



โปรแกรมคอมพิวเตอร์ช่วยในการออกแบบและวางผังโรงงาน
Computer-Aided Plant Layout and Design



โดย
นาย ธนารักษ์ ห่องสุข เลขที่ 32.1113
นาย มานพ ศรีพิพัฒน์ เลขที่ 32.1238

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
ปีการศึกษา 2535

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

032714

ปริญญานิพนธ์ปีการศึกษา 2535
ภาควิชา วิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง โปรแกรมคอมพิวเตอร์ช่วยในการออกแบบและวางผังโรงงาน
Computer-Aided Plant Layout and Design

ผู้จัดทำ นาย ธนารักษ์ ห่องสุ เลขที่ 32.1113
นาย มานพ ศรีพิพัฒน์ เลขที่ 32.1238

..... อาจารย์ที่ปรึกษา

(.....)

โปรแกรมคอมพิวเตอร์ช่วยในการออกแบบและวางผังโรงงาน

นาย ธนารักษ์ ห้องสุ

นาย มานพ ศรีพิพัฒน์

ดร. เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา
ปีการศึกษา 2535

บทคัดย่อ

โครงงานฉบับนี้ เป็นการศึกษาทฤษฎีในการออกแบบ และ การวางผังโรงงานอุตสาหกรรม เพื่อหาแนวทางในการนำคอมพิวเตอร์ประยุกต์ใช้งาน และ ทดลองสร้างโปรแกรมต้นแบบ อันจะเป็นเครื่องมือที่สามารถช่วยในการออกแบบและวางผังโรงงานในบางขั้นตอน มีความสะดวก รวดเร็วยิ่งขึ้น

ทฤษฎีในการออกแบบนั้นได้ศึกษาจาก การวางผังโรงงานอย่างเป็นระบบ (SYSTEMATIC LAYOUT PLANNING) โปรแกรมวางผังโรงงานเป็นโปรแกรมที่ช่วยในการวางผังโรงงาน โดยใช้ การวิเคราะห์ความสัมพันธ์ระหว่างแผนกในโรงงาน เพื่อคัดเลือกแผนกที่สัมพันธ์กันให้วางตำแหน่ง ใกล้กันตามค่าความใกล้ชิด (CLOSENESS RATING) เมื่อวางครบทุกแผนก ในที่สุดจะได้ ผังโรงงาน ในลักษณะ ผังโรงงานแบบบล็อก (BLOCK LAYOUT)

Computer-Aided Plant Layout and Design

Thanarak Hongtuk

Manop Sriphipattana

Dr. Ouen Pin-Ngeon Advisor

1992

ABSTRACT

This project is to study the theory of industrial plant layout and design, find out how to apply computer programming to plant layout and design, and build a program prototype that giving more convenience to some steps of plant planning.

Plant Layout Program supports the layout planning. It uses the relation among departments to analyze the closeness of departments. It uses closeness rating to select department that must be located closely to another department. When all the departments are located, finally, the block layout is completed.

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
สารบัญรูป	III
บทที่	
1 บทนำ	
1.1 ความเป็นมา	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขตของการทำโครงการ	1
1.4 ขั้นตอนในการดำเนินโครงการ	2
2 การวางแผนผังโรงงาน	
2.1 กฎแฉ่ PQRS (องค์ประกอบพื้นฐานของการวางแผนผังโรงงาน)	3
2.2 ขั้นตอนในการวางแผนผังโรงงาน	4
2.3 การวางแผนผังโรงงานอย่างเป็นระบบ (SLP)	
2.3.1 หลักการขั้นพื้นฐานสำหรับการวางแผนผังโรงงาน	6
2.3.2 รูปแบบของการวางแผนผังโรงงานอย่างมีระบบ	6
2.3.3 ความสำคัญของ PQRS	9
3 การวางแผนผังโรงงานด้วยคอมพิวเตอร์	
3.1 ข้อพิจารณาของการวางแผนผังโรงงานด้วยคอมพิวเตอร์	11
3.2 การเก็บข้อมูลในการวางแผนผังโรงงาน	
3.2.1 พื้นที่ที่ต้องการของขบวนการผลิต	11
3.2.2 ฟรอม - ทู ชาร์ต (From - To Chart)	12
3.2.3 ความสัมพันธ์ระหว่างแผนก (Relation Chart)	13
3.3 วิธีการในการใช้คอมพิวเตอร์ช่วยในการวางแผนผังโรงงาน	14

4	รายละเอียดโปรแกรม	
4.1	โครงสร้างของโปรแกรม	18
4.2	ขั้นตอนการทำงานของโปรแกรม	20
4.3	รายละเอียดของโปรแกรมส่วนต่าง ๆ	
4.3.1	ส่วนของข้อมูลเบื้องต้น	
4.3.1.2	Relation Chart	21
4.3.1.3	Department Data	29
4.3.1.4	From - To Chart	31
4.3.2	ส่วนของการวางแผน	38
4.4	องค์ประกอบของโปรแกรมและการสร้างโปรแกรม	48
4.5	ตัวอย่างการใช้โปรแกรม	49
สรุป		62
ภาคผนวก		
	โปรแกรม LISTING	
กิจกรรมประกาศ		
บรรณานุกรม		

สารบัญรูป

รูปที่		หน้า
2.1	กฎแจ PQRST เพื่อการไขปัญหาการวางผังโรงงาน	4
2.2	ขั้นตอนการวางผังโรงงานอย่างมีระบบ	5
2.3	แผนการเชิงปฏิบัติของการวางแผนผังโรงงานอย่างเป็นระบบ	7
2.4	แสดงถึงการนำแผนการเชิงปฏิบัติ SLP ไปประยุกต์ใช้งาน ในขั้นตอนการวางผังโรงงาน	10
3.1	แสดงแผนภูมิจากไป (From - To Chart)	13
3.2	แสดงลักษณะแผนภูมิความสัมพันธ์ (Relation Chart)	13
4.1	แสดงโครงสร้างของโปรแกรม	18
4.2	แสดงขั้นตอนการทำงานของโปรแกรม	21

บทที่ 1

บทนำ

1.1 ความเป็นมา

โรงงาน (Plant) คือสถานที่ซึ่งรวมเอาปัจจัยการผลิต (Input) เข้าด้วยกันเพื่อทำให้เกิดการผลิต (Output) ที่อยู่ในรูปของผลิตภัณฑ์ (Product) หรือ บริการ (Services)

การวางผังโรงงาน เป็นการจัดตำแหน่งของคน เครื่องจักร วัตถุดิบและสิ่งอำนวยความสะดวกในตำแหน่งที่เหมาะสม เป็นปัญหาของโรงงานทุกแห่ง การวางผังโรงงานที่ดีจะเป็นฐานสำคัญในการดำเนินงาน

การนำคอมพิวเตอร์เข้ามาช่วยในการวางผังโรงงานน่าจะเป็นประโยชน์ไม่น้อย ซึ่งในที่นี้ได้นำคอมพิวเตอร์มาช่วยในการวางแผนกต่าง ๆ ในโรงงาน โดยวิเคราะห์จากความสัมพันธ์ระหว่างแผนกเป็นขั้นตอนหนึ่งในการวางผังโรงงานอย่างเป็นระบบ (Systematic Layout Planning)

1.2 วัตถุประสงค์

1.2.1 ศึกษาทฤษฎีการออกแบบและการวางผังโรงงานเพื่อวิเคราะห์หาแนวทางในการนำคอมพิวเตอร์เข้ามาสนับสนุน การออกแบบและการวางผังโรงงาน

1.2.2 ศึกษาภาษาทางด้านโปรแกรมมิ่ง (Programming Language) เพื่อใช้ในการเขียนโปรแกรม

1.2.3 สร้างโปรแกรมที่เป็นเครื่องมือช่วยในการวางผังโรงงาน

1.3 ขอบเขตของการทำโครงการ

โครงการนี้ทำการศึกษาทฤษฎีการวางผังโรงงานเพื่อหาแนวทางในการนำคอมพิวเตอร์เข้าประยุกต์ใช้งาน เพื่อช่วยลดความซับซ้อนในการคำนวณ และ ช่วยประหยัดเวลา

โปรแกรมวางผังโรงงาน เป็นโปรแกรมที่ช่วยวางผังโรงงานวิเคราะห์จาก ความสัมพันธ์ของแผนกต่าง ๆ ในโรงงาน ให้ผลออกมาเป็น ผังโรงงานแบบบล็อก (Block Layout) ข้อมูลที่นำเข้าไปในโปรแกรม ได้แก่ พื้นที่ของแต่ละแผนก, ความสัมพันธ์ระหว่างแผนก และ ปริมาณการเคลื่อนที่ของวัสดุระหว่างแผนก

ความสัมพันธ์ระหว่างแผนก ป้อนเข้าโปรแกรมในรูปของ แผนภูมิความสัมพันธ์ (Relation Chart) จากนั้นผู้ใช้เลือกแผนกที่มีค่าความสัมพันธ์มาก วางลงในผังโรงงานจนครบทุกแผนก ต่อมา คำนวณหาค่าประสิทธิภาพของผังโรงงาน จาก ปริมาณการเคลื่อนที่ของวัสดุซึ่งป้อนเข้าโปรแกรม ในรูปของ แผนภูมิจาก-ไป (From-To Chart) กับ ระยะทางในการขนส่งระหว่างแผนก

สุดท้าย จะได้ผังโรงงานแบบบล็อกออกมา ซึ่งอยู่ในขั้นตอนที่ 2 Block Layout หรือ Overall Layout ของ การวางผังโรงงานอย่างเป็นระบบ (SYSTEMATIC LAYOUT PLANNING : SLP)

1.4 ขั้นตอนในการดำเนินโครงการ

1.4.1 ขั้นตอนการศึกษาและหาแนวทางประยุกต์ใช้

- ศึกษาในเรื่องการวางผังโรงงาน
- หาแนวทางประยุกต์ใช้ความรู้ในเรื่องการวางผังโรงงานเพื่อมาใช้สร้างโปรแกรม
- ศึกษาภาษาทางโปรแกรมมิ่ง

1.4.2 ขั้นตอนการสร้างโปรแกรมและทดลอง

- ศึกษาการเขียนโปรแกรม และ เทคนิคการเขียนโปรแกรม ในส่วนต่าง ๆ
- ออกแบบโปรแกรม
- สร้างส่วนการรับข้อมูล อันได้แก่ ส่วนรับพื้นที่ของแผนก ส่วนรับค่าความสัมพันธ์ (Relation Chart) ส่วนรับข้อมูลแผนภูมิ จาก-ไป (From-To Chart) และ แม้าส์
- สร้างโปรแกรมส่วนการคำนวณ และ การแสดงผลทางจอภาพ
- ประกอบโปรแกรมส่วนต่าง ๆ ให้สามารถทำงานร่วมกันได้อย่างเหมาะสม
- ทำการทดลองการใช้งาน

บทที่ 2

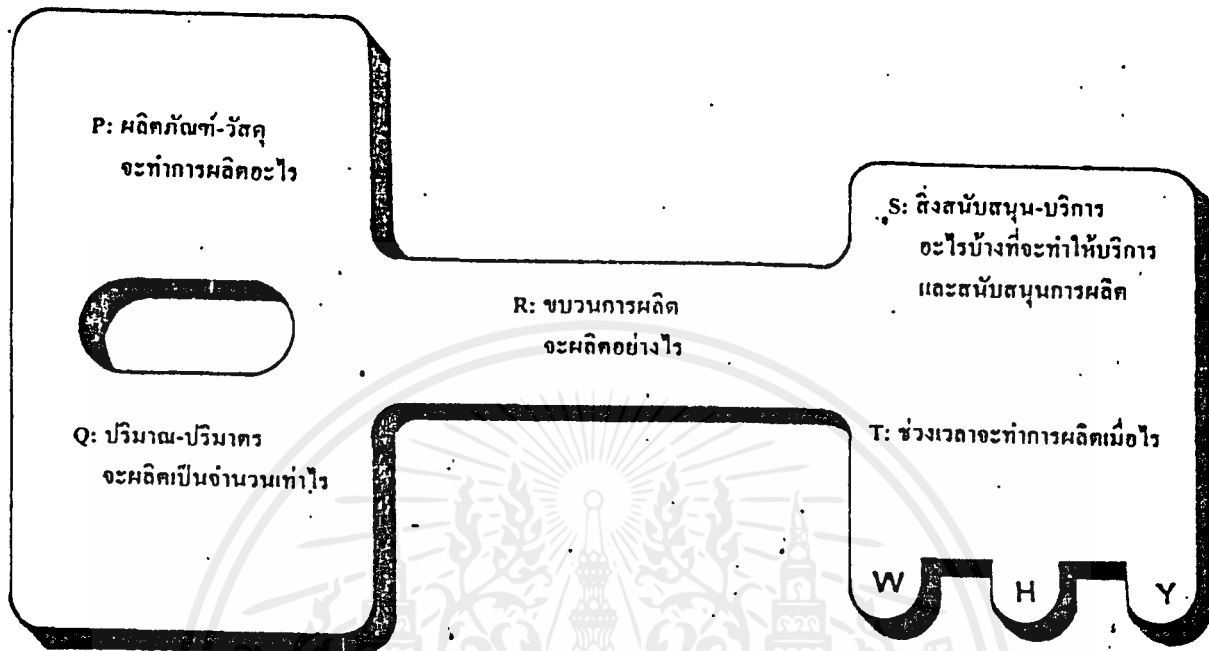
การวางแผนโรงงาน

2.1 ฤๅญแจ PORST (องค์ประกอบพื้นฐานของการวางแผนโรงงาน)

มีอยู่ 5 องค์ประกอบ คือ

1. ผลิตภัณฑ์ (Product) หมายถึง สินค้าที่เราผลิตนั้น เริ่มจากวัตถุดิบทั้งหมด หรือ ซื้อมาประกอบบางส่วน ทำการผลิตผลิตภัณฑ์สำเร็จรูป หรือ เฉพาะชิ้นส่วน ซึ่งอาจกำหนดว่าเป็นผลิตภัณฑ์ชนิดใด รุ่นไหน แบบใด หมายเลขรหัสอะไร ผลิตภัณฑ์กลุ่มใด หรือ กรณีที่เป็นวัสดุต้องแยกออกเป็นกลุ่ม ตามชนิดของวัสดุ
2. ปริมาณ (Quantity) หมายความว่า ผลิตภัณฑ์ที่จะทำการผลิตหรือใช้นั้นมีจำนวนมากน้อยเท่าใด
3. ขบวนการผลิต (Routing) หมายถึง กรรมวิธีหรือขบวนการผลิตที่ประกอบ ด้วยเครื่องจักร อุปกรณ์ วิธีการทำงาน และลำดับการทำงาน ซึ่งอาจบอกได้จาก แผนภูมิขบวนการผลิต และแผนภูมิการไหลของขบวนการ
4. สิ่งให้บริการและสนับสนุน (Supporting) หมายถึง สิ่งทีอำนวยการประโยชน์ เครื่องช่วย หรือเป็นตัวประสานระหว่างหน่วยงานในผังโรงงาน ทั้งนี้เพื่อให้การทำงานมี ประสิทธิภาพ สิ่งสนับสนุนการผลิตดังกล่าวได้รวมถึงการบำรุงรักษา การซ่อมแซมเครื่องจักร โกดังเก็บของ ห้องน้ำ ห้องอาหาร ห้องพยาบาล ฯลฯ
5. เวลา (Time) หมายถึง เวลา เมื่อไร นานเท่าไร บ่อยหรือไม่ จะทำการ ผลิตเมื่อไร ทำงานวันละกี่ชั่วโมง

ฤๅญแจไขปัญหาการวางแผนโรงงาน ดังแสดงในรูป 2.1 จะเห็นว่าองค์ประกอบ Product และ Quantity แทนด้วยมือจับฤๅญแจ ซึ่งเป็นส่วนที่เราต้องจับให้แน่นอนก่อนไข ปัญหา กล่าวคือ เราต้องรู้ให้แน่ชัดว่า จะทำการผลิตผลิตภัณฑ์ชนิดใด เป็นปริมาณเท่าไร หลังจากที่ได้ทราบ Product และ Quantity แล้ว ขั้นตอนต่อไปจะต้องทำการ ศึกษากรรมวิธีหรือขบวนการผลิต (Routing) เป็นก้านของฤๅญแจ หมายถึง จะทำการผลิต ผลิตภัณฑ์นั้นอย่างไร ต่อจากนั้น เป็นการศึกษา ส่วนที่สนับสนุนการผลิต (Supporting Service) และ เวลา (Time)



รูปที่ 2.1 กฎ ๖๕ PQRST เพื่อการไขปัญหาการวางผังโรงงาน

2.2 ขั้นตอนในการวางผังโรงงาน

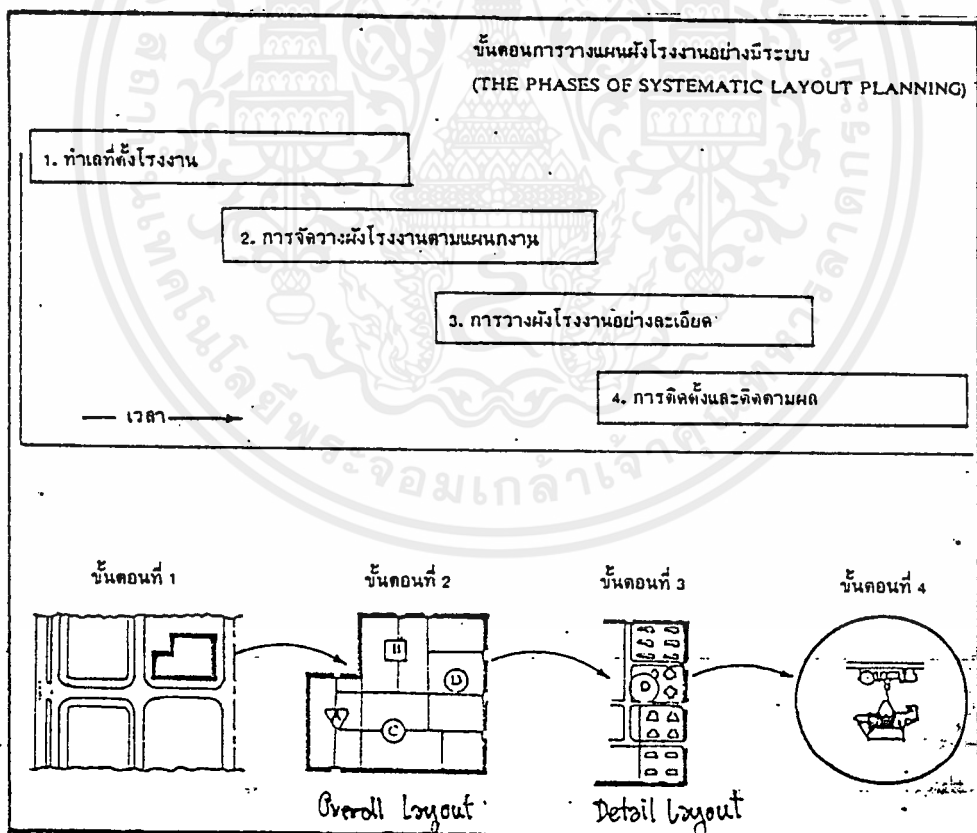
ขั้นตอนในการวางผังโรงงาน มีอยู่ด้วยกัน 4 ขั้นตอน คือ

ขั้นตอนที่ 1 การเลือกทำเลที่ตั้ง (Location) ใช้สำหรับพิจารณาหาทำเลที่ตั้งของพื้นที่ที่จะทำการวางผังโรงงาน ขั้นตอนนี้นอกจากเป็นปัญหาสำคัญในการเลือกสถานที่ตั้งโรงงานแห่งใหม่ แล้ว ยังเป็น สิ่งหนึ่งที่เราจะต้องหาสำหรับการจัดวางผังโรงงานใหม่ในเนื้อที่เดิมปัจจุบัน เช่น พื้นที่ห้องเก็บของสามารถใช้ได้หลายวัตถุประสงค์อย่างอิสระ หรือหลังจากที่เราได้พื้นที่ในอาคารใหม่ หรืออื่น ๆ ที่เรามีแนวโน้มที่จะหาพื้นที่ได้

ขั้นตอนที่ 2 การจัดวางผังโรงงานตามแผนงาน (Overall Layout) ใช้สำหรับการจัดพื้นที่ทั่ว ๆ ไป ที่ จะทำการวางผังโรงงาน ขั้นตอนนี้เป็นขั้นตอนที่แสดงแผนการปฏิบัติขั้นพื้นฐานที่แสดงถึงความสัมพันธ์และรูปร่างลักษณะพื้นที่หลักแต่ละพื้นที่หรือพื้นที่ที่แบ่งตามแผนงานที่เราจะต้องเขียนขึ้นมาอย่างกว้าง ๆ โดยไม่คำนึงถึงว่ามีอุปกรณ์หรือกิจกรรมใดบ้างอยู่ในพื้นที่หลักอันนั้น ขั้นตอนที่ 2 นี้บางครั้งเราเรียกว่า ผังโรงงานอย่างหยาบ หรือผังโรงงาน แบบบล็อก(Block Layout)

ขั้นตอนที่ 3 การวางผังโรงงานอย่างละเอียด (Detail Layout) ขั้นตอนนี้จะกำหนดรายละเอียดของเครื่องมือ และ เครื่องจักรต่าง ๆ ว่าอยู่ในตำแหน่งใดของพื้นที่หลักที่วางไว้ใน ขั้นตอนที่ 2 ในการวางแผนผังโรงงานอย่างละเอียด ต้องสร้างและกำหนดพื้นที่ของเครื่องจักรและเครื่องมือต่าง ๆ ทุกเครื่อง และรวมถึงสิ่งสนับสนุนการผลิตและการบริการ ทั้งนี้โดยการวางแผนลงในแบบหรืออาจสร้างเป็นแบบจำลองของเครื่องจักรแต่ละเครื่องลงไป

ขั้นตอนที่ 4 การติดตั้งและการติดตามงาน (Installation) หลังจากที่ได้ดำเนินการวางแผนผังโรงงานอย่างละเอียดสมบูรณ์แล้วในขั้นตอนที่ 3 ก็ต้อง ทำการพิจารณาถึงเรื่องการติดตั้งตามแบบ แสดงรายละเอียดพร้อมกับวางแผนด้านการเคลื่อน ย้ายตามที่ได้คิดกำหนดไว้ ต้องมีเงินทุนสำหรับการติดตั้งเพียงพอตลอดจนการขนย้าย เพื่อสำหรับการติดตั้งเครื่องจักร อุปกรณ์และสิ่งสนับสนุนการผลิตให้เป็นไปตามแผนที่วางไว้ นอกจากนั้นต้องติดตามผลหลังจากการติดตั้งและดำเนินงานแล้ว มีผลเสียต่อการดำเนินงาน อย่างไร



รูปที่ 2.2 ขั้นตอนการวางแผนผังโรงงานอย่างมีระบบ

2.3 การวางแผนผังโรงงานอย่างเป็นระบบ (The Systematic Layout Planning)

2.3.1 หลักการสำคัญขั้นพื้นฐานสำหรับการวางแผนผังโรงงาน

หลักการสำคัญขั้นพื้นฐานสำหรับการวางแผนผังโรงงาน สามารถแบ่งออกได้ 3 ประการ คือ

1. ความสัมพันธ์ (Relationship) เป็นการกำหนดความสัมพันธ์ของกิจกรรมต่างๆ โดยเริ่มจากกิจกรรมที่มีความสัมพันธ์มากมาหาน้อย โดยที่กิจกรรมใดมีความสัมพันธ์มาก ก็ให้อยู่ใกล้กัน

2. เนื้อที่ (Space) เป็นการพิจารณาเกี่ยวกับเนื้อที่ต่าง ๆ ทั้งชนิดและรูปร่าง หรือรูปทรงของเนื้อที่ ของกิจกรรมต่าง ๆ ในผังโรงงาน.

3. การจัดปรับตำแหน่งที่ตั้ง (Adjustment) เป็นการปรับหรือปรับตำแหน่งของ กิจกรรมต่าง ๆ ให้ได้เหมาะสมภายใต้ข้อจำกัดต่าง ๆ ที่ได้กำหนดในผังโรงงาน

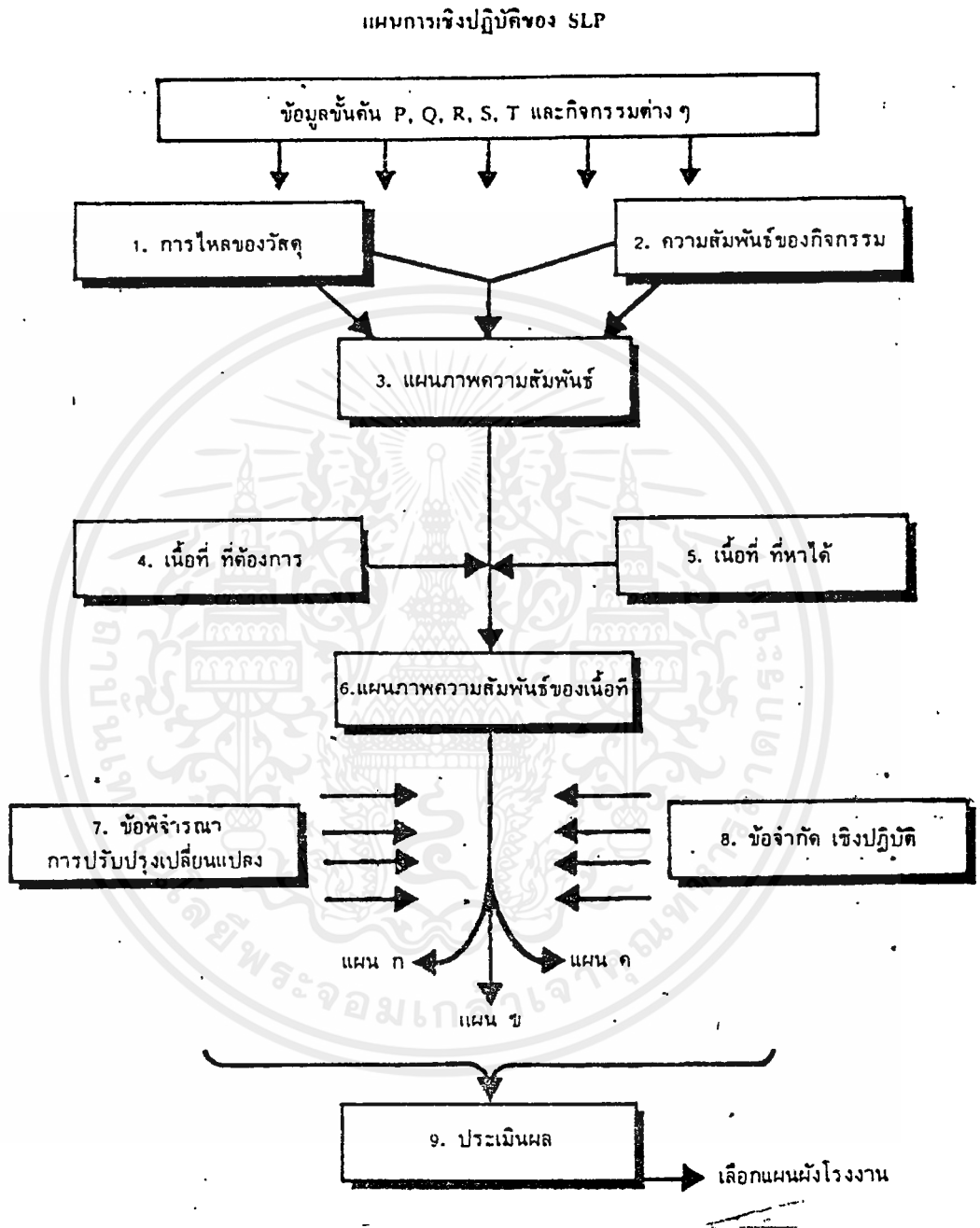
จากหลักการสำคัญขั้นพื้นฐาน 3 ข้อ ดังกล่าว เป็นหัวใจในการวางแผนผังโรงงานต่างๆ โดยไม่คำนึงถึงชนิดของผลิตภัณฑ์ ขบวนการผลิต หรือ ขนาดของโครงการแต่อย่างใด เราสามารถนำเอามาปรับเป็นแผนการเชิงปฏิบัติในการวางแผนผังโรงงานอย่างมีระบบ (Systematic Layout Planning.)

ได้ดังรูป 2.3

2.3.2 รูปแบบของการวางแผนผังโรงงานอย่างมีระบบ

ความสำคัญของผลิตภัณฑ์ (P) และปริมาณ (Q) ที่มีผลต่อผังโรงงาน การวิเคราะห์เกี่ยวกับความสำคัญของข้อมูลแต่ละตัวและทั้งหมดนั้น ต่างก็เป็นสิ่งสำคัญขั้นต้น สำหรับการ วางแผนผังโรงงาน ขบวนการผลิต และอุปกรณ์ (R) สิ่งสนับสนุนการผลิต (S) และเวลา (T) ตลอดจนกิจกรรมหรือพื้นที่ต่าง ๆ ที่รวมในผังโรงงาน ข้อมูลต่าง ๆ เหล่านี้ ต่างเป็น ข้อมูลพื้นฐานชั้นแรกที่จะใช้ประกอบการวางแผนผังโรงงาน

องค์ประกอบสำคัญของขบวนการผลิต และเป็นสิ่งสำคัญสูงสุดของการวางแผนผังโรงงาน ก็คือ การไหลของวัสดุ (Flow of Material) ซึ่งผู้วางแผนผังโรงงานจะต้อง ทำการวิเคราะห์ปริมาณการไหลของวัสดุทิศทางและลำดับขั้นตอนการไหล ตลอดจนพื้นที่ที่เกี่ยวข้อง ในพื้นที่การทำงานหรือพื้นที่สำหรับการผลิต และพื้นที่สำหรับสิ่งสนับสนุนการผลิตที่ได้กำหนดอยู่ในแผนนั้นจะเป็นข้อมูลอย่างดีที่จะพัฒนา และจัดทำแผนภาพแสดงความสัมพันธ์ ของกิจกรรมต่างๆ กิจกรรมใดที่มีความสัมพันธ์กับกิจกรรมการบริการ หรือกิจกรรมสนับสนุน การผลิต หรือลักษณะการทำงานต้องติดต่อกันบ่อยครั้ง จะมีความสำคัญมากกว่าความสัมพันธ์พื้นฐานเฉพาะการไหลของวัสดุแต่เพียงอย่างเดียว



รูปที่ 2.3 แผนเชิงปฏิบัติของการวางแผนผังโรงงานอย่างเป็นระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อนำผลการวิเคราะห์การไหลของวัสดุ และความสัมพันธ์ของกิจกรรม (Activity Relationship) ซึ่งเขียนอยู่ในรูปของแผนภูมิความสัมพันธ์ (Relationship Chart) มาพิจารณาร่วมกันก็สามารถเขียนเป็นแผนภาพความสัมพันธ์ (Relationship Diagram) โดยพิจารณาความสัมพันธ์กิจกรรมต่าง ๆ ของแต่ละแผนกในตำแหน่งและทิศทางที่เหมาะสม โดยไม่คำนึงถึงลักษณะรูปทรงของพื้นที่ที่เป็นจริงของแต่ละกิจกรรมว่าเป็นอย่างไร

ขั้นตอนต่อมาก็เป็นเรื่องของ เนื้อที่ที่ต้องการ (Space Requirement) เป็นผลมาจากการวิเคราะห์เนื้อที่ของขบวนการผลิต เครื่องจักร และอุปกรณ์ต่าง ๆ ที่สำคัญ ๆ และจากสิ่งอำนวยความสะดวก ในการสนับสนุนการผลิตที่เกี่ยวข้อง อย่างไรก็ตามเนื้อที่ที่ต้องการ จะต้องเป็นไปอย่างสอดคล้องกับเนื้อที่หาได้ด้วย (Space Available) เมื่อได้เนื้อที่สำหรับแต่ละกิจกรรมแล้ว ก็นำมาเขียนลงในแผนภาพความสัมพันธ์ก็จะได้เป็นแผนภาพขึ้นมาอีกอันหนึ่ง ซึ่งเรียกว่า

แผนภาพความสัมพันธ์ของเนื้อที่ (Space Relation Diagram) แผนภาพความสัมพันธ์ของเนื้อที่นี้เป็นสิ่งสำคัญของผังโรงงานเพราะเป็นแนวทาง ของการหาตำแหน่งของกิจกรรมได้อย่างเหมาะสม อย่างไรก็ตามแผนภาพความสัมพันธ์ของเนื้อที่ เป็นเพียงแค่แนวทางเท่านั้น ทั้งนี้จะต้องทำการปรับหาดำแหน่ง โยกย้าย หรือรวมเนื้อที่ นั้น คือต้องมีการปรับปรุงเปลี่ยนแปลง (Modifying) โดยพิจารณาถึง การขนถ่าย การปฏิบัติงาน การเก็บรักษา และอื่นหากแต่ละหน่วยงานของกิจกรรมมีแนวโน้มที่เป็นไปได้ หรือ กิจกรรมที่เราปรับปรุงเปลี่ยนแปลงตามแนวความคิดที่ดีแล้ว ก็ต้องคำนึงถึงข้อจำกัดเชิงปฏิบัติ (Practical Limitation) ต่าง ๆ อีกด้วย ทั้งนี้เพื่อให้การวางแผนมีความเป็นไปได้ มากที่สุด

การรวบรวมข้อมูลต่าง ๆ เพื่อประกอบการพิจารณาปรับปรุงเปลี่ยนแปลง ภายใต้ข้อ จำกัดเชิงปฏิบัติกีดเชิงปฏิบัติตามแนวความคิดที่เป็นไปได้ จากนั้นก็ควรทดลอง หรือทดสอบ ว่าวิธีไหน หรือแผนการใดที่เหมาะสมในเชิงปฏิบัติก็จะคงได้ เพื่อการพิจารณาเลือกประมาณ 2-5 แผนการจำนวนดังกล่าวสามารถนำไปใช้ได้แต่เราต้องมีการพิจารณาตัดสินใจเลือก แผนการใดแผนการหนึ่งที่เห็นว่าเหมาะสมที่สุด แผนการต่าง ๆ ที่เราได้จัดไว้เพื่อเลือกนั้น อาจให้ชื่อว่า แผน ก แผน ข แผน ค

มาถึงจุดนี้ก็ขั้นตอนของการประเมินผล (Evaluation) เพื่อพิจารณาตัดสินใจ เลือกแผนการที่เหมาะสมที่สุดโดยวิเคราะห์เกี่ยวกับค่าใช้จ่ายแต่ละประเภท เปรียบเทียบใน เชิงเหตุผล อันประกอบด้วยองค์ประกอบการพิจารณาต่าง ๆ ที่เป็นไปได้ การประเมินผลก็ สามารถตัดสินใจเลือกแผนการใดแผนการหนึ่งที่เหมาะสมที่สุด แผนการของผังโรงงานที่ถูกเลือกครั้งแรกนั้นเป็นแผนการของผังโรงงานในขั้นตอนการวางผังโรงงานตามแผนงาน หรือขั้นตอนที่ 2 ก็เป็นว่าเสร็จสิ้นในขั้นตอนนี้



ส่วนใน ขั้นตอนที่ 3 ของการวางแผนโรงงานอย่างละเอียด การวิเคราะห์ใช้หลักการพื้นฐานแบบเดียวกับ ขั้นตอนที่ 2 เพียงแต่พิจารณาในรายละเอียดเฉพาะภายในแต่ละแผนงานจนครบทุกแผนก ตามผังโรงงานตามแผนงานดังได้กล่าวในรายละเอียดมาแล้วข้างต้น

2.3.3 ความสำคัญของ P, Q, R, S และ T

จากแผนการเชิงปฏิบัติของ SLP ซึ่งมีความสัมพันธ์กับข้อมูลสำคัญพื้นฐานเริ่มต้น คือ P, Q, R, S และ T อันเป็นข้อมูลที่สำคัญสำหรับการวางแผนผังโรงงาน สำหรับการออกแบบผลิตภัณฑ์และการพยากรณ์การขายจะต้องประสานงานกัน

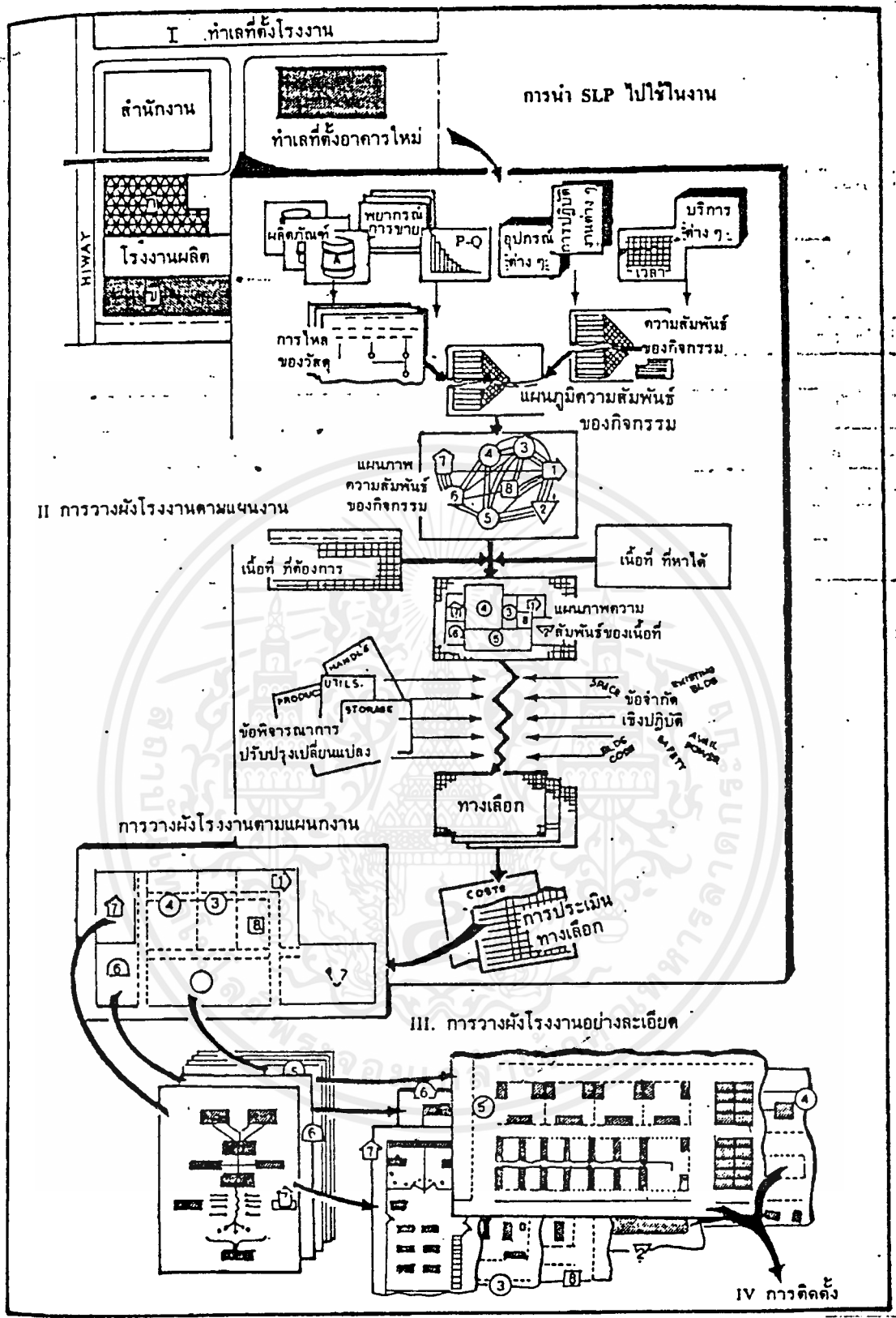
นอกจากนั้น ยังต้องทำการ วิเคราะห์ ผลิตภัณฑ์-ปริมาณ (Product-Quantity) ซึ่งบางครั้งเรียกว่า การวิเคราะห์ ปริมาตร-ชนิดผลิตภัณฑ์ ซึ่งรูปแบบของการแยก หรือรวมผลิตภัณฑ์แต่ละชนิด หรือจัดกลุ่มผลิตภัณฑ์ หรือจัดกลุ่มผังโรงงาน จะได้มาจากการวิเคราะห์ P-Q โดยเฉพาะการวิเคราะห์ผลิตภัณฑ์หลายๆชนิด ตลอดจนถึงการวิเคราะห์ ขบวนการผลิต (R : Routing) บริการ (S : Supporting) และเวลา (T : Time) เพื่อเป็นแนวทางที่จะบ่งชี้ถึงกิจกรรมต่างๆ ที่เกี่ยวข้องเพื่อจะให้ได้ผังโรงงานมา

P, Q และ R เป็นข้อมูลที่ต้องวิเคราะห์ร่วมกันเพื่อนำไปวิเคราะห์ในเรื่องการไหล ของวัสดุ (ในกรอบที่ 1 รูปที่ 2.3) P, Q และ S เป็นข้อมูลที่ต้องวิเคราะห์ร่วมกันเพื่อนำไปวิเคราะห์ในเรื่องของ ความสัมพันธ์ของกิจกรรม (ในกรอบที่ 2 รูปที่ 2.3)

ผลจากการวิเคราะห์การไหลของวัสดุ และความสัมพันธ์ของกิจกรรม เมื่อนำมา วิเคราะห์ร่วมกันจะเขียนได้เป็น แผนภาพความสัมพันธ์ของกิจกรรม (ในกรอบที่ 3 รูปที่ 2.3) เมื่อนำ R และ T มาวิเคราะห์รวมกัน ก็จะเป็นส่วนสำคัญสำหรับการหาเครื่องจักร และอุปกรณ์ที่ต้องการ

ในทำนองเดียวกัน การบริการ (S) เหมาะสำหรับการกำหนดสิ่งอำนวยความสะดวกและสนับสนุนการผลิตที่ต้องการเครื่องจักรและอุปกรณ์ในขบวนการผลิต และสิ่งอำนวยความสะดวกจะเป็นตัวกำหนดเนื้อที่ที่ต้องการ แล้วดำเนินการต่อไปตามแผน การเชิงปฏิบัติของ SLP

ตัวอย่างการนำ SLP ไปใช้งาน ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 แสดงถึงการนำแผนการเชิงปฏิบัติ SLP ไปประยุกต์ใช้งานใน
ขั้นตอนการวางแผนโรงงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ 10 ศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

การวางแผนโรงงานด้วยคอมพิวเตอร์ (Computerized Routines)

3.1 ข้อพิจารณาของการวางแผนโรงงานด้วยคอมพิวเตอร์

การนำคอมพิวเตอร์มาช่วยในการวางแผนโรงงานนั้น มีข้อพิจารณาที่สำคัญ 3 ข้อคือ

1. คอมพิวเตอร์ไม่สามารถล่วงรู้ถึงเงื่อนไขและหลักเกณฑ์ต่าง ๆ ในการวางแผนโรงงาน แต่คอมพิวเตอร์เป็นเพียงเครื่องคำนวณที่คำนวณได้รวดเร็ว และมีความจำที่ดีเลิศเท่านั้น
2. คนรู้ถึงเงื่อนไขและหลักเกณฑ์ในการวางแผนโรงงาน แต่คนคำนวณได้ไม่เร็ว และ แม่นยำเท่าคอมพิวเตอร์
3. คอมพิวเตอร์ไม่ควรนำมาใช้ให้ทำการวางแผนทั้งหมด แต่ควรนำมาใช้เป็นเครื่องมือในการช่วยวางแผนเท่านั้น

จากข้อพิจารณาทั้ง 3 ประการนี้ จึงทำให้นักออกแบบสามารถกำหนดบทบาทของคอมพิวเตอร์ในการช่วยวางแผนโรงงานได้อย่างถูกต้อง การใช้คอมพิวเตอร์ช่วยวางแผนโรงงานนี้มิได้เป็นแฟชั่นหรือเป็นการแสดงความหมายที่จะเอาเทคโนโลยีที่ทันสมัย เช่น คอมพิวเตอร์มาใช้วางแผน หากแต่เป็นความจำเป็น เนื่องจากโรงงานอุตสาหกรรมมีขนาดใหญ่และมีขบวนการผลิตที่ยุ่งยากซับซ้อนจนไม่สามารถใช้วิธีการอื่น ๆ นำมาแก้ปัญหาได้ นอกจากคอมพิวเตอร์

3.2 การเก็บข้อมูลในการวางแผนโรงงาน

ก่อนที่จะกล่าวถึงรายละเอียดของวิธีการนั้น จำเป็นอย่างยิ่งที่จะต้องรวบรวมข้อมูลที่จำเป็นในการวางแผนเสียก่อน เพื่อให้สามารถใช้คอมพิวเตอร์ช่วยในการวางแผนได้อย่างถูกต้อง ข้อมูลเหล่านี้ นับเป็นสิ่งที่สำคัญมากในการวางแผนโรงงานทั่วไป ไม่ใช่เฉพาะวิธีใช้ คอมพิวเตอร์เท่านั้น โดยทั่วไปข้อมูลที่จำเป็นในการวางแผนมีอยู่ 2 จำพวกคือ

3.2.1. พื้นที่โรงงานที่ต้องการของขบวนการผลิต (Area Requirement)

พื้นที่ที่ต้องการของแต่ละขบวนการผลิต ซึ่งจะเรียกในที่นี้ใหม่ว่า แผนกการผลิต(Department) จะประกอบด้วย ตำแหน่งงาน (Work Station) จำนวนตั้งแต่ 1 ตำแหน่งงานขึ้นไป โดยที่วัสดุที่จะมาทำการผลิตจะต้องผ่านตำแหน่งงานทั้งแผนก ตามลำดับ ทั้งหมด 100%

ยกตัวอย่างเช่น ถ้ามีชิ้นงานประกอบชนิดหนึ่งผ่านเข้ามาเข้าเครื่องขัด และจากนั้นชิ้นงานทั้งหมด

จะส่งผ่านเข้าไปในห้องพนัสนี้ ในที่นี้อาจกล่าวได้ว่า เครื่องขัดกับห้องพนัสนี้เป็นแผนกเดียวกัน แต่ถ้าหากสมมุติว่า มีเพียง 60% ของชิ้นงานเหล่านั้นถูกส่งเข้าห้องพนัสนี้แล้วจึงส่งไปเข้าสายงานประกอบอีก 40% ที่เหลือถูกส่งข้ามไปที่สายงานประกอบเลย ในกรณีนี้ไม่สามารถที่จะรวมทั้งสองตำแหน่งงานเข้าเป็นแผนกเดียวกันได้ ดังนั้น ในการที่จะจัดแบ่งแผนกการผลิตต่าง ๆ นั้น จำเป็นจะต้องแยกแยะตำแหน่งงานให้ชัดเจน ดังต่อไปนี้

- 1. ต้องกำหนดได้ว่า ตำแหน่งงานนี้ทำหน้าที่อะไร
- 2. มีชิ้นงานที่ต้องผ่านตำแหน่งงานนี้มากน้อยเท่าไร
- 3. ขั้นตอนการผลิตเป็นอย่างไร
- 4. มีเครื่องจักรอยู่เป็นจำนวนเท่าไร ใช้คนงานเท่าไร
- 5. พื้นที่ที่ต้องใช้งานของตำแหน่งงานนี้มีเท่ากับเท่าไร

จากนั้นนักวางแผนก็สามารถที่จะกำหนดพื้นที่ของแผนกการผลิตต่าง ๆ ได้ ดังตัวอย่าง

แผนก	พื้นที่ (ตารางเมตร)
แผนกรับวัสดุ	400
แผนกเก็บวัสดุ	800
แผนกกลึง	600
แผนกเจียรไน	400
แผนกขัดและแต่งผิว	400
แผนกประกอบ	300
โกดังเก็บผลิตภัณฑ์	600

3.2.2 ฟรอม-ทู ชาร์ท (From-To Chart)

ฟรอม-ทู ชาร์ท มีลักษณะเป็นตารางสี่เหลี่ยมที่แสดงให้เห็นถึงปริมาณการเคลื่อนที่ของวัสดุจากแผนกหนึ่งไปยังอีกแผนกหนึ่ง ปริมาณการเคลื่อนที่นี้อาจวัดเป็นชิ้นต่อชั่วโมง หรือถึงต่อสัปดาห์ก็ได้ แล้วแต่ความสะดวกในการนับ แต่สิ่งที่สำคัญที่ต้องระวังคือ หน่วยวัดเหล่านี้จะต้องเป็นหน่วยเดียวกัน ดังรูปที่ 3.1

From To	1	2	3	4	5	6	7	8	9
1		128				60	144		36
2	32		24	112			20		
3		48		128			44		48
4		56			16		120	70	
5				20					
6				18					
7				84					
8			120			20			
9			140			44			

รูปที่ 3.1 แสดง แผนภูมิ จาก-ไป (From-To Chart)

3.2.3 ความสัมพันธ์ระหว่างแผนก (Relationship among Department)

อยู่ในรูปของแผนภูมิความสัมพันธ์ (Relationship Chart) พิจารณาความสัมพันธ์กิจกรรมต่างๆ ของแต่ละแผนกโดยไม่คำนึงถึงลักษณะรูปทรงของพื้นที่ที่เป็นจริงของแต่ละกิจกรรมว่าเป็นอย่างไร การให้โค้ดใช้หลักสากล เป็น A,E,I,O,U และ X ซึ่งมีความหมายดังนี้

- A จำเป็นยิ่ง (Absolutely Necessary)

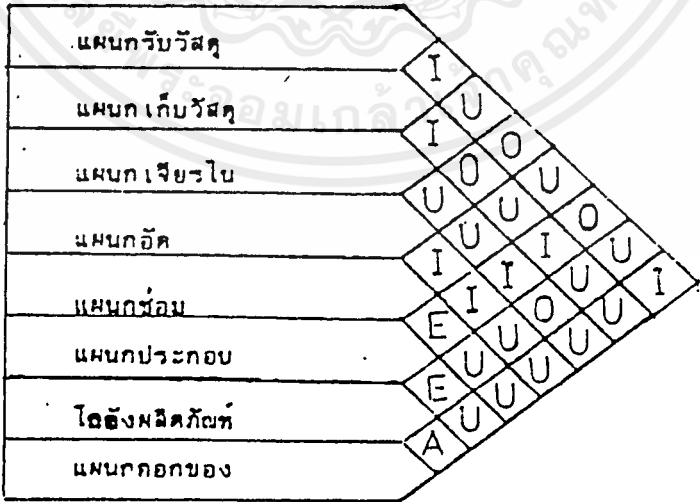
E สำคัญยิ่ง (Essentially Important)

I สำคัญ (Important)

O ปานกลาง (Ordinary)

U ไม่สำคัญ (Unimportant)

X ต้องห้าม (Not Desireable)



รูปที่ 3.2 แสดงลักษณะแผนภูมิความสัมพันธ์ (Relation Chart)

หลังจากนี้แล้ว ข้อมูลชุดนี้ก็จะพร้อมที่จะถูกป้อนเข้าเครื่องคอมพิวเตอร์ เพื่อทำการวางแผนโรงงานต่อไป การวางแผนด้านคอมพิวเตอร์ มี 2 กลุ่ม คือ

- กลุ่มที่ 1 เป็นการวางแผนโรงงานใหม่
- กลุ่มที่ 2 เป็นการปรับปรุงแผนผังโรงงานที่มีอยู่แล้ว

3.3 วิธีการในการใช้คอมพิวเตอร์ช่วยในการวางแผนโรงงาน

การใช้คอมพิวเตอร์ช่วยในการวางแผนโรงงานนี้ ได้มีผู้คิดค้นวิธีการขึ้นมามากมายหลายวิธี แต่วิธีที่เราใช้ในโปรแกรมนี้คือ คอร์แลป (CORLAP : Computerized Relationship Layout Planning) วิธีนี้เป็นการวางแผนโรงงานใหม่ชนิดหนึ่ง ซึ่งได้พัฒนาขึ้นมาในปี ค.ศ. 1967 ข้อมูลที่ต้องการของ CORLAP คือ พื้นที่ของแต่ละแผนกการผลิต และผังแสดงความสัมพันธ์ระหว่างแผนก

CORLAP ประกอบด้วยขั้นตอน 3 ขั้นตอนด้วยกันคือ

1. ขั้นตอนการแปลข้อมูล (Translation Phase)
2. ขั้นตอนการเลือก (Selection Phase)
3. ขั้นตอนการวางแผน (Placement Phase)

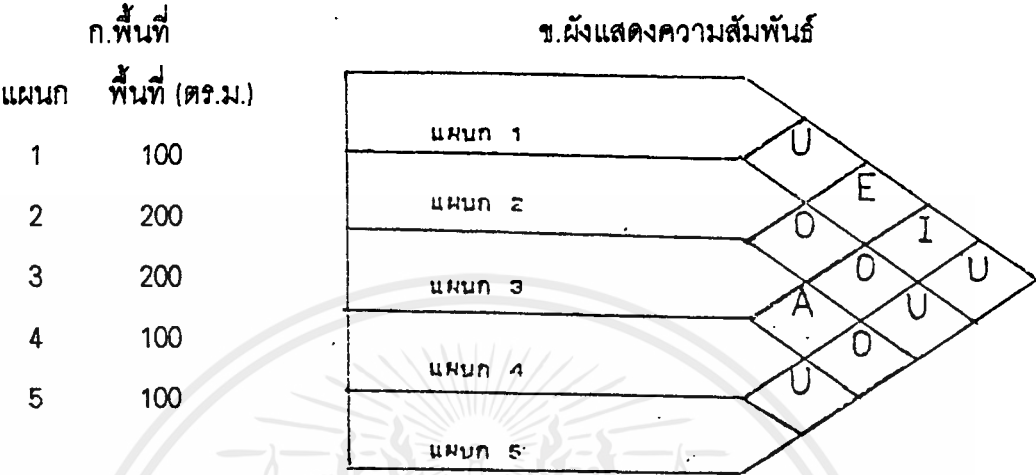
ในขั้นตอนการแปลข้อมูลนั้น คอมพิวเตอร์จะคำนวณอัตราความใกล้ชิด (Total Closeness Rating : TCR) ของแต่ละแผนก โดยการรวมค่าคะแนนตามโค้ตของแผนกนั้น ๆ เข้ากับคะแนนของแผนกอื่น ๆ ที่มีความสัมพันธ์ด้วย

ในขั้นตอนที่ 2 คอมพิวเตอร์จะเลือกแผนกที่มีค่า TCR สูงสุด เป็นอันดับแรก ต่อมาก็จะเลือกแผนกอื่น ที่มีความสัมพันธ์สูงที่สุดกับแผนกนี้ เป็นอันดับที่สอง และก็เลือกต่อไปเรื่อย ๆ ด้วยหลักการเดียวกันนี้จนครบ ในกรณีที่ความสัมพันธ์เท่ากัน ก็จะตัดสินด้วยค่า TCR

ในขั้นตอนสุดท้าย คอมพิวเตอร์จะเลือกแผนกแรกเข้าไปตั้งไว้ก่อน หลังจากนั้นก็จะให้คะแนนแต่ละแผนกใหม่ โดยใช้คะแนนเป็น $A = 60, E = 50, I = 40, O = 30, U = 20$ และ $X = 10$ แล้วทำการเลือกแผนกอื่น ๆ เข้าไปจัดวางเรียงทีละแผนก โดยแต่ละครั้งจะต้องพิจารณาให้มีความสัมพันธ์ รวมตามคะแนนใหม่สูงสุด จนครบทุกแผนก

ตัวอย่างของการวางแผนด้วยวิธี CORLAP

ข้อมูล



ขั้นที่ 1 การแปลข้อมูล (A=6,E=5,I=4,O=3,U=2,X=1)

แผนก	ความสัมพันธ์กับแผนกอื่น	คะแนนรวม	TCR
1	U E I U	2+5+4+2	13
2	U O O U	2+3+3+2	10
3	E O A O	5+3+6+3	17 สูงสุด
4	I O A U	4+3+6+2	15
5	U U O U	2+2+3+2	9

ขั้นที่ 2 การเลือกแผนก

1. จากขั้นที่ 1 จะเห็นว่า แผนก 3 มีค่า TCR สูงสุด คือ 17 จึงเลือกก่อนเป็นอันดับแรก
2. แผนก 4 มีความสัมพันธ์สูงสุดกับ แผนก 3 คือ "A" จึงเลือกเป็นอันดับ 2
3. แผนก 1 มีความสัมพันธ์ต่อมาจาก แผนก 3 คือ "E" จึงเลือกอยู่อันดับ 3
4. แผนก 2 และ แผนก 5 มีความสัมพันธ์ "O" กับ แผนก 3 แต่แผนก 2 มีค่า TCR สูงกว่า จึงเลือก แผนก 2 ก่อน
5. แผนก 5 เลือกเป็นอันดับสุดท้าย

ขั้นที่ 3 การวางแผน

1. แผนก 3 จะถูกเลือกมาตั้งไว้ตรงกลางเป็นอันดับแรก ต่อจากนั้น แผนก 4 จะเลือกมาวางไว้ด้านบนของแผนก 3 ดังรูป.

4	
3	3

2. ต่อมาแผนก 1 จะถูกเลือกเข้ามาเรียงตามตำแหน่งต่าง ๆ ดังนี้

4	1
3	3

1	4	
	3	3

4		
3	3	1

[1] คะแนนรวม 90

$$(4-1) = 1 \text{ (40)}$$

$$(3-1) = E \text{ (50)}$$

เลือกแบบที่ 1

[2] คะแนนรวม 40

$$(4-1) = 1 \text{ (40)}$$

[3] คะแนนรวม 50

$$(1-3) = E \text{ (50)}$$

3. ในขั้นต่อไป คอมพิวเตอร์จะเอาแผนก 2 เข้ามาแล้วจัดได้ 3 แบบ ดังนี้

2	4	1
2	3	3

2	2
4	1
~~3~~	~~3~~

4	1	2
3	3	2

[1] คะแนนรวม 60

$$(2-4) = O \text{ (30)}$$

$$(2-3) = O \text{ (30)}$$

เลือกแบบที่ 1

[2] คะแนนรวม 50

$$(2-4) = O \text{ (30)}$$

$$(2-1) = U \text{ (20)}$$

[3] คะแนนรวม 50

$$(2-1) = U \text{ (20)}$$

$$(3-2) = O \text{ (30)}$$

4. ในขั้นสุดท้าย แผนก 5 จะถูกจัดนำเข้ามาวางในผังดังนี้

2	4	1
2	3	3
	5	

	5	
2	4	1
2	3	3

2	4	1
2	3	3
5		

[1] คะแนนรวม 30

$(5-3) = O (30)$

[2] คะแนนรวม 50

$(5-4) = O (30)$

[3] คะแนนรวม 50

$(2-1) = U (20)$

ผลของการผังด้วยวิธี CORLAP คือแบบที่ 1

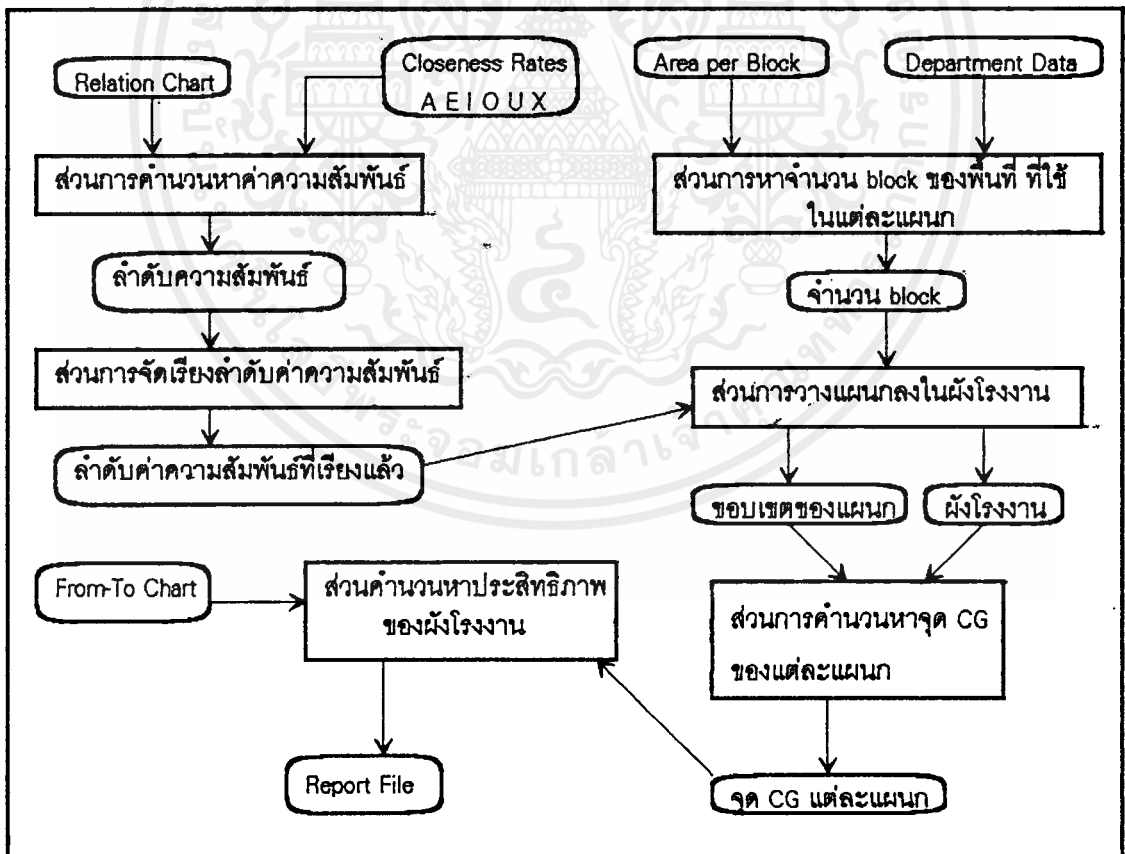
บทที่ 4

รายละเอียดโปรแกรม

โปรแกรมคอมพิวเตอร์ ช่วยในการออกแบบและวางผังโรงงาน จะใช้หลักการของวิธี CORLAP ดังที่ได้กล่าวไว้แล้วในบทที่ 3 โดยดูจากค่าความสัมพันธ์ที่อยู่ใน RELATION CHART เพื่อจัดลำดับแผนกที่มีค่าความสัมพันธ์มากไปน้อยให้ผู้ใช้ทำการเลือกแผนกและนำลงไปวางในผัง โดยที่ในการวางแผนกลงในผังนั้น จะใช้การกำหนดขอบเขตของแผนกที่มีรูปทรงเหลี่ยมประกอบ ขึ้นมาจากบล็อกจำนวนเล็ก ๆ หลาย ๆ บล็อกรวมกัน โดยที่จำนวนบล็อกนั้นคำนวณมาจากข้อมูล พื้นที่แต่ละแผนกนั่นเอง เมื่อวางครบทุกแผนกจะมีการคำนวณหาประสิทธิภาพของผังโรงงาน และสุดท้ายโปรแกรมจะสร้าง REPORT FILE ออกมาให้

4.1 โครงสร้างของโปรแกรม

โครงสร้างของโปรแกรมแสดงอยู่ในรูปที่ 4.1 ดังนี้



รูปที่ 4.1 แสดงโครงสร้างของโปรแกรม

ข้อมูลเบื้องต้นที่ใช้

RELATION CHART : เป็น INPUT ที่ใช้ในการหาค่าสัมพันธที่เป็น A E I O U และ X ผ่านไปให้ส่วนคำนวณหาค่าความสัมพันธ์

DEPARTMENT DATA : เป็น INPUT ที่เก็บข้อมูลแต่ละแผนกเองไว้ ซึ่งได้แก่ พื้นที่ของแผนกต่าง ๆ และชื่อของแผนกนั้น ข้อมูลนี้จะนำไปใช้คำนวณหาจำนวนบล็อกที่ต้องใช้ในแต่ละแผนก

FROM - TO CHART : เป็น INPUT เข้าไปในโปรแกรม เป็นค่าปริมาณการขนถ่ายวัสดุ (จำนวนครั้ง) ระหว่างแผนกต่าง ๆ ในโรงงาน ใช้ในส่วนของ การวัดประสิทธิภาพของผัง

CLOSENESS RATES : เป็นค่าความใกล้ชิด ที่แทนตัวอักษร A, E, I, O, U และ X ด้วยค่าตัวเลขทางคณิตศาสตร์ เช่น แทน

$A = 60, E = 50, I = 40, O = 30, U = 20$ และ $X = 10$

ค่าที่ใช้แทนต้องเรียงจากมากไปหาน้อย $A > E > I > O > U > X$

ใช้ในส่วนการหาค่าความสัมพันธ์เพื่อแปลงความสัมพันธ์เป็นจำนวนตัวเลข

AREAR PER BLOCK : เป็นค่าที่กำหนดว่า 1 BLOCK ที่อยู่ในผังแทนด้วยพื้นที่กี่หน่วย เช่น ถ้ากำหนด AREA PER BLOCK = 5 หมายถึง 1 BLOCK แทนพื้นที่ 5 หน่วย ใช้เป็นข้อมูลในส่วนการหาจำนวน BLOCK ของพื้นที่ที่ใช้ในแต่ละแผนก

จากนั้นข้อมูลเหล่านี้จะถูกใช้ในส่วนต่าง ๆ คือ

ส่วนการคำนวณหาค่าความสัมพันธ์

มีข้อมูลที่ใช้คือ RELATION CHART และ CLOSENESS RATES เป็นส่วนคำนวณความสัมพันธ์ เพื่อหาค่า TCR (Total Closeness Rating) และ PCR (Placed Closeness Rate) ตามหลักการ CORLAP ได้ผลออกมาในรูปของค่าความสัมพันธ์รวมของแผนกใดแผนกหนึ่งกับแผนกอื่น ๆ ที่เหลือทุกแผนก

โดยที่ค่า TCR คือค่าความสัมพันธ์รวม ,PCR คือค่าความสัมพันธ์ระหว่างแผนกที่ยังไม่ได้วางในผังกับแผนกที่วางลงในผังแล้ว

T
35231

ส่วนการจัดเรียงลำดับค่าความสัมพันธ์

เป็นส่วนเรียงลำดับค่าความสัมพันธ์จากมากไปหาน้อย ได้ผลออกมาเป็นลำดับค่าความสัมพันธ์ที่เรียงลำดับแล้ว เพื่อที่จะส่งค่าไปให้ส่วนของการวางแผนลงในผังโรงงาน

ส่วนการหาจำนวน BLOCK ของพื้นที่ที่ใช้ในแต่ละแผนก

ทำการหาจำนวน BLOCK ที่แต่ละแผนกต้องใช้ ส่งไปให้ส่วนของการวางแผนลงในผังโรงงาน

ส่วนของการวางแผนลงในผังโรงงาน

เป็นส่วนที่ให้ผู้ใช้โปรแกรมได้กำหนดขอบเขตของแผนกต่าง ๆ ในผัง ได้ผลออกมาเป็นขอบเขตของแผนกและผังโรงงาน

ส่วนการคำนวณหาจุด CG ของแต่ละแผนก

คำนวณหาจุดศูนย์กลางของแต่ละแผนก โดย INPUT ขอบเขตของแผนกและผังโรงงาน เข้ามาทำการคำนวณ

ส่วนการคำนวณหาประสิทธิภาพของผังโรงงาน

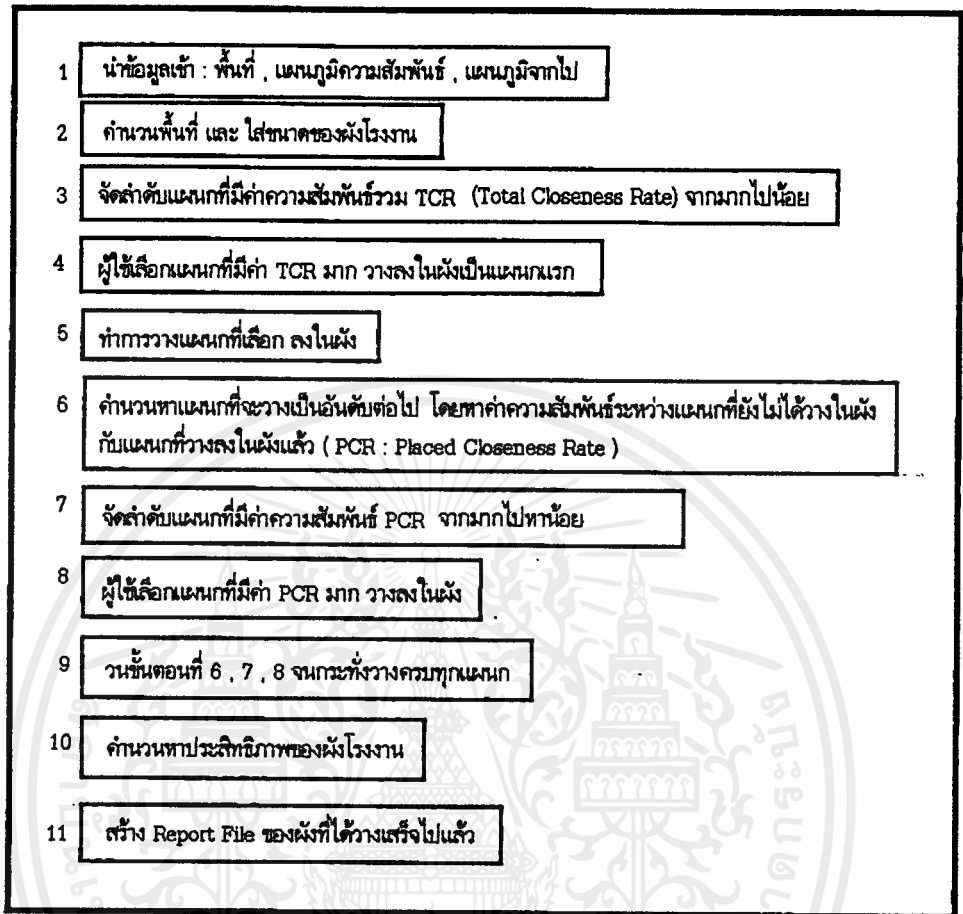
ค่าจาก FROM - TO CHART และจุด CG ของแต่ละแผนกเข้ามาคำนวณหาประสิทธิภาพของผังโรงงานนั้น และสร้าง REPORT FILE ออกมาซึ่งเป็นขั้นตอนสุดท้ายของโปรแกรม

4.2 ขั้นตอนการทำงานของโปรแกรม

โปรแกรมการวางแผนผังโรงงานโดยใช้การวิเคราะห์ความสัมพันธ์เป็นโปรแกรมที่ใช้ในการวางแผนผังโรงงานใหม่ โปรแกรมจะมีการนำข้อมูล 3 อย่างคือ

1. พื้นที่ของแต่ละแผนกในโรงงาน
2. แผนภูมิความสัมพันธ์ (Relation Chart)
3. แผนภูมิจาก--ไป (From-To Chart)

มีขั้นตอนโดยสรุปดังรูปที่ 4.2



รูปที่ 4.2 แสดงขั้นตอนการทำงานของโปรแกรม

เมื่อผ่านขั้นตอนดังกล่าวข้างต้น จะได้ ผังโรงงานที่เป็น Block Layout และ ผู้ใช้สามารถทดลองวางผังโรงงานได้ใหม่ จนเป็นที่พอใจ

4.3 รายละเอียดของโปรแกรมส่วนต่าง ๆ

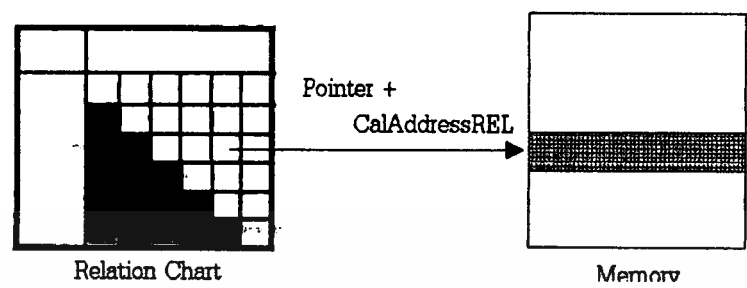
4.3.1 ส่วนของข้อมูลเบื้องต้น

4.3.1.1 Relation Chart

Relation Chart แสดงให้เห็นถึงความสัมพันธ์ระหว่างแผนก มีค่าที่ใส่ลงไปเป็น A,E,I,O,U และ X แผนกหนึ่งไปยังอีกแผนกหนึ่ง Relation Chart เป็นข้อมูลที่จำเป็นในการคำนวณหาค่าความสัมพันธ์ระหว่างแผนก มีลักษณะคล้าย Metrix ขนาด $(n-1) \times (n-1)$ โดย n คือ จำนวนแผนกหรือขนาดของ Relation Chart แต่จะใช้เฉพาะช่องที่อยู่ทางซ้ายบนเท่านั้น

ดังนั้น การเก็บข้อมูลลงใน Memory จะไม่ใช่ Array 2 มิติ แต่จะเป็น Array 1 มิติ เท่านั้น

การอ้างถึงข้อมูลที่เก็บใน Memory จึงต้องมีวิธีการ Map จาก Relation Chart ไปยัง Array 1 มิติ โดยบวกค่าที่ได้จากฟังก์ชัน CalAddressREL



การคำนวณหาตำแหน่งของฟังก์ชัน CalAddressREL

```
int CalAddressREL ( int n, int x, int y)
    n = จำนวนแผ่นกใน Relation หรือ ขนาดของ Relation Chart
    x = หมายเลขแผ่นกในแนวแกน X ของ Relation Chart
    y = หมายเลขแผ่นกในแนวแกน Y ของ Relation Chart
```

สมมติ ขนาดของ Relation Chart = 6 ดังนั้น n = 6 จะได้ การอ้าง Address ดังนี้

	2	3	4	5	6	
1	[0]	[1]	[2]	[3]	[4]	Pointer + CalAddressREL →
2		[5]	[6]	[7]	[8]	
3			[9]	[10]	[11]	
4				[12]	[13]	
5					[14]	

[0]

[1]

[2]

....

[13]

[14]

[] แสดงตำแหน่ง address ของ array 1 มิติ ที่เก็บค่าของ Relation Chart
 ขนาดของ Array คำนวณได้จากสูตร

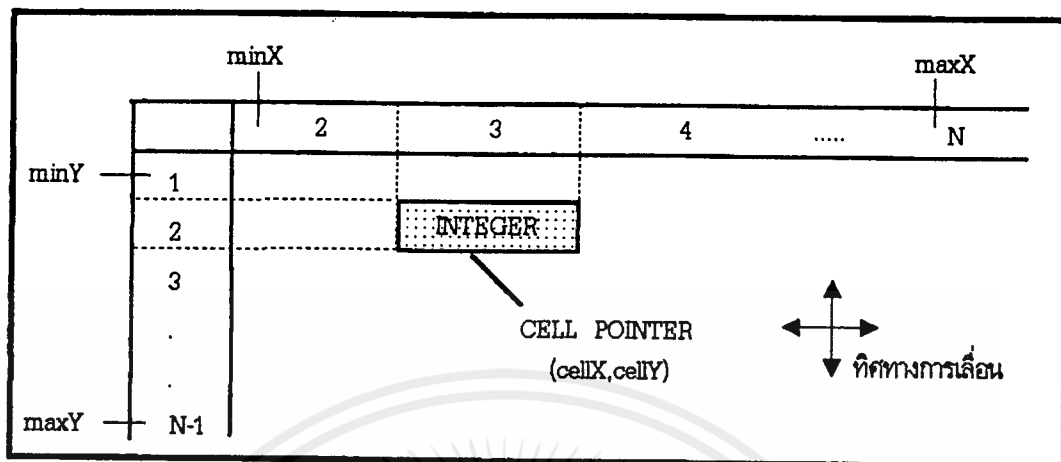
$$\text{ขนาดของ Relation Chart Array} = (N-1)*(N) / 2$$

ส่วน address ของ Array หาได้จากสูตร

เมื่อ $2 \leq X \leq N$, $1 \leq Y \leq N-1$ และ $X > Y$

$$\text{address} = (N-1)(N)/2 - (N-Y)(N-Y+1)/2 + X-Y-1$$

โครงสร้างข้อมูลของ Relation Chart



```
typedef struct {
    char filename[13]; int size;
    int minX; int maxX;
    int minY; int maxY;
    int cellX; int cellY;
    int *pRel;
} REL;
```

REL เป็นชื่อ TYPE ของ Relation Chart โดยประกอบด้วย

filename คือ ชื่อไฟล์ที่ใช้เก็บข้อมูลลงใน Disk

size คือ ขนาดของ Relation Chart

minX คือ ขอบเขตล่าง ของการแสดงผล ทางด้านแกน X โดยมีค่าอยู่ระหว่าง 2..size

maxX คือ ขอบเขตบน ของการแสดงผล ทางด้านแกน X โดยมีค่าอยู่ระหว่าง 2..size

minY คือ ขอบเขตล่าง ของการแสดงผล ทางด้านแกน Y โดยมีค่าอยู่ระหว่าง 1..size-1

maxY คือ ขอบเขตบน ของการแสดงผล ทางด้านแกน Y โดยมีค่าอยู่ระหว่าง 1..size-1

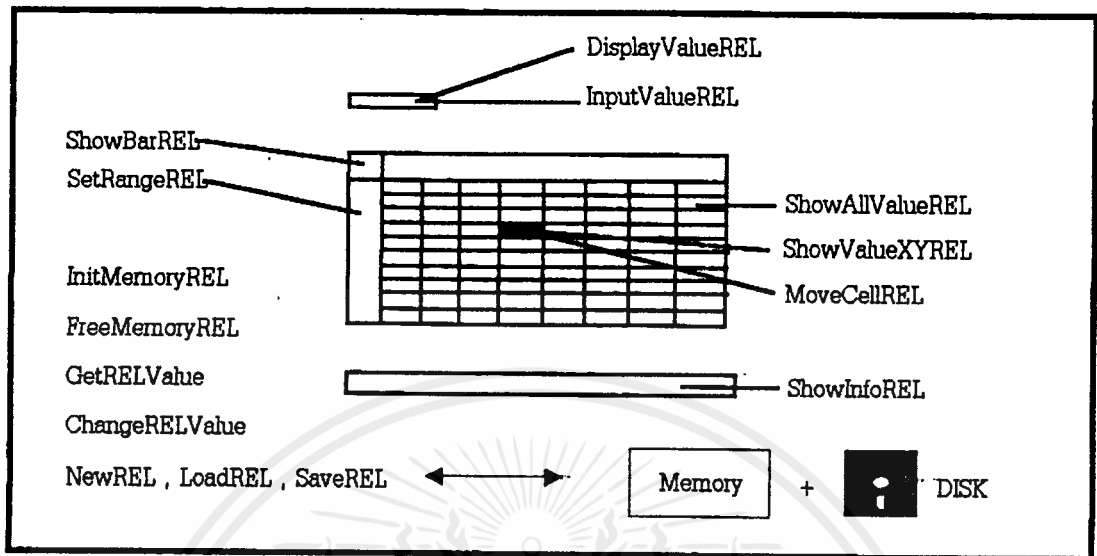
cellX, cellY คือ พิกัด (cellX, cellY) แสดงถึงตำแหน่งปัจจุบันของ Cell Pointer

ที่ปรากฏบนจอภาพ

pRel คือ pointer of integer ที่ชี้ไปยังเนื้อที่ของหน่วยความจำ ที่จองไว้เพื่อเก็บ

ค่าต่าง ๆ ใน Relation Chart

Relation Chart Functions



void RelationChart(REL *rel,int command)

ทำหน้าที่

เป็นส่วนหลักในการเรียกใช้งานของ Relation Chart

command เป็นคำสั่งให้เลือกว่า อยู่ใน mode สร้างใหม่ หรือ mode แก้ไข

command : cNEWREL = ให้ยกเลิกของเดิม และ สร้างใหม่

cEDITREL = ให้แก้ไขค่าใน Relation Chart

int far *InitMemoryREL(int rel_size)

ทำหน้าที่

จองเนื้อที่หน่วยความจำ เป็นค่า integer จำนวน $(rel_size-1) * (rel_size) / 2$

ถ้าจองได้ จะ return ค่า address ของหน่วยความจำที่จอง

ถ้าไม่ได้ จะ ขึ้นข้อความเตือน และ return ค่า NULL

void FreeMemoryREL(int far *prel)

ทำหน้าที่

Free memory หรือ ยกเลิก หน่วยความจำที่จองไว้ ที่ address prel

int GetRELValue(REL rel,int x,int y)

ทำหน้าที่

ดึงค่าข้อมูลจาก Relation Chart ที่ตำแหน่ง x,y

โดย $2 \leq X \leq rel.size$ $1 \leq Y \leq rel.size - 1$

void ChangeRELValue(int value,REL *rel,int x,int y)

ทำหน้าที่

เปลี่ยนค่าข้อมูลใน Relation Chart ที่ตำแหน่ง x,y

โดย $2 \leq X \leq rel.size$, $1 \leq Y \leq rel.size - 1$

void NewREL(REL *rel)

ทำหน้าที่ สร้าง Relation Chart ขึ้นมาใหม่

void LoadREL(REL *rel)

ทำหน้าที่ Load ข้อมูล Relation Chart จาก Disk เข้าสู่ หน่วยความจำ

void SaveREL(REL *rel)

ทำหน้าที่ Save ข้อมูล Relation Chart จาก หน่วยความจำ ลง Disk

void SetRangeREL(REL *rel,int minx,int miny,int maxx,int maxy)

ทำหน้าที่ ทำการ set ค่า minx,miny,maxX,maxY ที่อยู่ใน rel
ให้เป็นค่าของ minx,miny,maxx,maxy ตามลำดับ

void ShowBarREL(REL *rel,int StartCol,int StartRow)

ทำหน้าที่ แสดง Relation Chart's bar บนหน้าจอ โดยเริ่มแสดงค่าที่เก็บใน Relation Chart
ตั้งแต่พิกัด (StartCol,StartRow) จนกระทั่งสุดหน้าจอ
และมีการ Updates ค่าของ minx,miny,maxX,maxY ใน rel ด้วย

void ShowAllValueREL(REL rel)

ทำหน้าที่ แสดงค่าที่อยู่ใน Relation Chart ทุกค่าที่อยู่ในขอบเขต matrix ตั้งแต่
(minx,miny) ถึง (maxX,maxY)

void ShowValueXYREL(REL rel,int x,int y,int flag)

ทำหน้าที่ แสดงค่าของ Relation Chart พิกัด (x,y)

void ShowInfoREL(REL rel)

ทำหน้าที่ แสดงข้อความที่บรรทัดที่ 25 ของจอภาพ

void MoveCellREL(REL *rel,int direction)

ทำหน้าที่ เลื่อน Cell Pointer ไปมาบนจอภาพ ตามทิศทาง direction มีค่าบอกทิศทาง
kLEFT = เลื่อนซ้าย kRIGHT = เลื่อนขวา
kUP = เลื่อนขึ้น kDOWN = เลื่อนลง
kHOME = ไปยังตำแหน่งเริ่มต้น (1,1)
และมีการ Updates ค่าของ minx,miny,maxX,maxY ใน rel ด้วย

void DisplayValueREL(REL rel)

ทำหน้าที่ แสดงข้อความ "REL [.] AND [.] ." บนบรรทัดที่ 2

ตามด้วย ค่าของข้อมูล ของ Relation Chart ตำแหน่งที่ Cell Pointer ชี้อยุ

void InputValueREL(REL *rel,int message)

ทำหน้าที่ รับตัวอักษร A,E,I,O,U หรือ X ที่บรรทัดที่ 2 ของจอภาพ

ถ้ากด Enter จะทำการเปลี่ยนค่าใน Relation Chart ตำแหน่ง (celIX,celIY)

ถ้ากด Esc จะยกเลิกการเปลี่ยนแปลง

int CalAddressREL(int n,int x,int y)

ทำหน้าที่ คำนวณค่า address ของ Array ที่เก็บค่าของ Relation Chart

Return ค่า ที่จะนำไปบวกเพิ่มกับ pRel ในการอ้างถึง Memory

n = จำนวนแผ่นกใน Relation หรือ ขนาดของ Relation Chart

x = หมายเลขแผ่นกในแนวแกน X ของ Relation Chart

y = หมายเลขแผ่นกในแนวแกน Y ของ Relation Chart

วิธีการใช้งาน Relation Chart

การใช้งาน Relation Chart จะขอยกตัวอย่าง เริ่มจาก main() ในภาษา C โดยขั้นแรกต้องกำหนด ตัวแปรที่เป็นชนิด REL เสียก่อนดังนี้

```
main()
{
    REL rel;
    ...
}
```

หลังจากนั้น ทำการจองเนื้อที่ในหน่วยความจำ ใช้ฟังก์ชัน *InitMemoryREL(int rel_size) ผ่านค่า

rel_size คือ ขนาดของ Relation Chart และรับค่า pointer address ที่ return กลับ เก็บลงใน rel.pRel

```
rel.size = 10;
```

```
rel.pRel = InitMemoryREL(rel.size);
```

ต่อจากนั้น การเข้าสู่ Relation Chart ครั้งแรก ต้องใช้ ฟังก์ชันหลัก RelationChart(....)

```
RelationChart( &rel,cNEWREL) ;
```

จะปรากฏ Relation Chart บนหน้าจอ สามารถเลื่อน Cell Pointer ใส่ค่าได้ และ ออกจาก Relation Chart โดยการกด ปุ่ม ESC

หากต้องการเข้าไปแก้ไขค่าใน Relation Chart ภายหลัง สามารถทำได้โดยใช้คำสั่ง

```
RelationChart( &rel, cEDITREL);
```

สุดท้าย ก่อนที่จะออกจากโปรแกรม จำเป็นต้องมีการยกเลิกหน่วยความจำที่จองไว้เสียก่อน
โดยการใช้คำสั่ง FreeMemoryREL(...)

```
FreeMemoryREL( rel.pRel );
```

สรุปโปรแกรมได้ดังนี้

```
main()
```

```
{      REL rel;
```

```
...
```

```
rel.size = 10;
```

```
rel.pRel = InitMemoryREL( rel.size );
```

```
...
```

... ส่วนของ เมนู ทำการเลือก...

```
สร้างใหม่ :      RelationChart( &rel , cNEWREL );
```

```
แก้ไขค่า :      RelationChart( &rel , cEDITREL );
```

```
...
```

```
FreeMemoryREL( rel.pRel );
```

```
}
```

การอ่านค่าและการใส่ค่า Relation Chart

การดึงค่า หรือการอ่านค่าจาก Memory ที่เก็บ Relation Chart สามารถทำได้โดยใช้ ฟังก์ชัน
GetRELValue(...,...) อ่านค่าใน ตำแหน่ง Column;Row ที่ (x,y)

```
int value,x,y;
```

```
...
```

```
x = 2; y = 1;
```

```
value = GetRELValue( rel , x , y );
```

ตัวอย่างนี้ อ่านค่า Column=2, Row=1; จาก Relation Chart rel

การใส่ค่าลงใน Memory ของ Relation Chart ก็ใช้ ฟังก์ชัน ChangeRELValue(.....)

ChangeRELValue(value , &rel , x , y);

มีข้อจำกัดอยู่ก็คือ

- ค่า x มีค่าตั้งแต่ 2..rel.size

- ค่า y มีค่าตั้งแต่ 1..rel.size -1

การ LOAD และ SAVE Relation Chart

การ LOAD เป็นการอ่านข้อมูล Relation Chart จาก ไฟล์ที่เก็บใน DISK มีฟังก์ชันสำหรับการ LOAD คือ LoadREL(...) ต้องใช้หลังจากจองหน่วยความจำโดยฟังก์ชัน InitMemoryREL(..) แล้ว

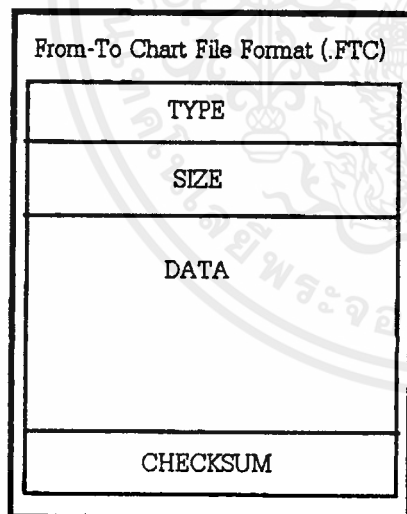
LoadREL(&rel);

การ SAVE ก็เป็นการเก็บข้อมูลลงไฟล์ใน DISK มีฟังก์ชันสำหรับการ SAVE คือ SaveFT(..)

ดังนี้

SaveREL(&rel);

ไฟล์ที่เก็บ Relation Chart จะมีสกุลเป็น *.REL " มี Format ในการเก็บดังนี้



TYPE เป็นเลข Integer มีค่าเท่ากับ 2 เพื่อแสดง
ว่าเป็น Relation Chart File

SIZE เป็นเลข Integer แสดงขนาดของ Relation
Chart

DATA เป็น Array of Integer มีขนาด
 $SIZE \times (SIZE - 1) / 2$

CHECKSUM เป็นผลรวมของค่าที่มีในไฟล์
เพื่อตรวจสอบว่ามี error หรือไม่

โปรแกรมในส่วนของ Relation chart นี้ อยู่ในไฟล์ชื่อ RELATION.C และ RELATION.H

4.3.1.2 Department Data

โปรแกรมส่วนของ Department Data นั้น อยู่ในไฟล์ PLANTLAY.C และ PLANTLAY.H
โครงสร้างข้อมูล (Data Structure) ที่ใช้เก็บข้อมูลของแผนกต่าง ๆ อันได้แก่ ชื่อแผนก และ
พื้นที่ของแผนก มีโครงสร้างดังนี้

```
typedef struct {  
    char dName[21];  
    int dArea;  
} AREA;
```

dName = ชื่อแผนก

dArea = พื้นที่ของแผนก

AREA เป็นชื่อ STRUCTURE TYPE

ส่วน Department Data มีโครงสร้างข้อมูลดังนี้

```
typedef struct {  
    char filename[13];  
    int size;  
    int top,bottom;  
    int cell;  
    int AreaPerBlock;  
    unsigned long total;  
    AREA far *pArea;  
} DEPT;
```

DEPT เป็นชื่อ STRUCTURE TYPE ของ Department Data โดยประกอบด้วย

filename คือ ชื่อไฟล์ที่ใช้เก็บข้อมูลลงใน disk

size คือ จำนวนของแผนก

top คือ ขอบเขตบนของการแสดงผล

bottom คือ ขอบเขตล่างของการแสดงผล

cell คือ ตำแหน่ง cell pointer

AreaPerBlock คือ ค่าของ พื้นที่ ต่อ block 1 block ในการวางผัง

total คือ ค่าผลรวมพื้นที่ของแผนกทั้งหมด

pArea คือ pointer of struct AREA ที่ไปยังหน่วยความจำที่จองไว้ เพื่อเก็บชื่อและพื้นที่ของแผนก

Department Data Functions

AREA *InitMemoryAREA(int items)

ทำหน้าที่ จองเนื้อที่หน่วยความจำ เป็น Array of Structure AREA เป็นจำนวน items
return ค่า pointer ที่ไปยังหน่วยความจำ

void FreeMemoryAREA(AREA far *pArea)

ทำหน้าที่ ยกเลิกหน่วยความจำที่ pArea

void FillNameArea(DEPT dpt)

ทำหน้าที่ แสดงผลข้อมูล Department Data บนจอภาพ

void EditAreaPerBlock(DEPT dpt)

ทำหน้าที่ แก้ไขค่า dpt.AreaPerBlock

void EnterNameArea(DEPT *dpt)

ทำหน้าที่ input ชื่อ และ พื้นที่ของแผนก

void EditNameArea(DEPT *dpt)

ทำหน้าที่ แก้ไข ชื่อ และ พื้นที่ของแผนก

void NewDEPT(DEPT dpt)

ทำหน้าที่ สร้าง Department Data ขึ้นมาใหม่

void LoadDEPT(DEPT dpt)

ทำหน้าที่ อ่านข้อมูล Department Data จาก disk เข้าสู่หน่วยความจำ

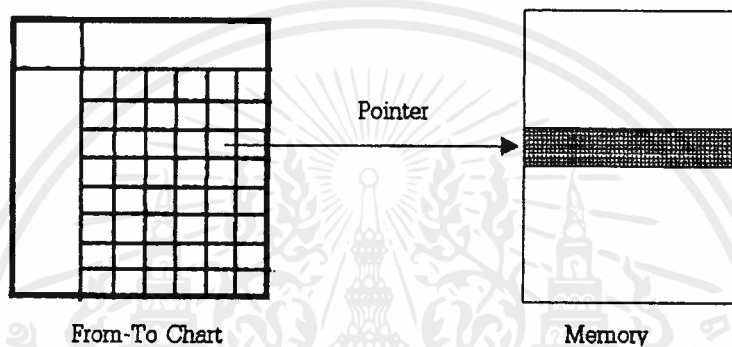
void SaveDEPT(DEPT dpt)

ทำหน้าที่ เขียนข้อมูล department Data จาก จากหน่วยความจำ ลงสู่ disk

4.3.1.3 From-To Chart

ฟอร์ม-ทู ชาร์ท มีลักษณะเป็นตารางสี่เหลี่ยมที่แสดงให้เห็นถึงปริมาณการเคลื่อนที่ของวัสดุจากแผนกหนึ่งไปยังอีกแผนกหนึ่ง ปริมาณการเคลื่อนที่นี้จะใช้หน่วยวัดที่เป็นหน่วยเดียวกัน

ฟอร์ม-ทู ชาร์ท เป็นข้อมูลที่จำเป็นต้องใช้ในการวิเคราะห์ หาประสิทธิภาพของผังโรงงาน ดังนั้น ส่วนของการ INPUT จึงต้องทำการเขียนโปรแกรมเพื่อเก็บค่าในรูปแบบของฟอร์ม-ทูชาร์ท โดยทำการจองเนื้อที่ ใน Memory ในลักษณะ Array of integer และใช้ Pointer ในการดูค่าและเก็บค่า



โครงสร้างข้อมูลของ From-To Chart

```
typedef struct {  
    char filename[13];    int size;  
    int minX;    int maxX;  
    int minY;    int maxY;  
    int cellX;    int cellY;  
    int *pFromto;  
} FROMTO;
```

FROMTO เป็นชื่อ TYPE ของ From-To Chart โดยประกอบด้วย

filename คือ ชื่อไฟล์ที่ใช้เก็บข้อมูลลงใน Disk

size คือ ขนาดของ From-To Chart มีลักษณะเป็น matrix ขนาด size x size

เช่น ถ้า size = 10 จะได้ matrix ขนาด 10 x 10

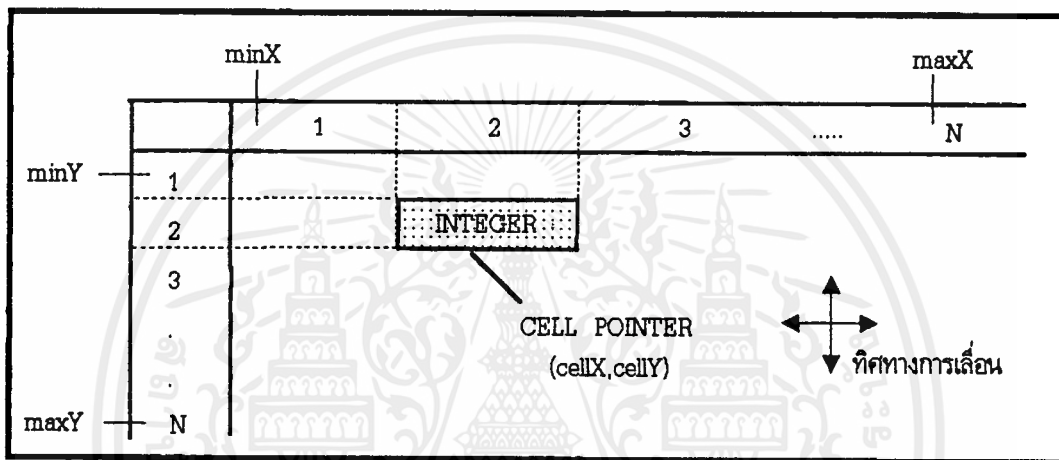
minX คือ ขอบเขตล่าง ของการแสดงผล ทางด้านแกน X โดยมีค่าอยู่ระหว่าง 1..size

maxX คือ ขอบเขตบน ของการแสดงผล ทางด้านแกน X โดยมีค่าอยู่ระหว่าง 1..size

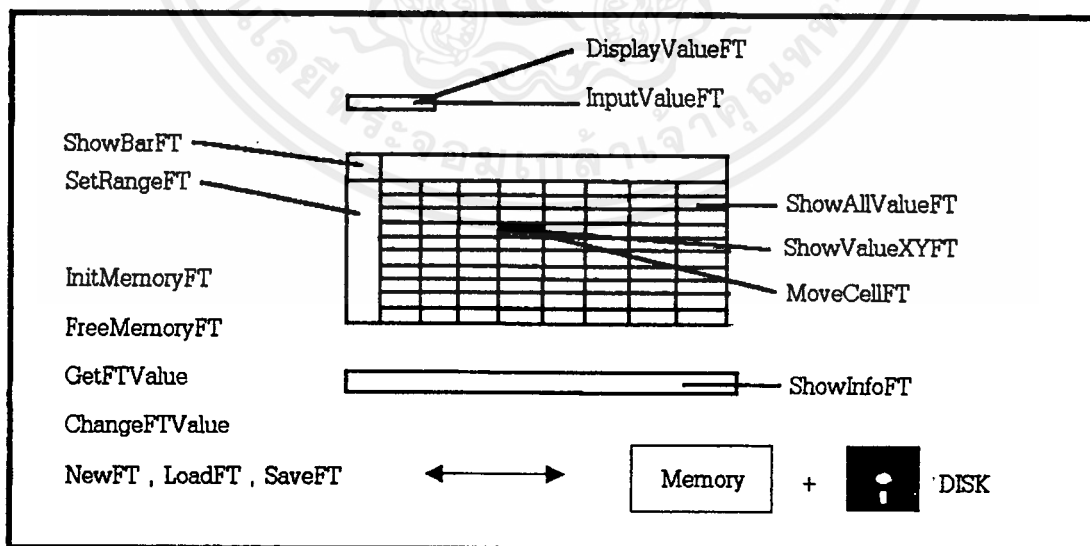
minY คือ ขอบเขตล่าง ของการแสดงผล ทางด้านแกน Y โดยมีค่าอยู่ระหว่าง 1..size
 maxY คือ ขอบเขตบน ของการแสดงผล ทางด้านแกน Y โดยมีค่าอยู่ระหว่าง 1..size
 cellX, cellY คือ พิกัด (cellX, cellY) แสดงถึงตำแหน่งปัจจุบันของ Cell Pointer

ที่ปรากฏบนจอภาพ

pFromto คือ pointer of integer ที่ชี้ไปยังเนื้อที่ของหน่วยความจำ ที่จองไว้เพื่อเก็บ
 ค่าต่าง ๆ ใน From-To Chart



From-To Chart Functions



void FromToChart(FROMTO *ft,int command)

ทำหน้าที่ เป็นส่วนหลักในการเรียกใช้งานของ From-To Chart
command เป็นคำสั่งให้เลือกว่า อยู่ใน mode สร้างใหม่ หรือ mode แก้ไข
command : cNEWFT = ให้ยกเลิกของเดิม และ สร้างใหม่
cEDITFT = ให้แก้ไขค่าใน From-To Chart

int *InitMemoryFT(int ft_size)

ทำหน้าที่ จองเนื้อที่หน่วยความจำ เป็นค่า integer จำนวน $ft_size * ft_size$
ถ้าจองได้ จะ return ค่า address ของหน่วยความจำที่จอง
ถ้าไม่ได้ จะ ขึ้นข้อความเตือน และ return ค่า NULL

void FreeMemoryFT(int *pfromto)

ทำหน้าที่ Free memory หรือ ยกเลิก หน่วยความจำที่จองไว้ ที่ address pfromto

int GetFTValue(FROMTO fromto,int x,int y)

ทำหน้าที่ ดึงค่าข้อมูลจาก From-To Chart ที่ตำแหน่ง x,y
โดย $1 \leq X \leq \text{fromto.size}$ $1 \leq Y \leq \text{fromto.size}$

void ChangeFTValue(int value,FROMTO *fromto,int x,int y)

ทำหน้าที่ เปลี่ยนค่าข้อมูลใน From-To Chart ที่ตำแหน่ง x,y
โดย $1 \leq X \leq \text{fromto.size}$, $1 \leq Y \leq \text{fromto.size}$

void NewFT(FROMTO *fromto)

ทำหน้าที่ สร้าง From-To Chart ขึ้นมาใหม่

void LoadFT(FROMTO *fromto)

ทำหน้าที่ Load ข้อมูล From-To Chart จาก Disk เข้าสู่ หน่วยความจำ

void SaveFT(FROMTO *fromto)

ทำหน้าที่ Save ข้อมูล From-To Chart จาก หน่วยความจำ ลง Disk

void SetRangeFT(FROMTO *fromto,int minx,int miny,int maxx,int maxy)

ทำหน้าที่ ทำการ set ค่า minX,minY,maxX,maxY ที่อยู่ใน fromto
ให้เป็นค่าของ minx,miny,maxx,maxy ตามลำดับ

void ShowBar(FROMTO *fromto,int StartCol,int StartRow)

ทำหน้าที่ แสดง From-To Chart's bar บนหน้าจอ โดยเริ่มแสดงค่าที่เก็บใน From-To Chart
ตั้งแต่พิกัด (StartCol,StartRow) จนกระทั่งสุดหน้าจอ
และมีการ Updates ค่าของ minX,minY,maxX,maxY ใน fromto ด้วย

void ShowAllValue(FROMTO fromto)

ทำหน้าที่ แสดงค่าที่อยู่ใน From-To Chart ทุกค่าที่อยู่ในขอบเขต matrix ตั้งแต่
(minX,minY) ถึง (maxX,maxY)

void ShowValueXY(FROMTO fromto,int x,int y)

ทำหน้าที่ แสดงค่าของ From-To Chart พิกัด (x,y)

void ShowInfo(FROMTO fromto)

ทำหน้าที่ แสดงข้อความที่บรรทัดที่ 25 ของจอภาพ

void MoveCell(FROMTO *fromto,int direction)

ทำหน้าที่ เลื่อน Cell Pointer ไปมาบนจอภาพ ตามทิศทาง direction มีค่าบอกทิศทาง
kLEFT = เลื่อนซ้าย kRIGHT = เลื่อนขวา
kUP = เลื่อนขึ้น kDOWN = เลื่อนลง
kHOME = ไปยังตำแหน่งเริ่มต้น (1,1)
และมีการ Updates ค่าของ minX,minY,maxX,maxY ใน fromto ด้วย

void DisplayValue(FROMTO fromto)

ทำหน้าที่ แสดงข้อความ "FROM [.] TO [.] ." บนบรรทัดที่ 2
ตามด้วย ค่าของข้อมูล ของ From-To Chart ตำแหน่งที่ Cell Pointer ชี้อยู่

void InputValue(FROMTO *fromto,int message)

ทำหน้าที่ รับเลข integer ที่บรรทัดที่ 2 ของจอภาพ
ถ้ากด Enter จะทำการเปลี่ยนค่าใน From-To Chart ตำแหน่ง (cellX,cellY)
ถ้ากด Esc จะยกเลิกการเปลี่ยนแปลง

วิธีการใช้งาน From-To Chart

การใช้งาน From-To Chart จะขอยกตัวอย่าง เริ่มจาก main() ในภาษา C
โดยขั้นแรกต้องกำหนด ตัวแปรที่เป็นชนิด FROMTO เสียก่อนดังนี้

```
main()
{
    FROMTO ft;
```

...

หลังจากนั้น ทำการจองเนื้อที่ในหน่วยความจำ ใช้ฟังก์ชัน `*InitMemoryFT(int ft_size)` ผ่านค่า `ft_size` คือ ขนาดของ From-To Chart และรับค่า pointer address ที่ return กลับ เก็บลงใน `ft.pFromto`

```
ft.size = 10;
```

```
ft.pFromto = InitMemoryFT(ft.size);
```

ต่อจากนั้น การเข้าสู่ From-To Chart ครั้งแรก ต้องใช้ ฟังก์ชันหลัก `FromToChart(....)`

```
FromToChart( &ft,cNEWFT) ;
```

จะปรากฏ From-To Chart บนหน้าจอ สามารถเลื่อน Cell Pointer ใส่ค่าได้ และ ออกจาก From-To Chart โดยการกด ปุ่ม ESC

หากต้องการเข้าไปแก้ไขค่าใน From-To Chart ภายหลัง สามารถทำได้โดยใช้คำสั่ง

```
FromToChart( &ft, cEDITFT);
```

สุดท้าย ก่อนที่จะออกจากโปรแกรม จำเป็นต้องมีการยกเลิกหน่วยความจำที่จองไว้เสียก่อน โดยการใช้คำสั่ง `FreeMemoryFT(..)`

```
FreeMemoryFT( ft.pFromto );
```

สรุปโปรแกรมได้ดังนี้

```
main()
```

```
{
    FROMTO ft;
```

...

```
ft.size = 10;
```

```
ft.pFromto = InitMemoryFT( ft.size );
```

...

... ส่วนของ เมนู ทำการเลือก...

สร้างใหม่ : `FromToChart( &ft , cNEWFT );`

แก้ไขค่า : `FromToChart( &ft , cEDITFT );`

```

...
FreeMemoryFT( ft.pFromto );
}

```

การอ่านค่าและการใส่ค่า From-To Chart

การดึงค่า หรือการอ่านค่าจาก Memory ที่เก็บ From-To Chart สามารถทำได้โดยใช้ ฟังก์ชัน GetFTValue(.....) อ่านค่าใน ตำแหน่ง Column,Row ที่ (x,y)

```
int value,x,y;
```

```
...
```

```
x = 1; y = 1;
```

```
value = GetFTValue( ft , x , y );
```

ตัวอย่างนี้ อ่านค่า Column=1, Row=1; จาก From-To Chart ft

การใส่ค่าลงใน Memory ของ From-To Chart ก็ใช้ ฟังก์ชัน ChangeFTValue(.....)

```
ChangeFTValue( value , &ft , x , y );
```

มีข้อจำกัดอยู่ก็คือ

- ค่าที่เก็บใน From-To Chart นั้น เป็น ชนิด integer มีขนาด -32767 ... 32767
- ค่า x,y มีค่าตั้งแต่ 1..ft.size ถ้าหากเกินจะรับค่าได้เท่ากับ 0

การ LOAD และ SAVE From-To Chart

การ LOAD เป็นการอ่านข้อมูล From-To Chart จาก ไฟล์ที่เก็บใน DISK มีฟังก์ชันสำหรับการ LOAD คือ LoadFT(...) ต้องใช้หลังจากจองหน่วยความจำโดยฟังก์ชัน InitMemoryFT(..) แล้ว

```
LoadFt( &ft );
```

การ SAVE ก็เป็นการเก็บข้อมูลลงไฟล์ใน DISK มีฟังก์ชันสำหรับการ SAVE คือ SaveFT(..)

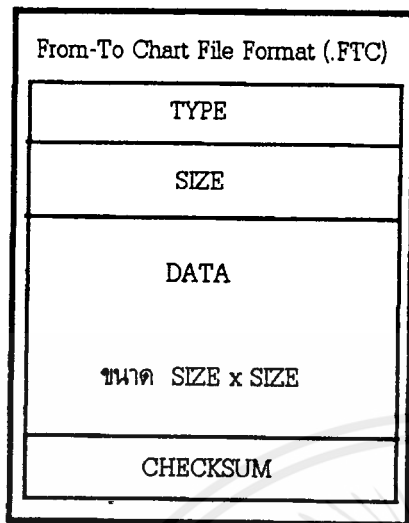
ดังนี้

```
SaveFT( &ft );
```

ไฟล์ที่เก็บ From-To Chart จะมีสกุลเป็น ".FTC" มี Format ในการเก็บดังนี้

3.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



TYPE เป็นเลข Integer มีค่าเท่ากับ 1 เพื่อแสดง
ว่าเป็น From-To Chart File

SIZE เป็นเลข Integer แสดงขนาดของ From-To
Chart

DATA เป็น Array of Integer มีขนาด SIZExSIZE

CHECKSUM เป็นผลรวมของค่าที่มีในไฟล์
เพื่อตรวจสอบว่ามี error หรือไม่

โปรแกรมในส่วนของ From-To chart นี้ อยู่ในไฟล์ชื่อ FROMTO.C และ FROMTO.H

4.3.1.4 Closeness Rates

เก็บค่าที่จะนำไปใช้แทนความสัมพันธ์ A,E,I,O,U และ X ให้เป็นค่าตัวเลขเพื่อใช้ในการคำนวณ
หาค่าความสัมพันธ์ มีโครงสร้างข้อมูลดังนี้

```
typedef struct {
    int A,E,I,O,U,X;
} CLOSENESS;
```

Closeness Rates Functions

void FillClosenessBox(CLOSENESS cr)

ทำหน้าที่ แสดงค่า Closeness Rate ทางจอภาพ

void InitClosenessRate(CLOSENESS *cr)

ทำหน้าที่ ใส่ค่าเริ่มต้นให้กับตัวแปร cr โดยให้ A=60,E=50,I=40,O=30,U=20 และ X=10

void EditClosenessRate(CLOSENESS *cr)

ทำหน้าที่ แก้ไขค่า Closeness Rate ของ ตัวแปร cr

int ClosenessRate(CLOSENESS cr, int c)

ทำหน้าที่ return ค่าตัวเลขที่เก็บอยู่ใน cr โดย input ตัวอักษร 'A','E','I','O','U' หรือ 'X'
ทางตัวแปร c

4.3.2 ส่วนของการวางผัง

ส่วนของการวางผังอยู่ใน PLANTLAY.C และ PLANTLAY.H โดยอยู่ในส่วนของฟังก์ชัน

RunLayout(...)

```
void RunLayout(REL *rel, FROMTO *ft, CLOSENESS cr, DEPT *dpt)
```

มีการผ่านข้อมูลเข้า คือ Relation Chart -rel, From-To Chart -ft, Closeness Rate -cr และ Department Data -dpt ต่อจากนั้นจะเป็นการกำหนดตัวแปร

ตัวแปรที่สำคัญมีดังนี้

```
SHEET sheet;
DPTCR *tcr; /* Total Closeness Rating */
DPTCR *rank; /* use in PCR to print relation */
int *remain; /* Remain of department which are not located in sheet */
int *blocks; /* Keep the number of blocks */
int *edge; /* keep department which adjacent to blank edge */
LIST *list;
CORDINATE *cg;
```

sheet เป็นตัวแปรประเภท struct SHEET เก็บรายละเอียดของแผนผังโรงงานซึ่งมีโครงสร้างดังนี้

```
typedef struct { /* Layout sheet */
    int length; /* length */
    int width; /* width */
    int gridsize; /* gridsize */
    unsigned long blocknum; /* number of sheet blocks */
    int *map; /* Point to sheet-map memory */
} SHEET;
```

length = ขนาดความยาวของ Layout Sheet

width = ขนาดความกว้างของ Layout Sheet

gridsize = ขนาดของ block สีเหลี่ยมจัตุรัส แต่ละ block มีหน่วยเป็น จุด (Pixel)

blocknum = จำนวน block ทั้งหมดที่ต้องใช้ คำนวนจากพื้นที่รวมของแผนกทั้งหมด กับ ค่า dpt.AreaPerBlock

map = pointer of integer ที่ชี้ไปยังแผนผัง ซึ่งแผนผังที่ใช้เก็บนั้น
เป็น Array of Integer ขนาด length x width

tcr และ rank เป็นตัวแปรประเภท struct DPTCR

tcr เก็บค่าความสัมพันธ์รวม TCR และ PCR

rank จะใช้ในการจัดเรียงความสัมพันธ์ที่แผนกหนึ่ง มีต่อแผนกที่ได้วางลงในผังไปแล้ว
ใช้ในช่วงของ การคำนวณ ค่า PCR

DPTCR มีโครงสร้างดังนี้

```
typedef struct {                                /* Total Closeness Rate table */
    int dptno;                                  /* department num. */
    int cr;                                     /* closeness rate */
} DPTCR;
```

dptno = หมายเลขของ department

cr = เก็บค่าความสัมพันธ์ TCR หรือ PCR

remain เป็น pointer ที่ชี้ไปยังหน่วยความจำ เป็น Array of integer มีขนาดเท่ากับจำนวน
แผนกเก็บค่า flag TRUE หรือ FALSE โดยที่ department ใดถูกวางลงในผังแล้ว
จะมีค่า TRUE

ถ้าไม่เช่นนั้น จะเป็น FALSE

block เป็น pointer ที่ชี้ไปยังหน่วยความจำ เป็น Array of integer มีขนาดเท่ากับจำนวน
แผนกเช่นกัน เก็บค่าจำนวน block ที่แต่ละแผนกจะต้องใช้ในการวางลงในผัง

edge เป็น pointer ที่ชี้ไปยังหน่วยความจำ เป็น Array of integer มีขนาดเท่ากับจำนวน
แผนก เก็บค่า flag TRUE หรือ FALSE โดยที่ department ใดถูกวางลงในผังแล้ว
และมีขอบด้านใดด้านหนึ่งเหลืออยู่ที่สามารถจะวางแผนกอื่นให้อยู่ติดกันได้
จะมีค่าเป็น TRUE ถ้าไม่เช่นนั้น จะเป็น FALSE

list เป็น pointer ที่ชี้ไปยังหน่วยความจำ มีโครงสร้างเป็นแบบ LIST โดยจะทำการจอง
เนื้อที่หน่วยความจำเป็นจำนวนเท่ากับจำนวนแผนก เพื่อเก็บ พิกัด x,y ของขอบเขต
แต่ละแผนกในรูปของ Linked-List

พิกัดจะเก็บในรูปของ node (Linked-List และ node สามารถหาอ่านได้จาก หนังสือ Data Structure) ดังนี้

```
struct node {                /* Node structure */
    COORDINATE c;            /* data of node */
    struct node *next;       /* next node pointer */
};

typedef struct node *NODEPTR; /* define type -> node pointer */
```

node ในที่นี้มีส่วนของ information เป็น COORDINATE หรือ พิกัด X,Y

```
typedef struct {
    int x;
    int y;
} COORDINATE;
```

แต่ละ node จะถูกเชื่อมไปยัง node ถัดไป โดย pointer next

ตัวที่ใช้ควบคุม node ต่าง ๆ ที่มี ก็คือ LIST ดังนี้

```
typedef struct {            /* LIST */
    int length;             /* length of list (or no. of nodes) */
    NODEPTR first,last;     /* first points at first node in list */
                                /* last points at last node in list */
} LIST;
```

length = จำนวนของ node หรือ ความยาวของ list

first = pointer ชี้ไปยัง node แรก

last = pointer ชี้ไปยัง node สุดท้าย

cg เป็น pointer ที่ชี้ไปยังหน่วยความจำ เป็น Array of integer มีขนาดเท่ากับจำนวน
แผนก

เก็บค่า COORDINATE หรือพิกัด ของจุดศูนย์กลางของแผนก (CG)

ต่อมาทำการจองหน่วยความจำ

```

size = dpt->size;

tcr = calloc(size,sizeof(DPTCR)); /* Total Closeness Rates */
rank = calloc(size,sizeof(DPTCR));
remain = calloc(size,sizeof(int)); /* Keep Un-located department flag*/
edge = calloc(size,sizeof(int));
blocks = calloc(size,sizeof(int));
cg = (CORDINATE *) calloc(size,sizeof(CORDINATE));
list = (LIST *) calloc(size,sizeof(LIST));

```

ขั้นตอนต่อไปทำการ set ค่าเริ่มต้น และ เปลี่ยนการแสดงผลเป็น graphic mode

```

sheet.length=0; /* Clear sheet variable */
sheet.width=0;
sheet.gridsize=10;
sheet.blocknum=0;
sheet.map=NULL;

for(i=0;i<size;i++) /* Initialize value */
{
    list[i].length=0;
    list[i].first =NULL;
    list[i].last =NULL;
    remain[i]=FALSE;
}

/***** use Graphic mode *****/
detectgraph(&graphdriver,&graphmode);
initgraph(&graphdriver,&graphmode,"");
clearviewport();
mouse_reset();
LayoutScreen();
ShowGrid(0,0,640,350,sheet.gridsize);

```

ขั้นตอนต่อไป ทำการหาจำนวน block ที่แต่ละแผนกต้องใช้ หาผลรวมพื้นที่ทั้งหมด และหาจำนวน block ทั้งหมด

```

/***** Blocks *****/
for(i=0;i<size;i++)      /* Find total area */
{
    sum= sum + dpt->pArea[i].dArea;
    blocks[i] = dpt->pArea[i].dArea / dpt->AreaPerBlock;
}
dpt->total = sum;          /* Set total area */
sheet.blocknum = dpt->total / dpt->AreaPerBlock; /* Calculate no. of blocks */

```

เมื่อคำนวณจำนวน block ที่ต้องใช้ทั้งหมดแล้ว ก็ทำการใส่ขนาดของผังโดยเรียกฟังก์ชัน InputSheetSize แล้วทำการจองหน่วยความจำ และ clear ค่าให้เป็น --1

```

message=InputSheetSize(dpt,&sheet); /* Input Sheet size */
....
/***** Alloc memory of map *****/
sheet.map = calloc(sheet.length*sheet.width,sizeof(int));
for(i=0;i<sheet.length*sheet.width;i++) /* Clear Map */
*(sheet.map+i) = -1;

```

จากนั้น เรียกใช้ mouse และคำนวณหาค่า TCR และทำการ เรียงลำดับค่า TCR

```

/***** TCR *****/
for(i=1;i<=size;i++)
{
    total=0;
    for(j=1;j<=size;j++)
    {
        if(i>j)          {a=i;b=j;}
        else             {a=j;b=i;}
        if(a != b)
        {
            value = GetRELValue(*rel,a,b);
            value = ClosenessRate(cr,value);
            total = total+value;
        }
    }
}

```

```

    }

    tcr[i-1].dptno = i-1;

    tcr[i-1].cr = total;

}

QsortTCR(tcr,0,size-1);

```

ให้ผู้ใช้เลือกแผนกที่มีค่า TCR มาก เมื่อเลือกแล้ว ทำการ set flag

```
remain[department]=TRUE;
```

จากนั้นทำการวางผัง และ รับค่าพิกัดบอกรอบเขตของแต่ละแผนก

```
/****** Input First dept. Rect *****/
```

```
check=InputPoly(&mx,&my,list,&sheet,blocks,cg,department,size);
```

ใส่ค่าลงในผัง ค่าที่ใส่ลงในผังนั้น เป็นค่าของ หมายเลขแผนกนั่นเอง

```
check=LocateToMap(list+department,sheet,department);
```

จากนั้น เป็นการหาจุด CG โดยมีการกำหนดขอบเขตของแผนก โดยใช้ฟังก์ชัน FindMin ,

FindMax

```
MIN=FindMin(list+department);
```

```
MAX=FindMax(list+department);
```

```
MIN.x = MIN.x/sheet.gridsize;
```

```
MIN.y = MIN.y/sheet.gridsize;
```

```
MAX.x = MAX.x/sheet.gridsize;
```

```
MAX.y = MAX.y/sheet.gridsize;
```

```
area=blocks[department];
```

```
*(cg+department) = FindCG(MIN.x,MAX.x,MIN.y,MAX.y,area,sheet,department);
```

ขั้นตอนต่อมาเป็นการเลือกแผนกอื่นวางลงในผัง จะทำการหาว่า มีแผนกใดที่วางลงไปแล้ว มีด้านที่ยังเหลือที่จะนำแผนกอื่นไปวางให้ชิดติดกันได้ โดยหาจุดที่ติดกับ -1 ฝั่งทั้งทางแนวแกน X และแนวแกน Y

```
/** Find edge **/
```

```
for(i=0;i<sheet.length;i++)
```

```
for(j=0;j<sheet.width-1;j++)
```

```
{ if( (*(sheet.map + sheet.length*j + i) != -1)
```

```
&& ( *(sheet.map + sheet.length*(j+1) + i) == -1) )
```

```

        edge[* (sheet.map + sheet.length*j + i)]=TRUE;
    if( ( *(sheet.map + sheet.length*j + i) == -1)
        && ( *(sheet.map + sheet.length*(j+1) + i) != -1) )
        edge[* (sheet.map + sheet.length*(j+1) + i)]=TRUE;
    }
    for(j=0;j<sheet.width;j++)
    for(i=0;i<sheet.length-1;i++)
    {
        if( ( *(sheet.map + sheet.length*j + i) != -1)
            && ( *(sheet.map + sheet.length*j +(i+1)) == -1) )
            edge[* (sheet.map + sheet.length*j + i)]=TRUE;

        if( ( *(sheet.map + sheet.length*j + i) == -1)
            && ( *(sheet.map + sheet.length*j +(i+1)) != -1) )
            edge[* (sheet.map + sheet.length*j +(i+1))]=TRUE;
    }
}

```

ต่อจากนั้น หาค่า PCR แล้วทำการเรียงลำดับ PCR ให้ผู้ใช้เลือก

```

/** PCR */
for(i=1;i<=size;i++)
{
    if(remain[i-1]==FALSE)
    {
        total=0;
        for(j=1;j<=size;j++)
        {
            if(edge[j-1]==TRUE)
            {
                if (i>j) {a=i;b=j;}
                else {a=j;b=i;}
                if(a != b)
                {
                    value = GetRELValue(*rel,a,b);
                    value = ClosenessRate(cr,value);
                    total = total+value;
                }
            }
        }
    }
}

```

```

    }
    tcr[i-1].dptno = i-1;
    tcr[i-1].cr    = total;
    }
}

QsortTCR(tcr,0,size-1);

```

จากนั้นก็เป็นการวางแผนลงในผัง คำนวณหาจุด CG ทำซ้ำจนวางครบทุกแผนกก็จะเป็น การหาประสิทธิภาพของผังโดยใช้ฟังก์ชัน Efficiency(...)

```
eff=Efficiency(ft,cg);
```

สุดท้ายจะเป็นการสร้าง Report File โดยการเรียกใช้ฟังก์ชัน OutputToTextFile(...)

```
OutputToTextFile(sheet,blocks,cg,dpt,rel,ft,eff);
```

Plant Layout Functions

void Plantlay(void)

ทำหน้าที่ เป็นส่วนเริ่มต้นเรียกใช้โปรแกรม

void MainDesktop(void)

ทำหน้าที่ Clear จอภาพ วาด BackGround

void PlantlayDesktop(void)

ทำหน้าที่ วาดกรอบด้านล่างของจอภาพ

int PlantlayMenu(REL *rel,FROMTO *ft,CLOSENESS *cr,DEPT *dept)

ทำหน้าที่ เป็นส่วนจัดการเมนูหลัก

int DeptMenu(void)

ทำหน้าที่ ส่วนจัดการเมนู Department Data

int RelMenu(void)

ทำหน้าที่ ส่วนจัดการเมนู Relation Chart

int FtMenu(FROMTO *ft)

ทำหน้าที่ ส่วนจัดการเมนู From-To Chart

int LayoutMenu(void)

ทำหน้าที่ ส่วนจัดการเมนู Layout

int ExitMenu(void)

ทำหน้าที่ ส่วนจัดการเมนู Exit

int Checking(REL rel, FROMTO ft, CLOSENESS cr, DEPT dpt)

ทำหน้าที่ ตรวจสอบความถูกต้องของข้อมูลก่อนเริ่มการวางผัง

void RunLayout(REL *rel, FROMTO *ft, CLOSENESS cr, DEPT *dpt)

ทำหน้าที่ เป็นส่วนหลักในการวางผัง

void FillStatusBox(REL rel, FROMTO ft, DEPT dpt)

ทำหน้าที่ แสดงสถานะของข้อมูล ในกรอบ Status

void LayoutScreen(void)

ทำหน้าที่ สร้างหน้าจอส่วนของการวางผัง Graphic Mode

void ShowGrid(int x1, int y1, int x2, int y2, int grid_size)

ทำหน้าที่ ตีตารางส่วนของการวางผัง Graphic Mode
เริ่มจาก (x1,y1) ถึง (x2,y2) โดยมีขนาดช่องห่าง grid_size

void read_mouse(int *mx, int *my, int deltax, int deltay, int grid_size)

ทำหน้าที่ อ่านค่าการเปลี่ยนแปลงของ mouse และทำการเลื่อน mouse cursor ไปยังตำแหน่งต่าง ๆ โดย (mx,my) = mouse cursor position
deltax = ค่าการเปลี่ยนแปลงในแนวแกน x
deltay = ค่าการเปลี่ยนแปลงในแนวแกน y

void ClearBox(void)

ทำหน้าที่ Clear ข้อความที่อยู่ใน message Box บนจอภาพ Graphic mode

int InputSheetSize(DEPT *dpt, SHEET *sheet)

ทำหน้าที่ Input ขนาดความยาวและความกว้างของผัง

void RedrawDepartment(LIST *lt, CORDINATE *cg, int size, int gridsizes, int department)

ทำหน้าที่ วาดขอบเขตของแผนกขึ้นมาใหม่
list = list ของพิกัดที่เป็นขอบเขตของ department
cg = จุด cg ของแผนกต่าง ๆ
size = จำนวนของแผนก
department = หมายเลขของ department

void drawp(LIST list)

ทำหน้าที่ วาดกรอบของแผ่นกตาม list ของพิกัดที่ input เข้ามา

void RubberPoly(int x1,int y1,int x2,int y2,int drawflag,void *topSave)

ทำหน้าที่ วาดเส้นตรงที่วาดไปแล้วจะไม่กระทบกระเทือน หรือ เปลี่ยนแปลง เส้นเดิมที่อยู่บนจอภาพ

int CalSelectPoly(LIST list)

ทำหน้าที่ หาจำนวน block ที่เกิดจากวงปิด list นั้น โดยใช้ทฤษฎีของ GREEN ในส่วนการ Integrate วงปิดเพื่อหาพื้นที่

int LocateToMap(LIST *list,SHEET sheet,int department)

ทำหน้าที่ ทำการใส่ค่าหมายเลขแผ่นกลงในผัง sheet->map

int InputPoly(int *mx,int *my,LIST *l,SHEET *sheet,int *blocks,CORDINATE *cg,int department,int size)

ทำหน้าที่ เป็นส่วนจัดการรับ ขอบเขตของแผ่นกเก็บลงใน list l

mx,my = mouse position

sheet = ผังที่ใช้วาง

blocks = จำนวน block ที่แต่ละแผ่นกต้องใช้

cg = จุด cg ของแผ่นกต่าง ๆ

size = จำนวนของแผ่นก

department = หมายเลขของ department

CORDINATE FindMin(LIST *list)

ทำหน้าที่ return ค่า x,y ที่น้อยที่สุดที่มีอยู่ใน list

CORDINATE FindMax(LIST *list)

ทำหน้าที่ return ค่า x,y ที่มากที่สุดที่มีอยู่ใน list

CORDINATE FindCG(int xmin,int xmax,int ymin,int ymax,int area,SHEET sheet,int department)

ทำหน้าที่ หาจุด CG ที่อยู่ในขอบเขต xmin,xmax,ymin,ymax

area = จำนวน block ที่ แผ่นก นั้นใช้

sheet = ผัง

department = หมายเลขของแผนก

int CalArea(int xmax,int xmin,int ymax,int ymin,SHEET sheet,int department)

ทำหน้าที่ คำนวณหาพื้นที่ที่อยู่ในขอบเขต xmin,xmax,ymin,ymax ใช้ในฟังก์ชัน FindCG

float Efficiency(FROMTO *ft,CORDINATE *cg)

ทำหน้าที่ หาประสิทธิภาพของผัง

void OutputToTextFile(SHEET sheet,int *blocks,CORDINATE *cg,DEPT *dpt,REL *rel,FROMTO *ft,float eff)

ทำหน้าที่ สร้าง Report file

4.4 องค์ประกอบของโปรแกรมและการสร้างโปรแกรม

โปรแกรมนี้นี้เขียนด้วยภาษา C โดยใช้ BorlandC version 2.0 หรือ version 3.1 ทำการ Compile และ Link โปรแกรมมีองค์ประกอบ 5 ส่วน คือ

1. PLANTLAY.C , PLANTLAY.H เป็นส่วนหลักของโปรแกรมวางผังโรงงาน
2. RELATION.C , RELATION.H เป็นส่วนของ Relation Chart
3. FROMTO.C , FROMTO.H เป็นส่วนของ From-To Chart
4. WIN.C , WIN.H เป็นส่วนของการแสดงผล และ รับ key
5. MOUSE.C , MOUSE.H เป็นส่วนของการใช้งาน mouse

โดยรายละเอียดของส่วนต่าง ๆ ได้กล่าวมาแล้วในข้างต้น ซึ่งมีส่วนของ WIN.C และ MOUSE.C จะไม่กล่าวถึง เนื่องจากไม่ใช่ส่วนสำคัญและสามารถอ่านได้จากหนังสือ การเขียนโปรแกรมภาษา C

*วิธีการ Compile ให้ทำการเปิด Project File และ set Code Generation ให้ เป็น Model LARGE และทำการ Link เป็น Execute file

*ก่อน Run Program ต้องทำการติดตั้ง mouse ก่อน

หมายเหตุ โปรแกรมที่ได้จัดทำขึ้นมานี้ ได้ทำบน

เครื่องรุ่น 386DX-33 Memory 4 MB

Display Color Monitor (VGA) and VGA Card

Microsoft Mouse (Two-Botton) and Microsoft Mouse Driver

4.4 ตัวอย่างการใช้โปรแกรม

หลังจากการที่ได้ทำการ Compile และ Link โปรแกรมส่วนต่าง ๆ เข้าด้วยกันแล้วจะได้ไฟล์ .EXE ที่สามารถ RUN ได้ ชื่อไฟล์ .EXE นี้เป็นชื่อเดียวกันกับชื่อไฟล์ .PRJ ที่สร้างขึ้น ในที่นี้ใช้ชื่อ Project File ว่า PLANTLAY.PRJ เมื่อทำการ Compile และ Link เป็น PLANTLAY.EXE

เริ่มต้นการใช้งานโปรแกรมจากแผ่น Disk

- 1. ใส่แผ่นใน Drive A :
- 2. พิมพ์ A: แล้วกดปุ่ม Enter
- 3. พิมพ์ DIR แล้วกดปุ่ม Enter จะปรากฏชื่อไฟล์ดังนี้

PLANTLAY.C , PLANTLAY.H , FROMTO.C , FROMTO.H RELATION.C , RELATION.H
MOUSE.C , MOUSE.H PLANTLAY.EXE EGAVGA.BGI MOUSE.COM

- 4. ทำการเรียกใช้ Mouse driver โดยพิมพ์

A:> mouse แล้วกดปุ่ม Enter

- 5. เรียกใช้โปรแกรม PLANT LAYOUT โดยพิมพ์

A:> PLANTLAY

- 6. ปรากฏเมนูหลักของโปรแกรม PLANTLAY ดังนี้

PLANT LAYOUT

Department data Relation Chart From-To Chart Layout Exit

STATUS		
Data	Filename	Size
Department data	Unloaded	n/a
Relation Chart	Unloaded	n/a
From-To Chart	Unloaded	n/a

Press ESC or F10 for menu

7. กด F10 หรือ ESC เพื่อทำการเลือกเมนู ในเมนูหลักจะประกอบด้วยเมนูย่อยคือ

- Department Data
- Relation Chart
- From-to Chart
- Layout
- และ Exit

PLANT LAYOUT

Department data Relation Chart From-To Chart Layout Exit

New Department data
Load Department data
Save Department data
Edit Department data

STATUS		
Data	Filename	Size
Department data	EX2.DPT	9
Relation Chart	Unloaded	n/a
From-To Chart	Unloaded	n/a

Press ESC or F10 for menu

โดยที่ใน Department Data Menu ยังประกอบด้วยคำสั่งย่อย

- NEW เป็นการสร้างข้อมูลที่เก็บค่าพื้นที่ของแต่ละแผนก
- LOAD เป็นการอ่านข้อมูล "*.DPT" จาก Disk
- SAVE เป็นการเก็บข้อมูล Department data ลงในไฟล์
- EDIT เป็นการแก้ไขค่า Department Data

Relation Chart Menu และ From-To Chart Menu ก็มีคำสั่ง New,Load,Save,Edit เช่นเดียวกัน

ส่วน Layout Menu มีคำสั่ง Run Plant Layout , Closeness Rates , Area per Block

Run Plant Layout เป็นคำสั่งเข้าสู่การวางแผนผัง ใน Graphic Mode

Closeness Rates เป็นคำสั่งเรียกดูค่า หรือ แก้ไขค่า Closeness Rates

Area per Block เป็นคำสั่งแก้ไขค่า Area per Block

8.ทำการใส่ค่าโดยการ New หรือ Load ค่าข้อมูลต่าง ๆ ที่ใช้ ซึ่งได้กล่าวมาแล้วข้างต้น หากเลือกคำสั่ง New โปรแกรมจะถาม จำนวนแผนก แต่ถ้าหากเลือก Load โปรแกรมจะถามชื่อไฟล์ โดยสกุลของไฟล์มีดังนี้

ไฟล์ข้อมูล Department Data มีสกุลเป็น " *.DPT "

ไฟล์ข้อมูล Relation Chart มีสกุลเป็น " *.REL "

ไฟล์ข้อมูล From-To Chart มีสกุลเป็น " *.FTC "

9. เลือกคำสั่ง New เพื่อใส่ค่าข้อมูลต่าง ๆ ดังนี้

เลือก New Department Data ใส่จำนวนแผนก = 9

PLANT LAYOUT				
Department data	Relation Chart	From-To Chart	Layout	Exit

NEW Department data

How many departments ? 9

STATUS		
Data	Filename	Size
Department data	EX2.DPT	9
Relation Chart	Unloaded	n/a
From-To Chart	Unloaded	n/a

Press ESC or F10 for menu

จากนั้นเข้าสู่ Edit Mode ดังรูป

PLANT LAYOUT

Department data Relation Chart From-To Chart Layout Exit

Department data

File Name : NONAME.DPT
Total Area= 0 No. of Depts.= 9

No.	NAME	AREA
1	DEPARTMENT #1	0
2	DEPARTMENT #2	0
3	DEPARTMENT #3	0
4	DEPARTMENT #4	0
5	DEPARTMENT #5	0
6	DEPARTMENT #6	0
7	DEPARTMENT #7	0
8	DEPARTMENT #8	0
9	DEPARTMENT #9	0

Press ESC or F10 for menu

การใส่ค่า Department Data ให้เลื่อนแถบสีไปยังแผนกที่ต้องการ โดยกดปุ่มลูกศรขึ้นลง จากนั้น กด Enter เพื่อต้องการแก้ไขค่าแผนกนั้น จะปรากฏแถบสีแดงใช้ในการเลือกที่จะแก้ไขข้อมูลตัวใด

แถบแดงใช้ลูกศร ซ้ายขวาทำการเลื่อน เมื่อต้องการใส่ค่า ให้กด Enter อีกครั้งหนึ่ง จะปรากฏ Cursor ให้ใส่ค่า

การออกจากแถบแดง ให้กด ESC และถ้าต้องการออกจาก Edit Mode ให้กด ESC อีกเช่นกัน

10.เลือก New Relation Chart ป้อนค่าดังนี้

จำนวนแผนก = 9

การเลื่อน Cell ให้ ปุ่มลูกศร ค่าที่ป้อนเข้าไปได้แก่ 'A','E','I','O', และ 'X'

กด ESC เมื่อทป้อนเสร็จ

REL [1] AND [2] : A

RELATION	2	3	4	5	6	7	8	9
1	A	O	E	X	X	X	U	X
2		A	X	X	X	X	O	X
3			X	O	I	X	A	X
4				X	X	U	X	E
5					X	O	X	X
6						U	X	O
7							X	I
8								A

Relation Chart | NONAME.REL | SIZE 9 * 9 | ESC:Exit

11. เลือกคำสั่ง New From-To Chart จำนวนแผนก = 9 และทำการป้อนค่าตามนี้

การเลือก Cell ใช้ปุ่มลูกศร กด ESC เพื่อออกจาก From-To Chart

FROM [1] TO [1] : UNUSED

FROM-TO	1	2	3	4	5	6	7	8	9
1	UNUSED	324	60	204	0	0	0	20	0
2	0	UNUSED	270	4	0	0	0	50	0
3	0	0	UNUSED	0	60	70	0	250	4
4	0	0	0	UNUSED	4	0	24	0	180
5	0	0	4	0	UNUSED	0	60	0	0
6	0	0	0	0	0	UNUSED	20	0	70
7	0	0	0	0	0	0	UNUSED	0	104
8	0	0	50	0	0	20	0	UNUSED	250
9	0	0	0	0	0	0	0	0	UNUSED

From-To Chart | NONAME.FTC | SIZE 9 * 9 | ESC:Exit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 53
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

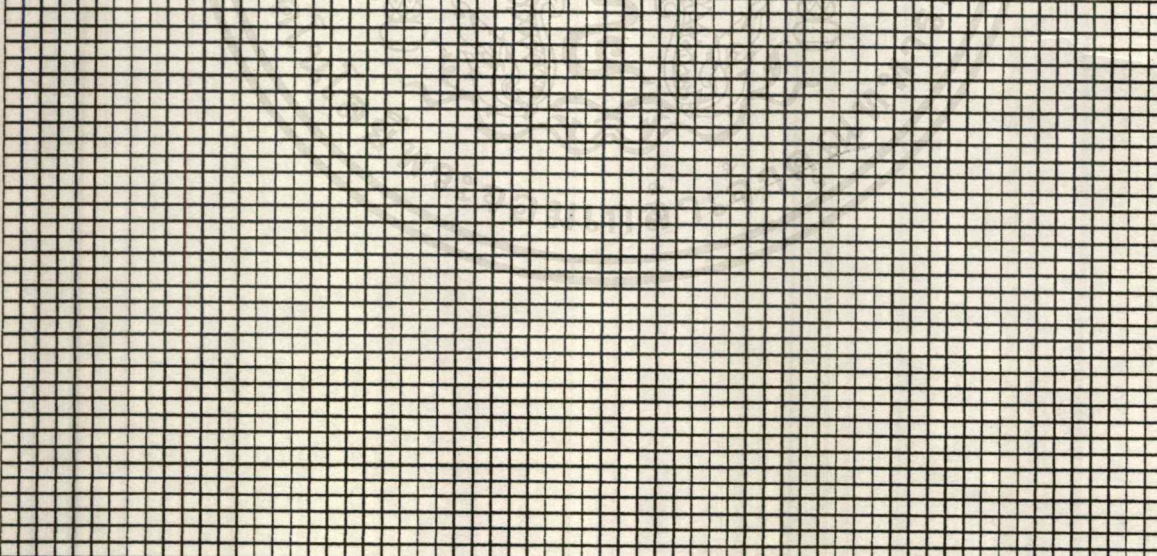
12. เลือก Closeness Rate ใน Layout Menu ใส่ค่าตามนี้

PLANT LAYOUT		
Department data Relation Chart From-To Chart Layout Exit		
Closeness Rates A (Absolutely Necessary) = 60 E (Essentially Important) = 50 I (Important) = 40 O (Ordinary) = 30 U (Unimportant) = 20 X (Not Desirable) = 10		
Department data	NONAME.DPT	9
Relation Chart	NONAME.REL	9
From-To Chart	NONAME.FTC	9

Press ESC or F10 for menu

13. เลือก Area per Block และใส่ค่า Area Per Block = 1

14. เข้าสู่การวางผัง โดยเลือก Run Plant Layout ที่อยู่ใน Layout Menu ปรากฏดังรูป



Total area = 900 , Area per block = 1 The number of blocks = 900

Please choose the length and width of the layout sheet :

1. Square (default value)
2. Length-to-width ratio of 2:1
3. Enter other Length-to-width ratio x:y
4. Enter size manually
5. Go back to change department area

Which configuration do you desire ?

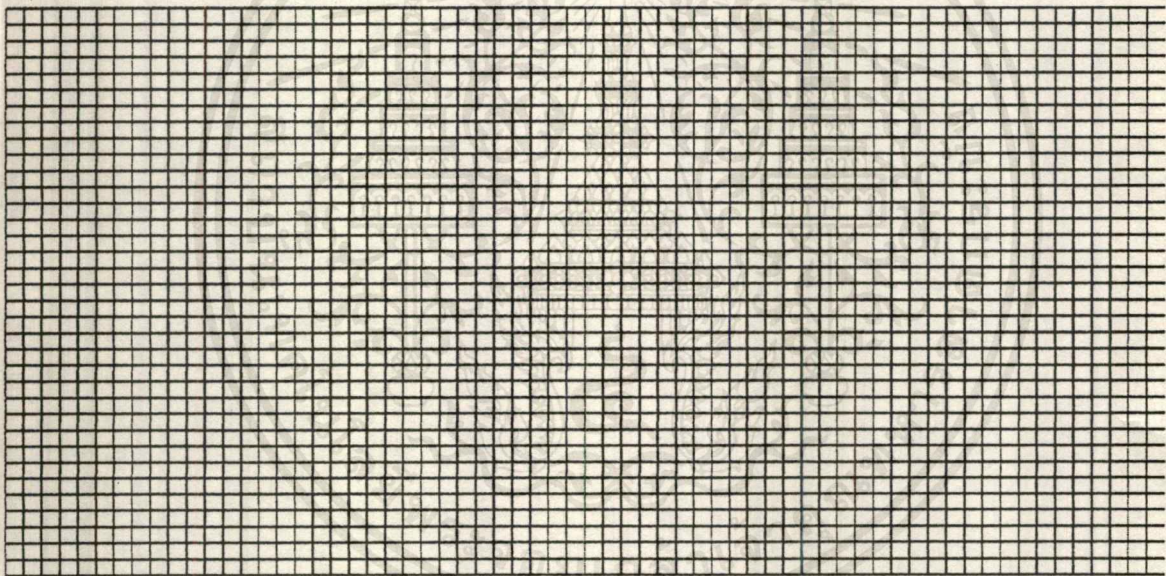
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ 54 ศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมจะให้ใส่ค่าความกว้าง ความยาวของผัง มีความกว้างยาวเป็นจำนวน Block ซึ่งผังที่ใหญ่ที่สุดมีขนาด 63 Blocks X 35 Blocks

มีตัวเลือกคือ

- 1. ให้ผังเป็น สีเหลี่ยมจัตุรัส
- 2. อัตราส่วน กว้าง : ยาว เท่ากับ 2:1
- 3. อัตราส่วน กว้าง : ยาว เท่ากับ x:y
- 4. ใส่ความยาวด้านใดด้านหนึ่ง อีกด้านหนึ่งโปรแกรมจะคำนวณให้
- 5. ย้อนกลับไป menu

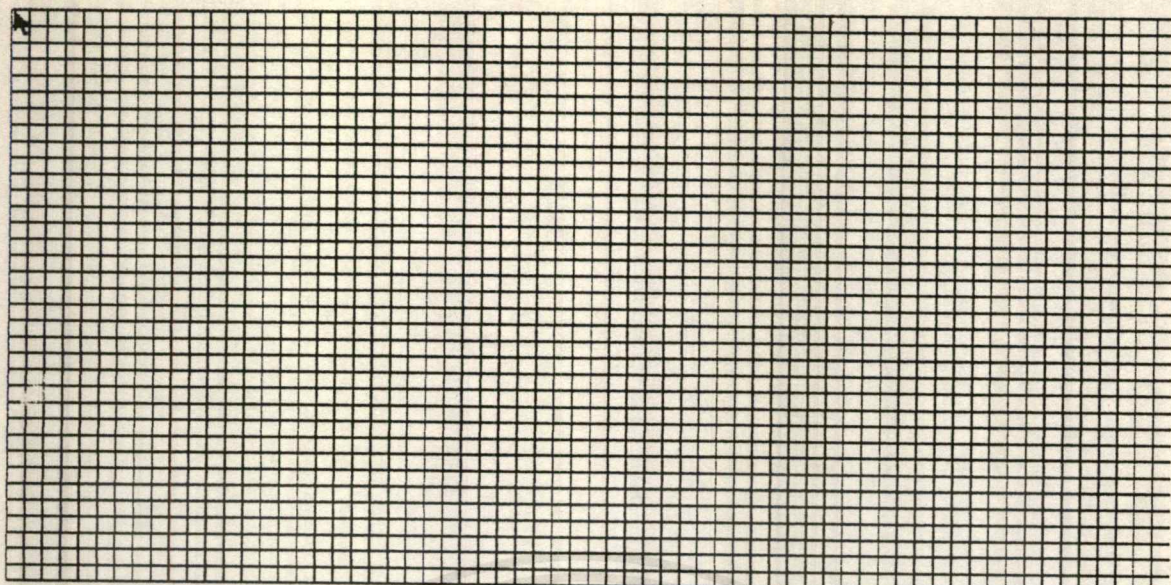
15. เลือกหมายเลขหนึ่ง คือให้ผังเป็น สีเหลี่ยมจัตุรัส จากนั้น โปรแกรมคำนวณหาค่า TCR และแสดงลำดับจากมากไปน้อย 5 อันดับ ให้ผู้ใช้เลือกแผนกหนึ่งแผนกใด นำไปวางลงในผัง



The following departments have the highest TCRs -		
Dept.	TCR	Area
3	250	100
9	220	100
8	210	100
1	200	100
2	200	100

Which department would you like to place first ?

เลือก แผนก 3 ลงไปวางก่อน

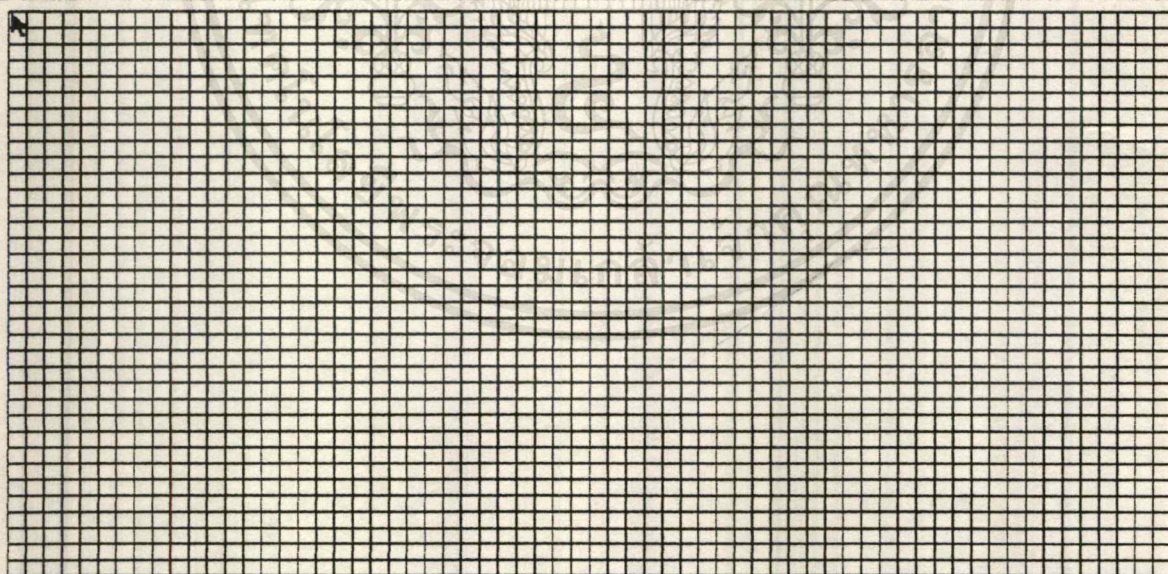


Locate department : 3
Area = 100 Blocks.

COMMAND> (P):place (D):delete (E):end

(C):Clear sheet

16.จากนั้นทำการ mark จุดที่เป็นมุมหัก เพื่อบอกขอบเขตของแผนที่วาง โดยจะมีการแสดง พิกัด หรือ node ที่ได้เก็บลงในโปรแกรม สิ่งสำคัญคือ ขอบเขตของแผนที่นั้นต้องเป็นวงปิดรูปเหลี่ยม และ ทิศทางการ mark จุด ควรจะมีทิศทางตามเข็มนาฬิกา ดังรูป



Locate department : 3
Area = 100 Blocks.

Position(0, 0)

Selected Node(s) = 5
Selected Blocks = 100

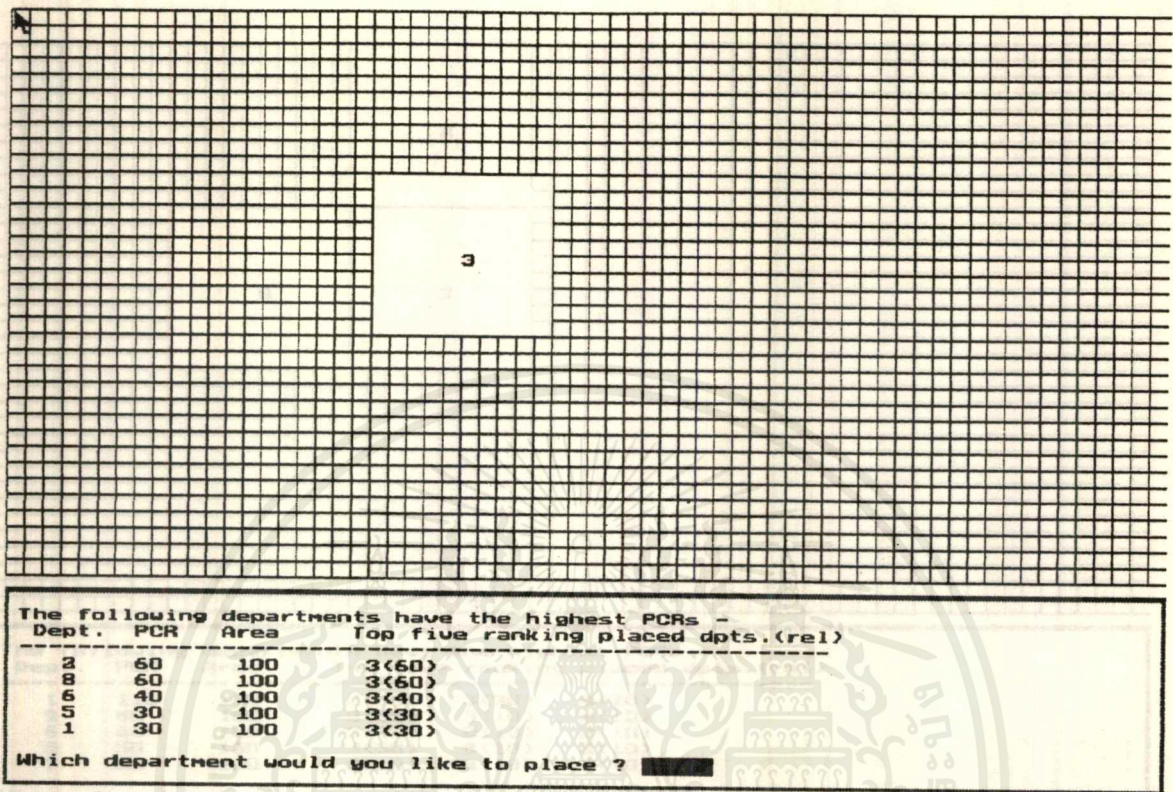
(20,10) (30,10) (30,20) (20,20) (20,10)

COMMAND> (P):place (D):delete (E):end

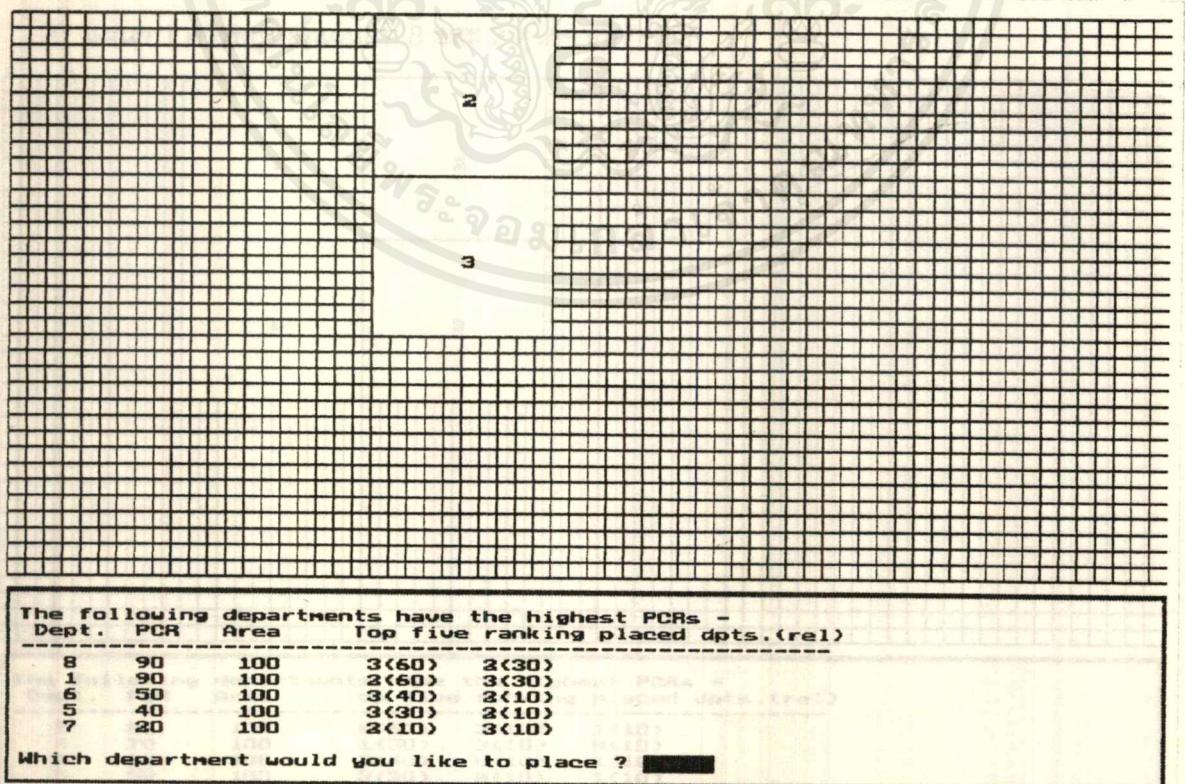
(C):Clear sheet

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ 56 วิชาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

17. หลังจากวางแผนก 3 ลงไปแล้ว โปรแกรมจะแสดง แผนกที่มีความสัมพันธ์กับ แผนกที่วางลงในผังแล้ว ซึ่งเรียกว่า ค่า PCR ดังที่ได้กล่าวไว้ข้างต้น ดังรูป



18. เลือกแผนก 2 ลงไปวางในผัง จากนั้น โปรแกรมจะแสดงค่า PCR สัมพันธ์กับ แผนก 2 และ 3



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเที57รศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[Layout]

123456789012345678901234567890
1DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
2DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
3DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
4DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
5DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
6DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
7DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
8DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
9DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
0DDDDDDDDDDAAAAAAAAAABBBBBBBBBB
1IIIIIIIIIIHHHHHHHHHCCCCCCCCC
2IIIIIIIIIIHHHHHHHHHCCCCCCCCC
3IIIIIIIIIIHHHHHHHHHCCCCCCCCC
4IIIIIIIIIIHHHHHHHHHCCCCCCCCC
5IIIIIIIIIIHHHHHHHHHCCCCCCCCC
6IIIIIIIIIIHHHHHHHHHCCCCCCCCC
7IIIIIIIIIIHHHHHHHHHCCCCCCCCC
8IIIIIIIIIIHHHHHHHHHCCCCCCCCC
9IIIIIIIIIIHHHHHHHHHCCCCCCCCC
0IIIIIIIIIIHHHHHHHHHCCCCCCCCC
1GGGGGGGGGGFFFFFFFEEEEEEEEEE
2GGGGGGGGGGFFFFFFFEEEEEEEEEE
3GGGGGGGGGGFFFFFFFEEEEEEEEEE
4GGGGGGGGGGFFFFFFFEEEEEEEEEE
5GGGGGGGGGGFFFFFFFEEEEEEEEEE
6GGGGGGGGGGFFFFFFFEEEEEEEEEE
7GGGGGGGGGGFFFFFFFEEEEEEEEEE
8GGGGGGGGGGFFFFFFFEEEEEEEEEE
9GGGGGGGGGGFFFFFFFEEEEEEEEEE
0GGGGGGGGGGFFFFFFFEEEEEEEEEE

[Efficiency]
Efficiency = 85.59

[Department] Filename : 'NONAME.DPT'
Area per block = 1
Total area = 900

Department	Name	Area	Blocks	CG
1=[A]	DEPARTMENT #1	100	100	(15, 5)
2=[B]	DEPARTMENT #2	100	100	(25, 5)
3=[C]	DEPARTMENT #3	100	100	(25,15)
4=[D]	DEPARTMENT #4	100	100	(5, 5)
5=[E]	DEPARTMENT #5	100	100	(25,25)
6=[F]	DEPARTMENT #6	100	100	(15,25)
7=[G]	DEPARTMENT #7	100	100	(5,25)
8=[H]	DEPARTMENT #8	100	100	(15,15)
9=[I]	DEPARTMENT #9	100	100	(5,15)

[Relation Chart] Filename : 'NONAME.REL'

	2	3	4	5	6	7	8	9
1	A	O	E	X	X	X	U	X
2		A	X	X	X	X	O	X
3			X	O	I	X	A	X
4				X	X	U	X	E
5					X	O	X	X

6 U X O
7 X I
8 A

[From-To Chart] Filename : 'NONAME.FTC'

	1	2	3	4	5	6	7	8	9
1		324	60	204	0	0	0	20	0
2	0		270	4	0	0	0	50	0
3	0	0		0	60	70	0	250	4
4	0	0	0		4	0	24	0	180
5	0	0	4	0		0	60	0	0
6	0	0	0	0	0		20	0	70
7	0	0	0	0	0	0		0	104
8	0	0	50	0	0	20	0		250
9	0	0	0	0	0	0	0	0	

[Distance among department]

	1	2	3	4	5	6	7	8	9
1	0	1	2	1	3	2	3	1	2
2	1	0	1	2	2	3	4	2	3
3	2	1	0	3	1	2	3	1	2
4	1	2	3	0	4	3	2	2	1
5	3	2	1	4	0	1	2	2	3
6	2	3	2	3	1	0	1	1	2
7	3	4	3	2	2	1	0	2	1
8	1	2	1	2	2	1	2	0	1
9	2	3	2	1	3	2	1	1	0

End of Layout text file.

สรุป

จากการศึกษาทฤษฎีในการออกแบบและการวางผังของโรงงานโดยที่กลุ่มผู้จัดทำมีวัตถุประสงค์เพื่อหาแนวทางที่จะประยุกต์นำเอาคอมพิวเตอร์เข้ามาช่วยใช้งาน ซึ่งจะก่อให้เกิดประโยชน์คือจะช่วยลดระยะเวลาในการออกแบบ และลดความยุ่งยากซับซ้อนในการออกแบบโดยเฉพาะการออกแบบโรงงานที่มีขนาดใหญ่ ๆ ซึ่งประกอบด้วยแผนกหรือหน่วยการผลิตจำนวนมาก

ภายหลังจากการศึกษาแล้วได้ทดลองทำโปรแกรมต้นแบบออกมาโดยอาศัยวิธีการ CORLAP ซึ่งเป็นวิธีการสำหรับวางผังโรงงานใหม่โดยจะอาศัยความสัมพันธ์ระหว่างแผนกต่าง ๆ เป็นแนวทางในการทำ โดยปรับปรุงให้การวางผังมีความยืดหยุ่นสูงขึ้นคือรูปร่างของแผนกหรือหน่วยการผลิตไม่ได้ถูกจำกัดเฉพาะรูปทรงสี่เหลี่ยมเท่านั้นแต่สามารถที่จะวางเป็นรูปเหลี่ยมอื่น ๆ ได้ และ ผู้ใช้งานจะเป็นผู้ทำการเลือกวางแผนกโดยมีโปรแกรมช่วยแนะนำว่าควรที่จะลงแผนกใด ซึ่งจะทำให้แบบของผังที่ออกมาเป็นผังที่ค่อนข้างตรงกับความต้องการและเป็นผังที่มีความเป็นไปได้มากขึ้น

สำหรับแนวทางในการพัฒนาโปรแกรมช่วยในการวางผังโรงงานต่อไปคือ พัฒนาในโปรแกรมให้สามารถเก็บ KNOWLEDGE BASE เพื่อช่วยตัดสินใจว่าแผนกใดสามารถวางอยู่ใกล้กันได้หรือไม่

ภาคผนวก



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



LISTING PROGRAM

PLANTLAY.H

PLANTLAY.C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*                               File Name : PLANTLAY.H                               */
/* ===== */

```

```

/* ***** Include Files ***** */

```

```

#include "relation.h"
#include "fromto.h"

```

```

/* ***** Define STRUCT ***** */

```

```

typedef struct {
    int x;
    int y;
} CORDINATE;

```

```

typedef struct {
    char dName[21];          /* Department name */
    int dArea;               /* Department area */
} AREA;

```

```

typedef struct {
    char filename[13];       /* Department filename */
    int size;                /* no. of Departments */
    int top,bottom;          /* Display range (top,bottom) */
    int cell;                /* Cell pointer */
    int AreaPerBlock;        /* Area per blocks */
    unsigned long total;     /* Total area */
    AREA far *pArea;         /* Pointer to department data memory */
} DEPT;

```

```

typedef struct {             /* Closeness rate value */
    int A,E,I,O,U,X;
} CLOSENESS;

```

```

typedef struct {             /* Layout sheet */
    int length;              /* length */
    int width;               /* width */
    int gridsize;            /* gridsize */
    unsigned long blocknum;   /* number of sheet blocks */
    int *map;                /* Point to sheet-map memory */
} SHEET;

```

```

typedef struct {             /* Total Closeness Rate table */
    int dptno;               /* department num. */
    int cr;                  /* closeness rate */
} DPTCR;

```

```

struct node {                /* Node structure */
    CORDINATE c;             /* data of node */
    struct node *next;       /* next node pointer */
};

```

```

typedef struct node *NODEPTR; /* define type -> node pointer */

```

```

typedef struct {             /* LIST */
    int length;              /* length of list (or no. of nodes) */
    NODEPTR first,last;      /* first points at first node in list */
                                /* last points at last node in list */
} LIST;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```

/* ***** Function Prototypes ***** */

```

```

/* Department data functions */
AREA *InitMemoryAREA(int items);
void FreeMemoryAREA(AREA far *pArea);

```

```

void ClearDeptValue(DEPT *dpt);
void FillNameArea(DEPT dpt);
void EditAreaPerBlock(DEPT *dpt);
void EnterNameArea(DEPT *dpt);
void EditNameArea(DEPT *dpt);
void NewDEPT(DEPT *dpt);
void LoadDEPT(DEPT *dpt);
void SaveDEPT(DEPT *dpt);
/*****/
/* Closeness Rate Functions */
void FillClosenessBox(CLOSENESS cr);
void InitClosenessRate(CLOSENESS *cr);
void EditClosenessRate(CLOSENESS *cr);
int ClosenessRate(CLOSENESS cr,int c);

/* Linked-List Functions */
NODEPTR getnode(void);
void freenode(NODEPTR p);
void freelist(LIST *list);
int insafter(NODEPTR p,CORDINATE n);
int delafter(NODEPTR p,CORDINATE *n);
int getinfo(LIST list,int node_no,CORDINATE *n);
NODEPTR getptr(LIST list,int node_no);

/*****/
/* Plantlay Functions */
void Plantlay(void);
void MainDesktop(void);
void PlantlayDesktop(void);
int PlantlayMenu(REL *rel,FROMTO *ft,CLOSENESS *cr,DEPT *dept);
int DeptMenu(void);
int RelMenu(void);
int FtMenu(FROMTO *ft);
int LayoutMenu(void);
int ExitMenu(void);
int Checking(REL rel,FROMTO ft,CLOSENESS cr,DEPT dpt);
void RunLayout(REL *rel,FROMTO *ft,CLOSENESS cr,DEPT *dpt);
void FillStatusBox(REL rel,FROMTO ft,DEPT dpt);
void LayoutScreen(void);
void ShowGrid(int xl,int yl,int x2,int y2,int grid_size);
void read_mouse(int *mx,int *my,int deltax,int deltay,int grid_size);
void ClearBox(void);
int InputSheetSize(DEPT *dpt,SHEET *sheet);
void ConvertToMap(CORDINATE n1,CORDINATE n2,CORDINATE *m1,CORDINATE *m2
,int gridsize);
void RedrawSheet(SHEET sheet,int department);
void RedrawDepartment(LIST *lt,CORDINATE *cg,int size,int gridsize
,int department);
void drawp(LIST list);
void RubberPoly(int xl,int yl,int x2,int y2,int drawflag,void *topSave);
int CalSelectPoly(LIST list);
int LocateToMap(LIST *list,SHEET sheet,int department);
int InputPoly(int *mx,int *my,LIST *l,SHEET *sheet,int *blocks,CORDINATE *cg
,int department,int size);
CORDINATE FindMin(LIST *list);
CORDINATE FindMax(LIST *list);
CORDINATE FindCG(int xmin,int xmax,int ymin,int ymax,int area,SHEET sheet
,int department);
int CalArea(int xmax,int xmin,int ymax,int ymin,SHEET sheet,int department);
float Efficiency(FROMTO *ft,CORDINATE *cg);
void OutputToTextFile(SHEET sheet,int *blocks,CORDINATE *cg,DEPT *dpt,REL *rel
,FROMTO *ft,float eff);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* =====
Program PLANTLAY.C : the main plant layout program

<OPEN PROJECT FILE>: PLANTLAY.PRJ
                    combine with files :  PLANTLAY.C
                                         FROMTO.C
                                         RELATION.C
                                         WIN.C
                                         MOUSE.C

*Use BorlandC Version 2.0 or 3.1

Set Option
<COMPILER> : /Code Generation/Model = Large
              /Code Generation/Option = Treat enums as ints
              /Code Generation/Assume SS equals DS = Default
              /Advance/Floating Point = Emulation
              /Advance/Instruction Set = 8088
              /Advance/Option = Generates underbars,
                               Debug info in OBJs,
                               Fast Floating Point.
              /Advance/Far Data Threshold = 32767

<Linker>      *only BC 3.1
              /Libraries/libraries = Graphics Library

<Directories>
              Add Project Directory in Include Directories.

<WRITTEN BY>   Manop Sriphiphattana  32.1238
                Thanarak Hongasuk    32.1113

===== */
/***** Include File *****/
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>

#include "plantlay.h"
#include "win.h"
#include "mouse.h"

/***** Define CONSTANT *****/
#define PLTXT          F_BLACK|B_LIGHTGRAY /* PLANTLAY normal text color */
#define PLHI          F_YELLOW|B_LIGHTGRAY /* PLANTLAY hilite text color */
#define MENUBARCOLOR  F_BLACK|B_CYAN       /* Menubar color */
#define CNRTXT        F_WHITE|B_BLUE       /* Closenessrate window text color */
#define CNRBOR        F_YELLOW|B_BLUE      /* Closenessrate window border color */
#define PLMENUTXT     F_WHITE|B_BLUE       /* PLANTLAY menu text color */
#define PLMENUBOR     F_YELLOW|B_BLUE      /* PLANTLAY menu border color */
#define PLMENUBAR     F_YELLOW|B_MAGENTA   /* PLANTLAY menubar color */
#define PLMENUHOT     F_LIGHTRED|B_BLUE    /* PLANTLAY Hotkey color */
#define DEPTTXT       F_WHITE|B_BLUE       /* Department window text color */
#define DEPTBOR       F_YELLOW|B_BLUE      /* Department window border color */
#define DEPTLINE      F_YELLOW|B_BLUE      /* Department window line color */

/***** Variable Declaration *****/
int PlantlayMenu_count=0; /* Menu counter */
int been_saved; /* Save flag : used by DrawPoly() */

/***** Function Prototypes *****/

```

```

void QsortTCR(DPTCR *tcr,int lb,int ub);    /* Quick Sort the
                                             TCR (Total Closeness Rate) */

/*-----*/
NODEPTR getnode(void)
    : Allocate memory 1 node (use in linked-list).
    Return node pointer NODEPTR.
/*-----*/

NODEPTR getnode(void)
{
    NODEPTR p;

    p= (NODEPTR) malloc(sizeof(struct node));
    return(p);
}

/*-----*/
void freenode(NODEPTR p)
    : Free node memory at (NODEPTR) p.
/*-----*/

void freenode(NODEPTR p)
{
    free(p);
}

/*-----*/
void freelist(LIST *list)
    : Free all nodes in list (LIST).
    Set list.length = 0
    set list.first,list.last = NULL.
/*-----*/

void freelist(LIST *list)
{
    NODEPTR q,p;
    int i;

    q=list->first;
    if (list->length!=0)
        for(i=0;i < list->length;i++)
        {
            p=q->next;
            freenode(q);
            q=p;
        }
    list->length=0;
    list->first=NULL;
    list->last=NULL;
}

/*-----*/
int insafter(NODEPTR p,COORDINATE n)
    : Insert new node with data (COORDINATE n) after node (NODEPTR) p.
    If p==NULL then return FALSE.
/*-----*/

int insafter(NODEPTR p,COORDINATE n)
{
    NODEPTR q;

    if (p==NULL) /* Check */
        printf("void insertion\n");
    return(FALSE);
}

```

```

q=getnode();                /* get new node */
q->c.x = n.x;                /* write data */
q->c.y = n.y;
q->next = p->next;           /* insert node after p */
p->next = q;

return(TRUE);
}

/*-----
int delafter(NODEPTR p,COORDINATE *n)
: Delete node after node p,
and return the data of deleted node in (COORDINATE) n.
If p==NULL or p->next ==NULL ,then return FALSE.
-----*/

int delafter(NODEPTR p,COORDINATE *n)    /* Return info */
{
    NODEPTR q;

    if ( (p==NULL) || (p->next==NULL) )    /* check */
    {
        printf("void deletion\n");
        return(FALSE);
    }
    q=p->next;                             /* point at next node */
    n->x = q->c.x;                          /* return data in n.x,n.y */
    n->y = q->c.y;
    p->next = q->next;                     /* cut node */
    freenode(q);                          /* delete the after node */
    return(TRUE);
}

/*-----
int getinfo(LIST list,int node_no,COORDINATE *n)
: Get data of node_no(th) node in list,
return data in (COORDINATE) n.
-----*/

int getinfo(LIST list,int node_no,COORDINATE *n)
{
    NODEPTR q;
    int i;

    if ((node_no>list.length)|| (list.length==0))    /* Check length */
        return(FALSE);

    q=list.first;
    for (i=0;i < node_no;i++)
        q=q->next;                                /* get data */
    n->x = q->c.x;                                /* return data in n.x,n.y */
    n->y = q->c.y;

    return(TRUE);
}

/*-----
NODEPTR getptr(LIST list,int node_no)
: Get pointer of node_no(th) node in list,
return node pointer.
-----*/
เอกลสารนี้เป็นเอกลสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น  เมื่ออนุญาตเห็นาเบไซประโยชน์ด้านการค้า
NODEPTR getptr(LIST list,int node_no)
{
    NODEPTR q;
    int i;

```

```

    if ((node_no>list.length)!!(list.length==0)) /* check */
        return(NULL);
    q=list.first;
    for (i=0;i < node_no;i++) /* get node pointer */
        q=q->next;
    return(q); /* return node pointer */
}

/*-----
void MainDesktop()
: Make the first screen of program layout.
-----*/

void MainDesktop()
{
    int i,j;

    window(1,1,80,25);
    textattr(PLTXT);
    clrscr();
    gotoxy(1,6);

    for(j=5;j<=24;j++)
    for(i=0;i<=1;i++)
        Fastputs(i*40+1,j,
            PLTXT,"");

    textattr(F_WHITE|B_RED);
    Box(1,1,80,3,DOUBLE); /* Draw Frame */
    gotoxy(2,2);
    cprintf("%-78s","PLANT LAYOUT");

    textattr(PLTXT);
    window(20,16,60,22);
    clrscr();
    window(1,1,80,25);
    Box(20,16,60,22,DOUBLE);
    Fastputs(22,16,PLTXT," STATUS ");
    HLine(18,20,60,HEADDOUBLE);
    ILine(38,16,22,HEADDOUBLE);
    ILine(53,16,22,HEADDOUBLE);
    Fastputs(27,17,PLTXT,"Data");
    Fastputs(42,17,PLTXT,"Filename");
    Fastputs(55,17,PLTXT,"Size");
    Fastputs(22,19,PLTXT,"Department data");
    Fastputs(22,20,PLTXT,"Relation Chart");
    Fastputs(22,21,PLTXT,"From-To Chart");
}

/*-----
void FillStatusBox(REL rel,FROMTO ft,DEPT dpt)
: Write status in the status box
-----*/

void FillStatusBox(REL rel,FROMTO ft,DEPT dpt)
{
    if((dpt.size<2) !! (dpt.size>30))
    {
        Fastputs(40,19,PLHI," Unloaded ");
        Fastputs(54,19,PLHI," n/a ");
    }
    else
    {
        Fastprintf(40,19,PLHI,"%12s",dpt.filename);
    }
}

```

```

        Fastprintf(54,19,PLHI," %2d ",dpt.size);
    }

    if((rel.size<2) || (rel.size>30))
    {
        Fastputs(40,20,PLHI," Unloaded ");
        Fastputs(54,20,PLHI," n/a ");
    }
    else
    {
        Fastprintf(40,20,PLHI,"%12s",rel.filename);
        Fastprintf(54,20,PLHI," %2d ",rel.size);
    }

    if((ft.size<2) || (ft.size>30))
    {
        Fastputs(40,21,PLHI," Unloaded ");
        Fastputs(54,21,PLHI," n/a ");
    }
    else
    {
        Fastprintf(40,21,PLHI,"%12s",ft.filename);
        Fastprintf(54,21,PLHI," %2d ",ft.size);
    }
}

/*-----
void PlantlayDesktop()
:   Make PLANTLAY Screen.
-----*/

void PlantlayDesktop()
{
    /****** Write Menu Head line******/
    textattr(MENUBARCOLOR);
    gotoxy(1,4);
    cprintf("%-80s",
    " Department data Relation Chart From-To Chart Layout Exit ");
}

/*-----
void Plantlay()
:   This is the head subroutine to call other routine.
-----*/

void Plantlay()
{
    REL rel;           /* Relation Chart variable */
    FROMTO ft;         /* From-To Chart variable */
    DEPT dept;         /* Department Data variable */
    CLOSENESS cr;      /* Closeness Rating variable */
                      /* See the structure in PLANTLAY.H */

    int key;
    int message;
    int loop;

    PlantlayDesktop(); /* Make first screen */
    ClearRELValue(&rel); /* Clear Relation Chart variable */
    ClearFTValue(&ft); /* Clear From-To Chart variable */
    InitClosenessRate(&cr); /* Set value of Closeness Rates */
    ClearDeptValue(&dept); /* Clear Department variable */

```

```

/* loop to get key pressed */
loop=TRUE;
do{
    FillStatusBox(rel,ft,dept);

    setcursor(CURSROFF);

    key = InputKey();
    switch(key)
    {
        /* Press ESC or F10 for menu */
        case kESC :
        case kF10 :      /* PLANTLAY Menu */
            message = PlantlayMenu(&rel,&ft,&cr,&dept);
            if (message == kALTX)
                loop=FALSE;      /* Exit? */
            break;

        /* Exit PLANTLAY Program */
        case kALTX:
            loop=FALSE;
            break;
    }
}while(loop==TRUE);

FreeMemoryREL(rel.pRel);      /* Free memory */
FreeMemoryFT(ft.pFromto);
FreeMemoryAREA(dept.pArea);
}

/*-----
int PlantlayMenu(REL *rel,FROMTO *ft,CLOSENESS *cr,DEPT *dpt)
: PLANTLAY main Menu.
-----*/

int PlantlayMenu(REL *rel,FROMTO *ft,CLOSENESS *cr,DEPT *dpt)
{
    int menunum = 5;      /* The number of sub-menu */

    int count;
    int loop;
    int message;
    int *scr;      /* Save All Screen pointer*/

    loop=TRUE;
    do{
        /* switches among the menus */
        switch(PlantlayMenu_count)
        {
            case 0: message = DeptMenu();
                    break;
            case 1: message = RelMenu();
                    break;
            case 2: message = FtMenu(ft);
                    break;
            case 3: message = LayoutMenu();
                    break;
            case 4: message = ExitMenu();
                    break;
        }

        /* Switch selected menu message */
        switch(message)
        {
            case kLEFT:      /* Left */
                PlantlayMenu_count--;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใด ๆ ก็ตาม หากมีข้อผิดพลาดหรือต้องการแจ้งแก้ไข กรุณาแจ้งไปยังเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        if (PlantlayMenu_count < 0)
            PlantlayMenu_count=menunum-1;
        break;
case kRIGHT:                                /* Right */
    PlantlayMenu_count ++;
    if (PlantlayMenu_count>menunum-1)
        PlantlayMenu_count = 0;
    break;
case kALTX:                                /* Alt-X = Exit */
    loop=FALSE;
    break;
case kESC :
    loop=FALSE;
    break;

default :
    loop=FALSE;

    switch(message)
    {

        /***** Department Menu *****/
        case 10: /* NEW Department Data */
            NewDEPT(dpt);
            EditNameArea(dpt);
            break;
        case 11: /* LOAD DEPARTMENT DATA */
            LoadDEPT(dpt);
            break;
        case 12: /* SAVE DEPARTMANT DATA */
            SaveDEPT(dpt);
            break;
        case 13: /* Edit Department Names and areas */
            EditNameArea(dpt);
            break;

        /***** Relation Chart Menu *****/
        case 20: /* New Relation Chart */
            scr=SaveAllScreen();
            RelationChart(rel,cNEWREL);
            /* call Relation Chart */
            RestoreAllScreen(scr);
            break;

        case 21: /* load Relation Chart */
            LoadREL(rel);
            break;

        case 22: /* Save Relation Chart */
            SaveREL(rel);
            break;

        case 23: /* Edit Relation Chart */
            scr = SaveAllScreen();
            RelationChart(rel,cEDITREL);
            RestoreAllScreen(scr);
            break;

        /***** From-To Chart Menu *****/
        case 30: /* New From-To Chart */
            scr=SaveAllScreen();
            /* Call From-To Chart */
            FromToChart(ft,cNEWFT);
            RestoreAllScreen(scr);
            break;

        case 31: /* Load From-To Chart */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในหน่วยงานเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้เผยแพร่ข้อมูลใดๆ ของเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        LoadFT(ft);
        break;

    case 32:/* Save From-To Chart */
        SaveFT(ft);
        break;

    case 33:/* Edit From-To Chart */
        scr = SaveAllScreen();
        /* Call From-To Chart */
        FromToChart(ft,cEDITFT);
        /* Restore Screen */
        RestoreAllScreen(scr);
        break;

        /***** Layout Menu *****/
    case 40:/* Run Layout */
        if(Checking(*rel,*ft,*cr,*dpt)==TRUE) /* check */
            RunLayout(rel,ft,*cr,dpt);
        break;

    case 41:/* Edit Closeness Rates */
        EditClosenessRate(cr);
        break;

    case 42:/* Edit Area per Block value */
        EditAreaPerBlock(dpt);
        break;
    }
    break;
} /* switch */
}while(loop==TRUE);

return(message);
}

/*-----
int DeptMenu()
: Department Data Menu
-----*/

int DeptMenu()
{
    WIN wFile = {2,5,27,10,"",DOUBLE};          /* window */

    char cFile[4][25] = {                          /* list of commands */
        " New  Department data ",
        " Load Department data ",
        " Save  Department data ",
        " Edit  Department data "
    };

    int commandnum=4;                                /* No. of Commands */
    int a,oldcount,count,message,loop;
    int *scr;                                          /* Screen Saver pointer */

    textattr(F_WHITE|B_BLACK);
    gotoxy(2,4);
    cputs(" Department data ");

    ShowWin(&wFile,PLMENUBOR,PLMENUTXT);          /* Show window */
    for(a=0;a<commandnum;a++)                        /* Print Commands */
    {
        gotoxy(2,a+1);
        PrintCommand(cFile[a],PLMENUBOR,PLMENUTXT);
    }
}

```

```

}

gotoxy(2,1);                                /* print menu-bar */
PrintCommandBar(cFile[0],PLMENUBAR,PLMENUTXT);

count=0;
loop=TRUE;
do{
    oldcount=count;                          /* keep old count */
    message = InputKey();                     /* Input keyboard */
    switch(message)                           /* switch message */
    {
        case KUP :
            count--;
            if( count < 0) count = commandnum-1;
            break;

        case KDOWN :
            count++;
            if( count > commandnum-1) count = 0;
            break;

        case KCR :
            loop=FALSE;
            switch(count)
            {
                /* NEW */
                case 0:
                    count = 0;
                    break;

                /* LOAD */
                case 1:
                    count = 1;
                    break;

                /* SAVE */
                case 2:
                    count = 2;
                    break;

                /* EDIT */
                case 3:
                    count = 3;
                    break;

            }

            count +=10;
            break;

        case KRIGHT:                          /* Right */
            loop = FALSE;
            count= KRIGHT;
            break;

        case KLEFT:                           /* Left */
            loop = FALSE;
            count= KLEFT;
            break;

        case KESC :                           /* Escape */
            loop = FALSE;
            count= KESC;
            break;

        case KALTX:                           /* Exit */
            loop= FALSE;
            count= KALTX;
            break;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีเมล: pl@pl.com ปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }

    if( loop == TRUE )          /* Print Menubar */
    {
        gotoxy(2,oldcount+1);
        PrintCommand(cFile[oldcount],PLMENUHOT,PLMENUTXT);
        gotoxy(2,count+1);
        PrintCommandBar(cFile[count],PLMENUBAR,PLMENUTXT);
    }

    }while( loop == TRUE);

    EraseWin(&wFile);          /* Close Menu */

    textattr(MENUBARCOLOR);
    gotoxy(2,4);
    cputs(" Department data ");

    return(count);
}

/*-----
int RelMenu()
    : Relation Chart Menu
-----*/

int RelMenu()
{
    WIN wFile = {20,5,44,10,"",DOUBLE};          /* window */

    char cFile[4][25] = {                          /* list of commands */
        " New Relation Chart ",
        " Load Relation Chart ",
        " Save Relation Chart ",
        " Edit Relation Chart "
    };

    int commandnum=4;          /* No. of Commands */
    int a,oldcount,count,message,loop;
    int *scr;                  /* Screen Saver pointer */

    textattr(F_WHITE|B_BLACK);
    gotoxy(20,4);
    cputs(" Relation Chart ");

    ShowWin(&wFile,PLMENUBOR,PLMENUTXT);          /* Show window */
    for(a=0;a<commandnum;a++)                      /* Print Commands */
    {
        gotoxy(2,a+1);
        PrintCommand(cFile[a],PLMENUHOT,PLMENUTXT);
    }

    gotoxy(2,1);          /* print menu-bar */
    PrintCommandBar(cFile[0],PLMENUBAR,PLMENUTXT);

    count=0;
    loop=TRUE;
    do{
        oldcount=count;          /* keep old count */
        message = InputKey();      /* Input keyboard */
        switch(message)            /* switch message */
        {
            case KUP:
                count--;
                if( count < 0 ) count = commandnum-1;
                break;

```

```

case kDOWN :
    count++;
    if( count > commandnum-1) count = 0;
    break;

case kCR :
    loop=FALSE;
    switch(count)
    {
        /* NEW RELATION CHART */
        case 0:
            count = 0;
            break;

        /* LOAD RELATION CHART */
        case 1:
            count = 1;
            break;/*LoadREL();*/

        /* SAVE RELATION CHART */
        case 2:
            count = 2;
            break;

        /* EDIT RELATION CHART */
        case 3:
            count = 3;
            break;

    }

    count +=20;
    break;

case kRIGHT: /* Right */
    loop = FALSE;
    count= kRIGHT;
    break;

case kLEFT: /* Left */
    loop = FALSE;
    count = kLEFT;
    break;

case kESC : /* Escape */
    loop = FALSE;
    count= kESC;
    break;

case kALTX: /* EXIT */
    loop = FALSE;
    count= kALTX;
    break;

}

if( loop == TRUE ) /* Print Menubar */
{
    gotoxy(2,oldcount+1);
    PrintCommand(cFile[oldcount],PLMENUHOT,PLMENUTXT);
    gotoxy(2,count+1);
    PrintCommandBar(cFile[count],PLMENUBAR,PLMENUTXT);
}

}

while( loop == TRUE);

EraseWin(&wFile); /* Close Menu */

textattr(MENUBARCOLOR);

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

textattr(MENUBARCOLOR);

```

gotoxy(20,4);
cputs(" Relation Chart ");

return(count);
}

/*-----*/
int FtMenu(FROMTO *ft)
: From-To Chart Menu
/*-----*/

int FtMenu()
{
    WIN wFile = {37,5,60,10,"",DOUBLE};          /* window */

    char cFile[4][25] = {                          /* list of commands */
        " New From-to Chart ",
        " Load From-to Chart ",
        " Save From-To Chart ",
        " Edit From-To Chart "
    };

    int commandnum=4;                               /* No. of Commands */
    int a,oldcount,count,message,loop;
    int *scr;                                       /* Screen Saver pointer */

    textattr(F_WHITE;B_BLACK);
    gotoxy(37,4);
    cputs(" From-To Chart ");

    ShowWin(&wFile,PLMENUBOR,PLMENUTXT);          /* Show window */
    for(a=0;a<commandnum;a++)                     /* Print Commands */
    {
        gotoxy(2,a+1);
        PrintCommand(cFile[a],PLMENUHOT,PLMENUTXT);
    }

    gotoxy(2,1);                                  /* print menu-bar */
    PrintCommandBar(cFile[0],PLMENUBAR,PLMENUTXT);

    count=0;
    loop=TRUE;
    do{
        oldcount=count;                          /* keep old count */
        message = InputKey();                     /* Input keyboard */
        switch(message)                           /* switch message */
        {
            case KUP :
                count--;
                if( count < 0) count = commandnum-1;
                break;

            case KDOWN :
                count++;
                if( count > commandnum-1) count = 0;
                break;

            case KCR :
                loop=FALSE;
                switch(count)
                {
                    case 0:
                        /* NEW FROM-TO CHART */
                        count = 0;
                        break;
                }
            }
        }
    } while(loop);
}

```

เอกสารนี้เป็นเอกสารที่หน่วยงานนี้จัดทำขึ้นเพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* LOAD FROM-TO CHART */
case 1:
    count = 1;
    break;

/* SAVE FROM-TO CHART */
case 2:
    count = 2;
    break;
/* EDIT FROM-TO CHART */
case 3:
    count = 3;
    break;
}
count +=30;
break;

case kRIGHT:                                /* Right */
    loop = FALSE;
    count= kRIGHT;
    break;

case kLEFT:
    loop = FALSE;                            /* Left */
    count = kLEFT;
    break;

case kESC :                                /* Escape */
    loop = FALSE;
    count= kESC;
    break;

case kALTX:                                /* Exit */
    loop = FALSE;
    count= kALTX;
    break;

}

if( loop == TRUE )                          /* Print Menubar */
{
    gotoxy(2,oldcount+1);
    PrintCommand(cFile[oldcount],PLMENUHOT,PLMENUTXT);
    gotoxy(2,count+1);
    PrintCommandBar(cFile[count],PLMENUBAR,PLMENUTXT);
}

}while( loop == TRUE);

EraseWin(&wFile);                            /* Close Menu */

textattr(MENUBARCOLOR);
gotoxy(37,4);
cputs(" From-To Chart ");

return(count);
}

```

/*-----
 int LayoutMenu()
 เอกสารนี้เป็นลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
 -----*/

```

int LayoutMenu()
{
    WIN wFile = {53,5,74,9,"",DOUBLE};        /* window */

```

```

char cFile[3][25] = {                                     /* list of commands */
    " Run Plant Layout ",
    " Closeness Rates ",
    " Area per Block "
};

```

```

int commandnum=3;                                         /* No. of Commands */
int a,oldcount,count,message,loop;
int *scr;                                                 /* Screen Saver pointer */

```

```

textattr(F_WHITE|B_BLACK);
gotoxy(53,4);
cputs(" Layout ");
ShowWin(&wFile,PLMENUMBOR,PLMENUTXT); /* Show window */
for(a=0;a<commandnum;a++) /* Print Commands */
{
    gotoxy(2,a+1);
    PrintCommand(cFile[a],PLMENUHOT,PLMENUTXT);
}
gotoxy(2,1); /* print menu-bar */
PrintCommandBar(cFile[0],PLMENUBAR,PLMENUTXT);

```

```

count=0;
loop=TRUE;
do{
    oldcount=count; /* keep old count */
    message = InputKey(); /* Input keyboard */
    switch(message) /* switch message */
    {
        case kUP :
            count--;
            if( count < 0) count = commandnum-1;
            break;

        case kDOWN :
            count++;
            if( count > commandnum-1) count = 0;
            break;

        case kCR :
            loop=FALSE;
            switch(count)
            {
                case 0: /* Run Layout */
                    count = 0;
                    break;
                case 1: /* Edit Closeness Rates */
                    count = 1;
                    break;
                case 2: /* Edit Area Per Block */
                    count = 2;
                    break;
            }
            count +=40;
            break;

        case kRIGHT: /* Right */
            loop = FALSE;
            count= kRIGHT;
            break;

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น count = kLEFT; ลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
break;

```

case kESC :

```

```

        loop = FALSE;
        count= kESC;
        break;

    case kALTX:
        loop = FALSE;
        count= kALTX;
        break;

    }

    if( loop == TRUE )        /* Print Menubar */
    {
        gotoxy(2,oldcount+1);
        PrintCommand(cFile[oldcount],PLMENUHOT,PLMENUTXT);
        gotoxy(2,count+1);
        PrintCommandBar(cFile[count],PLMENUBAR,PLMENUTXT);
    }

    }while( loop == TRUE);

    EraseWin(&wFile);          /* Close Menu */
    textattr(MENUBARCOLOR);
    gotoxy(53,4);
    cputs(" Layout ");
    return(count);
}

/*-----
int ExitMenu()
: Exit Menu.
-----*/

int ExitMenu()
{
    int message,loop;

    do{
        textattr(F_WHITE|B_BLACK);
        gotoxy(62,4);
        cputs(" Exit ");

        loop=TRUE;
        message = InputKey();

        textattr(MENUBARCOLOR);
        gotoxy(62,4);
        cputs(" Exit ");

        switch(message)
        {
            case kCR : loop=FALSE;message=kALTX;break;
            case kLEFT : loop=FALSE;message=kLEFT;break;
            case kRIGHT: loop=FALSE;message=kRIGHT;break;
            case kESC : loop=FALSE;message=kESC;break;
            default : loop=TRUE;break;
        }
    }while(loop==TRUE);

    return(message);
}
/*-----

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void InitClosenessRate(CLOSENESS *cr)
:   Set value to Closeness Rates.
```

```
*/
```

```
void InitClosenessRate(CLOSENESS *cr)
{
    cr->A = 60;
    cr->E = 50;
    cr->I = 40;
    cr->O = 30;
    cr->U = 20;
    cr->X = 10;
}
```

```
/*-----
void FillClosenessBox(CLOSENESS cr)
:   Fill value into Closeness Rates Box.
```

```
*/
```

```
void FillClosenessBox(CLOSENESS cr)
{
    gotoxy(32,1);printf("%-5d",cr.A);
    gotoxy(32,2);printf("%-5d",cr.E);
    gotoxy(32,3);printf("%-5d",cr.I);
    gotoxy(32,4);printf("%-5d",cr.O);
    gotoxy(32,5);printf("%-5d",cr.U);
    gotoxy(32,6);printf("%-5d",cr.X);
}
```

```
/*-----
void EditClosenessRate(CLOSENESS *cr)
:   Edit Closeness Rates value
```

```
*/
```

```
void EditClosenessRate(CLOSENESS *cr)
{
    WIN w = {20,10,60,17," Closeness Rates ",DOUBLE};
    int value;
    int count;
    int message;
    int loop;
```

```
ShowWin(&w,CNRBOR,CNRTXT);
gotoxy(3,1);cputs("A (Absolutely Necessary) = ");
gotoxy(3,2);cputs("E (Essentially Important) = ");
gotoxy(3,3);cputs("I (Important) = ");
gotoxy(3,4);cputs("O (Ordinary) = ");
gotoxy(3,5);cputs("U (Unimportant) = ");
gotoxy(3,6);cputs("X (Not Desirable) = ");
```

```
count = 0;
loop = TRUE;
do{
```

```
    textattr(CNRTXT);
    FillClosenessBox(*cr);           /* Fill all value */
```

เอกสารนี้เป็นเอกสารเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆก็ตาม ผู้ใช้ต้องแจ้งเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    textattr(F_BLACK|B_LIGHTGRAY); /* print Edit-bar */
    gotoxy(32,1+count);
    switch(count)
    {
        case 0: value = cr->A ;break;
```

```

        case 1: value = cr->E ;break;
        case 2: value = cr->I ;break;
        case 3: value = cr->O ;break;
        case 4: value = cr->U ;break;
        case 5: value = cr->X ;break;
    }
    cprintf("%-5d",value);

    message = InputKey();          /* Input key */
    switch(message)
    {
        case kUP : count--;
            if (count <0) count = 0;
            break;
        case kDOWN: count ++;
            if (count >5) count = 5;
            break;
        case kCR : setcursor(CURSORON);
            gotoxy(32,1+count);
            cputs(" ");
            gotoxy(32,1+count);

            value = GetInt(); /* input value */
            switch(count)
            {
                case 0: cr->A = value;break;
                case 1: cr->E = value;break;
                case 2: cr->I = value;break;
                case 3: cr->O = value;break;
                case 4: cr->U = value;break;
                case 5: cr->X = value;break;
            }
            break;
        case kESC : loop=FALSE;
            break;
    }
    }while(loop==TRUE);

    EraseWin(&w);
}

/*-----
int Checking(REL rel,FROMTO ft,CLOSENESS cr,DEPT dpt)
:
-----*/

int Checking(REL rel,FROMTO ft,CLOSENESS cr,DEPT dpt)
{
    int a,b;
    long sum;
    WIN chk={15,12,65,16," Checking ",DOUBLE};

    ShowWin(&chk,rMSGWINBORDER,rMSGWINTXT);

    Beep();

    if ((cr.A < cr.E)!!(cr.E < cr.O)!!(cr.I<cr.O)!!(cr.O<cr.U)!!(cr.U<cr.X))
    {
        gotoxy(4,2);
        cputs("Closeness Rates Error! Rule: A>E>I>O>U>X");
        getch();
        EraseWin(&chk);
        return(FALSE);
    }
}

```

```

if ((rel.size != dpt.size) || (rel.size != ft.size) || (dpt.size != ft.size))
{
    gotoxy(2,2);
    cputs("Size Mismatch ! Check the size of input data.");
    getch();
    EraseWin(&chk);
    return(FALSE);
}

```

```

if ((rel.size < 2) || (rel.size > 30))
{
    gotoxy(7,2);
    cputs("Relation Chart hasn't been loaded.");
    getch();
    EraseWin(&chk);
    return(FALSE);
}

```

```

if((ft.size < 2) || (ft.size > 30))
{
    gotoxy(8,2);
    cputs("From-To Chart hasn't been loaded.");
    getch();
    EraseWin(&chk);
    return(FALSE);
}

```

```

if((dpt.size < 2) || (dpt.size > 30))
{
    gotoxy(7,2);
    cputs("Department Data hasn't been loaded.");
    getch();
    EraseWin(&chk);
    return(FALSE);
}

```

```

for(a=0;a<dpt.size;a++)
{
    if(dpt.pArea[a].dArea <= 0)
    {
        gotoxy(1,2);
        cputs("Department Area Error! Negative or zero value.");
        window(1,1,80,25);
        getch();
        EraseWin(&chk);
        return(FALSE);
    }
}

```

```

sum=0;
for(a=1;a<=ft.size;a++)
for(b=1;b<=ft.size;b++)
    sum += GetFTValue(ft,a,b);
if(sum <= 0)
{
    gotoxy(5,2);
    cputs("From-To Chart Error! All data are zero.");
    window(1,1,80,25);
    getch();
    EraseWin(&chk);
    return(FALSE);
}

```

```

gotoxy(16,2);
cputs("Checking Passed !");
getch();
EraseWin(&chk);
return(TRUE);
}

```

```

/*-----
int ClosenessRate(CLOSENESS cr,int c)
: return Closeness rates (cr) value
(c) is input character = A,E,I,O,U,X
-----*/

int ClosenessRate(CLOSENESS cr,int c)
{
    int value;

    switch(c)
    {
        case 'A' : value = cr.A;break;
        case 'E' : value = cr.E;break;
        case 'I' : value = cr.I;break;
        case 'O' : value = cr.O;break;
        case 'U' : value = cr.U;break;
        case 'X' : value = cr.X;break;
        default : value = 0; break;
    }
    return(value);
}

/*-----
void LayoutScreen()
: draw message box in graphic mode .
-----*/

void LayoutScreen()
{
    setlinestyle(SOLID_LINE,0,THICK_WIDTH);
    setcolor(RED);
    rectangle(0,359,639,479);
}

/*-----
AREA *InitMemoryAREA(int items)
: Allocate memory for Department Data variable
'items' is the number of departments.
-----*/

AREA *InitMemoryAREA(int items)
{
    AREA far *address; /* address of memory allocation */

    char str[21],buff[5];
    int c;

    if (items<1) return(NULL); /* allocation */
    address = (AREA far *) farcalloc(items,sizeof(AREA));

    if(address==NULL) /* Error */
    {
        gotoxy(5,2);
        cputs("InitMemoryAREA Error : Not enough memory !");
        getch();
    }
    else
    {
        /* OK */
        /* Initial values */
        for(c=0;c<items;c++)
        {
            strcpy(str,"DEPARTMENT #");
            itoa(c+1,buff,10);
            strcat(str,buff);
            strcpy((address+c)->dName,str);
            (address+c)->dArea= 0;

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ หากมีการนำเอกสารนี้ไปเผยแพร่โดยไม่ได้รับอนุญาตจะถือว่าผิดกฎหมายและต้องแจ้งถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }
    return(address);
}

/*-----
void FreeMemoryAREA(AREA far *pArea)
:   Free memory of Department Data.
-----*/

void FreeMemoryAREA(AREA far *pArea)
{
    farfree(pArea);
}

/*-----
void ClearDeptValue(DEPT *dpt)
:   Clear the value of Department Data variable 'dpt'.
-----*/

void ClearDeptValue(DEPT *dpt)
{
    strcpy(dpt->filename,"NONAME.DPT");
    dpt->size=0;
    dpt->top =0;
    dpt->bottom = 0;
    dpt->cell=0;
    dpt->AreaPerBlock = 5;
    dpt->total = 0;
    dpt->pArea = NULL;
}

/*-----
void FillNameArea(DEPT dpt)
:   Fill value into Department Data Box.
-----*/

void FillNameArea(DEPT dpt)
{
    int a;
    unsigned long sum=0;

    for(a=0;a<dpt.size;a++)                /* Find total area */
        sum= sum + dpt.pArea[a].dArea;

    for(a=dpt.top;a<=dpt.bottom;a++)        /* Print Name and Area */
    {
        textattr(DEPTTXT);
        gotoxy(2,6+a-dpt.top);
        cprintf(" %2d    %-20s    %-5d ",
            a+1,dpt.pArea[a].dName,dpt.pArea[a].dArea);
        textattr(DEPTLINE);
        gotoxy(6,6+a-dpt.top); putchar(' ');
        gotoxy(31,6+a-dpt.top);putchar(' ');
    }

    if( dpt.bottom <10)
    {
        for(a=dpt.bottom+1;a<=10;a++)
            gotoxy(2,6+a-dpt.top);
        cprintf("ยังมีอีก %-20s และต้องอ", "");
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ กรุณาติดต่อ %20s และต้องอ";

```

/* Fill Filename and Total Area */
dpt.total = sum;
textattr(DEPTTXT);
gotoxy(13,1);
cprintf("%12s",dpt.filename);
gotoxy(13,2);
cprintf("%8lu",dpt.total);
gotoxy(37,2);
cprintf("%2d",dpt.size);

```

```

}

```

```

/*-----
void EditNameArea(DEPT *dpt)
:   Edit Department Data ... Names and Areas
-----*/

```

```

void EditNameArea(DEPT *dpt)
{

```

```

    int value;
    int count;
    int message;
    int loop;
    WIN wDpt = { 20,6,60,23," Department data ",DOUBLE};
    WIN msg = { 15,12,65,16," MESSAGE ",DOUBLE};

```

```

    if((dpt->size <2) || (dpt->size >30))
    {
        Beep();
        ShowWin(&msg,RMSGWINBORDER,RMSGWINTXT);
        gotoxy(5,2);
        cputs(" Department Data hasn't been loaded !");
        getch();
        EraseWin(&msg);
        return;
    }

```

```

    ShowWin(&wDpt,DEPTBOR,DEPTTXT);
    textattr(DEPTLINE);
    gotoxy(2,1);
    cputs("File Name : ");
    gotoxy(2,2);
    cputs("Total Area=          No. of Depts.=");
    gotoxy(1,3);
    cputs("_____");
    gotoxy(2,4);
    cputs("No.");
    gotoxy(17,4);
    cputs("NAME");
    gotoxy(33,4);
    cputs("AREA");
    gotoxy(1,5);
    cputs("_____");
    ILine(6,3,12,SINGLE);
    ILine(31,3,12,SINGLE);

```

```

    count = 0;
    loop = TRUE;
    do{
        setcursor(CURSROFF);
        textattr(PLTXT);
        FillNameArea(*dpt);

```

```

    }while(loop); /* Fill all value into DEPT Data Box */
    textattr(F_WHITE;B_CYAN); /* Edit Bar */
    gotoxy(2,6+dpt->cell- dpt->top);

```

```

    cprintf(" %2d | %-20s | %-5d ",

```

```
dpt->cell+1,dpt->pArea[dpt->cell].dName,dpt->pArea[dpt->cell].dArea);
```

```
message = InputKey();          /* input key */
```

```
switch(message)
```

```
{
```

```
case KUP :                      /* up */
```

```
    count= dpt->cell - 1;
```

```
    if (count < dpt->top)      /* check limit */
```

```
    {
```

```
        if(count>=0)
```

```
        {    --dpt->top;
```

```
            --dpt->bottom;
```

```
            dpt->cell=count;
```

```
        }
```

```
        else Beep();
```

```
    }
```

```
    else
```

```
        dpt->cell=count;
```

```
    break;
```

```
case kDOWN:                     /* down */
```

```
    count = dpt->cell + 1;
```

```
    if (count > dpt->bottom)     /* check limit */
```

```
    {
```

```
        if(count <= dpt->size-1)
```

```
        {    ++dpt->top;
```

```
            ++dpt->bottom;
```

```
            dpt->cell=count;
```

```
        }
```

```
        else Beep();
```

```
    }
```

```
    else
```

```
        dpt->cell = count;
```

```
    break;
```

```
case kCR :                      /* Enter */
```

```
    EnterNameArea(dpt); /* Input Name and area */
```

```
    break;
```

```
case kESC :
```

```
    loop=FALSE;
```

```
    break;
```

```
}
```

```
}while(loop==TRUE);
```

```
EraseWin(&wDpt);
```

```
}
```

```
/*-----  
void EnterNameArea(DEPT *dpt)  
    :    Input Department Data ... Name and Area  
-----*/
```

```
void EnterNameArea(DEPT *dpt)
```

```
{
```

```
    char str[21];
```

```
    int message;
```

```
    int value;
```

```
    int count;
```

เอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    count = 0;
```

```
    loop=TRUE;
```

```
    do{
```

```

setcursor(CURSROFF);
textattr(F_WHITE;B_CYAN);          /* Print Edit bar */
gotoxy(2,6+dpt->cell- dpt->top);

cprintf(" %2d | %-20s | %-5d ",
dpt->cell+1,dpt->pArea[dpt->cell].dName,dpt->pArea[dpt->cell].dArea);

textattr(F_YELLOW;B_RED);          /* Print hilite-Edit bar */
if(count==0)
{
    gotoxy(9,6+dpt->cell- dpt->top);
    cprintf("%-20s",dpt->pArea[dpt->cell].dName);
}
else
{
    gotoxy(33,6+dpt->cell- dpt->top);
    cprintf("%-5d",dpt->pArea[dpt->cell].dArea);
}

message = InputKey();               /* input key */
switch(message)
{
case kRIGHT:
case kTAB :    count++;
                if(count>1) count=0;
                break;
case kLEFT :   count--;
                if(count<0) count=1;
                break;
case kCR :
                switch(count)
                {
case 0:
                    gotoxy(9,6+dpt->cell- dpt->top);
                    cprintf("%20s"," ");
                    setcursor(CURSROFF);
                    gotoxy(9,6+dpt->cell- dpt->top);

                    message = getstr(21,str); /* Input Name */
                    if((message==kCR))
                        strcpy(dpt->pArea[dpt->cell].dName,str);
                    break;
case 1:
                    gotoxy(33,6+dpt->cell- dpt->top);
                    cputs(" ");
                    gotoxy(33,6+dpt->cell- dpt->top);
                    value = GetInt();          /* Input Area */
                    dpt->pArea[dpt->cell].dArea = value;
                    break;
                }
                break;

case kESC :
                loop=FALSE;
                break;
        }
    }while(loop==TRUE);
}

/*-----
void EditAreaPerBlock(DEPT *dpt)
    : Edit Area per block value in dpt.AreaPerBlock
    -----*/
void EditAreaPerBlock(DEPT *dpt)
{
    WIN w = {20,11,60,14," Edit Area per Block ",DOUBLE};

```

```
int value;
```

```
ShowWin(&w,CNRBOR,CNRTXT);
```

```
do{
```

```
    gotoxy(1,1);
```

```
    cprintf("Area per block = %5d",dpt->AreaPerBlock);
```

```
    gotoxy(1,2);
```

```
    cprintf("Enter new value : ");
```

```
    gotoxy(19,2);
```

```
    value = GetInt();
```

```
    } while(value<=0);
```

```
    dpt->AreaPerBlock = value;
```

```
    EraseWin(&w);
```

```
}
```

```
/*-----  
void LoadDEPT(DEPT *dpt)  
    : Load Department data  
-----*/
```

```
void LoadDEPT(DEPT *dpt)
```

```
{
```

```
    FILE *fpLoad;
```

```
    WIN load = {20,7,60,9," LOAD Department Data ",SINGLE};
```

```
    WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
```

```
    char name[9],oldname[13],key;
```

```
    int value,length,loop,x,y,size;
```

```
    long chksum,sum;
```

```
    int m,type;
```

```
    loop=TRUE;
```

```
    do{
```

```
        ShowWin(&load,DEPTBOR,DEPTTXT);
```

```
        setcursor(CURSORON);
```

```
        cputs(" File Name : ");
```

```
        /* Input File name */
```

```
        m=getstr(9,name);
```

```
        if(m==kESC)
```

```
        {
```

```
            EraseWin(&load);
```

```
            return;
```

```
        }
```

```
        strcpy(oldname,dpt->filename);
```

```
        /* Add file extension */
```

```
        strcpy(dpt->filename,name);
```

```
        strcat(dpt->filename,".DPT");
```

```
        /* *.DPT */
```

```
        ShowWin(&msg,rMSGWINBORDER,rMSGWINTXT);
```

```
        fpLoad = fopen(dpt->filename,"rb");
```

```
        /* Open file */
```

```
        if(fpLoad != NULL)
```

```
        { /* Open file */
```

```
            loop=FALSE;
```

```
        }
```

```
        else
```

```
        { /* Can't Open file */
```

```
            gotoxy(10,2);
```

```
            cprintf("File %-12s not found !",dpt->filename);
```

```
            getch();
```

```
            EraseWin(&msg);
```

```

        EraseWin(&load);
        strcpy(dpt->filename,oldname);
    }
}while(loop==TRUE);

/***** LOAD *****/
clrscr();
gotoxy(10,2);
cputs(" Loading File ... ");

fread(&type,sizeof(int),1,fpLoad);      /* read Type of file */

if (type != 3)                          /* Type 3 */
{
    fclose(fpLoad);
    clrscr();
    gotoxy(6,2);
    cprintf("%-12s is not a Department Data file !",dpt->filename);
    getch();
    EraseWin(&msg);
    EraseWin(&load);
    strcpy(dpt->filename,oldname);
    return;
}

fread(&length,sizeof(int),1,fpLoad);    /* read size of department data */

dpt->size=length;                        /* set size of department */
dpt->top  = 0;                            /* set display range: top,bottom */

if(dpt->size <= 10) dpt->bottom = dpt->size-1;
else dpt->bottom = 10;

dpt->cell = 0;                           /* set cell cursor */
FreeMemoryAREA(dpt->pArea);              /* Clear old memory */
dpt->pArea = InitMemoryAREA(dpt->size);    /* Allocate new memory */

fread(dpt->pArea,sizeof(AREA),dpt->size,fpLoad); /* read department data */
fclose(fpLoad);                          /* Close file */

cputs("Completed !");
getch();
EraseWin(&msg);
EraseWin(&load);
setcursor(CURSROFF);
}

/*-----
void SaveDEPT(DEPT *dpt)
    : Save Department data
-----*/

void SaveDEPT(DEPT *dpt)
{
    FILE *fpSave;

    WIN save = {20,7,60,9," SAVE Department Data ",SINGLE};
    WIN msg  = {15,12,65,16," MESSAGE ",DOUBLE};
    char name[9],key;
    int value,length,loop,x,y;
    long chksum;
    int m,type;

    if((dpt->size <2)!!(dpt->size>30))
    {

```

```

    Beep();
    ShowWin(&msg,rMSGWINBORDER,rMSGWINTXT);
    gotoxy(5,2);
    cputs(" Department Data hasn't been loaded ! ");
    getch();
    EraseWin(&msg);
    return;
}

type=3;                                /* set type = 3 */
loop=TRUE;

do{                                     /* Input file name */
    ShowWin(&save,RELBOR,RELTXT);

    setcursor(CURSORON);
    clrscr();
    cputs(" File Name : ");
    m=getstr(9,name);

    if((m==kESC) || (strlen(name)==0))
    {
        EraseWin(&save);
        return;
    }

    strcpy(dpt->filename,name);          /* add file extension */
    strcat(dpt->filename, ".DPT");        /* *.DPT */

    fpSave = fopen(dpt->filename,"rb");   /* Open file */

    ShowWin(&msg,rMSGWINBORDER,rMSGWINTXT);

    if(fpSave != NULL)                   /* check existed file */
    {
        gotoxy(6,1);
        cprintf(" File %-12s is already existed !",dpt->filename);
        gotoxy(15,2);
        cputs("Overwrite (Y/N) ? ");

        do{
            key=toupper(getche());
        }while( !(key=='Y' || key=='N'));

        if(key=='Y')
        {
            loop=FALSE;
            fclose(fpSave);
        }
        else
        {
            loop=TRUE;
            fclose(fpSave);
            EraseWin(&msg);
            EraseWin(&save);
        }
    }
    else loop=FALSE;

}while(loop==TRUE);

/***** SAVE *****/
clrscr();
gotoxy(10,2);
cputs(" Saving File ... ");

fpSave = fopen(dpt->filename,"wb");

```

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต
 ไม่ว่าในรูปแบบใดก็ตาม หากท่านมีให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

fwrite(&type, sizeof(int), 1, fpSave);    /* write type into file */

length = dpt->size;
fwrite(&length, sizeof(int), 1, fpSave); /* write length into file */

fwrite(dpt->pArea, sizeof(AREA), dpt->size, fpSave); /* write department data */

fclose(fpSave);                          /* Close file */

cputs("Completed !");
getch();
EraseWin(&msg);
EraseWin(&save);
setcursor(CURSROFF);
}

/*-----
void NewDEPT(DEPT *dpt)
: Create new department data, and allocate memory.
Clear dpt variable.
-----*/

void NewDEPT(DEPT *dpt)
{
    WIN w = {20,7,60,9," NEW Department data ",SINGLE};    /* window */
    int value;

    FreeMemoryAREA(dpt->pArea);    /* Free old memory */
    ClearDeptValue(dpt);    /* Clear department variable */

    ShowWin(&w,DEPTBOR,DEPTTXT);

    do{
        gotoxy(5,1);    /* Input No. of Department */
        cputs("How many departments ? ");
        value = GetInt();
    }while((value <2) || (value >30));

    EraseWin(&w);

    dpt->size = value;    /* Set department variable */
    dpt->top = 0;
    if(dpt->size <= 10)
        dpt->bottom = dpt->size-1;
    else dpt->bottom = 10;
    dpt->cell = 0;
    dpt->pArea = (AREA *) InitMemoryAREA(dpt->size);    /* Allocate memory */
}

/*-----
void ShowGrid(int xl,int yl,int x2,int y2,int grid_size)
: Draw grid line on the screen (graphic mode).
start from (xl,yl) to (x2,y2) step grid_size.
-----*/

void ShowGrid(int xl,int yl,int x2,int y2,int grid_size)
{
    int i,j;

    setcolor(DARKGRAY);
    setlinestyle(SOLID_LINE,0,NORM_WIDTH);
    for(i=xl;i<=x2;i+=grid_size)
        line(i,yl,i,y2);
    for(j=yl;j<=y2;j+=grid_size)

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
 ไม่ควรนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย

```
line(x1,j,x2,j);
```

```
}
```

```
/*-----  
void read_mouse(int *mx,int *my,int deltax,int deltay,int grid_size)  
: Read mouse move and set mouse cursor position.  
  mx,my = mouse position  
  deltax = delta x  
  deltay = delta y  
-----*/
```

```
void read_mouse(int *mx,int *my,int deltax,int deltay,int grid_size)  
{
```

```
  if(deltax>0){  
    *mx = *mx+grid_size;  
    if( *mx >= 639) *mx = 639;  
  }
```

```
  if(deltax<0){  
    *mx = *mx-grid_size;  
    if(*mx<0) *mx=0;  
  }
```

```
  if(deltay>0){  
    *my = *my+grid_size;  
    if(*my>479) *my=479;  
  }
```

```
  if(deltay<0){  
    *my = *my-grid_size;  
    if(*my<0) *my=0;  
  }
```

```
  set_mouse_position(*mx,*my);  
}
```

```
/*-----  
void ClearBox()  
: Clear message box (graphic mode)  
-----*/
```

```
void ClearBox()  
{  
  setfillstyle(SOLID_FILL,BLACK);  
  floodfill(5,364,RED);  
}
```

```
/*-----  
int InputSheetSize(DEPT *dpt,SHEET *sheet)  
: Input Layout Sheet size  
-----*/
```

```
int InputSheetSize(DEPT *dpt,SHEET *sheet)  
{
```

```
  int loop,loop2;  
  int value;  
  int x,y;  
  double r;  
  char string[12];
```

```
  if(sheet->blocknum>63*35)
```

```
  {  
    ClearBox();
```

```
    printf(2,38,WHITE,BLACK,"Total area = %-10lu, Area per block = %-5d",  
      dpt->total,dpt->AreaPerBlock);
```

```
    printf(48,38,YELLOW,BLACK,"The number of blocks = %5lu",
```

```

        sheet->blocknum);
    gprintf(2,40,WHITE,BLACK,
    "Error! The number of blocks must not greater than 2205 blocks.");
    gprintf(2,41,WHITE,BLACK,
    "Please changes the value of department area or the area per block value.");
    getch();
    return(FALSE);
}

```

```

loop2=TRUE;

```

```

do{
    ClearBox();
    gprintf(2,38,WHITE,BLACK,"Total area = %-10lu, Area per block = %-5d "
    ,dpt->total,dpt->AreaPerBlock);
    gprintf(48,38,YELLOW,BLACK,"The number of blocks = %5lu"
    ,sheet->blocknum);
    gprintf(2,40,WHITE,BLACK,
    "Please choose the length and width of the layout sheet :");
    gprintf(2,41,WHITE,BLACK,"1. Square (default value)");
    gprintf(2,42,WHITE,BLACK,"2. Length-to-width ratio of 2:1");
    gprintf(2,43,WHITE,BLACK,"3. Enter other Length-to-width ratio x:y");
    gprintf(2,44,WHITE,BLACK,"4. Enter size manually ");
    gprintf(2,45,WHITE,BLACK,"5. Go back to change department area ");
    gprintf(2,47,WHITE,BLACK,"Which configuration do you desire ? ");

    do{
        value = gGetInt(38,47,WHITE,DARKGRAY);
    }while((value<1)!!(value>5));

    ltoa(sheet->blocknum,string,10);
    r=atof(string);
    switch(value)
    {
    case 1: sheet->length=sqrt(r);
            sheet->width=sheet->blocknum/sheet->length;
            break;
    case 2: sheet->width=sqrt(r/2);
            sheet->length=sheet->blocknum/sheet->width;
            break;
    case 3:
            loop = TRUE;
            do{

                ClearBox();
                gprintf(2,38,YELLOW,BLACK,
                "The number of blocks = %5lu",sheet->blocknum);
                gputs(2,39,WHITE,BLACK,
                "Enter the length-to-width ratio , x:y ");
                gputs(2,40,WHITE,BLACK,
                "REMARK! x must greater than y ");
                gputs(2,42,WHITE,BLACK,
                "Please enter the length ratio (x) = ");

                do{
                    x=gGetInt(38,42,WHITE,DARKGRAY);
                }while(x<=0);

                gputs(2,44,WHITE,BLACK,
                "Please enter the width ratio (y) = ");

                do{
                    y=gGetInt(38,44,WHITE,DARKGRAY);
                }while(y<=0);

                if (x>y)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ หากมีการนำเอกสารนี้ไปดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    sheet->length=sqrt((x*r)/y);
    sheet->width= sheet->blocknum/sheet->length;
    loop=FALSE;
}

}while(loop==TRUE);
break;
case 4:
    loop=TRUE;
    do{
        ClearBox();
        gprintf(2,38,YELLOW,BLACK,
            "The number of blocks = %5lu",sheet->blocknum);

        gputs(2,39,WHITE,BLACK,
            "Enter one of the sides,then the other is automatically calulated.");
        gputs(2,41,WHITE,BLACK,
            "Please enter one side length : ");
        do{
            sheet->length=gGetInt(33,41,WHITE,DARKGRAY);
        }while(sheet->length<=0);
        sheet->width= sheet->blocknum/sheet->length;
        if(sheet->length<sheet->width)
        {
            value=sheet->length;
            sheet->length = sheet->width;
            sheet->width = value;
        }

        if(sheet->width==0)
        {
            gputs(2,43,YELLOW,BLUE,
                "Invalid Value ! Plese enter again....");
            getch();
        }
        else loop=FALSE;

    }while(loop==TRUE);
    break;
case 5: return(FALSE);
}

ClearBox();
gprintf(2,38,YELLOW,BLACK,"The number of blocks = %5d Blocks ",
    sheet->blocknum);
gprintf(2,41,WHITE,BLACK,"Sheet Length = %5d",sheet->length);
gprintf(2,42,WHITE,BLACK,"Sheet Width = %5d",sheet->width);
gprintf(2,43,LIGHTCYAN,BLACK,"Sheet Area = %5d",
    sheet->length*sheet->width);
if(sheet->length>63)
{
    gputs(2,45,YELLOW,BLUE,
        "Sheet length must not greater than 63.");
    gputs(2,46,WHITE,BLACK,"Please select again...");
}
else
if(sheet->width>35)
{
    gputs(2,45,YELLOW,BLUE,
        "Sheet width must not greater than 35.");
    gputs(2,46,WHITE,BLACK,"Please select again...");
}
else
if(sheet->length*sheet->width != sheet->blocknum)
{

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์โดยมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ไม่ว่ากรณีใดๆ ห้ามมิให้คัดลอกหรือเผยแพร่ข้อมูลใดๆ จากเอกสารนี้โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย

if(sheet->length*sheet->width != sheet->blocknum)

```

        gputs(2,45,YELLOW,BLUE,
        "Selected blocks aren't equal to total blocks !");
        gputs(2,46,WHITE,BLACK,"Please select again...");
    }
    else
        loop2=FALSE;

    getch();

    }while(loop2==TRUE);
    return(TRUE);
}

/*-----
void QsortTCR(DPTCR *tcr,int lb,int ub)
    : Quick sort the TCR table from great value to less value
-----*/

void QsortTCR(DPTCR *tcr,int lb,int ub)
{
    int i,j,flag=1;
    int fac_temp;
    float key,temp;

    if(lb>=ub) return;

    i=lb;
    j=ub+1;

    key=tcr[lb].cr;
    while(flag)
    {
        i++;
        /* scan the key from left to right */
        while(tcr[i].cr>key)
            i++;
        j--;
        /* scan the key from left to right */
        while(tcr[j].cr<key)
            j--;
        /* interchange records */
        if(i<j)
        {
            temp=tcr[i].cr;
            fac_temp =tcr[i].dptno;
            tcr[i].cr=tcr[j].cr;
            tcr[i].dptno=tcr[j].dptno;
            tcr[j].cr=temp;
            tcr[j].dptno=fac_temp;
        }
        else
            flag=0;
    }
    temp=tcr[lb].cr;
    fac_temp =tcr[lb].dptno;
    tcr[lb].cr=tcr[j].cr;
    tcr[lb].dptno=tcr[j].dptno;
    tcr[j].cr=temp;
    tcr[j].dptno=fac_temp;

    QsortTCR(tcr,lb,j-1);
    QsortTCR(tcr,j+1,ub);
}

/*-----

```

เอกสารนี้เป็นทรัพย์สินของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
 ไม่ควรนำเอกสารนี้ไปใช้ในการเรียนการสอนโดยไม่ได้รับอนุญาต
 ไม่ควรนำเอกสารนี้ไปใช้ในการทำวิจัยโดยไม่ได้รับอนุญาต
 ไม่ควรนำเอกสารนี้ไปใช้ในการทำธุรกิจโดยไม่ได้รับอนุญาต
 ไม่ควรนำเอกสารนี้ไปใช้ในการทำกิจกรรมโดยไม่ได้รับอนุญาต
 ไม่ควรนำเอกสารนี้ไปใช้ในการทำกิจกรรมโดยไม่ได้รับอนุญาต

```
void RunLayout(REL *rel, FROMTO *ft, CLOSENESS cr, DEPT *dpt)
: Run layout
```

-----*/

```
void RunLayout(REL *rel, FROMTO *ft, CLOSENESS cr, DEPT *dpt)
{
    SHEET sheet;
    CORDINATE N1, N2;
    CORDINATE M1, M2;
    CORDINATE MIN, MAX;

    DPTCR *tcr;    /* Total Closeness Rating */
    DPTCR *rank;   /* use in PCR to print relation */
    int *remain;   /* Remain of department which are not located in sheet */
    int *blocks;   /* Keep the number of blocks */
    int *edge;     /* keep department which adjacent to blank edge */
    LIST *list;
    CORDINATE *cg;
    int area;

    int pcr;
    int check;
    int department;
    int graphdriver, graphmode;
    int *address;
    int value;
    int mx, my;
    int a, b, size, total;
    char string[12];
    int loop, loop2;
    int deltax, deltay;
    int message;
    int i, j, x, y;
    unsigned long sum=0;
    double r;
    float eff;

    address = SaveAllScreen();

    size = dpt->size;
    tcr = calloc(size, sizeof(DPTCR)); /* Total Closeness Rates */
    rank = calloc(size, sizeof(DPTCR));
    remain = calloc(size, sizeof(int)); /* Keep Un-located department flag*/
    edge = calloc(size, sizeof(int));
    blocks = calloc(size, sizeof(int));
    cg = (CORDINATE *) calloc(size, sizeof(CORDINATE));
    list = (LIST *) calloc(size, sizeof(LIST));

    sheet.length=0; /* Clear sheet variable */
    sheet.width=0;
    sheet.gridsize=10;
    sheet.blocknum=0;
    sheet.map=NULL;

    for(i=0; i<size; i++) /* Initialize value */
    {
        list[i].length=0;
        list[i].first =NULL;
        list[i].last  =NULL;
        remain[i]=FALSE;
    }

    /***** use Graphic mode *****/
    detectgraph(&graphdriver, &graphmode);
    initgraph(&graphdriver, &graphmode, "");
    clearviewport();
    mouse_reset();
    LayoutScreen();
```

```
ShowGrid(0,0,640,350, sheet.gridsize);
```

```
/****** Blocks *****/
```

```
for(i=0;i<size;i++) /* Find total area */
```

```
{
    sum= sum + dpt->pArea[i].dArea;
    blocks[i] = dpt->pArea[i].dArea / dpt->AreaPerBlock;
}
```

```
dpt->total = sum; /* Set total area */
```

```
sheet.blocknum = dpt->total / dpt->AreaPerBlock; /* Calculate no. of blocks */
```

```
message=InputSheetSize(dpt,&sheet); /* Input Sheet size */
```

```
if(message==FALSE)
```

```
{
    closegraph();
    RestoreAllScreen(address);
    free(tcr);
    free(rank);
    free(edge);
    free(remain);
    free(blocks);
    free(cg);
    for (i=0;i<size;i++)
        freelist(list+i);
    free(list);
    return;
}
```

```
setcolor(LIGHTGREEN);
```

```
rectangle(0,0,sheet.length*sheet.gridsize,sheet.width*sheet.gridsize);
```

```
/****** Alloc memory of map *****/
```

```
sheet.map = calloc(sheet.length*sheet.width,sizeof(int));
```

```
if ( sheet.map == NULL )
```

```
{
    printf("Not enough memory : Run Layout step ");
    getch();
    closegraph();
    RestoreAllScreen(address);
    free(tcr);
    free(rank);
    free(edge);
    free(remain);
    free(blocks);
    free(cg);
    for(i=0;i<size;i++)
        freelist(list+i);
    free(list);
    return;
}
```

```
for(i=0;i<sheet.length*sheet.width;i++) /* Cleat Map */
```

```
*(sheet.map+i) = -1;
```

```
/****** Set mouse *****/
```

```
set_mouse_bound(0,0,sheet.length*sheet.gridsize,
    sheet.width*sheet.gridsize);
```

```
mx=0;my=0;
```

```
set_mouse_position(mx,my);
```

```
ClearBox();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ให้นำไปเผยแพร่โดยไม่ได้รับอนุญาตให้ตัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
    total=0;
```

```

        if(i>j)      {a=i;b=j;}
        else        {a=j;b=i;}

        if(a != b)
        {
            value = GetRELValue(*rel,a,b);
            value = ClosenessRate(cr,value);

            total = total+value;
        }
    }
    tcr[i-1].dptno = i-1;
    tcr[i-1].cr    = total;
}

QsortTCR(tcr,0,size-1);

gputs(2,38,WHITE,BLACK,
      "The following departments have the highest TCRs - ");
gputs(2,39,CYAN,BLACK,"      Dept.      TCR      Area");
gputs(2,40,WHITE,BLACK,"      -----");

for(i=0;i<5;i++)
if(i<size)
if(tcr[i].dptno!=-1)
    gprintf(2,41+i,YELLOW,BLACK,"      %2d      %5d      %-5d",
            tcr[i].dptno+1,tcr[i].cr,dpt->pArea[tcr[i].dptno].dArea);

gputs(2,47,WHITE,BLACK,
      "Which department would you like to place first ? ");

loop=TRUE;
do{
    value = gGetInt(52,47,WHITE,DARKGRAY);
    for(i=0;i<5;i++)
        if (tcr[i].dptno+1 == value)
            loop=FALSE;
    if (value<=0)
        loop=TRUE;
} while(loop==TRUE);

department=value-1;
remain[department]=TRUE;

/***** Input First dept. Rect *****/
ClearBox();

check=InputPoly(&mx,&my,list,&sheet,blocks,cg,department,size);
if(check==999)
{

```

```

    ClearBox();
    cursor_off();
    closegraph();
    RestoreAllScreen(address);
    free(tcr);
    free(rank);
    free(edge);
    free(remain);
    free(blocks);
    free(cg);
    for(i=0;i<size;i++)
        freelist(list+i);
    free(list);
    return;
}

```

เอกสารนี้เป็นเอกสารเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ หากมีให้ตัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

}
check=LocateToMap(list+department, sheet, department);

MIN=FindMin(list+department);
MAX=FindMax(list+department);
MIN.x = MIN.x/sheet.gridsize;
MIN.y = MIN.y/sheet.gridsize;
MAX.x = MAX.x/sheet.gridsize;
MAX.y = MAX.y/sheet.gridsize;

area=blocks[department];
*(cg+department) = FindCG(MIN.x,MAX.x,MIN.y,MAX.y,
                           area,sheet,department);
itoa(department+1,string,10);
setcolor(YELLOW);
outtextxy(cg[department].x*sheet.gridsize,
          cg[department].y*sheet.gridsize,string);
setcolor(WHITE);
ClearBox();

loop=TRUE;
do{
/***** PCR *****/

/* Clear TCR */
for(i=0;i<size;i++)
{
    tcr[i].dptno = -1;
    tcr[i].cr = 0;
    edge[i]=FALSE;
}

/** Find edge */
for(i=0;i<sheet.length;i++)
for(j=0;j<sheet.width-1;j++)
{
    if(    ( *(sheet.map + sheet.length*j +i)      != -1)
        && ( *(sheet.map + sheet.length*(j+1) +i) == -1) )
        edge[*(sheet.map + sheet.length*j +i)]=TRUE;

    if(    ( *(sheet.map + sheet.length*j +i)      == -1)
        && ( *(sheet.map + sheet.length*(j+1) +i) != -1) )
        edge[*(sheet.map + sheet.length*(j+1) +i)]=TRUE;
}

for(j=0;j<sheet.width;j++)
for(i=0;i<sheet.length-1;i++)
{
    if(    ( *(sheet.map + sheet.length*j +i)      != -1)
        && ( *(sheet.map + sheet.length*j +(i+1)) == -1) )
        edge[*(sheet.map + sheet.length*j +i)]=TRUE;

    if(    ( *(sheet.map + sheet.length*j +i)      == -1)
        && ( *(sheet.map + sheet.length*j +(i+1)) != -1) )
        edge[*(sheet.map + sheet.length*j +(i+1))]=TRUE;
}

/**** PCR ****/
for(i=1;i<=size;i++)
{
    if(remain[i-1]==FALSE)
    {
        total=0;
        for(j=1;j<=size;j++)
        {
            if(edge[j-1]==TRUE)
            {

```

```

        if (i>j) {a=i;b=j;}
        else    {a=j;b=i;}

        if(a != b)
        {
            value = GetRELValue(*rel,a,b);
            value = ClosenessRate(cr,value);

            total = total+value;
        }
    }
    tcr[i-1].dptno = i-1;
    tcr[i-1].cr    = total;
}

}

QsortTCR(tcr,0,size-1);

gputs(2,38,WHITE,BLACK,
      "The following departments have the highest PCR's - ");
gputs(2,39,CYAN,BLACK,
      " Dept.  PCR   Area      Top five ranking placed dpts.(rel)");
gputs(2,40,WHITE,BLACK,
      "-----");

for(i=0;i<5;i++)
{
    if(i<size)
    if(tcr[i].dptno!=-1)
    {
        gprintf(2,41+i,YELLOW,BLACK,"  %2d    %-5d  %-5d",
                tcr[i].dptno+1,tcr[i].cr,dpt->pArea[tcr[i].dptno].dArea);

        /* clear rank */
        for(j=0;j<size;j++)
        {
            rank[j].dptno = -1;
            rank[j].cr = 0;
        }

        /* Rank */
        for(j=0;j<size;j++)
        {
            if(edge[j]==TRUE)
            {
                x=tcr[i].dptno+1;
                y=j+1;

                if (x>y)    {a=x;b=y;}
                else       {a=y;b=x;}

                if(a != b)
                {
                    value = GetRELValue(*rel,a,b);
                    value = ClosenessRate(cr,value);

                    rank[j].dptno = j;
                    rank[j].cr = value;
                }
            }
        }
    }
}

QsortTCR(rank,0,size-1);

for(j=0;j<5;j++)
if(j<size)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามแก้ไขเพิ่มเติมเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

11 {rank[j].dptno := -1;
    printf(25+j*8,41+i,WHITE,BLACK,"%2d(%d)",
        rank[j].dptno+1,rank[j].cr);
}
}
gputs(2,47,WHITE,BLACK,"Which department would you like to place ? ");

loop2=TRUE;
do{
    value = gGetInt(45,47,WHITE,DARKGRAY);
    for(i=0;i<5;i++)
        if (tcr[i].dptno+1 == value)
            {pcr=tcr[i].cr;loop2=FALSE;break;}
    if(value<=0) loop=TRUE;
}while(loop2==TRUE);

department=value-1;
remain[department]=TRUE;

/***** Input dept. Rect *****/
do{
    ClearBox();

    check=InputPoly(&mx,&my,list,&sheet,blocks,cg,department,size);
    if(check==999)
    {
        ClearBox();
        cursor_off();

        closegraph();
        RestoreAllScreen(address);
        free(tcr);
        free(rank);
        free(edge);
        free(remain);
        free(blocks);
        free(cg);
        for (i=0;i<size;i++)
            freelist(list+i);
        free(list);
        return;
    }
    check=LocateToMap(list+department,sheet,department);
    if (check==FALSE)
    {
        ClearBox();
        freelist(list+department);
        Beep();
        gputs(2,38,YELLOW,BLACK,
            "Locate Error! Overlapped department. Please input again...");
        getch();
        ShowGrid(0,0,sheet.length*sheet.gridsize,
            sheet.width*sheet.gridsize,sheet.gridsize);
        setcolor(LIGHTGREEN);
        rectangle(0,0,sheet.length*sheet.gridsize,
            sheet.width*sheet.gridsize);
    }
}while(check==FALSE);

MIN=FindMin(list+department);
MAX=FindMax(list+department);
MIN.x = MIN.x/sheet.gridsize;
MIN.y = MIN.y/sheet.gridsize;
MAX.x = MAX.x/sheet.gridsize;
MAX.y = MAX.y/sheet.gridsize;

```

```

area=blocks[department];
*(cg+department) = FindCG(MIN.x,MAX.x,MIN.y,MAX.y,
                           area,sheet,department);

```

```

itoa(department+1,string,10);

```

```

setcolor(YELLOW);
outtextxy(cg[department].x*sheet.gridsize,
          cg[department].y*sheet.gridsize,string);
setcolor(WHITE);

```

```

ClearBox();

```

```

for(i=0;i<size;i++)
{
    loop=FALSE;
    if (remain[i]==FALSE)
        { loop=TRUE;break; }
}
}while(loop==TRUE);

```

```

eff=Efficiency(ft,cg);

```

```

gprintf(2,40,WHITE,BLACK,"Efficiency of layout = %5.2f ",eff);

```

```

gputs(2,42,YELLOW,BLACK,"Output Plant Layout to text-file ? Yes/No");

```

```

do{
value=toupper(getch());
}while(!((value=='Y')||(value=='N')));

```

```

if(value=='Y')
{
    gputs(2,43,WHITE,BLACK,"Yes ! Press any key to continue...");
    getch();
    OutputToTextFile(sheet,blocks,cg,dpt,rel,ft,eff);
}
else
{
    gputs(2,43,WHITE,BLACK,"No! Press any key to return menu...");
    getch();
}
ClearBox();
cursor_off();

```

```

closegraph();
RestoreAllScreen(address);
free(tcr);
free(rank);
free(edge);
free(remain);
free(blocks);
free(cg);
for(i=0;i<size;i++)
    freelist(list+i);
free(list);

```

```

}

```

```

/*-----
float Efficiency(FROMTO *ft,CORDINATE *cg) นั้น ไม่นอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
: Calculate efficiency of layout
ไม่มีการรับประกันว่าฟังก์ชันนี้จะทำงานได้อย่างถูกต้องและอาจส่งผลให้เกิดข้อผิดพลาดได้หากมีการนำโค้ดนี้ไปใช้ */

```

```

float Efficiency(FROMTO *ft,CORDINATE *cg)

```

```

{
    int value;
    int i,j,size;
    float ft_moment,cg_moment,eff;

    ft_moment=cg_moment=0;
    size=ft->size;
    for(i=1;i<=size;i++)
        for(j=1;j<=size;j++)
        {
            value= GetFTValue(*ft,i,j);
            ft_moment += value;
            cg_moment += value*(abs(cg[i-1].x-cg[j-1].x) +
                                abs(cg[i-1].y-cg[j-1].y))/10; /* 10 is gridsize */
        }
    eff = (ft_moment*100)/cg_moment;

    return(eff);
}

```

```

/*-----
CORDINATE FindMin(LIST *list)
: Find minimum value of nodes in list
-----*/

```

```

CORDINATE FindMin(LIST *list)
{
    int i;
    CORDINATE min;
    NODEPTR q;

    min.x=min.y=9999;
    q=list->first;
    for(i=0;i<list->length;i++)
    {
        if(q->c.x<min.x) min.x=q->c.x;
        if(q->c.y<min.y) min.y=q->c.y;
        q=q->next;
    }

    return(min);
}

```

```

/*-----
CORDINATE FindMax(LIST *list)
: find maximum value of nodes in list
-----*/

```

```

CORDINATE FindMax(LIST *list)
{
    int i;
    CORDINATE max;
    NODEPTR q;

    max.x=max.y=0;
    q=list->first;
    for(i=0;i<list->length;i++)
    {
        if(q->c.x>max.x) max.x=q->c.x;
        if(q->c.y>max.y) max.y=q->c.y;
        q=q->next;
    }

    return(max);
}

```

เอกสารนี้เป็นเอกสารของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
 ไม่ควรนำเอกสารนี้ไปใช้ประโยชน์ด้านการค้า
 ไม่ควรนำเอกสารนี้ไปทำสิ่งอื่นใดที่ขัดต่อวัตถุประสงค์ของเอกสาร

```

}

/*-----
CORDINATE FindCG(int xmin,int xmax,int ymin,int ymax,int area,
    SHEET sheet,int department)
    : Find CG point of department
-----*/

```

```

CORDINATE FindCG(int xmin,int xmax,int ymin,int ymax,int area,
    SHEET sheet,int department)
{
    int c,i,j,sum;
    int Xi,Yi;
    CORDINATE pos;

    sum=c=0;
    for (i=xmin;i<=xmax;i++)
    {
        Xi=0;
        for (j=ymin;j<=ymax;j++)
            if (*(sheet.map+sheet.length*j + i)==department)
                Xi++;

        sum=sum+Xi*(++c);
    }
    pos.x = sum/area+xmin;

    sum=c=0;
    for (j=ymin;j<=ymax;j++)
    {
        Yi=0;
        for (i=xmin;i<=xmax;i++)
            if (*(sheet.map + sheet.length*j + i)==department)
                Yi++;
        sum=sum+Yi*(++c);
    }
    pos.y = sum/area+ymin;

    return(pos);
}

```

```

/*-----
int CalArea(int xmax,int xmin,int ymax,int ymin,
    SHEET sheet,int department)
    : Calulate area of department
-----*/

```

```

int CalArea(int xmax,int xmin,int ymax,int ymin,
    SHEET sheet,int department)
{
    int x,y;
    int sum=0;

    for (y=ymin;y<=ymax;y++)
    for (x=xmin;x<=xmax;x++)
        if (*(sheet.map + sheet.length*y+x) == department)
            sum++;

    return(sum);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/*-----
int LocateToMap(LIST *list,SHEET sheet,int department)
    : Locate department into layout map

```

```

int LocateToMap(LIST *list,SHEET sheet,int department)
{
    int i,j,a,b;

    setfillstyle(SOLID_FILL,LIGHTGRAY);
    floodfill(list->first->c.x+1,list->first->c.y+1,WHITE);

    for(i=0;i<sheet.length;i++)
        for(j=0;j<sheet.width;j++)
        {
            if(getpixel(i*sheet.gridsize+5,j*sheet.gridsize+5) == LIGHTGRAY)
            {
                if(*(sheet.map +sheet.length*j +i)== -1)
                {
                    *(sheet.map +sheet.length*j +i)=department;
                }
                else
                {
                    for(a=0;a<sheet.length;a++)
                    for(b=0;b<sheet.width;b++)
                        if(*(sheet.map +sheet.length*b +a)==department)
                            *(sheet.map +sheet.length*b +a) = -1;

                    setfillstyle(SOLID_FILL,BLACK);
                    floodfill(list->first->c.x+1,list->first->c.y+1,WHITE);
                    return(FALSE);
                }
            }
        }
    setfillstyle(SOLID_FILL,BLACK);
    floodfill(list->first->c.x+1,list->first->c.y+1,WHITE);

    return(TRUE);
}

/*-----*/
int InputPoly(int *mx,int *my,LIST *l,SHEET *sheet,int *blocks,
    CORDINATE *cg,int department,int size)
    : Input the frame of department in layout
/*-----*/

int InputPoly(int *mx,int *my,LIST *l,SHEET *sheet,int *blocks,
    CORDINATE *cg,int department,int size)
{
    NODEPTR p;
    CORDINATE n1,n2,m;
    LIST *list;
    int blocknum;

    int message;
    int ins_flag,draw;
    int value,select;
    int deltax,deltay;
    int menu,i;
    char ch;
    void *topsave;

    RedrawDepartment(l,cg,size,sheet->gridsize,department);
    list=l+department;
    blocknum=blocks[department];
    gprintf(2,38,YELLOW,BLACK,"Locate department : %-5d ",department+1);

```

```

gprintf(2,39,CYAN,BLACK,"Area = %5d Blocks.",blocknum);
gprintf(2,47,WHITE,BLACK,
"COMMAND> (P):place (D):delete (E):end (C):Clear sheet");

```

```

topsave =(void *) calloc(imagesize(0,0,getmaxx(),1),1);

```

```

message=FALSE;
setcolor(WHITE);
nl.x=n2.x=0;
n2.y=n2.y=0;

```

```

do{
    draw=FALSE;
    cursor_on();
    mouse_motion(&deltax,&deltay);

```

```

/***** Mouse Move *****/

```

```

if( deltax || deltay)

```

```

{
    read_mouse(mx,my,deltax,deltay,sheet->gridsize);

```

```

    if(*mx>sheet->length*sheet->gridsize)

```

```

        *mx=sheet->length*sheet->gridsize;

```

```

    if(*my>sheet->width*sheet->gridsize)

```

```

        *my=sheet->width*sheet->gridsize;

```

```

    gprintf(60,38,WHITE,BLACK,"Position(%2d,%2d)",

```

```

        *mx/sheet->gridsize,*my/sheet->gridsize);

```

```

}

```

```

/***** Left button pressed *****/

```

```

if(leftb_pressed())

```

```

{

```

```

    if(list->length==0)

```

```

    {

```

```

        nl.x = n2.x = *mx;

```

```

        nl.y = n2.y = *my;

```

```

        putpixel(nl.x,nl.y,LIGHTRED);

```

```

        p=getnode();

```

```

        list->first=list->last=p;

```

```

        p->c.x =nl.x;

```

```

        p->c.y =nl.y;

```

```

        p->next=NULL;

```

```

        list->length++;

```

```

        for (i=0;i<list->length;i++)

```

```

        {

```

```

            getinfo(*list,i,&m);

```

```

            gprintf(2+8*(i%9),44+i/9,WHITE,BLACK,"(%2d,%2d)",

```

```

                m.x/sheet->gridsize,m.y/sheet->gridsize);

```

```

        }

```

```

    }

```

```

}

```

```

else

```

```

{

```

```

    if ((nl.x==*mx)&&(nl.y!=*my)) || ((nl.x!=*mx)&&(nl.y==*my)) )

```

```

    {

```

```

        draw=TRUE;

```

```

        nl.x = *mx;

```

```

        nl.y = *my;

```

```

        p=list->last;

```

```

        n2.x = p->c.x;

```

```

        n2.y = p->c.y;

```

```

        ins_flag=insafter(p,nl);

```

```

        if (ins_flag)

```

```

            list->last = p->next;

```

```

            list->length++;

```

```

        }

```

```

    }

```

```

else

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น ผู้ใช้ต้องรับผิดชอบต่อเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        printf("ERROR : insafter ");
    }
}
for(i=0;i<list->length;i++)
{
    getinfo(*list,i,&m);
    gprintf(2+8*(i%9),44+i/9,WHITE,BLACK,"(%2d,%2d)",
        m.x/sheet->gridsize,m.y/sheet->gridsize);
}
}

/***** Keyboard hit *****/
if(kbhit())
{
    gprintf(60,38,WHITE,BLACK,"Position(%2d,%2d)",
        *mx/sheet->gridsize,*my/sheet->gridsize);

    message=InputKey();
    switch(message)
    {

    case KRIGHT: *mx+=sheet->gridsize;
        set_mouse_position(*mx,*my);
        break;
    case KLEFT:  *mx-=sheet->gridsize;
        set_mouse_position(*mx,*my);
        break;
    case KUP   : *my-=sheet->gridsize;
        set_mouse_position(*mx,*my);
        break;
    case KDOWN: *my+=sheet->gridsize;
        set_mouse_position(*mx,*my);
        break;
    case KALTX: message='a';
        break;

    case kCR   : /* Place node */
    case kSPC :
    case 'p'  :
    case 'P'  :
        if(list->length==0)
        {
            nl.x = n2.x = *mx;
            nl.y = n2.y = *my;
            putpixel(nl.x,nl.y,LIGHTRED);
            p=getnode();
            list->first=list->last=p;
            p->c.x =nl.x;
            p->c.y =nl.y;
            p->next=NULL;
            list->length++;
            gprintf(2,41,WHITE,BLACK,
                "Selected Node(s) = %5d",list->length);
            for(i=0;i<list->length;i++)
            {
                getinfo(*list,i,&m);
                gprintf(2+8*(i%9),44+i/9,WHITE,BLACK,
                    "(%2d,%2d)",m.x/sheet->gridsize,
                    m.y/sheet->gridsize);
            }
        }
        else
        {
            if( ((nl.x==*mx)&&(nl.y!=*my)) ||
                ((nl.x!=*mx)&&(nl.y==*my)) )

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น

```

        draw=TRUE;
        nl.x = *mx;
        nl.y = *my;

        p=list->last;
        n2.x = p->c.x;
        n2.y = p->c.y;
        ins_flag=insafter(p,nl);
        if(ins_flag)
        {
            list->last = p->next;
            list->length++;
        }
        else
            printf("ERROR : insafter ");
    }

    for(i=0;i<list->length;i++)
    {
        getinfo(*list,i,&m);
        gprintf(2+8*(i%9),44+i/9,WHITE,BLACK,
            "(%2d,%2d)",m.x/sheet->gridsize,m.y/sheet->gridsize);
    }

    }
    break;

case 'd' : /* Delete last node */
case 'D' :
case kDEL :
    if (list->length==1)
    {
        putpixel(list->first->c.x,list->first->c.y,DARKGRAY);
        freelist(list);
        gprintf(2,41,WHITE,BLACK,"Selected Node(s) = %5d",list->length);
        gprintf(2,44,WHITE,BLACK,"%75s"," ");
        gprintf(2,45,WHITE,BLACK,"%75s"," ");
    }
    if (list->length >1)
    {
        p=getptr(*list,list->length-2);
        delafter(p,&m);
        list->length--;
        list->last=p;
        list->last->next=NULL;
        draw=TRUE;
        cursor_off();

        n2.x=m.x;
        n2.y=m.y;
        nl.x=list->last->c.x;
        nl.y=list->last->c.y;

        ShowGrid(0,0,sheet->length*sheet->gridsize,
            sheet->width*sheet->gridsize,sheet->gridsize);
        setcolor(LIGHTGREEN);
        setlinestyle(SOLID_LINE,0,1);
        rectangle(0,0,sheet->length*sheet->gridsize,
            sheet->width*sheet->gridsize);
        setcolor(WHITE);

        RedrawDepartment(1,cg,size,sheet->gridsize,department);
        gprintf(2,44,WHITE,BLACK,"%75s"," ");
        gprintf(2,45,WHITE,BLACK,"%75s"," ");
        for (i=0;i<list->length;i++)
        {

```

```

        getinfo(*list,i,&m);
        gprintf(2+8*(i%9),44+i/9,WHITE,BLACK,"(%2d,%2d)",
            m.x/sheet->gridsize,m.y/sheet->gridsize);
    }

    }
    break;
case 'e': /* End of input department frame */
case 'E':
if(list->length >2)
    if(select==blocknum)
    {
        if((list->first->c.x == list->last->c.x)
            && (list->first->c.y == list->last->c.y))
            message = kALTX;

        else
        {
            gputs(2,46,YELLOW,BLACK,"Unclosed Area Frame!");
            getch();
            gprintf(2,46,WHITE,BLACK,"%70s"," ");
        }
    }
    else
    {
        gputs(2,46,YELLOW,BLACK,
            "Selected area is not equal to department area.");
        getch();
        gprintf(2,46,WHITE,BLACK,"%70s"," ");
    }
    break;

case 'c': /* Clear layout sheet */
case 'C':
    gprintf(2,46,YELLOW,RED,
        "Are you sure to clear layout sheet (Y/N)?");
    do{
        ch=toupper(getch());
    } while( !( (ch=='Y')||(ch=='N')));

    if(ch=='Y') return(999);
    gprintf(2,46,WHITE,BLACK,"%70s"," ");
    break;
}

if(*mx<0) *mx=0;
if(*my<0) *my=0;
if(*mx>sheet->length*sheet->gridsize)
    *mx=sheet->length*sheet->gridsize;
if(*my>sheet->width*sheet->gridsize)
    *my=sheet->width*sheet->gridsize;
gprintf(60,38,WHITE,BLACK,"Position(%2d,%2d)",
    *mx/sheet->gridsize,*my/sheet->gridsize);
}

/***** Draw Poly *****/
if(draw==TRUE)
{
    setcolor(WHITE);
    drawp(*list);

    if((list->first->c.x != list->last->c.x)
        &&(list->first->c.y != list->last->c.y))
    {
        setlinestyle(DOTTED_LINE,0,1);
        cursor_off();
    }
}

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ไม่ควรเผยแพร่โดยไม่ได้รับอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```

        RubberPoly(list->first->c.x,list->first->c.y,
                    n2.x,n2.y,0,topsave);
        n2.x=list->last->c.x;
        n2.y=list->last->c.y;
        RubberPoly(list->first->c.x,list->first->c.y,
                    n2.x ,n2.y,1,topsave);
    }
    select = CalSelectPoly(*list);
    gprintf(2,41,WHITE,BLACK,"Selected Node(s) = %5d",list->length);
    gprintf(2,42,YELLOW,BLACK,"Selected Blocks = %5d",select);
}

}while(message!=kALTX);

cursor_off();
return(0);
}

```

```

/*-----
int CalSelectPoly(LIST list)
: Calculate area of selected department frame
-----*/

```

```

int CalSelectPoly(LIST list)
{
    int total,i;
    CORDINATE n,m;

    if(list.length<3)
        return(0);
    total=0;
    for(i=0;i<list.length-1;i++)
    {
        getinfo(list,i,&n);
        getinfo(list,i+1,&m);
        total += (n.x*(m.y-n.y) - n.y*(m.x-n.x));
    }
    n.x=list.first->c.x;
    n.y=list.last->c.y;

    total += ( list.last->c.x * (n.y - list.last->c.y)
              - list.last->c.y * (n.x - list.last->c.x) );
    total += ( n.x * (list.first->c.y - n.y)
              - n.y * (list.first->c.x - n.x) );
    total = abs(total/200);

    return(total);
}

```

```

/*-----
void drawp(LIST list)
: Draw lines among node in list
-----*/

```

```

void drawp(LIST list)
{
    int i,j,k;
    CORDINATE n,m;

```

```

    cursor_off();
    if(list.length==1)
        putpixel(list.first->c.x,list.first->c.y,YELLOW);
    for(i=0;i<list.length-1;i++)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ

```

    {
        getinfo(list,i,&n);
        getinfo(list,i+1,&m);

        //setcolor(WHITE);
        setlinestyle(SOLID_LINE,0,1);
        line(n.x,n.y,m.x,m.y);
    }

}

/*-----*/
void RubberPoly(int xl,int yl,int x2,int y2,
                int drawflag,void *topSave)
: draw line that overlapped the old screen
  but not change anything in old screen
/*-----*/

void RubberPoly(int xl,int yl,int x2,int y2,int drawflag,void *topSave)
{
    int sx,fx;
    int value;

    if (xl>x2) {sx=x2;fx=x1;} else {sx=x1;fx=x2;}

    if (drawflag) /* draw flag == 1*/
    { /* must use a two pixel high rectangle */
        getimage(sx,y2,fx,y2+1,topSave);
        been_saved = 1;
        line(x2,y2,x1,y2);

        outport(0x3CE,0x3); /* select function register */
        outport(0x3CF,0x18); /* set it to XOR */
        line(x1,yl,x1,y2); /* then draw the line */
        outport(0x3CE,0x3); /* set it back to REPLACE */
        outport(0x3CF,0x0);

    }
    else {

        outport(0x3CE,0x3); /* select function register */
        outport(0x3CF,0x18); /* set it to XOR */
        line(x1,yl,x1,y2); /* then draw the line */
        outport(0x3CE,0x3); /* set it back to REPLACE */
        outport(0x3CF,0x0);

        if(been_saved)
            putimage(sx,y2,topSave,COPY_PUT);

        been_saved=0;
    }
}

/*-----*/
void RedrawDepartment(LIST *lt,COORDINATE *cg,
                     int size,int gridsize,int department)
: Re-draw department frame
/*-----*/

void RedrawDepartment(LIST *lt,COORDINATE *cg,
                     int size,int gridsize,int department)
{
    int i,j;
    LIST *p;
    char string[12];

```

```

for(i=0;i<size;i++)
{
p=lt+i;
if(i!=department)
    if(p->length != 0)
    {
        setcolor(YELLOW);
        drawp(*p);

        setfillstyle(SOLID_FILL,BLACK);
        floodfill(cg[i].x*gridsize,cg[i].y*gridsize,YELLOW);

        itoa(i+1,string,10);
        setcolor(YELLOW);
        outtextxy(cg[i].x*gridsize,cg[i].y*gridsize,string);
        setcolor(WHITE);
    }
}
}

```

```

/*-----
void OutputToTextFile(SHEET sheet,int *blocks,COORDINATE *cg,
    DEPT *dpt,REL *rel,FROMTO *ft,float eff)
: Make report file after layout finished.
-----*/

```

```

void OutputToTextFile(SHEET sheet,int *blocks,COORDINATE *cg,
    DEPT *dpt,REL *rel,FROMTO *ft,float eff)
{
    FILE *fp;
    char str[13],fn[13];
    int i,j;
    int size,value;
    int key;
    int loop;

    size=dpt->size;
    do{
        ClearBox();
        gputs(2,38,CYAN,BLACK,"OUTPUT TO TEXT FILE ...");

        gputs(2,40,WHITE,BLACK,"Enter only filename (*.LAY) : ");
        gGetStr(35,40,8,str);
        strcpy(fn,str);
        strcat(fn,".LAY");

        fp = fopen(fn,"rt");

        if(fp != NULL)
        {
            gprintf(2,42,YELLOW,BLACK,"File %-12s is already existed !",fn);
            gputs(2,43,WHITE,BLACK,"Overwrite ? Yes/No ");

            do{
                key=toupper(getche());
            } while( !(key=='Y' || key=='N'));

            if(key=='Y')
            {
                gputs(20,43,WHITE,BLACK,"Yes");
                loop=FALSE;
            }
            else
            {

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น ยกเว้นหากไม่เกิดผลและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        gputs(20,43,WHITE,BLACK,"No");
        loop=TRUE;
        fclose(fp);
    }
else loop=FALSE;

    }while(loop==TRUE);

fclose(fp);

if((fp=fopen(fn,"wt"))== NULL)
{
    gputs(2,47,YELLOW,BLACK,"Cannot open output file.");
    return;
}
fprintf(fp,
"
fprintf(fp,
" PLANT LAYOUT FILE  [ Filename : '%s' ]\n",fn);
fprintf(fp,
"
fprintf(fp,"\n[Layout]\n\n");
fprintf(fp," ");

for(i=0;i<sheet.length;i++)
{
    fprintf(fp,"%c",'0'+(i+1)%10);
}
fprintf(fp,"\n");

for(j=0;j<sheet.width;j++)
{
    fprintf(fp,"%2c",'0'+(j+1)%10);

    for(i=0;i<sheet.length;i++)
        fprintf(fp,"%c",'A'+ *(sheet.map +sheet.length*j +i));

    fprintf(fp,"\n");
}
fprintf(fp,"\n[Efficiency]\n");
fprintf(fp,"Efficiency = %5.2f\n",eff);

fprintf(fp,"\n[Department] Filename : '%s' \n",dpt->filename);
fprintf(fp,"Area per block = %5d\n",dpt->AreaPerBlock);
fprintf(fp,"Total area      = %5d\n",dpt->total);
fprintf(fp,
"
fprintf(fp,
" Department          Name          Area          Blocks          CG \n");
fprintf(fp,
"
for(i=0;i<size;i++)
fprintf(fp,"      %2d=[%c]      %-21s %5d          %5d          (%2d,%2d)\n",
i+1,'A'+i,dpt->pArea[i].dName,dpt->pArea[i].dArea,blocks[i],cg[i].x,cg[i].y);
fprintf(fp,
"
fprintf(fp,"\n[Relation Chart] Filename : '%s'\n",rel->filename);
fprintf(fp," ");
for(i=2;i<=size;i++)
    fprintf(fp," %2d ",i);
fprintf(fp,"\n");
fprintf(fp,"_____");
for(i=2;i<=size;i++)
    fprintf(fp,"_____");

```

เอกสารนี้เป็นเอกสารสำหรับการใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ควรนำเอกสารนี้ไปใช้เพื่อการค้า หากมีให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

fprintf(fp, "\n");

for(j=1; j<=size-1; j++)
{
    fprintf(fp, "%-3d|", j);
    for(i=2; i<=size; i++)
    {
        if(i>j)
        {
            value = GetRELValue(*rel, i, j);
            fprintf(fp, " %c ", value);
        }
        else
            fprintf(fp, "███");
    }
    fprintf(fp, "\n");
}

fprintf(fp, "\n[From-To Chart] Filename : '%s'\n", ft->filename);
fprintf(fp, " ");
for(i=1; i<=size; i++)
    fprintf(fp, " %2d ", i);
fprintf(fp, "\n");
fprintf(fp, "——");

for(i=1; i<=size; i++)
    fprintf(fp, "——");
fprintf(fp, "\n");

for(j=1; j<=size; j++)
{
    fprintf(fp, "%-3d|", j);
    for(i=1; i<=size; i++)
    {
        if(i!=j)
        {
            value = GetFTValue(*ft, i, j);
            fprintf(fp, " %5d ", value);
        }
        else
            fprintf(fp, "███");
    }
}

fprintf(fp, "\n");

fprintf(fp, "\n[Distance among department]\n");
fprintf(fp, " ");
for(i=1; i<=size; i++)
    fprintf(fp, " %3d ", i);
fprintf(fp, "\n");
fprintf(fp, "——");

for(i=1; i<=size; i++)
    fprintf(fp, "——");
fprintf(fp, "\n");

for(j=1; j<=size; j++)
{
    fprintf(fp, "%-3d|", j);
    for(i=1; i<=size; i++)
    {
        value = (abs(cg[i-1].x-cg[j-1].x) +
        abs(cg[i-1].y-cg[j-1].y))/10; /* 10 is gridSize */
        fprintf(fp, " %3d ", value);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ห้ามทำซ้ำหรือดัดแปลงโดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }

    fprintf(fp, "\n");
}

fprintf(fp, "\n");
fprintf(fp,
"-----\n
fprintf(fp, "End of Layout text file.\n");
fprintf(fp,
"-----\n

fclose(fp);

gputs(2,44,WHITE,BLACK,
    "Output to text file...OK! Press anykey to menu... ");
getch();
}

/*-----
void main()
    : MAIN!
-----*/

void main()
{
    setcursor(CURSOROFF);
    MainDesktop();
    Comment(" Press ESC or F10 for menu");
    Plantlay();
    textattr(F_LIGHTGRAY;B_BLACK);
    setcursor(CURSORON);
    clrscr();
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



LISTING PROGRAM

FROMTO.H

FROMTO.C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* =====
File Name : FROMTO.H
===== */

/***** From to STRUCT *****/

typedef struct {
    char filename[13]; /* Filename */
    int size; /* size */
    int minX; /* Range to display values */
    int maxX; /* (minX,minY)-(maxX,maxY) */
    int minY;
    int maxY;
    int cellX; /* Cell pointer (cellX,cellY) */
    int cellY;
    int far *pFromto; /* Pointer to memory */
} FROMTO;

/***** Define CONSTANT *****/

/* COLOR */
#define NORMALTXT F_WHITE|B_MAGENTA /* Normal Text Color */
#define HILITE F_RED|B_LIGHTGRAY /* Hilite Text Color */
#define BARCOLOR F_BLUE|B_LIGHTGRAY /* Bar Text Color */
#define CELLCOLOR F_YELLOW|B_BLACK|BLINK /* Cell Text Color */
#define UNUSED_TXT F_WHITE|B_BLUE /* 'UNUSED' Text Color */
#define ERRORBORDER F_WHITE|B_RED /* Error Window Color */
#define ERRORTXT F_YELLOW|B_RED
#define FTBOR F_YELLOW|B_MAGENTA /* Fromto Chart window color */
#define FTTXT F_WHITE|B_MAGENTA
#define MSGWINBORDER F_YELLOW|B_RED /* Message window */
#define MSGWINTXT F_WHITE|B_RED

/* < MENU > */
#define FTMENUTXT F_BLACK|B_CYAN /* Menu's Text Color */
#define FTMENUHOT F_LIGHTRED|B_CYAN /* Menu's Hotkey Color */
#define FTMENUBORDER F_WHITE|B_CYAN /* Menu's Border Color */
#define FTMENUBAR F_WHITE|B_BLACK /* Menu's Bar Color */

/* Command : use to command function FromToChart() */
#define cNEWFT 500
#define cEDITFT 501

/***** Function Prototypes *****/
int *InitMemoryFT(int ft_size);
void FreeMemoryFT(int far *pfromto);
void SetRangeFT(FROMTO *fromto,int minx,int miny,int maxx,int maxy);
void ShowBarFT(FROMTO *fromto,int StartCol,int StartRow);
void ShowAllValueFT(FROMTO fromto);
void ShowValueXYFT(FROMTO fromto,int x,int y);
void ShowInfoFT(FROMTO fromto);
void MoveCellFT(FROMTO *fromto,int direction);
void DisplayValueFT(FROMTO fromto);
void InputValueFT(FROMTO *fromto,int message);
int GetFTValue(FROMTO fromto,int x,int y);
void ChangeFTValue(int value,FROMTO *fromto,int x,int y);
void LoadFT(FROMTO *fromto);
void SaveFT(FROMTO *fromto);
void NewFT(FROMTO *fromto);
void FromToChart(FROMTO *ft,int command);
void ClearFTValue(FROMTO *fromto);

```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*
/*      Program FROMTO.C : Make From-To chart and save into disk      */
/*
/* ===== */

```

```

/***** Include Orthers File *****/

```

```

#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <dos.h>

```

```

#include "win.h"
#include "fromto.h"

```

```

/***** Define Constant Value *****/

```

```

/* ORIGIN X,Y */
#define STARTX 1      /* Start X of Bar */
#define STARTY 4      /* Start Y of Bar */

```

```

/***** Variable Declaration *****/

```

```

int cell_size=8;      /* Cell Size */

```

```

/***** From-To Chart Functions *****/

```

```

/*-----
int *InitMemoryFT(int ft_size)
: Allocates ft_size*ft_size bytes of memory
  to keep value of From-To Chart.
  Returns address if successful.
  If unsuccessful, then shows message and returns. NULL.
-----*/

```

```

int *InitMemoryFT(int ft_size)
{
    WIN error = { 15,10,65,14," ERROR ",SINGLE};
    int far *ft;

    ft = farcalloc(ft_size*ft_size,sizeof(int));
    if(ft==NULL) /* Error */
    {
        setcursor(CURSOROFF);
        ShowWin(&error,ERRORBORDER,ERRORTXT); /* Show Window */
        gotoxy(5,2);
        cputs("From-To Chart Error : Not enough memory !");
        getch();
        EraseWin(&error); /* Erase Window */
    }
    return(ft);
}

```

```

/*-----
void FreeMemoryFT(int *pfromto)
: Free From-To Chart memory
-----*/

```

```

void FreeMemoryFT(int far *pfromto)
{
    farfree(pfromto);
}

```

```

/*-----*/
void SetRangeFT(FROMTO *fromto,int minx,int miny,int maxx,int maxy)
: Set From-To Chart display range minX,minY,maxX,maxY to the
minx,miny,maxx,maxy value.
/*-----*/

void SetRangeFT(FROMTO *fromto,int minx,int miny,int maxx,int maxy)
{
    fromto->minX=minx;
    fromto->minY=miny;
    fromto->maxX=maxx;
    fromto->maxY=maxy;
}

/*-----*/
void ShowBarFT(FROMTO *fromto,int StartCol,int StartRow)
: Shows the From-To Chart's bar at STARTX column and STARTY row
StartCol= start No. of departments by X-axis.
StartRow= start No. of departments by Y-axis.
! Updates the display ranges minX,minY,maxX,maxY of the *fromto.
/*-----*/

void ShowBarFT(FROMTO *fromto,int StartCol,int StartRow)
{
    int i; /* Counter */
    int maxx,maxy; /* max of x and y */
    int ft_size; /* size of From-To Chart */

    window(1,1,80,25);

    /***** Draw Bar *****/

    ft_size = fromto->size; /* size */

    /***** Draw Horizontal Bar *****/

    Fastputs(STARTX,STARTY,HILITE,"FROM-TO ");

    for(i=StartCol;i<=ft_size;i++)
    {
        /* check limit of 80 columns */
        if( (i-StartCol) * cell_size +10 < 80)
        {
            /* print number from StartCol ...end of column */
            Fastprintf(STARTX +(i-StartCol+1)*cell_size,STARTY,BARCOLOR,
                " %2d ",i);
        }
        else break;
    }
    maxx= --i; /* Keep Range of X-axis */

    /***** Draw Vertical Bar *****/

    for(i=StartRow;i<=ft_size;i++)
    {
        /* check limit of 25 rows */
        if( (i-StartRow)+STARTY+1 < 24)
        {
            Fastprintf(STARTX,STARTY+1+(i-StartRow),BARCOLOR," %2d ",i);
        }
        else break;
    }
    maxy= --i; /* Keep Range Y-axis */

    gotoxy(1,1);
}

```

เอกสารนี้เป็นเอกสารตัวอย่าง ไม่ควรนำออกเผยแพร่โดยไม่ได้รับอนุญาต
 ไม่ว่า gotoxy(1,1); ก็ทั้งห้ามมิให้ดัดแปลงเนื้อเรื่อง Go Home * / เจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SetRangeFT(fromto,StartCol,StartRow,maxx,maxy); /* Call SetRangeFT */
textattr(NORMALTXT);
}
```

```
/*-----
void ShowAllValueFT(FROMTO fromto)
: Show all the value that in the range of
(minX,minY)-(maxX,maxY).
-----*/
```

```
void ShowAllValueFT(FROMTO fromto)
{
    int a,b;
    int value;

    for(a=fromto.minX;a<=fromto.maxX;a++)
    for(b=fromto.minY;b<=fromto.maxY;b++)
    {
        if(a==b) /* print unused */
        {
            Fastputs((a-fromto.minX+1)*cell_size+STARTX,
                (b-fromto.minY+1)+STARTY,NORMALTXT," UNUSED");
        }
        else /* print value */
        {
            value = GetFTValue(fromto,a,b); /* Get valuat at (a,b) */
            Fastprintf((a-fromto.minX+1)*cell_size+STARTX,
                (b-fromto.minY+1)+STARTY,NORMALTXT,"%8d",value);
        }
    }
}
```

```
/*-----
void ShowValueXYFT(FROMTO fromto,int x,int y)
: Show value at X,Y on From-To chart
-----*/
```

```
void ShowValueXYFT(FROMTO fromto,int x,int y)
{
    int value;

    /* check range */
    if( (x<fromto.minX) || (x>fromto.maxX)) return;
    if( (y<fromto.minY) || (y>fromto.maxY)) return;

    gotoxy((x-fromto.minX+1)*cell_size+STARTX,(y-fromto.minY+1)+STARTY);
    if(x==y) /* print unused */
    {
        cputs(" UNUSED");
    }
    else /* print value */
    {
        value = *( fromto.pFromto + fromto.size*(y-1) + (x-1) );
        cprintf("%8d",value);
        textattr(NORMALTXT);
    }
}
```

```
/*-----
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่า void ShowInfoFT(FROMTO fromto)
: Show the statement at the 25th row.
-----*/
```

```

void ShowInfoFT(FROMTO fromto)
{
    char string[10];

    itoa(fromto.size,string,10);    /* From-To chart filename */

    Fastprintf(1,25,COMMENTCOLOR,
    " From-To Chart | %12s | SIZE %3s * %-3s |          ESC:Exit",
    fromto.filename,string,string);
}

```

```

/*-----
void MoveCellFT(FROMTO *fromto,int direction)
    : Move cell from current position to left,right,up,down,home
    by direction :- kLEFT,kRIGHT,kUP,kDOWN,kHOME
    and update range of display.
-----*/

```

```

void MoveCellFT(FROMTO *fromto,int direction)
{
    int x,y;

    switch(direction)
    {
    case kHOME :    /* Goto x=1,y=1 */
        fromto->cellX = 1;
        fromto->cellY = 1;
        ShowBarFT(fromto,1,1);    /* Change Bar */
        ShowAllValueFT(*fromto);    /* Display all */
        textattr(CELLCOLOR);
        ShowValueXYFT(*fromto,1,1);    /* Display Cell */
        break;
    /* Move right */
    case kRIGHT :
        x = fromto->cellX +1;    /* Increase x */
        y = fromto->cellY ;
        if ( x > fromto->maxX)    /* Check Range-X */
        {    /* Out of Range-X */
            if(x <= fromto->size){
                ++fromto->minX;
                ShowBarFT(fromto,fromto->minX,fromto->minY);
                ShowAllValueFT(*fromto);
                textattr(CELLCOLOR);
                ShowValueXYFT(*fromto,x,y);
                fromto->cellX = x;
            }
            else {
                Beep();
            }
        }
        else    /* In the Range-X */
        {
            textattr(NORMALTXT);
            ShowValueXYFT(*fromto,x-1,y);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto,x,y);
            fromto->cellX = x;
        }
        break;
    /* Move left */
    case kLEFT :
        x = fromto->cellX -1;    /* Decrease x */
        y = fromto->cellY ;
        if ( x < fromto->minX)    /* Check Range-X */
        {    /* Out of Range-X */
            if(x >= 1){
                --fromto->minX;
                ShowBarFT(fromto,fromto->minX,fromto->minY);
                ShowAllValueFT(*fromto);
                textattr(CELLCOLOR);
                ShowValueXYFT(*fromto,x,y);
                fromto->cellX = x;
            }
            else {
                Beep();
            }
        }
        else    /* In the Range-X */
        {
            textattr(NORMALTXT);
            ShowValueXYFT(*fromto,x+1,y);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto,x,y);
            fromto->cellX = x;
        }
        break;
    }
}

```

เอกสารนี้เป็นส่วนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ที่สืบย้อนกลับไปในเอกสารและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    {
        --fromto->minX;
        ShowBarFT(fromto, fromto->minX, fromto->minY);
        ShowAllValueFT(*fromto);
        textattr(CELLCOLOR);
        ShowValueXYFT(*fromto, x, y);
        fromto->cellX = x;
    }
    else
        Beep();
}
else /* In the Range-X */
{
    textattr(NORMALTXT);
    ShowValueXYFT(*fromto, x+1, y);
    textattr(CELLCOLOR);
    ShowValueXYFT(*fromto, x, y);
    fromto->cellX = x;
}
break;
/* Move Down */
case kDOWN :
    x = fromto->cellX ;
    y = fromto->cellY +1; /* Increase Y */
    if ( y > fromto->maxY) /* Check Range Y */
    { /* Out of range Y */
        if(y <= fromto->size)
        {
            ++fromto->minY;
            ShowBarFT(fromto, fromto->minX, fromto->minY);
            ShowAllValueFT(*fromto);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto, x, y);
            fromto->cellY = y;
        }
        else
            Beep();
    }
    else /* In the Range-y */
    {
        textattr(NORMALTXT);
        ShowValueXYFT(*fromto, x, y-1);
        textattr(CELLCOLOR);
        ShowValueXYFT(*fromto, x, y);
        fromto->cellY = y;
    }
    break;
/* Move up */
case kUP :
    x = fromto->cellX ;
    y = fromto->cellY -1; /* Decrease y */
    if ( y < fromto->minY) /* Check range Y */
    { /* Out of Range-Y */
        if (y >= 1)
        {
            --fromto->minY;
            ShowBarFT(fromto, fromto->minX, fromto->minY);
            ShowAllValueFT(*fromto);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto, x, y);
            fromto->cellY = y;
        }
        else
            Beep();
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีให้คัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        else
        {
            textattr(NORMALTXT);
            ShowValueXYFT(*fromto,x,y+1);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto,x,y);
            fromto->cellY = y;
        }
        break;
    } /* end of switch */
    textattr(NORMALTXT);
}

/*-----
void DisplayValueFT(FROMTO fromto)
: Display statement and value at the 2nd row.
-----*/

void DisplayValueFT(FROMTO fromto)
{
    int value;

    gotoxy(1,2);
    textattr(NORMALTXT);

    if(fromto.cellX==fromto.cellY)
    {
        cprintf("FROM [%2d] TO [%2d] : UNUSED      ",fromto.cellY,
            fromto.cellX);
    }
    else
    {
        value = GetFTValue(fromto,fromto.cellX,fromto.cellY);/* get value */
        cprintf("FROM [%2d] TO [%2d] : %d      ",fromto.cellY,
            fromto.cellX,value);
    }
}

/*-----
void InputValueFT(FROMTO *fromto,int message)
: Input value at the 2nd row
  Change value at (cellx,cellY) of From-To Chart
  Press ESC to cancel the change.
-----*/

void InputValueFT(FROMTO *fromto,int message)
{
    int ch;
    int a;
    int count,loop;
    char string[6];
    int value;

    strcpy(string,"      "); /* Init string */

    gotoxy(21,2);
    cputs("      "); /* clear display */
    gotoxy(21,2);
    setcursor(CURSORON); /* cursor on */

    if(message !=kCR) /* if != ENTER */
    {
        cprintf("%c",message);
        count = 1;
        string[0]=message;
    }
}

```

```

}
else
    count = 0;

loop=TRUE;

do{
    ch = InputKey();
    switch(ch)
    {
        case '0': /* if == '0'...'9' */
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
            if(count<5)
            {
                /* Check limit */
                string[count] = ch;
                count++;
                gotoxy(21,2);
                cprintf("%s",string);
                gotoxy(21+count,2);
            }
            else
            {
                Beep();
            }
            break;

        case KBSP : /* Back Space */
            count--;
            if(count <0)
            {
                count=0;
                Beep();
            }
            for(a=count;a<5;a++)
                string[a]=string[a+1];
            gotoxy(21,2);
            cprintf("%s",string);
            gotoxy(21+count,2);
            break;

        case KCR :
            string[6] = '\0';
            ChangeFTValue(atoi(string),fromto,
                fromto->cellX,fromto->cellY);
            setcursor(CURSROFF);
            textattr(CELLCOLOR);
            ShowValueXYFT(*fromto,fromto->cellX,
                fromto->cellY);
            loop=FALSE;
            break;

        case KESC: loop=FALSE;
            setcursor(CURSROFF);
            break;
    }
}

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
 ไม่ควรนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจากมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
 ไม่ควรนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจากมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

/*-----

```
int GetFTValue(FROMTO fromto,int x,int y)
: Get value from From-To chart at (x,y)
```

-----*/

```
int GetFTValue(FROMTO fromto,int x,int y)
{
    int value;

    /* check range */

    value = *( fromto.pFromto + fromto.size*(y-1) + (x-1) );
    return(value);
}
```

```
/*-----
void ChangeFTValue(int value,FROMTO *fromto,int x,int y)
: Change the value at (x,y) of From-To Chart.
```

-----*/

```
void ChangeFTValue(int value,FROMTO *fromto,int x,int y)
{
    /* check range */
    if( (x<fromto->minX) || (x>fromto->maxX)) return;
    if( (y<fromto->minY) || (y>fromto->maxY)) return;

    *( fromto->pFromto + fromto->size*(y-1) + (x-1) ) = value;
}
```

```
/*-----
void LoadFT(FROMTO *fromto)
: Load From-To chart from disk, then put into memory.
```

-----*/

```
void LoadFT(FROMTO *fromto)
{
    FILE *fpLoad;

    WIN load = {20,7,60,9," LOAD From-To Chart ",SINGLE};
    WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
    char name[9],oldname[13],key;
    int value,length,loop,x,y;
    long chksum,sum;
    int m,type;

    /***** Input Filename *****/
    loop=TRUE;
    do{
        ShowWin(&load,FTBOR,FTTXT);

        setcursor(CURSORON);

        cputs(" File Name : ");          /* Filename */
        m=getstr(9,name);

        if(m==kESC)                      /* Press ESC to exit */
        {
            EraseWin(&load);
            return;
        }

        strcpy(oldname,fromto->filename); /* add file extension */
        strcpy(fromto->filename,name);
        strcat(fromto->filename,".FTC");

        ShowWin(&msg,MSGWINBORDER,MSGWINTXT);
```

```

fpLoad = fopen(fromto->filename,"rb"); /* Check the exist file */

if(fpLoad != NULL) /* file is existed */
{
    loop=FALSE;
}
else /* file isn't existed */
{
    /* Can't Open file */
    gotoxy(10,2);
    cprintf("File %-12s not found !",fromto->filename);
    getch();
    EraseWin(&msg);
    EraseWin(&load);
    strcpy(fromto->filename,oldname);
}
}while(loop==TRUE);

/***** Load *****/
clrscr();
gotoxy(10,2);
cputs(" Loading File ... ");

fread(&type,sizeof(int),1,fpLoad); /* Read type of file */

if (type != 1) /* Check Type (First byte) */
{
    fclose(fpLoad);
    clrscr();
    gotoxy(6,2);
    cprintf("%-12s is not a From-To Chart file !",
        fromto->filename);
    getch();
    EraseWin(&msg);
    EraseWin(&load);
    strcpy(fromto->filename,oldname);
    return;
}

fread(&length,sizeof(int),1,fpLoad); /* read size of From-To chart */
fromto->size=length; /* update size */

FreeMemoryFT(fromto->pFromto); /* Clear old memory */
fromto->pFromto = InitMemoryFT(length); /* allocate new memory */

sum = 0; /* checksum */

for(y=1;y<=length;y++) /* read value */
for(x=1;x<=length;x++)
{
    fread(&value,sizeof(int),1,fpLoad);
    sum = sum + value;
    *( fromto->pFromto + fromto->size*(y-1) + (x-1) ) = value;
}

fread(&chksum,sizeof(long),1,fpLoad); /* read checksum */
fclose(fpLoad);

if(chksum==sum)
    cputs("Completed !"); /* OK! checksum==sum */
else /* ERROR! */
{
    clrscr();
    gotoxy(10,2);
    cprintf("Data Error in file %-12s !",fromto->filename);
}
getch();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ หากมีให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
EraseWin(&msg);
EraseWin(&load);
setcursor(CURSROFF);
```

```
}
```

```
/*-----
void SaveFT(FROMTO *fromto)
: Save From-To chart from memory into disk.
-----*/
```

```
void SaveFT(FROMTO *fromto)
```

```
{
```

```
FILE *fpSave;
```

```
WIN save = {20,7,60,9," SAVE From-To Chart ",SINGLE};
```

```
WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
```

```
char name[9],key;
```

```
int value,length,loop,x,y;
```

```
long chksum;
```

```
int m,type;
```

```
/* Check there are values in Fromto chart? */
```

```
if((fromto->size <2)!!(fromto->size>30))
```

```
{
```

```
Beep();
```

```
ShowWin(&msg,MSGWINBORDER,MSGWINTXT);
```

```
gotoxy(5,2);
```

```
cputs(" From-To Chart hasn't been loaded !");
```

```
getch();
```

```
EraseWin(&msg);
```

```
return;
```

```
}
```

```
type=1; /* set type =1 */
```

```
/****** Input Filename *****/
```

```
loop=TRUE;
```

```
do{
```

```
ShowWin(&save,FTBOR,FTTXT);
```

```
setcursor(CURSROFF);
```

```
clrscr();
```

```
cputs(" File Name : ");
```

```
m=getstr(9,name);
```

```
if((m==kESC))
```

```
{
```

```
EraseWin(&save);
```

```
return;
```

```
}
```

```
strcpy(fromto->filename,name);
```

```
/* Add file extension */
```

```
strcat(fromto->filename,".FTC");
```

```
/* *.FTC */
```

```
fpSave = fopen(fromto->filename,"rb"); /* check existed file */
```

```
ShowWin(&msg,MSGWINBORDER,MSGWINTXT);
```

```
if(fpSave != NULL)
```

```
/* File is existed */
```

```
{
```

```
gotoxy(6,1);
```

```
cprintf("File %-12s is already existed!",
```

```
fromto->filename);
```

```
gotoxy(15,2);
```

```
cputs("Overwrite (Y/N) ? ");
```

```
/* Overwrite? */
```

เอกสารนี้เป็นเอกสารของกรมส่งเสริมการค้าระหว่างประเทศ กระทรวงพาณิชย์
 ไม่ว่ากรณีใดๆ ทั้งสิ้น ลิขสิทธิ์เป็นของกรมส่งเสริมการค้าระหว่างประเทศ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

do{
    key=toupper(getche());
}while( !(key=='Y' || key=='N'));

if(key=='Y')                                /* Yes */
{
    loop=FALSE;
    fclose(fpSave);
}
else                                        /* No */
{
    loop=TRUE;
    fclose(fpSave);
    EraseWin(&msg);
    EraseWin(&save);
}
}
else loop=FALSE;

}while(loop==TRUE);

/***** Save *****/
clrscr();
gotoxy(10,2);
cputs(" Saving File ... ");

fpSave = fopen(fromto->filename,"wb");      /* Open file */

fwrite(&type, sizeof(int), 1, fpSave);      /* Write type */

length = fromto->size;
fwrite(&length, sizeof(int), 1, fpSave);    /* Write Length */

chksum = 0;                                /* clear checksum */

for(y=1;y<=length;y++)                    /* Write value */
for(x=1;x<=length;x++)
{
    value = *( fromto->pFromto + fromto->size*(y-1) + (x-1) );
    chksum = chksum + value;
    fwrite(&value, sizeof(int), 1, fpSave);
}
    fwrite(&chksum, sizeof(long), 1, fpSave); /* Write checksum */

fclose(fpSave);                            /* Close file */

cputs("Completed !");
getch();
EraseWin(&msg);
EraseWin(&save);
setcursor(CURSROFF);
}

-----
void NewFT(FROMTO *fromto)
    : Make new From-To Chart
-----*/

void NewFT(FROMTO *fromto)
{
    WIN new = {20,7,60,9," NEW From-To Chart ",SINGLE};
    int value;
    int *v,a,b;
    FreeMemoryFT(fromto->pFromto);          /* Free old memory */

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่ให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าในรูปแบบใด ๆ และหากมีการนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจากมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
 จะถือว่าผิดกฎหมายและต้องรับผิดชอบต่อเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/ ***** Input Department number *****/
do{
    ShowWin(&new,FTBOR,FTTXT);
    gotoxy(5,1);
    cputs("How many departments ? ");
    value=GetInt();
    EraseWin(&new);
}while( (value <2) || (value>30));

strcpy(fromto->filename,"NONAME.FTC");          /* Filename = 'NONAME' */
fromto->size = value;                             /* Size */
fromto->pFromto = InitMemoryFT(value);           /* Allocate memory */

fromto->cellX = 1;                                /* Initial Cell (1,1) */
fromto->cellY = 1;

textattr(NORMALTXT);
setcursor(CURSROFF);
clrscr();
ShowBarFT(fromto,1,1);                          /* Display */
ShowAllValueFT(*fromto);
textattr(CELLCOLOR);
ShowValueXYFT(*fromto,fromto->cellX,fromto->cellY);
}

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*-----
void FromToChart(FROMTO *ft,int command)
    : Main Function of From-To Chart
-----*/

void FromToChart(FROMTO *ft,int command)
{
    WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
    int message;
    int *v,a,b;

    ft->cellX = 1;    /* Initial cell (1,1) */
    ft->cellY = 1;

    switch(command)
    {
    case cNEWFT :    /* if command = NEWFT then NEWFT() */
        NewFT(ft);
        break;

    case cEDITFT :    /* Edit */
        if((ft->size <2)|| (ft->size>30))
        {
            Beep();
            ShowWin(&msg,MSGWINBORDER,MSGWINTXT);
            gotoxy(5,2);
            cputs(" From-To Chart hasn't been loaded ! ");
            getch();
            EraseWin(&msg);
            return;
        }

        textattr(NORMALTXT);
        setcursor(CURSROFF);
        clrscr();
        ShowBarFT(ft,1,1);
        ShowAllValueFT(*ft);
        textattr(CELLCOLOR);
        ShowValueXYFT(*ft,ft->cellX,ft->cellY);
        ShowInfoFT(*ft);
        break;

```

```

}

/***** Input key to move cell and edit value *****/
do{
    ShowInfoFT(*ft);
    DisplayValueFT(*ft);
    message = InputKey();
    switch(message)
    {
        case kLEFT : MoveCellFT(ft,kLEFT);break;
        case kRIGHT : MoveCellFT(ft,kRIGHT);break;
        case kUP : MoveCellFT(ft,kUP);break;
        case kDOWN : MoveCellFT(ft,kDOWN);break;
        case kHOME : MoveCellFT(ft,kHOME);break;
        case '0' :
        case '1' :
        case '2' :
        case '3' :
        case '4' :
        case '5' :
        case '6' :
        case '7' :
        case '8' :
        case '9' :
        case kCR :
            if(ft->cellX != ft->cellY)
                InputValueFT(ft,message);
            break;
    }
}while(message != kESC);
}

/*-----
void ClearFTValue(FROMTO *fromto)
: Initialize the FROMTO variable.
-----*/

void ClearFTValue(FROMTO *fromto)
{
    strcpy(fromto->filename,"NONAME.FTC");
    fromto->size = 0;
    fromto->minX = 0;
    fromto->minY = 0;
    fromto->maxX = 0;
    fromto->maxY = 0;
    fromto->cellX = 1;
    fromto->cellY = 1;
    fromto->pFromto = NULL;
}

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/

```



LISTING PROGRAM

RELATION.H

RELATION.C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*                               File Name : RELATION.H                               */
/* ===== */

```

```

/***** Define STRUCT *****/

```

```

typedef struct {
    char    filename[13];    /* Filename */
    int     size;            /* Relation Chart size */
    int     minX;           /* Range to display value */
    int     maxX;           /* (minX,minY)-(maxX,maxY) */
    int     minY;
    int     maxY;
    int     cellX;          /* Cell Pointer (cellX,cellY) */
    int     cellY;
    int far *pRel;          /* Pointer to memory */
} REL;

```

```

/***** Define CONSTANT *****/

```

```

/* COLOR */
#define rNORMALTXT    F_BLACK;B_CYAN        /* Normal Text Color */
#define rHILITE       F_RED;B_LIGHTGRAY     /* Hilite Text Color */
#define rBARCOLOR     F_WHITE;B_BLUE        /* Bar Text Color */
#define rCELLCOLOR    F_BLACK;B_LIGHTGRAY;BLINK /* Cell Text Color */
#define rUNUSED_TXT   F_WHITE;B_BLUE        /* 'UNUSED' Text Color */
#define rERRORBORDER   F_WHITE;B_RED        /* Error Window Color */
#define rERRORTXT     F_YELLOW;B_RED        /* Error Window Color */
#define RELBOR        F_YELLOW;B_CYAN       /* REL Chart Window Color */
#define RELTXT        F_BLACK;B_CYAN
#define rMSGWINBORDER  F_YELLOW;B_RED       /* Message window color */
#define rMSGWINTXT    F_WHITE;B_RED

/* < MENU > */
#define RELMENUTXT    F_BLACK;B_LIGHTGRAY    /* Menu's Text Color */
#define RELMENUHOT    F_LIGHTRED;B_LIGHTGRAY /* Menu's Hotkey Color */
#define RELMENUBORDER  F_WHITE;B_LIGHTGRAY    /* Menu's Border Color */
#define RELMENUBAR    F_WHITE;B_BLACK        /* Menu's Bar Color */

#define ACOLOR        F_LIGHTRED;B_BLACK     /* Character 'A' color */
#define ECOLOR        F_YELLOW;B_BLACK      /* Character 'E' color */
#define ICOLOR        F_LIGHTGREEN;B_BLACK  /* Character 'I' color */
#define OCOLOR        F_LIGHTCYAN;B_BLACK   /* Character 'O' color */
#define UCOLOR        F_WHITE;B_BLACK       /* Character 'U' color */
#define XCOLOR        F_LIGHTGRAY;B_BLACK   /* Character 'X' color */

```

```

/* Command : use to command function RelaionChart() */
#define cNEWREL       500
#define cEDITREL      501

```

```

/* ORIGIN X,Y */
#define STARTX        1    /* Start X of Bar */
#define STARTY        4    /* Start Y of Bar */

```

```

/***** Function Prototypes *****/

```

```

int far *InitMemoryREL(int rel_size);
void FreeMemoryREL(int far *prel);
void SetRangeREL(REL *rel,int minx,int miny,int maxx,int maxy);
void ShowBarREL(REL *rel,int StartCol,int StartRow);
void ShowAllValueREL(REL rel);
void ShowValueXYREL(REL rel,int x,int y,int flag);
void ShowInfoREL(REL rel);
void MoveCellREL(REL *rel,int direction);
void DisplayValueREL(REL rel);

```

```
void InputValueREL(REL *rel,int message);
int CalAddressREL(int n,int x,int y);
int GetRELValue(REL rel,int x,int y);
void ChangeRELValue(int value,REL *rel,int x,int y);
void LoadREL(REL *rel);
void SaveREL(REL *rel);
void NewREL(REL *rel);
void RelationChart(REL *rel,int command);
void ClearRELValue(REL *rel);
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*
/*      Program REALTION.C : Make Relation Chart and save to disk.
/*
/*
/* ===== */

/***** Include File *****/

#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <dos.h>

#include "win.h"
#include "relation.h"

/***** Variable Declaration *****/

int cell_sizeREL=8;          /* Cell Size */

/***** Relation Chart Functions *****/

/*-----
int far *InitMemoryREL(int rel_size)
: Allocates rel_size*rel_size bytes of memory
  to keep value of Relation Chart.
  Returns address if successful.
  If unsuccessfull,then shows message and returns NULL.
-----*/

int far *InitMemoryREL(int rel_size)
{
    int far *rel;
    int size;
    int c;

    size = ((rel_size-1)*(rel_size))/2 ;
    rel = farcalloc( size , sizeof(int)); /* Allocate memory */
    if(rel==NULL) /* Error */
    {
        setcursor(CURSROFF);
        gotoxy(5,2);
        cputs("Relation Chart Error : Not enough memory !");
        getch();
    }
    else {
        for(c=0;c<size;c++)
            *(rel+c) = 'X';
    }
    return(rel);
}

/*-----
void FreeMemoryREL(int far *prel)
: Free Relation Chart memory
-----*/

void FreeMemoryREL(int far *prel)
{
    farfree(prel);
}

```

```

/*-----*/
void SetRangeREL(REL *rel,int minx,int miny,int maxx,int maxy)
: Set Relation Chart display range minX,minY,maxX,maxY to
the minx,miny,maxx,maxy value.
/*-----*/

void SetRangeREL(REL *rel,int minx,int miny,int maxx,int maxy)
{
    rel->minX=minx;
    rel->minY=miny;
    rel->maxX=maxx;
    rel->maxY=maxy;
}

/*-----*/
void ShowBarREL(REL *rel,int StartCol,int StartRow)
: Shows the Relation Chart's bar at STARTX column & STARTY row
StartCol= start No. of departments by X-axis.
StartRow= start No. of departments by Y-axis.
! Updates the display ranges minX,minY,maxX,maxY of the *rel.
/*-----*/

void ShowBarREL(REL *rel,int StartCol,int StartRow)
{
    int i;
    int maxx,maxy;                /* max of x and y */
    int rel_size;                  /* size of Relation Chart */

    window(1,1,80,25);

    /****** Draw Bar *****/
    rel_size = rel->size;          /* size */

    /****** Draw Horizontal Bar *****/
    Fastputs(STARTX,STARTY,rHILITE,"RELATION");

    for(i=StartCol;i<=rel_size;i++)
    {
        /* check limit of 80 columns */
        if( (i-StartCol) * cell_sizeREL +10 < 80)
        {
            /* print number from StartCol ...end of column */
            Fastprintf(STARTX +(i-StartCol+1)*cell_sizeREL,STARTY,
                rBARCOLOR," %2d ",i);
        }
        else break;
    }
    maxx= --i;                    /* Keep Range of X-axis */

    /****** Draw Vertical Bar *****/
    for(i=StartRow;i<rel_size;i++)
    {
        /* check limit of 25 rows */
        if( (i-StartRow)+STARTY+1 < 24)
        {
            Fastprintf(STARTX,STARTY+1+(i-StartRow),
                rBARCOLOR," %2d ",i);
        }
        else break;
    }
    gotoxy(1,1);                  /* Go Home */

    maxy= --i;                    /* Keep Range Y-axis */
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ควรเผยแพร่ ห้ามนำออกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SetRangeREL(rel, StartCol, StartRow, maxx, maxy); /* Call SetRange */
textattr(rNORMALTXT);
}

/*-----
void ShowAllValueREL(REL rel)
: Show all the value that in the range of
(minX,minY)-(maxX,maxY).
-----*/

void ShowAllValueREL(REL rel)
{
    int a,b,c;
    int value,color;

    for(a=rel.minX;a<=rel.maxX;a++)
    for(b=rel.minY;b<=rel.maxY;b++)
    {

        if(a>b)
        {
            value = GetRELValue(rel,a,b); /* Get valut at (a,b) */
            switch(value)
            {
                case 'A': color=ACOLOR;break;
                case 'E': color=ECOLOR;break;
                case 'I': color=ICOLOR;break;
                case 'O': color=OCOLOR;break;
                case 'U': color=UCOLOR;break;
                case 'X': color=XCOLOR;break;
            }
            Fastprintf((a-rel.minX+1)*cell_sizeREL+STARTX,
                (b-rel.minY+1)+STARTY,color," %2c ",value);
        }
        else
        {
            Fastputs((a-rel.minX+1)*cell_sizeREL+STARTX,
                (b-rel.minY+1)+STARTY,rNORMALTXT,"[X]");
        }
    }
    textattr(rNORMALTXT);
}

/*-----
void ShowValueXY(REL rel,int x,int y)
: Show value at X,Y on Relation chart
-----*/

void ShowValueXYREL(REL rel,int x,int y,int flag)
{
    int value;
    int half,c;
    struct text_info text;

    half=(cell_sizeREL-2)/2;
    /* check range */
    if( (x<rel.minX) || (x>rel.maxX)) return;
    if( (y<rel.minY) || (y>rel.maxY)) return;

    gotoxy((x-rel.minX+1)*cell_sizeREL+STARTX,(y-rel.minY+1)+STARTY);
    if(x>y)
    {
        value = GetRELValue(rel,x,y); /* Get value at (a,b) */
        gettextinfo(&text);
    }
    else
    {
        value = GetRELValue(rel,y,x); /* Get value at (a,b) */
        gettextinfo(&text);
    }
    Fastprintf((x-rel.minX+1)*cell_sizeREL+STARTX,
        (y-rel.minY+1)+STARTY,color," %2c ",value);
}

```

```

if(flag == ON)
{
switch(value)
{
case 'A': textattr(ACOLOR);break;
case 'E': textattr(ECOLOR);break;
case 'I': textattr(ICOLOR);break;
case 'O': textattr(OCOLOR);break;
case 'U': textattr(UCOLOR);break;
case 'X': textattr(XCOLOR);break;
}
}
else textattr(rCELLCOLOR);

for(c=1;c<=half;c++) putchar(' ');
cprintf("%2c",value);
for(c=1;c<=half;c++) putchar(' ');

/* cprintf("%8c",value);*/
textattr(rNORMALTXT);
}
else cputs("███");
}

/*-----
void ShowInfoREL(REL rel)
: Show the statement at the 25th row.
-----*/

void ShowInfoREL(REL rel)
{
char string[10];

itoa(rel.size,string,10); /* Filename */

Fastprintf(1,25,COMMENTCOLOR,
" Relation Chart| %12s | SIZE %3s * %-3s |          ESC:Exit",
rel.filename,string,string);
}

/*-----
void MoveCellREL(REL *rel,int direction)
: Move cell from current position to lrel,right,up,down,home
by direction :- kLEFT,kRIGHT,kUP,kDOWN,kHOME
and update range of display.
-----*/

void MoveCellREL(REL *rel,int direction)
{
int x,y;

switch(direction)
{
case kHOME : /* Goto x=1,y=1 */
rel->cellX = 2;
rel->cellY = 1;
ShowBarREL(rel,2,1); /* Change Bar */
ShowAllValueREL(*rel); /* Display all */
ShowValueXYREL(*rel,2,1,OFF); /* Display Cell */
break;
/* Move right */
case kRIGHT :
x = rel->cellX +1; /* Increase x */
y = rel->cellY ;

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
 ไม่ควรนำเอกสารนี้ไปใช้ในการเรียนการสอนโดยไม่ได้รับอนุญาต
 ไม่ควรนำเอกสารนี้ไปใช้ในการทำรายงานหรือสิ่งพิมพ์โดยไม่ได้รับอนุญาต

```

if ( x > rel->maxX)          /* Check Range-X */
{
    /* Out of Range-X */
    if(x <= rel->size)
    {
        ++rel->minX;
        ShowBarREL(rel,rel->minX,rel->minY);
        ShowAllValueREL(*rel);
        ShowValueXYREL(*rel,x,y,OFF);
        rel->cellX = x;
    }
    else {
        Beep();
    }
}
else /* In the Range-X */
{
    ShowValueXYREL(*rel,x-1,y,ON);
    ShowValueXYREL(*rel,x,y,OFF);
    rel->cellX = x;
}
break;
/* Move left */
case kLEFT :
    x = rel->cellX -1;          /* Decrease x */
    y = rel->cellY ;
    if (x>y)
    {
        if ( x < rel->minX)      /* Check Range-X */
        {
            /* Out of Range-X */
            if (x >= 1)
            {
                --rel->minX;
                ShowBarREL(rel,rel->minX,rel->minY);
                ShowAllValueREL(*rel);
                ShowValueXYREL(*rel,x,y,OFF);
                rel->cellX = x;
            }
            else {
                Beep();
            }
        }
        else /* In the Range-X */
        {
            ShowValueXYREL(*rel,x+1,y,ON);
            ShowValueXYREL(*rel,x,y,OFF);
            rel->cellX = x;
        }
    }
    else Beep();
    break;
/* Move Down */
case kDOWN :
    x = rel->cellX ;
    y = rel->cellY +1;          /* Increase Y */

    if (x>y)
    {
        if ( y > rel->maxY)      /* Check Range Y */
        {
            /* Out of range Y */
            if (y <= rel->size)
            {
                ++rel->minY;
                ShowBarREL(rel,rel->minX,rel->minY);
                ShowAllValueREL(*rel);
                ShowValueXYREL(*rel,x,y,OFF);
                rel->cellY = y;
            }
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังขอสงวนสิทธิ์ในเอกสารทุกครั้งที่มีการนำไปใช้

```

        else{
            Beep();
        }
    }
    else
        /* In the Range-y */
        {
            ShowValueXYREL(*rel,x,y-1,ON);
            ShowValueXYREL(*rel,x,y,OFF);
            rel->cellY = y;
        }
    }
    else Beep();
    break;
/* Move up */
case kUP :
    x = rel->cellX ;
    y = rel->cellY -1;
    /* Decrease y */
    if (x>y)
    {
        if ( y < rel->minY)
            /* Check range Y */
            /* Out of Range-Y */
            {
                if(y >= 1)
                {
                    --rel->minY;
                    ShowBarREL(rel,rel->minX,rel->minY);
                    ShowAllValueREL(*rel);
                    ShowValueXYREL(*rel,x,y,OFF);
                    rel->cellY = y;
                }
                else{
                    Beep();
                }
            }
        else
        {
            ShowValueXYREL(*rel,x,y+1,ON);
            ShowValueXYREL(*rel,x,y,OFF);
            rel->cellY = y;
        }
    }
    else Beep();
    break;
} /* switch */
textattr(rNORMALTXT);
},

/*-----
void DisplayValueREL(REL rel)
    : Display statement and value at the 2nd row.
-----*/

void DisplayValueREL(REL rel)
{
    int value;

    gotoxy(1,2);
    textattr(rNORMALTXT);

    if(rel.cellX==rel.cellY)
    {
        cprintf("REL [%2d] AND [%2d]: UNUSED",rel.cellY,rel.cellX);
    }
    else
    {
        value = GetRELValue(rel,rel.cellX,rel.cellY); /* get value */
    }
}

```

```

    cprintf("REL [%2d] AND [%2d] : %c",rel.cellY,rel.cellX,value);
}

}

/*-----
void InputValueREL(REL *rel,int message)
: Input value at the 2nd row
Change value at (cellx,celly) of Relation Chart
Press ESC to cancel the change.
-----*/

void InputValueREL(REL *rel,int message)
{
    int ch;
    int a;
    int count,loop;
    char string[6];
    int value;

    value=0;
    strcpy(string,""); /* Init string */

    gotoxy(21,2);
    cputs(""); /* clear display */
    gotoxy(21,2);
    setcursor(CURSORON); /* cursor on */

    if(message !=kCR) /* if != ENTER */
    {
        switch(message)
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'x':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
            case 'X':
                value =toupper(message);
                cprintf("%c",value);
                count = 1;
                break;
            default : Beep();
                break;
        }
    }
    else
        count = 0;

    loop=TRUE;

    do{
        ch=InputKey();
        switch(ch)
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':

```

เอกสารนี้เป็นส่วนหนึ่งของการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าการแก้ไขหรือเปลี่ยนแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

case 'x':
case 'A':
case 'E':
case 'I':
case 'O':
case 'U':
case 'X':
    if(count<1) {                /* Check limit */
        count++;
        gotoxy(21,2);
        ch=toupper(ch);
        value=ch;
        cprintf("%c    ",ch);
        gotoxy(21+count,2);
    }
    else
    {
        Beep();
    }
    break;

```

```

case kBSP :                /* Back Space */

```

```

    count--;
    if(count <0)
    {
        count=0;
        Beep();
    }
    gotoxy(21,2);
    cputs(" ");
    value = 0;
    gotoxy(21+count,2);
    break;

```

```

case kCR :
    if (value==0) value='X';
    ChangeRELValue(value,rel,rel->cellX,rel->cellY);
    setcursor(CURSOROFF);
    ShowValueXYREL(*rel,rel->cellX,rel->cellY,OFF);
    loop=FALSE;
    break;

```

```

case kESC: loop=FALSE;
    setcursor(CURSOROFF);
    break;

```

```

}
}while(loop==TRUE);
}

```

```

/*-----
int CalAddressREL(int n,int x,int y)
: Calculate the address of Relation Chart.
n = size of relation chart
x = first department number
y = second department number
-----*/

```

```

int CalAddressREL(int n,int x,int y)
{
    int address;
    if(x<=y) return(0);
    address = ((n-1)*(n))/2 - ((n-y)*(n-y+1))/2 +x-y-1;
    return(address);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่อนุญาตให้นำไปใช้เพื่อการค้าโดยไม่ได้รับอนุญาตจากศูนย์วิจัยและพัฒนาเอกสารทุกครั้งที่มีการนำไปใช้

```

/*-----*/
int GetRELValue(REL rel,int x,int y)
: Get value from Relation chart at (x,y)
/*-----*/

int GetRELValue(REL rel,int x,int y)
{ int value;

/* check range */

value = *( rel.pRel + CalAddressREL(rel.size,x,y) );
return(value);
}

/*-----*/
void ChangeRELValue(int value,REL *rel,int x,int y)
: Change the value at (x,y) of Relation Chart.
/*-----*/

void ChangeRELValue(int value,REL *rel,int x,int y)
{
/* check range */
if( (x<rel->minX) || (x>rel->maxX)) return;
if( (y<rel->minY) || (y>rel->maxY)) return;

*( rel->pRel +CalAddressREL(rel->size,x,y) ) = value;
}

/*-----*/
void LoadREL(REL *rel)
: Load Relation Chart from disk.
/*-----*/

void LoadREL(REL *rel)
{

FILE *fpLoad;

WIN load = {20,7,60,9," LOAD Relation Chart ",SINGLE};
WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
char name[9],oldname[13],key;
int value,length,loop,x,y,size;
long chksum,sum;
int m,type;

/***** Input Filename *****/
loop=TRUE;
do{
ShowWin(&load,RELBOR,RELTXT);

setcursor(CURSORON);

cputs(" File Name : ");
m=getstr(9,name); /* Filename */

if(m==kESC) /* Press ESC to exit */
{
EraseWin(&load);
return;
}

strcpy(oldname,rel->filename); /* Add file extension */
strcpy(rel->filename,name);
strcat(rel->filename, ".REL"); /* *.REL */

ShowWin(&msg,rMSGWINBORDER,rMSGWINTXT);

```

```

fpLoad = fopen(rel->filename,"rb");          /* Check existed file */

if(fpLoad != NULL)                          /* Have that file */
{
    /* Open file */
    loop=FALSE;
}
else
{
    /* Can't Open file */                /* File not found */
    gotoxy(10,2);
    cprintf("File %-12s not found !",rel->filename);
    getch();
    EraseWin(&msg);
    EraseWin(&load);
    strcpy(rel->filename,oldname);
}
}while(loop==TRUE);

/***** Load *****/
clrscr();
gotoxy(10,2);
cputs(" Loading File ... ");

fread(&type,sizeof(int),1,fpLoad);          /* read file type */

if (type != 2)                             /* type==2 (First byte) */
{
    fclose(fpLoad);
    clrscr();
    gotoxy(6,2);
    cprintf("%-12s is not a Relation Chart file !",rel->filename);
    getch();
    EraseWin(&msg);
    EraseWin(&load);
    strcpy(rel->filename,oldname);
    return;
}

fread(&length,sizeof(int),1,fpLoad);        /* read REL chart size */

rel->size=length;                          /* set size to rel->size */
size = ((length-1)*(length))/2;
rel->cellX = 2;                            /* set cell pointer (2,1) */
rel->cellY = 1;
FreeMemoryREL(rel->pRel);                  /* free old memory */
rel->pRel = InitMemoryREL(size);           /* allocate new memory */

sum = 0;                                  /* initialize checksum */

for(y=1;y<=length-1;y++)                  /* read REL Chart value */
for(x=y+1;x<=length;x++)
{
    fread(&value,sizeof(int),1,fpLoad);
    sum = sum + value;
    *( rel->pRel + CalAddressREL(rel->size,x,y) ) = value;
}

fread(&chksum,sizeof(long),1,fpLoad);      /* read checksum */
fclose(fpLoad);

if(chksum==sum) cputs("Completed !");      /* compare checksum==sum */
else
{
    clrscr();
    gotoxy(10,2);
    cprintf("Data Error in file %-12s !",rel->filename);
}
getch();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใด ๆ ขอสงวนสิทธิ์ในสิ่งที่ปรากฏและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

EraseWin(&msg);
EraseWin(&load);
setcursor(CURSOROFF);
}

/*-----
void SaveREL(REL *rel)
: Save Relation Chart to disk.
-----*/

void SaveREL(REL *rel)
{
    FILE *fpSave;

    WIN save = {20,7,60,9," SAVE Relation Chart ",SINGLE};
    WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
    char name[9],key;
    int value,length,loop,x,y;
    long chksum;
    int m,type;

    /* check there are value in REL Chart ?*/
    if((rel->size <2)|| (rel->size>30))
    {
        Beep();
        ShowWin(&msg,RMSGWINBORDER,RMSGWINTXT);
        gotoxy(5,2);
        cputs(" Relation Chart hasn't been loaded !");
        getch();
        EraseWin(&msg);
        return;
    }

    type=2;          /* set type =2 */
    loop=TRUE;       /* input filename */
    do{
        ShowWin(&save,RELBOR,RELTXT);

        setcursor(CURSORON);
        clrscr();
        cputs(" File Name : ");
        m=getstr(9,name);

        if((m==kESC) || (strlen(name)==0)) /* Press ESC to exit */
        {
            EraseWin(&save);
            return;
        }

        strcpy(rel->filename,name);          /* Add file extension */
        strcat(rel->filename,".REL");        /* *.REL */

        fpSave = fopen(rel->filename,"rb");  /* Open file */

        ShowWin(&msg,RMSGWINBORDER,RMSGWINTXT);

        if(fpSave != NULL)                  /* Check existed file */
        {
            gotoxy(6,1);
            cprintf(" File %-12s is already existed !",rel->filename);
            gotoxy(15,2);
            cputs("Overwrite (Y/N) ? ");    /* Overwrite?*/
            do{
                key=toupper(getche());
            }while( !(key=='Y' || key=='N'));
        }
    }while(1);
}

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของสำนักงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาต
 ไม่ว่ากรณีใดๆ ทั้งสิ้น

do{ ห้ามมิให้คัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
 key=toupper(getche());
 }while(!(key=='Y' || key=='N'));

```

        if(key=='Y')
        {
            loop=FALSE;
            fclose(fpSave);
        }
        else
        {
            loop=TRUE;
            fclose(fpSave);
            EraseWin(&msg);
            EraseWin(&save);
        }
    }
    else loop=FALSE;

}while(loop==TRUE);

/***** Save *****/
clrscr();
gotoxy(10,2);
cputs(" Saving File ... ");

fpSave = fopen(rel->filename,"wb");    /* Open to write */

fwrite(&type, sizeof(int), 1, fpSave);    /* write file type */

length = rel->size;
fwrite(&length, sizeof(int), 1, fpSave);    /* write REL Chart size */

chksum = 0;    /* Clear checksum */

for(y=1;y<=length-1;y++)    /* write REL Chart value */
for(x=y+1;x<=length;x++)
{
    value = *( rel->pRel + CalAddressREL(rel->size,x,y) );
    chksum = chksum + value;
    fwrite(&value, sizeof(int), 1, fpSave);
}

fwrite(&chksum, sizeof(long), 1, fpSave);    /* Write Checksum */

fclose(fpSave);    /* Close file */

cputs("Completed !");
getch();
EraseWin(&msg);
EraseWin(&save);
setcursor(CURSROFF);
}

/*-----
void NewREL(REL *rel)
    : Create new Relation Chart.
-----*/

void NewREL(REL *rel)
{
    WIN new = { 20,7,60,9," NEW Relation Chart ",SINGLE};
    int value;
    int *v,a,b;

    /*Input department number*/
    do{
        ShowWin(&new,RELBOR,RELTXT);
        gotoxy(5,1);
        cputs("How many departments ? ");
    }while(1);
}

```

เอกสารนี้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        EraseWin(&new);
    }while( (value <2) || (value>30));

    strcpy(rel->filename,"NONAME.REL");
    rel->size = value;
    FreeMemoryREL(rel->pRel);
    rel->pRel = InitMemoryREL(value);
    rel->cellX = 2;
    rel->cellY = 1;

    textattr(rNORMALTXT);
    setcursor(CURSOROFF);
    clrscr();
    ShowBarREL(rel,2,1);
    ShowAllValueREL(*rel);
    ShowValueXYREL(*rel,rel->cellX,rel->cellY,OFF);
}

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*-----
void RelationChart(REL *rel,int command)
: Main Function of Relation Chart
    command = cNEWREL -> make new REL CHART
             cEDITREL -> edit REL CHART
-----*/

void RelationChart(REL *rel,int command)
{
    WIN msg = {15,12,65,16," MESSAGE ",DOUBLE};
    int message;
    int *v,a,b;

    switch(command)
    {
    case cNEWREL :
        NewREL(rel);
        ShowInfoREL(*rel);
        break;

    case cEDITREL :
        if((rel->size <2)|| (rel->size>30))
        {
            Beep();
            ShowWin(&msg,rMSGWINBORDER,rMSGWINTXT);
            gotoxy(5,2);
            puts(" Relation Chart hasn't been loaded !");
            getch();
            EraseWin(&msg);
            return;
        }

        textattr(rNORMALTXT);
        setcursor(CURSOROFF);
        clrscr();
        ShowBarREL(rel,2,1);
        ShowAllValueREL(*rel);
        ShowValueXYREL(*rel,rel->cellX,rel->cellY,OFF);
        ShowInfoREL(*rel);
        break;
    }

    DisplayValueREL(*rel);
    message = InputKey();
    switch(message)
    {
    case 'n':
        NewREL(rel);
        ShowInfoREL(*rel);
        break;
    case 'e':
        EditREL(rel);
        ShowInfoREL(*rel);
        break;
    case 'q':
        return;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆก็ตาม หากมีข้อสงสัยหรือต้องการข้อมูลเพิ่มเติม กรุณาติดต่อเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        case kLEFT : MoveCellREL(rel,kLEFT);break;
        case kRIGHT : MoveCellREL(rel,kRIGHT);break;
        case kUP : MoveCellREL(rel,kUP);break;
        case kDOWN : MoveCellREL(rel,kDOWN);break;
        case kHOME : MoveCellREL(rel,kHOME);break;

        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'x':
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U':
        case 'X':
        case kCR:
            if(rel->cellX != rel->cellY)
                InputValueREL(rel,message);
            break;
    }

    }while(message != kESC);
}

/*-----
void ClearRELValue(REL *rel)
    : Initialize REL variable
-----*/

void ClearRELValue(REL *rel)
{
    strcpy(rel->filename,"NONAME.REL");
    rel->size = 0;
    rel->minX = 0;
    rel->minY = 0;
    rel->maxX = 0;
    rel->maxY = 0;
    rel->cellX = 2;
    rel->cellY = 1;
    rel->pRel = NULL;
}
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



LISTING PROGRAM

WIN.H

WIN.C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*                               File Name : WIN.H                               */
/* ===== */

```

```

/* ----- */

```

COLOR:

<FOREGROUND>

F_BLACK	F_BLUE	F_GREEN	F_CYAN	F_RED
F_MAGENTA	F_BROWN	F_LIGHTGRAY	F_DARKGRAY	F_LIGHTBLUE
F_LIGHTGREEN	F_LIGHTCYAN	F_INTEN	F_LIGHTRED	F_LIGHTMAGENTA
F_YELLOW	F_WHITE			

F_LIGHTGRAY!F_INTEN

F_CYAN!F_INTEN.

<BACKGROUND>

B_BLACK	B_BLUE	B_GREEN	B_CYAN
B_RED	B_MAGENTA	B_BROWN	B_LIGHTGRAY

```

/* COLOR */

```

```

#define B_BLACK      0
#define B_BLUE       0x10
#define B_GREEN      0x20
#define B_CYAN       0x30
#define B_RED        0x40
#define B_MAGENTA    0x50
#define B_BROWN     0x60
#define B_LIGHTGRAY  0x70

```

```

#define F_BLACK      0
#define F_BLUE       1
#define F_GREEN      2
#define F_CYAN       3
#define F_RED        4
#define F_MAGENTA    5
#define F_BROWN     6
#define F_LIGHTGRAY  7
#define F_INTEN      8

```

```

#define F_DARKGRAY   F_BLACK!F_INTEN
#define F_LIGHTBLUE  F_BLUE!F_INTEN
#define F_LIGHTGREEN F_GREEN!F_INTEN
#define F_LIGHTCYAN  F_CYAN!F_INTEN
#define F_LIGHTRED   F_RED!F_INTEN
#define F_LIGHTMAGENTA F_MAGENTA!F_INTEN
#define F_YELLOW     F_BROWN!F_INTEN
#define F_WHITE      F_LIGHTGRAY!F_INTEN

```

```

#define COMMENTCOLOR F_BLACK!B_LIGHTGRAY /* Comment Text Color */

```

```

/* WIN.H : prototypes and typedefs for WIN.C library */

```

```

#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

```

```

#define FALSE      0
#define TRUE       1
#define OFF        0
#define ON         1
#define NONE       0

```

```

#define SINGLE      1
#define DOUBLE      2
#define XSINGLE      3
#define XDOUBLE     4

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define HEADSINGLE 1
#define HEADDDOUBLE 3
```

```
/* KEYCODE Lead-in byte is 0, follow by...*/
```

```
#define AltQ 16
#define AltW 17
#define AltE 18
#define AltR 19
#define AltT 20
#define AltY 21
#define AltU 22
#define AltI 23
#define AltO 24
#define AltP 25
#define AltA 30
#define AltS 31
#define AltD 32
#define AltF 33
#define AltG 34
#define AltH 35
#define AltJ 36
#define AltK 37
#define AltL 38
#define AltZ 44
#define AltX 45
#define AltC 46
#define AltV 47
#define AltB 48
#define AltN 49
#define AltM 50
#define F1 59
#define F2 60
#define F3 61
#define F4 62
#define F5 63
#define F6 64
#define F7 65
#define F8 66
#define F9 67
#define F10 68
#define HOME 71
#define UP 72
#define P_UP 73
#define LE 75
#define RI 77
#define END 79
#define DN 80
#define P_DN 81
```

```
/* KEYCODE :-No lead by 0 */
```

```
#define INS 82
#define DEL 83
#define CR 13
#define ESC 27
#define SPC 32
#define TAB 9
#define SHF_TAB 15
#define BSP 8
#define C_P_UP 132
#define C_P_DN 118
#define SCRL 0x46
#define ALT_1 120
```

```
/* Key define */
```

```
#define kESC 299
#define kLEFT 300
#define kRIGHT 301
```

```

#define kUP          302
#define kDOWN        303
#define kF1          304
#define kF2          305
#define kF3          306
#define kF4          307
#define kF10         310
#define kPGUP        311
#define kPGDN        312
#define kHOME        313
#define kCR          314
#define kINS         315
#define kDEL         316
#define kBSP         317
#define kSPC         318
#define kALTX        319
#define kALTQ        320
#define kTAB         321

#define CURSOROFF    0x2000          /* Cursor OFF */
#define CURSORON     0x0607          /* Cursor ON */

/***** Define STRUCT *****/

extern struct text_info;              /* from CONIO.H */
typedef struct{
    int  left,top,right,bottom;        /* border loc */
    char title[30];                   /* title of window */
    int  style;                       /* border style */
    void *save;                       /* pointer to save buffer */
    struct text_info prev;            /* old video state */
} WIN;

/***** Function Prototypes *****/

void ILine(int col,int top,int bottom,int type);
void HLine(int row,int left,int right,int type);
void Center(int x, int y, int type);
void Box(int l, int t, int r, int b, int type);
void ShowWin(WIN *win,unsigned int BorderColor,unsigned int TextColor);
void EraseWin(WIN *win);
void RewriteWin(WIN *win,int row, char attrib);
int  InputKey(void);
void setcursor(unsigned int shape);
void Beep(void);
void Comment(char *str);
void putConner(char ch);
void setcursor(unsigned int shape);
int  getstr(int length,char *str);
int  GetInt(void);
int  *SaveAllScreen(void);
void RestoreAllScreen(int *address);
void VScroll(int col,int top,int bottom,int value,int size);
void PrintCommandBar(char *command,int BarColor,int TxtColor);
void PrintCommand(char *command,int HotColor,int TxtColor);
void Fastputs(int x,int y,int attr,char *str);
void Fastprintf(int x,int y,int attr,char *msg, ...);
void gprintf(int x,int y,int txtcolor,int bkcolor,char *arg,...);
void gputs(int x,int y,int txtcolor,int bkcolor,char *str);
int  gGetInt(int startx,int starty,int txtcolor,int bkcolor);
int  gGetStr(int xl,int yl,int length,char *str);

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์หรือการเชิงงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/*
/*
/*      Program WIN.C :
/*
/*
/* ===== */

```

```

/***** Include File *****/

```

```

#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include <stdarg.h>
#include <graphics.h>
#include "win.h"

```

```

/***** Variable Declaration *****/

```

```

char far *mono =MK_FP(0xB000,0);
char far *color=MK_FP(0xB800,0);

```

```

/*-----
void ILine(int col,int top,int bottom,int type)
: Vertical line in (col) column from row (top) to (bottom)
type = SINGLE      ->single line,single-lined head
DOUBLE            ->double line,double-lined head
HEADDDOUBLE       ->single line,double-lined head
HEADSINGEL        ->double line,single-lined head
/*-----*/

```

```

void ILine(int col,int top,int bottom,int type)
{
int c;
int framtab[4][3]={
    {194,179,193}, /*  T  |  T  */
    {203,186,202}, /*  T  |  T  */
    {209,179,207}, /*  T  |  T  */
    {210,186,208}  /*  T  |  T  */
};

```

```

if ((type <0) || (type >4)) return;

```

```

type--;

```

```

gotoxy(col,top);
cprintf("%c",framtab[type][0]);
for(c=top+1;c<bottom;++c){
    gotoxy(col,c);
    cprintf("%c",framtab[type][1]);
}
gotoxy(col,bottom);
cprintf("%c",framtab[type][2]);
}

```

```

/*-----
void HLine(int row,int left,int right,int type)
: Horizontal line on (row) row from column (left) to (right)
type = SINGLE      ->single line,single-lined head
DOUBLE            ->double line,double-lined head
HEADDDOUBLE       ->single line,double-lined head
HEADSINGEL        ->double line,single-lined head
/*-----*/

```

```

void HLine(int row,int left,int right,int type)

```

```

{
    int c;
    int framtab[4][3]={
        {195,196,180}, /* | - | */
        {204,205,185}, /* | = | */
        {199,196,182}, /* | - | */
        {198,205,181}  /* | = | */
    };

    if ((type <0) || (type >4)) return;

    type--;

    gotoxy(left,row);
    cprintf("%c",framtab[type][0]);

    for(c=left+1;c<right;++c){
        gotoxy(c,row);
        cprintf("%c",framtab[type][1]);
    }
    gotoxy(right,row);
    cprintf("%c",framtab[type][2]);
}

/*-----
void Center(int x, int y, int type)
: put Cross line or center character at (x,y)
type = SINGLE    ->single x-line,single y-line
DOUBLE          ->double x-line,double y-line
XSINGLE          ->single x-line,double y-line
XDOUBLE         ->double x-line,single y-line
-----*/

void Center(int x, int y, int type)
{
    int framtab[]={197,206,215,216}; /* + + + + */

    if ((type <0) || (type >4)) return;

    type--;

    gotoxy(x,y);
    cprintf("%c",framtab[type]);
}

/*-----
void Box(int l, int t, int r, int b, int type)
: Draw box from (l,t) to (r,b)
type = SINGLE    ->single x-line,single y-line
DOUBLE          ->double x-line,double y-line
XSINGLE          ->single x-line,double y-line
XDOUBLE         ->double x-line,single y-line
-----*/

void Box(int l, int t, int r, int b, int type)
{
    int a;
    int framtab[5][6]={
        {32,32,32,32,32,32},
        {218,196,191,179,192,217}, /* ┌ ─ ─ ─ ─ ┐ */
        {201,205,187,186,200,188}, /* │ │ │ │ │ │ */
        {214,196,183,186,211,189}, /* └ ─ ─ ─ ─ ┘ */
        {213,205,184,179,212,190}  /* ┌ ─ ─ ─ ─ ┐ */
    };

```

```

if ((type <0) || (type >5)) return;

gotoxy(l,t);
cprintf("%c",framtab[type][0]);

for(a=1;a<r-1;a++)
    cprintf("%c",framtab[type][1]);

cprintf("%c",framtab[type][2]);

for(a=t+1;a<b;a++)
{
    gotoxy(l,a);
    cprintf("%c",framtab[type][3]);
    gotoxy(r,a);
    cprintf("%c",framtab[type][3]);
}

gotoxy(l,b);
cprintf("%c",framtab[type][4]);
for(a=1;a<r-1;a++)
    cprintf("%c",framtab[type][1]);
cprintf("%c",framtab[type][5]);
}

/*-----
void ShowWin(WIN *win,unsigned int BorderColor,unsigned int TextColor)
: Display pop up window
-----*/

void ShowWin(WIN *win,unsigned int BorderColor,unsigned int TextColor)
{
    struct text_info text;
    int bufsizex,y;
    char far *video;
    int shadow,shade_x,shade_y;

    gettextinfo(&(win->prev)); /* save old video state */
    window(1,1,80,25); /* goto full screen mode */

    /* make sure boundaries are in correct order */
    if (win->left > win->right) {
        x=win->left;
        win->left = win->right;
        win->right = x;
    }
    if(win->top > win->bottom) {
        x=win->top;
        win->top = win->bottom;
        win->bottom = x;
    }
    if((win->bottom >23) || (win->right>78))
    {
        shadow= OFF;
        shade_x=shade_y=0;
    }
    else
    {
        shadow= ON;
        shade_x=2;shade_y=1;
    };

    /* save area under new window */
    bufsizex = (win->right - win->left +1+shade_x) *
               (win->bottom - win->top +1+shade_y) * 2; /* buf size */
    win->save = malloc(bufsizex); /* get buffer */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่อาจรับผิดชอบต่อความเสียหายใดๆที่เกิดจากการใช้เอกสารนี้โดยไม่ถูกต้องตามลิขสิทธิ์ของเอกสารฉบับนี้

```

if (win->save==NULL)
{
    printf("ERROR ! Not enough memory");
    getch();
    return;
}
gettext(win->left,win->top,win->right+shade_x,
        win->bottom+shade_y,win->save);
/* save image */

/* draw the border */
textattr(BorderColor); /* border color */
Box(win->left,win->top,win->right,win->bottom,win->style); /* box */
if((win->title != ""))
{
    gotoxy((win->right-win->left-strlen(win->title))/2+win->left,win->top);
    cprintf("%s",win->title);
}
if(shadow==ON) {
    gettextinfo(&text);
    switch(text.currmode) {
        case C80: video=(char far*) color;break;
        case BW80:
        case MONO: video=(char far*) mono;break;
    }
    y=win->bottom+1;
    for(x=win->left+2;x<=win->right+2;x++)
        video[x*2-1+(y-1)*160]=(char)F_LIGHTGRAY;B_BLACK;

    for(x=win->right+1;x<=win->right+2;x++)
        for(y=win->top+1;y<=win->bottom+1;y++)
            video[x*2-1+(y-1)*160]=(char)F_LIGHTGRAY;B_BLACK;
}
/* open the window */
textattr(TextColor); /* set text foreground */
window(win->left+1,win->top+1,win->right-1,win->bottom-1);
clrscr();
}

/*-----
void EraseWin(WIN *win)
    : Erase pop-up window , restoring overlaid image
-----*/

void EraseWin(WIN *win)
{
    int shade_x,shade_y;

    /* make sure the window exists */
    if (win->save == NULL )
        return; /* quit if it doesn't */
    /* restore previous window characteristics */
    window(win->prev.winleft,win->prev.wintop,
           win->prev.winright,win->prev.winbottom);
    gotoxy(win->prev.curx,win->prev.cury); /* cursor pos */
    textcolor(win->prev.attribute & 0x0F); /* old color */
    textbackground(win->prev.attribute >> 4); /* and b/g */

    if((win->bottom >23) || (win->right>78)) {shade_x=shade_y=0; }
    else {shade_x=2;shade_y=1;};
    /* restore overlaid area */
    puttext (win->left,win->top,win->right+shade_x,
            win->bottom+shade_y,win->save);
    free(win->save);
    win->save = NULL;
}

/*-----

```

```
void RewriteWin(WIN *win,int row, char attrib)
: Rewrite pop-up row with new character attribute
```

```
void RewriteWin(WIN *win,int row, char attrib)
```

```
{
    struct text_info text;
    int p,nchars;
    union REGS reg;

    gettextinfo(&text);          /* get current settings */
    nchars = win->right - win->left; /* pop-up width */
    for(p=1;p<nchars;p++) {
        gotoxy(p,row);
        reg.h.ah = 8;             /* get character */
        reg.h.bh = 0;             /* in page 0 */
        int86(0x10,&reg,&reg);     /* via ROM BIOS */
        reg.h.ah = 9;             /* write back out with */
        reg.h.bl = attrib;        /* hilite attribute */
        reg.h.bh = 0;             /* page 0 */
        reg.x.cx = 1;             /* one char */
        int86(0x10,&reg,&reg);
    }
    gotoxy(text.curx,text.cury);  /* restore cursor */
}
```

```
/*-----*/
void Beep()
: a short sound.
/*-----*/
```

```
void Beep()
{
    sound(1000);
    delay(10);
    nosound();
}
```

```
/*-----*/
void Comment(char *str)
: Write comment string at 25th row.
/*-----*/
```

```
void Comment(char *str)
{
    Fastprintf(1,25,COMMENTCOLOR,"%-80s",str);
}
```

```
/*-----*/
void putConner(char ch)
: write a char at row=25,column=80.
/*-----*/
```

```
void putConner(char ch)
{
    char far *screen;
    struct text_info text;
```

```
    gettextinfo(&text);          /* get text information */
    if (text.currmode == C80)    /* check screen mode */
        screen = MK_FP(0xB800, 0); /* color */
    else
        screen = MK_FP(0xB000, 0); /* monochrome */
    /* put char */
    screen[80*2-1+(25-1)*160]=(char) COMMENTCOLOR & 0x00F0;
    screen[80*2-2+(25-1)*160]=(char) ch;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่สามารถแก้ไขหรือเปลี่ยนแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/*-----
int InputKey()
: Input key and return key-message.
-----*/

int InputKey()
{
    char key1, key2;

    key1=getch();          /* Read first byte */
    if( key1 == 0 )        /* Control Key ? */
    {
        /* control key */
        key2 = getch();    /* get second byte */
        switch(key2)
        {
            case UP       : return(kUP);
            case DN       : return(kDOWN);
            case LE       : return(kLEFT);
            case RI       : return(kRIGHT);
            case P_UP     : return(kPGUP);
            case P_DN     : return(kPGDN);
            case HOME     : return(kHOME);
            case F1       : return(kF1);
            case F2       : return(kF2);
            case F3       : return(kF3);
            case F4       : return(kF4);
            case F10      : return(kF10);
            case AltX     : return(kALT_X);
            case AltQ     : return(kALT_Q);

            default       : return(0);
        }
    }
    else
    {
        switch(key1)
        {
            case ESC     : return(kESC);
            case CR      : return(kCR);
            case INS     : return(kINS);
            case DEL     : return(kDEL);
            case BSP     : return(kBSP);
            case SPC     : return(kSPC);
            case TAB     : return(kTAB);
            default      : return(key1);
        }
    }
}

/*-----
void setcursor(unsigned int shape)
: set cursor shape.
-----*/

void setcursor(unsigned int shape)
{
    union REGS reg;

    reg.h.ah = 1;
    reg.x.cx = shape;
    int86(0x10, &reg, &reg);
}

/*-----

```

เอกสารนี้เป็นทรัพย์สินทางปัญญาของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
 ไม่ควรเผยแพร่โดยไม่ได้รับอนุญาตจากทางมหาวิทยาลัย
 ไม่ควรนำเอกสารนี้ไปใช้ในการเรียนการสอนโดยไม่ได้รับอนุญาตจากทางมหาวิทยาลัย

```
int getstr(int length,char *str)
: Input string message at cursor position.
Length is number of chars to input.
```

```
int getstr(int length,char *str)
{
    struct text_info text;
    int message,ret,loop;
    int count;
    int x,y;
```

```
    gettextinfo(&text);
    x=text.curs;
    y=text.curs;
    count=0;loop=TRUE;
```

```
do{
```

```
    gotoxy(x+count,y);
    message = InputKey();
```

```
    if(message==kTAB) {loop=FALSE;ret=kTAB;}
    if(message==kESC) {loop=FALSE;ret=kESC;}
    if(message==kCR) {loop=FALSE;ret=kCR;}
    if(message==kBSP)
```

```
    {
        count--;
        if (count<0) count=0;
        gotoxy(x+count,y);cprintf(" ");
        str[count]='\0';
    }
```

```
    if( ((message>='a') && (message<='z'))
        || ((message>='A') && (message<='Z'))
        || ((message>='0') && (message<='9'))
    )
```

```
    {
        if(count>=length-1) count=length-1;
        else
        { str[count] = toupper(message);
          cprintf("%c",str[count]);
          count++;
        }
    }
```

```
while(loop==TRUE);
```

```
str[count]='\0';
return(ret);
}
```

```
/*-----*/
int GetInt()
: Input integer at cursor position
```

```
int GetInt()
```

```
{
    int ch;
    int a;
    int count,loop;
    int startx,starty;
    char string[6];
```

struct text_info text; รับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

gettextinfo(&text);

```
strcpy(string," ");
```

```
/* Init string */
```

```

setcursor(CURSORON);          /* cursor on */
startx = text.curx;
starty = text.cury;

```

```

count=0;
loop=TRUE;
do{

```

```

    ch = InputKey();
    switch(ch)
    {
        case '0':          /* if == '0'...'9' */
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
        case '-':

```

```

            if (count<5) /* Check limit */
            {

```

```

                string[count] = ch;
                count++;
                gotoxy(startx,starty);
                cprintf("%-5s",string);
                gotoxy(startx+count,starty);
            }

```

```

        else
        {
            Beep();
        }
        break;

```

```

    case KBSP :          /* Back Space */

```

```

        count--;
        if(count <0)
        {
            count=0;
            Beep();
        }

```

```

        for(a=count;a<5;a++)
            string[a]=string[a+1];
        gotoxy(startx,starty);
        cprintf("%-5s",string);
        gotoxy(startx+count,starty);
        break;

```

```

    case kCR :
        string[6] = '\0';

```

```

        loop = FALSE;
        break;

```

```

    }
}while(loop == TRUE);
return(atoi(string));
}

```

```

/*-----
int *SaveAllScreen()
เอกสารนี้เป็นเอกสารเพื่อการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
return address pointer of memory that saved screen
ไม่อนุญาตให้นำไปเผยแพร่ หรือทำซ้ำโดยไม่ได้รับอนุญาตจากเจ้าของลิขสิทธิ์ทุกครั้งที่มีการนำไปใช้-----*/

```

```

int *SaveAllScreen()
{

```

```

int bufsize;
int *address;

window(1,1,80,25);          /* goto full screen mode */

bufsize = (80) * (25) * 2;
address = malloc(bufsize);   /* get buffer */
if (address == NULL)
{
    printf("ERROR ! Not enough memory");
    getch();
    return(address);
}
gettext(1,1,80,25,address); /* Save all screen into memory */
return(address);
}

/*-----
void RestoreAllScreen(int *address)
: Restore screen from memory
  Input address pointer of memory
-----*/

void RestoreAllScreen(int *address)
{
    puttext(1,1,80,25,address); /* Restore screen from memory */
    free(address);
}

/*-----
void VScroll(int col,int top,int bottom,int value,int size)
:
-----*/

void VScroll(int col,int top,int bottom,int value,int size)
{
    int a,count;

    for(a=top+1;a<=bottom-1;a++)
    {
        gotoxy(col,a);
        cprintf("█");
    }
    gotoxy(col,top);
    cprintf("%c",30);
    gotoxy(col,bottom);
    cprintf("%c",31);
    count = bottom-top-1;
    gotoxy(col, top + (value*count)/size + 1);
    cprintf("%c",254);
}

/*-----
void PrintCommand(char *command)
: print command in the menu's window.
-----*/

void PrintCommand(char *command,int HotColor,int TxtColor)
{
    int a,l;
    char ch;

    l = strlen(command);
    for(a=0;a<l;a++)

```

```

{
    ch = command[a];
    if ( ch == '&') textattr(HotColor);    /* check hotkey */
    else
    {   cprintf("%c",ch);
        textattr(TxtColor);
    }
}

}

/*-----
void PrintCommandBar(char *command)
    : Print menu's bar
-----*/

void PrintCommandBar(char *command,int BarColor,int TxtColor)
{
    int a,l;
    char ch;

    l = strlen(command);    /* string length */
    for(a=0;a<l;a++)
    {
        ch = command[a];
        if ( ch != '&')
        {   textattr(BarColor);
            cprintf("%c",ch);
        }
    }
    textattr(TxtColor);
}

/*-----
void Fastputs(int x,int y,int attr,char *str)
    : Direct put string into display memory
-----*/

void Fastputs(int x,int y,int attr,char *str)
{
    register int Addr,pos=2*(x-1)+160*(y-1);

    _AH=15;
    geninterrupt(0x10);
    Addr=(_AL=='\x3')?0xB800:0xB000;

    while(*str!='\0'){
        pokeb(Addr,pos++,*str++);
        pokeb(Addr,pos++,(char)attr);
    }
}

/*-----
void Fastprintf(int x,int y,int attr,char *msg, ...)
    : Direct printf into display memory
-----*/

void Fastprintf(int x,int y,int attr,char *msg, ...)
{
    เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
    char str[256];
    ไม่ว่า va_list ap; อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

    va_start(ap,msg);
    vsprintf(str,msg,ap);
}

```

```

    va_end(ap);
    Fastputs(x,y,attr,str);
}

```

```

/*-----
void gprintf(int x,int y,int txtcolor,int bkcolor,char *arg,...)
: Graphic mode printf,
column:      1<= x <=80
row:         1<= y <=48
-----*/

```

```

void gprintf(int x,int y,int txtcolor,int bkcolor,char *arg,...)
{
    char str[256];
    va_list ap;
    int length,i;
    /* x is between 1 and 80
       y is between 1 and 48 */

    va_start(ap,arg);
    vsprintf(str,arg,ap);
    va_end(ap);
    setcolor(bkcolor);
    setfillstyle(SOLID_FILL,bkcolor);
    length=strlen(str);
    bar((x-1)*8,(y-1)*10,(x-1+length)*8-1,y*10); /* background */
    setcolor(txtcolor);
    outtextxy((x-1)*8,(y-1)*10+1,str); /* text */
}

```

```

/*-----
void gputs(int x,int y,int txtcolor,int bkcolor,char *str)
: Graphic mode puts
-----*/

```

```

void gputs(int x,int y,int txtcolor,int bkcolor,char *str)
{
    int length;
    /* x is between 1 and 80
       y is between 1 and 48 */

    setcolor(bkcolor);
    setfillstyle(SOLID_FILL,bkcolor);
    length=strlen(str);
    bar((x-1)*8,(y-1)*10,(x-1+length)*8-1,y*10); /* Background */
    setcolor(txtcolor);
    outtextxy((x-1)*8,(y-1)*10+1,str); /* Text */
}

```

```

/*-----
int gGetInt(int startx,int starty,int txtcolor,int bkcolor)
: Graphic mode GetInt
Input integer at (startx,starty) with txtcolor (txtcolor)
and background color (bkcolor).
Range: 1 <= startx <= 80
       1 <= starty <= 48
-----*/

```

```

int gGetInt(int startx,int starty,int txtcolor,int bkcolor)
{
    int ch;
    int a;
    int count,loop;
    char string[6];

```

ไม่ว่า int ch; ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

strcpy(string,""); /* Initialize string */

count=0;
loop=TRUE;
gprintf(startx,starty,txtcolor,bkcolor,"%-5s",string);
do{
    ch = InputKey();
    switch(ch)
    {
        case '0': /* if == '0'...'9' */
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
        case '-':
            if(count<5) { /* Check limit */
                string[count] = ch;
                count++;
                gprintf(startx,starty,txtcolor,bkcolor,
                    "%-5s",string);
            }
            break;

        case KBSP : /* Back Space */
            count--;
            if(count < 0)
            {
                count=0;
                Beep();
            }
            for(a=count;a<5;a++)
                string[a]=string[a+1];
            gprintf(startx,starty,txtcolor,bkcolor,
                "%-5s",string);
            break;

        case KCR :
            string[6] = '\0';
            loop = FALSE;
            break;

    }

} while(loop == TRUE);
return(atoi(string));
}

```

```

/*-----
int gGetStr(int xl,int yl,int length,char *str)
: Graphic mode GetStr
  Input string at (xl,yl), length = string length limit
  Output to *str.
  Range: 1 <= xl <= 80
         1 <= yl <= 48
-----*/

```

เอกสารนี้เป็นเอกสารที่ลงพิมพ์ไว้สำหรับเอาไว้ใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```

int gGetStr(int xl,int yl,int length,char *str)
{
    int message,ret,loop;
    int count;
    int x,y;

```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
x=xl;  
y=yl;
```

```
gputs(x,y,WHITE,DARKGRAY,"-");  
count=0;  
loop=TRUE;
```

```
do{
```

```
    message = InputKey();
```

```
    if(message==kTAB) {loop=FALSE;ret=kTAB;}
```

```
    if(message==kESC) {loop=FALSE;ret=kESC;}
```

```
    if(message==kCR) {loop=FALSE;ret=kCR;}
```

```
    if(message==kBSP)
```

```
    {
```

```
        count--;
```

```
        if (count<0) count=0;
```

```
        gputs(x+count,y,WHITE,DARKGRAY,"-");
```

```
        str[count]='\0';
```

```
    }
```

```
    if( ((message>='a') && (message<='z'))
```

```
        || ((message>='A') && (message<='Z'))
```

```
        || ((message>='0') && (message<='9'))
```

```
    )
```

```
    {
```

```
        if(count>=length-1) count=length-1;
```

```
        else
```

```
        { str[count] = toupper(message);
```

```
          gprintf(x+count,y,WHITE,DARKGRAY,"%c-",str[count]);
```

```
          count++;
```

```
        }
```

```
    }
```

```
} while(loop==TRUE);
```

```
str[count]='\0';
```

```
return(ret);
```

```
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



LISTING PROGRAM

MOUSE.H

MOUSE.C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/*                                     File Name : MOUSE.H                                     */
/* =====                                                                    */
/*
/***** Define CONSTANT *****/

#define mNOT_MOVED 0
#define mRIGHT 1
#define mLEFT 2
#define mUP 3
#define mDOWN 4

/***** Function Prototypes *****/

void set_mouse_position(int x, int y);
void set_mouse_bound(int left,int top,int right,int bottom);
void mouse_position(int *x, int *y);
void mouse_motion(int *deltax, int *deltay);
void mouse(int *ax, int *bx, int *cx, int *dx);
void mouse_reset(void);
void mode(int mode_code);
void cursor_off(void), cursor_on(void);
void goto_xy(int x, int y);
int leftb_pressed(void);
int rightb_pressed(void);

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* ===== */
/*
/*          Program MOUSE.C : Mouse Interfacing
/*
/*
/* ===== */

```

```

/***** Include File *****/

```

```

#include "dos.h"
#include "stdio.h"
#include "stdlib.h"
#include "mouse.h"

```

```

/*-----
void mode(int mode_code)
    : Set the video mode.
-----*/

```

```

void mode(int mode_code)
{
    union REGS r;

    r.h.al = mode_code;
    r.h.ah = 0;
    int86(0x10, &r, &r);
}

```

```

/*-----
void goto_xy(int x, int y)
    :Send the cursor to the specified X,Y position.
-----*/

```

```

void goto_xy(int x, int y)
{
    union REGS r;

    r.h.ah = 2;    /* cursor addressing function */
    r.h.dl = x;    /* column coordinate */
    r.h.dh = y;    /* row coordinate */
    r.h.bh = 0;    /* video page */
    int86(0x10, &r, &r);
}

```

```

/*-----
void cursor_off(void)
    : Turn off the mouse cursor.
-----*/

```

```

void cursor_off(void)
{
    union REGS r;

    r.x.ax = 2;    /* remove the cursor */
    int86(0x33, &r, &r);
}

```

```

/*-----
void cursor_on(void)
    : Turn on the mouse cursor.
-----*/

```

```

void cursor_on(void)

```

```

{
    union REGS r;

    r.x.ax = 1;    /* show the cursor */
    int86(0x33, &r, &r);
}

/*-----*/
    rightb_pressed(void)
        : Return true if right button is pressed,
          false otherwise.
/*-----*/

rightb_pressed(void)
{
    union REGS r;

    r.x.ax = 3;    /* get position and button status */
    int86(0x33, &r, &r);
    return r.x.bx & 2;
}

/*-----*/
    leftb_pressed(void)
        : Return true if left button is pressed,
          false otherwise.
/*-----*/

leftb_pressed(void)
{
    union REGS r;

    r.x.ax = 3;    /* get position and button status */
    int86(0x33, &r, &r);
    return r.x.bx & 1;
}

/*-----*/
    void set_mouse_position(int x, int y)
        : Set mouse cursor coordinates.
/*-----*/

void set_mouse_position(int x, int y)
{
    union REGS r;

    r.x.ax = 4;    /* set position */
    r.x.cx = x;
    r.x.dx = y;
    int86(0x33, &r, &r);
}

/*-----*/
    void mouse_position(int *x, int *y)
        : Return mouse cursor coordinates.
/*-----*/

void mouse_position(int *x, int *y)
{
    union REGS r;
    r.x.ax = 3;    /* get position and button status */
    int86(0x33, &r, &r);
    *x = r.x.cx;

```

```

    *y = r.x.dx;
}

/*-----
void mouse_motion(int *deltax, int *deltay)
    : Return the direction of travel.
-----*/

void mouse_motion(int *deltax, int *deltay)
{
    union REGS r;

    r.x.ax = 11; /* get direction of motion */
    int86(0x33, &r, &r);

    *deltax = r.x.cx;
    *deltay = r.x.dx;
}

/*-----
void mouse_reset(void)
    : Reset the mouse.
-----*/

void mouse_reset(void)
{
    union REGS r;

    r.x.ax = 0; /* reset the mouse */
    int86(0x33, &r, &r);
    if((int) r.x.ax != -1) {
        printf("mouse hardware or software not installed");
        exit(1);
    }
    if(r.x.bx != 2) {
        printf("two-button mouse required");
        exit(1);
    }
}

/*-----
void set_mouse_bound(int left,int top,int right,int bottom)
    : Set mouse boundary.
-----*/

void set_mouse_bound(int left,int top,int right,int bottom)
{
    union REGS r;

    r.x.ax = 7;
    r.x.cx = left;
    r.x.dx = right;
    int86(0x33,&r,&r);

    r.x.ax = 8;
    r.x.cx = top;
    r.x.dx = bottom;
    int86(0x33,&r,&r);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงการและวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีได้เนื่องด้วยความกรุณาและความช่วยเหลือจาก ดร. เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำและข้อเสนอแนะแนวทางการแก้ปัญหา ในช่วงระหว่างการทำโครงการ ตลอดจนคณะกรรมการสอบวิทยานิพนธ์ อาจารย์ บุญธีร เครือตราฐ, อาจารย์ กฤตวรรณ เครือตราฐ และ อาจารย์ ประสาร ดังตีสานนท์ ที่กรุณาตรวจสอบความถูกต้องของโครงการและให้คำแนะนำแนวทางในการปรับปรุงของโครงการนี้

นอกจากนี้กลุ่มผู้ทำโครงการขอขอบพระคุณ บิดา มารดาที่สนับสนุนและช่วยเหลือตลอดเวลา ที่ทำการศึกษาใน สถาบันเทคโนโลยี พระจอมเกล้า ลาดกระบัง แห่งนี้ , ขอขอบพระคุณอาจารย์ทุกท่าน ที่กรุณาประสิทธิ์ประสาทวิชาต่าง ๆ ให้กลุ่มผู้ทำโครงการ จนกระทั่งทำโครงการและวิทยานิพนธ์ สำเร็จลุล่วงได้ด้วยดี

ท้ายสุดนี้ กลุ่มผู้ทำโครงการขอขอบคุณ เต่า ที่ได้ให้คำแนะนำในการเขียนโปรแกรม อีก, โต, โอ้ และ ยิว ที่ให้ความช่วยเหลือโดยให้ยืมเครื่องพิมพ์ในการพิมพ์รายงาน รวมทั้งเพื่อนๆ ในคณะวิศวกรรมศาสตร์ที่สนับสนุนและให้กำลังใจในการทำโครงการนี้ .

นาย ธนารักษ์ ห้องสุข

นาย มานพ ศรีพิพัฒน์

บรรณานุกรม

1. สมศักดิ์ ตรีสัตย์, "การออกแบบ และ วางผังโรงงาน" ,
สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 245 หน้า, 2534.
2. ผศ.ดร. สันต์ชัย กลิ่นพิกุล, อ.พิจิตร พิศสุวรรณ, "การออกแบบโรงงาน",
ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์,
มหาวิทยาลัยสงขลานครินทร์ หาดใหญ่.
3. RICHARD L. FRANCIS, JOHN A. WHITE, "FACILITY LAYOUT AND LOCATION :
AN ANALYTICAL APPROACH" , PRENTICE-HALL INC., 585 p., 1974.
4. AARON M. TENENBAUM, YEDIDYAH LANGSAM, MOSHE J. AUGENSAM,
"DATA STRUCTURES USING C", PRENTICE-HALL INC., 662 p., 1990.
5. DONNA MOSICH, NAMIR SHAMMAS, BRYAN FLAMIG,
"ADVANCE TURBO C PROGRAMMER'S GUIDE" , JOHN WILEY & SONS, INC,
339 p., 1988.
6. HERBERT SCHILDT, "C : POWER USER'S GUIDE", MCGRAW-HILL, 382 p., 1988.
7. BORLAND INTERNATIONAL INC. "BORLAND C++ 2.0 : LIBRARY REFERENCE",
BORLAND INTERNATIONAL INC, 593 p., 1991.