display
right1:
 GOTO exitip1
'-------------------------------- speed up ---------------------------------'
fast:
 DELAY (.25)
 bufx = deltx
 deltx = deltx * 5
 IF deltx > 125 THEN deltx = bufx
 delty = deltx
 speed = 1
 GOSUB clrkbd
fast1:
  GOTO exitip
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'-------------------------------- speed down --------------------------------'
slow:
 DELAY (.25)
 bufy = delty
 delty = delty / 5
 IF delty < 1 THEN delty = bufy
 deltx = delty
 speed = 1
 GOSUB clrkbd
slow1:
  GOTO exitip
'------------------------------- cursor display -----------------------------'
display:
 LOCATE 1, 61: PRINT SPC(19);
 LOCATE 1, 61: PRINT o; ","; p
 PUT (old0, oldP + ym), cur%, OR
 PUT (old0, oldP + ym), rep%, PSET
 GET (o, p + ym)-(o + xm, p), rep%
 PUT (o, p + ym), cur%, OR
RETURN
'-------------------------------- exit input --------------------------------'
exitip1:
 cursor = 1
 x = o:          ' give value of x-coordinate
 y = p:          ' give value of y-coordinate
 argtype$ = "num"
 inputkey$ = ""
GOSUB clrkbd
'------------------------------------------------------------------------------'
exitip:
RETURN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ctrlkey:

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                                              '
'       This routine is for checking end input data and send back value of    '
'       required parameters which consist of:                                 '
'               (1) argt$ : general string argument                            '
'               (2) argt# : general numeric argument                           '
'               (3)   x   : x-coordinate position                              '
'               (4)   y   : y-coordinate position                              '
'               (5) aval# : angle value                                        '
'               (6) flag%: flag argument which:                                '
'                     0 ---> No set anything                                   '
'                     1 ---> Cancel Routine                                    '
'                     2 ---> SpaceBar Pressed                                  '
'                     3 ---> Enter Pressed                                     '
'                     4 ---> Invalid Input                                     '
'                                                                              '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
flag% = 0
ctrl$ = inputkey$
SELECT CASE ctrl$
  CASE = CHR$(&H03):              '      Ctrl C checking
    GOTO cancel
  CASE = CHR$(&H0D):              '      Carry Return checking
    flag% = 3
    GOTO ctrlchk
  CASE = CHR$(&H20):              '      Space Bar checking
    flag% = 2
    GOTO ctrlchk
END SELECT
GOTO exitctrl
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'------------------------- argument type checking --------------------------'
ctrlchk:
 IF cursor = 1 THEN
  PUT (oldO, oldP + ym), rep%, PSET
  GOTO exitctrl
 END IF
 SELECT CASE argtype$
   CASE = ""
     GOTO cancel
   CASE = "num"
     charchk:
      char$ = LEFT$(arg$, 1)
      arg$ = RIGHT$(arg$, LEN(arg$) - 1)
      SELECT CASE ASC(char$)
        CASE = &H2C
          comma = 1
          x = VAL(argt$)
          test# = x
          GOSUB invalidchk
          IF flag% = 4 THEN
            GOTO exitctrl
           ELSE
            GOTO chrchk2
          END IF
        CASE = &H3C
          great = 1
          argt# = VAL(argt$)
          test# = argt#
          GOSUB invalidchk
          IF flag% = 4 THEN
            GOTO exitctrl
           ELSE
            GOTO chrchk2
          END IF
      END SELECT
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      argt$ = argt$ + char$
      IF LEN(arg$) > 0 THEN charchk
      argt$ = LEFT$(argt$, LEN(argt$) - 1)
      argt# = VAL(argt$)
      test# = argt#
      GOSUB invalidchk
      argt$ = ""
      GOTO exitctrl
      chrchk2:
       arg$ = LEFT$(arg$, LEN(arg$) - 1)
       argt$ = arg$
       IF comma = 1 THEN
        comma = 0
        y = VAL(argt$)
        test# = y
        GOSUB invalidchk
       END IF
       IF great = 1 THEN
        great = 0
        aval# = VAL(argt$)
        test# = aval#
        GOSUB invalidchk
       END IF
     argt$ = ""
     GOTO exitctrl
   CASE = "chr"
     argt$ = LEFT$(arg$, LEN(arg$) - 1)
     IF argt$ = "cancel" OR argt$ = "CANCEL" THEN
       GOTO cancel
     END IF
     GOTO exitctrl
 END SELECT
GOTO exitctrl
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'------------------------------ invalid input --------------------------------'
invalidchk:
 SELECT CASE LEFT$(argt$, 1)
   CASE "+", "-", " "
     differ = 0
   CASE ELSE
     differ = 1
 END SELECT
 numbf = LEN(argt$)
 test# = VAL(argt$)
 numaf = LEN(STR$(test#)) - differ
 test# = 0
 IF numbf < > numaf THEN flag% = 4
 RETURN
'---------------------------- cancel processing ----------------------------'
cancel:
 flag% = 1
'------------------------ exit control input routine ------------------------'
exitctrl:
 IF flag% <> 0 THEN
   IF direct = 1 THEN
     direct = 0
     cursor = 0
     PUT (old0, oldP + ym), rep%, PSET
     LOCATE 1, 61: PRINT SPC(19)
     LOCATE 1, 61: PRINT x; ","; y
   END IF
   LOCATE 1, 61: PRINT x; ","; y
   arg$ = ""
   argtype$ = ""
 END IF
 IF cursor = 1  AND direct = 0 THEN
   locate rowc, columnc
 END IF
RETURN
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'----------------------------- Instruction Set -----------------------------'

'---------------------------------- QUIT ----------------------------------'
break:
 locate 25,1: print "Do you really want to exit this program ? ";
 bkloop:
  inputkey$ = inkey$
  gosub directip
  gosub ctrlkey
  if flag% = 0 then bkloop
 quit$ = argt$
 select case ucase$(quit$)
   case "Y", "YES"
     end
   case "N", "NO"
     flag% = 2
     exit select
   case else
     if quit$ < > "" then flag% = 4
 end select
 argt$ = ""
'-----------------------------------------------------------------------'
return
'---------------------------------- STATUS ---------------------------------'
stsint:
 locate 25,1: print "Current LineType: "; style$;
 gosub scrollup
 locate 25,1: print "Current Colour: "; colour$; " <"; colour; ">";
 argt$ = ""
'-----------------------------------------------------------------------'
return
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'--------------------------------- LINETYPE ---------------------------------'
ltpint:
 oldstyle$ = style$
 locate 25,1: print "Name of linetype<"; style$; ">: ";
 ltloop:
  inputkey$ = inkey$
  gosub directip
  gosub ctrlkey
  if flag% = 0 then ltloop
 ltype$ = ucase$(argt$)
 style% = FNltype%(ltype$)
 if flag% = 4 then style$ = oldstyle$ else style$ = ltype$
 argt$ = ""
'--------------------------------------------------------------------------'
return
'---------------------------------- COLOUR ----------------------------------'
clrint:
 oldcolour$ = colour$
 locate 25,1: print "Current colour<"; colour$; ">";
 gosub scrollup
 locate 25,1: print "New colour: ";
 clrloop:
  inputkey$ = inkey$
  gosub directip
  gosub ctrlkey
  if flag% = 0 then clrloop
  if flag% = 1 then
    flag% = 3
    colour$ = oldcolour$
    goto colour1
  end if
 if argt$ = "" then
   argt$ = str$(argt#)
   argt$ = right$(argt$, len(argt$) - 1)
 end if
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 newcolour$ = ucase$(argt$)
 colour = FNcolor(newcolour$)
 if flag% = 4 then colour$ = oldcolour$ else colour$ = newcolour$
 colour1:
  argt$ = ""
'-------------------------------------------------------------------------'
return
'---------------------------------- LINE ---------------------------------'
linint:
 locate 25,1: print "From point: ";
 line1:
  inputkey$ = inkey$
  gosub directip
  gosub movecur
  gosub ctrlkey
  if flag% = 0 then line1
  if flag% = 1 then line5
 x1 = x: y1 = y
 xstart = x: ystart = y
 gosub scrollup
 CALL mark(x1, y1)
 line2:
  locate 25,1: print "To point: ";
  line3:
   inputkey$ = inkey$
   gosub directip
   gosub movecur
   gosub ctrlkey
   CALL mark(x1, y1)
   if flag% = 0 then line3
   if flag% = 1 then
     flag% = 3
     goto line5
   end if
  if ucase$(argt$) = "C" or ucase$(argt$) = "CLOSED" then line4
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  x2 = x: y2 = y
  CALL mark(x2, y2)
  if x1 = x2 and y1 = y2 then line5
 CALL linef(x1, y1, x2, y2)
 x1 = x2: y1 = y2
 gosub scrollup
 goto line2
 line4:
  CALL linef(x1, y1, xstart, ystart)
 line5:
  argt$ = ""
'-------------------------------------------------------------------------'
return
'------------------------------- CIRCLE ----------------------------------'
cirint:
 locate 25,1: print "Center/Start point<C/S>: ";
 circle1:
  inputkey$ = inkey$
  gosub directip
  gosub ctrlkey
  if flag% = 0 then circle1
 gosub scrollup
 select case ucase$(argt$)
   case "C", "CENTER"
     locate 25,1: print "Center: ";
     circle2:
      inputkey$ = inkey$
      gosub directip
      gosub movecur
      gosub ctrlkey
      if flag% = 0 then circle2
     x0 = x: y0 = y
     gosub scrollup
     CALL mark(x0, y0)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
locate 25,1: print "Radius/Diameter<R/D>: ";
circle3:
 inputkey$ = inkey$
 gosub directip
 gosub ctrlkey
 CALL mark(x0, y0)
 if flag% = 0 then circle3
gosub scrollup
select case ucase$(argt$)
  case "R", "RADIUS"
    locate 25,1: print "Radius: ";
    circle4:
     inputkey$ = inkey$
     gosub directip
     gosub movecur
     gosub ctrlkey
     CALL mark(x0, y0)
     if flag% = 0 then circle4
    CALL mark(x, y)
    if oldargt# = argt# then
      r = sqr(((x - x0) ^ 2) + ((y - y0) ^ 2))
     else
      r = argt#
    end if
    exit select
  case "D", "DIAMETER"
    locate 25,1: print "Diameter: ";
    circle5:
     inputkey$ = inkey$
     gosub directip
     gosub ctrlkey
     CALL mark(x0, y0)
     if flag% = 0 then circle5
    r = argt# / 2
     exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    case else
      flag% = 4
      goto circle11
  end select
  exit select
case "S", "START"
  locate 25,1: print "Start point: ";
  circle6:
   inputkey$ = inkey$
   gosub directip
   gosub movecur
   gosub ctrlkey
   if flag% = 0 then circle6
  x1 = x: y1 = y
  gosub scrollup
  CALL mark(x1, y1)
  locate 25,1
  print "End point in direction of diameter/Second point<E/S>: ";
  circle7:
   inputkey$ = inkey$
   gosub directip
   gosub ctrlkey
   CALL mark(x1, y1)
   if flag% = 0 then circle7
  gosub scrollup
  select case ucase$(argt$)
    case "E", "END"
      locate 25,1: print "End point: ";
      circle8:
       inputkey$ = inkey$
       gosub directip
       gosub movecur
       gosub ctrlkey
       CALL mark(x1, y1)
       if flag% = 0 then circle8
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    x2 = x: y2 = y
    CALL mark(x2, y2)
    r  = sqr(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2)) / 2
    x0 = (x1 + x2) / 2: y0 = (y1 + y2) / 2
    exit select
  case "S", "SECOND"
    locate 25,1: print "Second point: ";
    circle9:
     inputkey$ = inkey$
     gosub directip
     gosub movecur
     gosub ctrlkey
     CALL mark(x1, y1)
     if flag% = 0 then circle9
    x2 = x: y2 = y
    gosub scrollup
    CALL mark(x2, y2)
    locate 25,1: print "End point: ";
    circle10:
     inputkey$ = inkey$
     gosub directip
     gosub movecur
     gosub ctrlkey
     CALL mark(x2, y2)
     if flag% = 0 then circle10
    x3 = x: y3 = y
    CALL mark(x3, y3)
    CALL threepoint(x1, y1, x2, y2, x3, y3)
    r = sqr(((x1 - x0) ^ 2) + ((y1 - y0) ^ 2))
    exit select
  case else
    flag% = 4
    goto circle11
end select
exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   case else
     flag% = 4
     goto circle11
 end select
 CALL arcf(x0, y0, r, 0, 360)
 circle11:
  argt$ = ""
'-------------------------------------------------------------------------------'
return
'---------------------------------- ARC ----------------------------------'
arcint:
 s = 0: e = 0
 locate 25,1: print "Center/Start point<C/S>: ";
 arc1:
  inputkey$ = inkey$
  gosub directip
  gosub ctrlkey
  if flag% = 0 then arc1
 gosub scrollup
 select case ucase$(argt$)
   case "C", "CENTER"
     locate 25,1: print "Center: ";
     arc2:
      inputkey$ = inkey$
      gosub directip
      gosub movecur
      gosub ctrlkey
      if flag% = 0 then arc2
     x0 = x: y0 = y
     gosub scrollup
     CALL mark(x0, y0)
     locate 25,1: print "Start point: ";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
arc3:
 inputkey$ = inkey$
 gosub directip
 gosub movecur
 gosub ctrlkey
 CALL mark(x0 ,y0)
 if flag% = 0 then arc3
x1 = x: y1 = y
s = FNangle#(x0, y0, x1, y1)
r = sqr(((x1 - x0) ^ 2) + ((y1 - y0) ^ 2))
gosub scrollup
CALL mark(x1, y1)
locate 25,1: print "End point/Included angle/Lenght of chord<E/I/L>: ";
arc4:
 inputkey$ = inkey$
 gosub directip
 gosub ctrlkey
 CALL mark(x0 ,y0)
 if flag% = 0 then arc4
gosub scrollup
select case ucase$(argt$)
  case "E", "END"
    locate 25,1: print "End point: ";
    arc5:
     inputkey$ = inkey$
     gosub directip
     gosub movecur
     gosub ctrlkey
     CALL mark(x1, y1)
     if flag% = 0 then arc5
    x2 = x: y2 = y
    e = FNangle#(x0, y0, x2, y2)
    CALL mark(x2, y2)
    exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  case "I", "INCLUDED"
    locate 25, 1: print "Included angle(in degree): ";
    arc6:
     inputkey$ = inkey$
     gosub directip
     gosub ctrlkey
     CALL mark(x1, y1)
     if flag% = 0 then arc6
     e = cdbl(s + argt#)
     if s > e then
      buff = e
      e = s
      s = buff
     end if
    exit select
  case "L", "LENGHT"
    locate 25, 1: print "Lenght of chord: ";
    arc11:
     inputkey$ = inkey$
     gosub directip
     gosub ctrlkey
     CALL mark(x1, y1)
     if flag% = 0 then arc11
     lc# = argt#
     s = FNangle#(x0, y0, x1, y1): e = 0
     r = (((x1 - x0) ^ 2) + ((y1 - y0) ^ 2)) ^ 0.5
     CALL sclarc(s, e, r, lc#)
     exit select
  case else
    flag% = 4
    goto arc11
end select
exit select
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
case "S", "START"
  locate 25,1: print "Start point: ";
  arc7:
   inputkey$ = inkey$
   gosub directip
   gosub movecur
   gosub ctrlkey
   if flag% = 0 then arc7
  x1 = x: y1 = y
  gosub scrollup
  CALL mark(x1, y1)
  locate 25,1: print "Second point: ";
  arc8:
   inputkey$ = inkey$
   gosub directip
   gosub movecur
   gosub ctrlkey
   CALL mark(x1, y1)
   if flag% = 0 then arc8
  x2 = x: y2 = y
  gosub scrollup
  CALL mark(x2, y2)
  locate 25,1: print "End point: ";
  arc9:
   inputkey$ = inkey$
   gosub directip
   gosub movecur
   gosub ctrlkey
   CALL mark(x2, y2)
   if flag% = 0 then arc9
  x3 = x: y3 = y
  CALL mark(x3, y3)
  CALL threepoint(x1, y1, x2, y2, x3, y3)
  r = (((x1 - x0) ^ 2) + ((y1 - y0) ^ 2)) ^ 0.5
  s = FNangle#(x0, y0, x1, y1)
  e = FNangle#(x0, y0, x3, y3)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'------------------------------ sequancy angle ------------------------------'
 cent = FNangle#(x0, y0, x2, y2)
 if sgn(cent) < 0 then cent = 360 + cent
 if cent < s then
   buffer = s
   s = e
   e = buffer
 end if
'----------------------------------------------------------------------------'
     exit select
   case else
     flag% = 4
     goto arc10
 end select
 CALL arcf(x0, y0, r, s, e)
 arc10:
  argt$ = ""
'----------------------------------------------------------------------------'
return
'----------------------------------------------------------------------------'



SUB mark(xx,yy)
'---------------------------- mark select position --------------------------'
 LINE (xx, yy) - (xx + 5, yy + 2), 1
 LINE - (xx + 2, yy + 5), 1
 LINE - (xx, yy), 1
 LINE (xx + 4, yy + 4)-(xx + 9, yy + 9), 1
'----------------------------------------------------------------------------'
END SUB
'----------------------------------------------------------------------------'
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้