เอกสารอ้างอิง

1. Efroymson , M.A., Md T.L.Ray, " Branch and Bound Algasithm for plant Location ," Operation Research , V14 N3, 1966 , 361 - 368.

2. Khumawala, B.M.,"An Efficient Branch and Bound Augorithm for warehouse Location Problem," Management Science , V18,1972 718-731.

3. Erlenkotter , Donald A., A dual - Based procedure for Uncapacitated Facility Location ; operation Research. V26 N6 , Nov - Dec 1978 , 1002 - 1009.

4. Khuehn , A.A., and M.J. Humburger , " A Heuristic program for Loading warehouse ," Management Science , VA , 643 - 606.

5. Khumawala, B.m.,"An Effficient Heuristic Procedure for the Capacitated Warehouse Location Problem," Naval Research Logistic Review Quarterly, V21 N4 , Dec.1974 , 609-624.

6. Ballou,Ronald H.,"Dinamic Warehouse Location Analysis,"Journal of Marketing Research, V5, Aug. 1968.

7. Sa,Graciano, "Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem, "Operations Research", V17 N6, 1969 , 1005-1016.

8. Lee , Sang M., Goal programming for Decision Analysis , Anerbach , philadephia, 1972.

9. Tantasuth , "Goal Programming and Long Range Planning in Under developed Countries " Ph.D. Dissertation, Texas Tech University.

10. Ignizio, Goal programming and Extensions, Lexington Books , Lexington , MA , 1976.

11. เสาวลักษณ์ สุรพลชัย , " การวิเคราะห์โปรแกรมออกแบบทางสถาปัตยกรรมด้วยทฤษฎีโปรแกรมเชิงเส้น : การหาสัดส่วนพื้นที่ใช้สอยของโครงการอเนกหน้าที่ใช้สอย," วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถาปัตยกรรมศาสตร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย , 2525.

12. วีระ อินพันทัง ," อิทธิพลของสภาพแวดล้อมกายภาพภายในสำนักงานที่มีต่อพฤติกรรมผู้ใช้ : การศึกษากรณีตัวอย่างอาคารสำนักงานใหญ่ของธนาคารกสิกรไทย และ ธนาคารศรีนคร," วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถาปัตยกรรมศาสตร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย , 2524.

13. Francis , R.L., and A while , Facility Layout and Location - An Analytical Approach , Prentice Hall , Inc., 1974.

14. The Steelcase National Study of Environment ,Louise Harris Associate,Inc., 1978.

15. Muther ,R., Systematic Layout Planning , Industrial Education Institue,Boston,Mass.,1961.

16. Simon, H.a., and Allen Newell,"Heuristic Problem Solving:The next advance in Operation Research, " Operations Research", Jan-Feb 1958, 1-10.

17. Lee ,R.L. and J. M.Moor, "CORELAP-Computerized Relational Layout Planning, " Jounal of Industrial Engineering," V18 N3, 1967, 196-200.

18.Seehof,J. M., and W. O. Evans , "Automated Layout Design Program," The Jounal of Industrial Engineering", V18, N12,1967,690-695.

19.Balas, Egon, "An Additive Algorithm for Solving Linear Programming with Zero-one Variable," Operation Research, V13, 1965, 517-546.

ภาคผนวก

ภาคผนวก ก.

การดำเนินงานของบริษัท ฐานเศรษฐกิจ จำกัด

การดำเนินงาน ของบริษัท ฐานเศรษฐกิจ จำกัด

บริษัท ฐานเศรษฐกิจ จำกัด แบ่งแยกหน่วยงานรับผิดชอบทั้งหมด 7 กอง คือ

1. กองบรรณาธิการ
2. กองโฆษณา
3. กองการเงิน และ ธุรการ
4. กองจัดการ
5. กองการผลิต
6. กองจัดจำหน่าย
7. กองกฎหมาย และ บุคคล

1. กองบรรณาธิการ

กองบรรณาธิการทำหน้าที่รับผิดชอบด้านเนื้อหาและลักษณะรูปเล่ม การจัดหน้าของหนังสือพิมพ์ทั้งหมด

1.1. โครงสร้างการบริหารของกองบรรณาธิการ

กองบรรณาธิการแบ่งออกได้เป็น 6 ฝ่าย

1. ฝ่ายธุรการกอง
2. ฝ่ายข่าวเช็คชั่น 1 ประกอบด้วย 3 สายงาน
   - สายที่ 1 เกษตรการค้า
   - สายที่ 2 ภูมิภาค
   - สายที่ 3 เศรษฐกิจการเมือง
3. ฝ่ายข่าวเช็คชั่น 2 ประกอบด้วยสายงานข่าว 4 สายงาน
   - สายงานที่ 1 ลงทุน - อุตสาหกรรม
   - สายงานที่ 2 งานก่อสร้าง - ที่ดิน
   - สายงานที่ 3 คมนาคม
   - สายงานที่ 4 ต่างประเทศ

4.ฝ่ายข่าวเช็คชั่น 3 ประกอบด้วย 2 สายงาน

สายงานที่ 1 สายการตลาด

สายงานที่ 2 ท่องเที่ยว

5.สายงานข่าวการเงิน (เช็คชั่น 4)

6.ฝ่ายบรรณาธิกรณ์ ประกอบด้วย 6 สายงาน

สายงานที่ 1 พิสูจน์อักษร

สายงานที่ 2 งานศิลป์

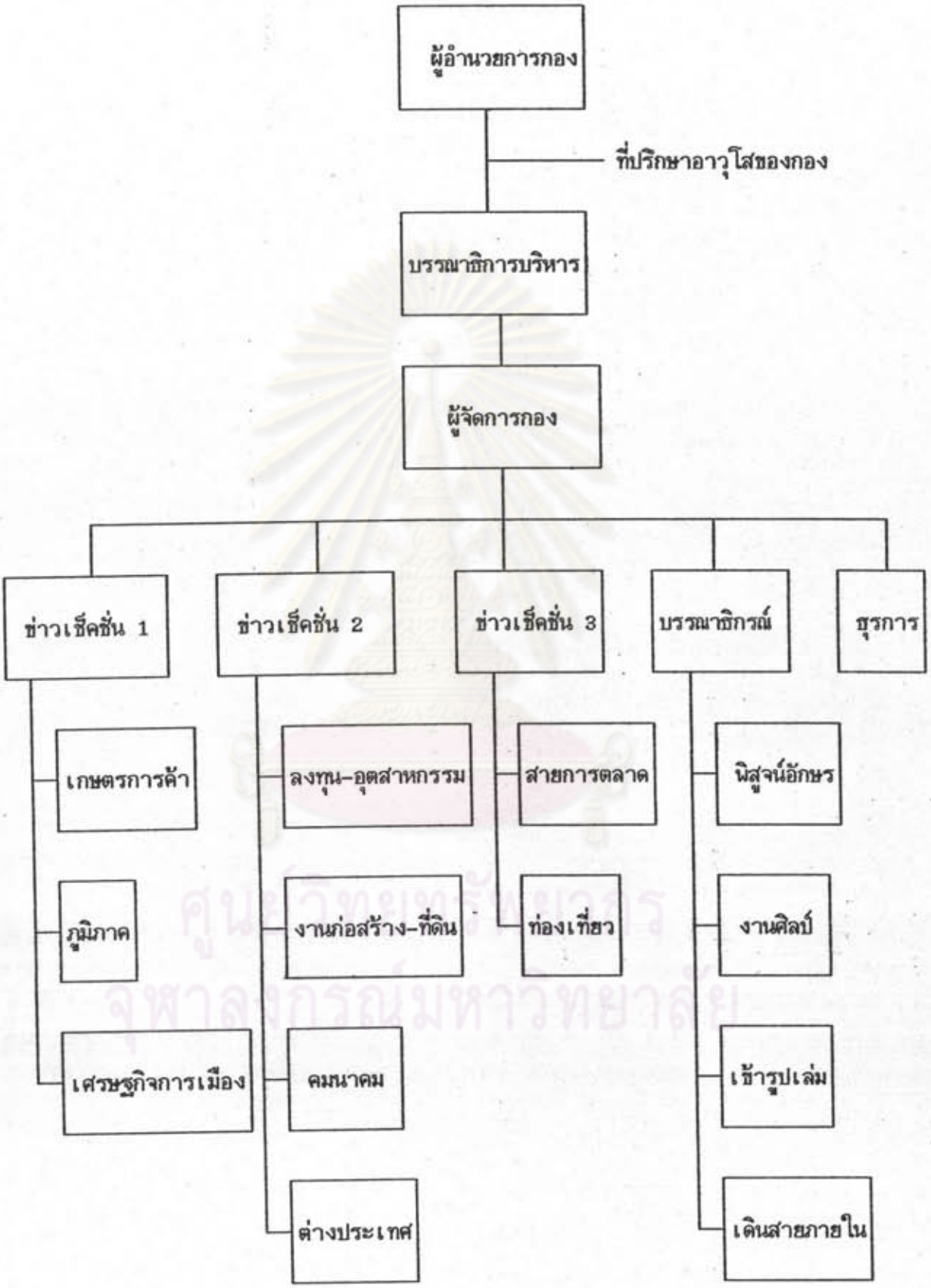
สายงานที่ 3 เข้ารูปเล่ม

สายงานที่ 4 เดินสายภายใน

สายงานที่ 5 กลุ่ม Rewriter

สายงานที่ 6 คอมพิวเตอร์

โครงสร้างของกองบรรณาธิการ แสดงดังรูปที่ 1


ผู้อำนวยการกอง
ที่ปรึกษาอาวุโสของกอง
บรรณาธิการบริหาร
ผู้จัดการกอง
ข่าวเช็คชั่น 1
ข่าวเช็คชั่น 2
ข่าวเช็คชั่น 3
บรรณาธิกรณ์
ธุรการ
เกษตรการค้า
ภูมิภาค
เศรษฐกิจการเมือง
ลงทุน-อุตสาหกรรม
งานก่อสร้าง-ที่ดิน
คมนาคม
ต่างประเทศ
สายการตลาด
ท่องเที่ยว
พิสูจน์อักษร
งานศิลป์
เข้ารูปเล่ม
เดินสายภายใน

1.2. จำนวนพนักงาน และ เนื้อที่ที่ต้องการ แสดงดังตารางที่ 1

	จำนวนพนักงาน	เนื้อที่(ตรม.)/คน	รวมเนื้อที่(ตรม.)
1.ผู้อำนวยการกอง	1	20	20
2.ที่ปรึกษาอาวุโสของกอง	3	20	60
3.บรรณาธิการบริหาร	1	20	20
4.บรรณาธิการเชคชั่น	6	20	120
5.ผู้จัดการกอง	1	20	20
6.สายธุรการ	7	4	28
7.ฝ่ายข่าวเช็คชั่น 1	15	4	60
8.ฝ่ายข่าวเช็คชั่น 2	14	4	56
9.ฝ่ายข่าวเช็คชั่น 3	13	4	52
10.สายงานข่าวการเงิน	8	4	32
11.ฝ่ายบรรณาธิกรณ์	27	4	108
12.ส่วนคอมพิวเตอร์	1 ห้อง	25	25
13.ห้องสมุด	1 ห้อง	50	50
14.ห้องประชุม	1 ห้อง	20	20
15.ที่เก็บเอกสาร & เครื่องใช้สำนักงาน		20	20
รวม	100 คน		695 ตรม.

ตารางที่ 1 จำนวนพนักงาน และเนื้อที่ทำงาน ของกองบรรณาธิการ

จำนวนพนักงานรวม 100 คน จำนวนเนื้อที่รวม 695 ตารางเมตร

2. กองโฆษณา

กองโฆษณา ทำหน้าที่รับผิดชอบด้านโฆษณาทั้งหมด ตั้งแต่การหาโฆษณาตลอดจนถึงการจัดทำโฆษณาในบับพิเศษ

2.1. โครงสร้างการบริหาร

กองโฆษณา แบ่งการทำงานเป็น 3 แผนก คือ

1. แผนกโฆษณาดิสเพลย์ หน้าที่ ออกไปเสนอขายสินค้า

แบ่งออกเป็น 5 ทีมงาน

1. ฝ่ายความคิด
2. ทีมเอเจนซี่
3. ทีมสายส่ง
4. ทีมแวดวงธุรกิจ
5. ทีมของผู้ช่วยผู้จัดการโฆษณาดิสเพลย์

2. แผนกบริการทอง หน้าที่ ออกไปติดต่อขายบริการ เช่น ขายที่ดิน รถยนต์ สมัครงาน และการศึกษา ประกอบด้วย 4 ทีมงาน

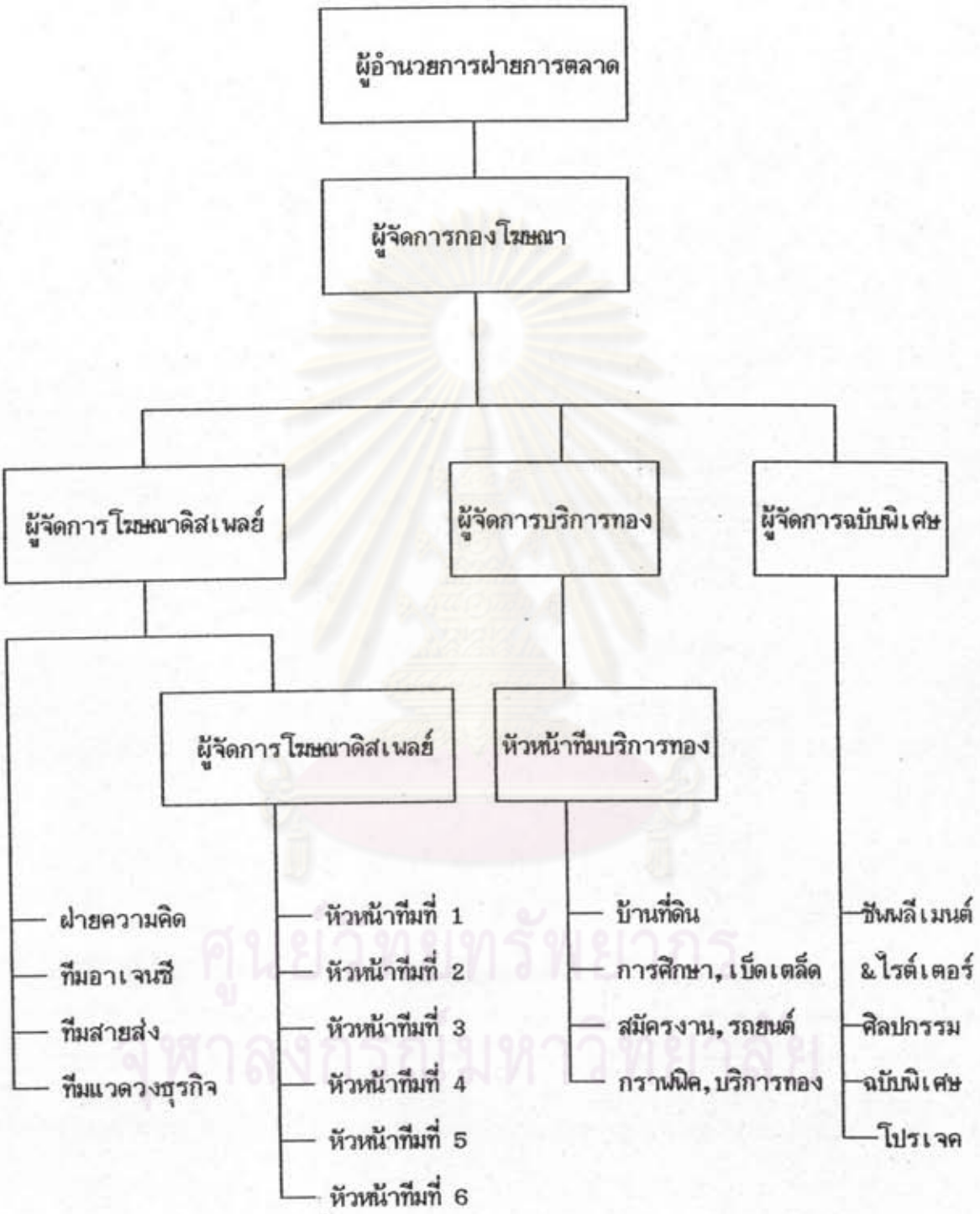
1. บ้านที่ดิน
2. การศึกษา, เบ็ดเตล็ด
3. สมัครงาน, รถยนต์
4. กราฟฟิค บริการทอง

3. แผนกฉบับพิเศษ (Supplyment) หน้าที่หาโอกาสพิเศษจัดทำ Supplement เช่น ครบรอบปีของบริษัทของลูกค้า

ประกอบด้วย 4 ส่วน

1. ส่วนซัพพลีเม้นต์-ไรท์เตอร์
2. ส่วนศิลปกรรม
3. ส่วนฉบับพิเศษ
4. ส่วนโปรเจค

โครงสร้างของกองโฆษณาแสดงได้ดังรูปที่ 2



รูปที่ 2 โครงสร้างการบริหารงานของกองโฆษณา

2.2. จำนวนพนักงาน และเนื้อที่ที่ต้องการ แสดงดังตารางที่ 2

	จำนวนพนักงาน	เนื้อที่/คน (ตรม.)	รวมเนื้อที่ (ตรม.)
1.ผู้อำนวยการกอง	1	20	20
2.ผู้จัดการกอง	4	16	64
3.แผนกดิสเพลย์	39	4	156
4.แผนกบริการทอง	17	4	68
5.แผนกฉบับพิเศษ	32	4	128
6.ที่เก็บเอกสาร & อุปกรณ์สำนักงาน		20	20
รวม	93		456

ตารางที่ 2 จำนวนพนักงาน และเนื้อที่ ของกองโฆษณา

รวมพนักงาน = 93 คน

พื้นที่ที่ต้องการรวม = 456 ตารางเมตร

3. กองการเงินและธุรการ

กองการเงินและธุรการ ทำหน้าที่ด้านการจัดการเกี่ยวกับการเงินของบริษัท รวมทั้งหน้าที่ด้านการจัดซื้อ

3.1. โครงสร้างการบริหาร

กองการเงินและธุรกิจ แบ่งออกได้ 4 ส่วน

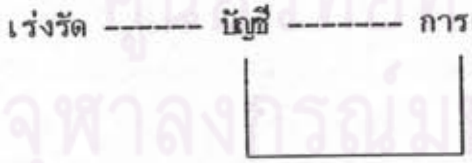
1. ส่วนบัญชี ทำหน้าที่บันทึกเกี่ยวกับค่าใช้จ่าย รายได้ต่าง ๆ เช่น บันทึกบัญชีลูกหนี้ค่าโฆษณา

2. ส่วนการเงิน ทำหน้าที่ ควบคุมการใช้เงิน และทำ cash flow

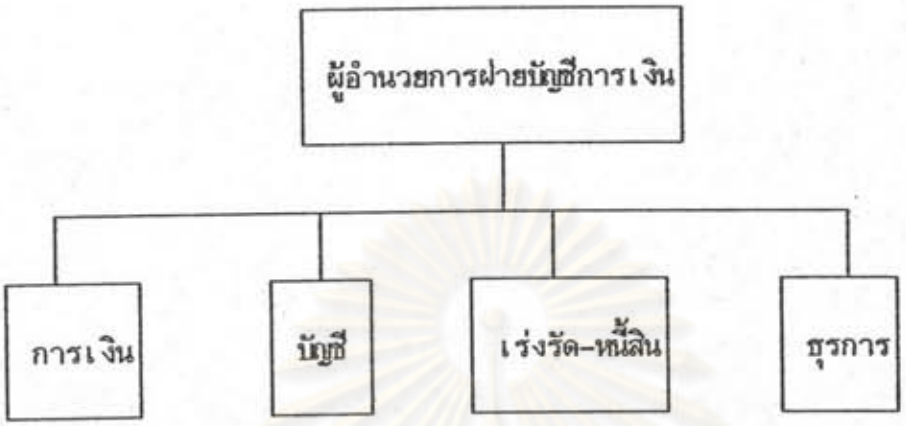
3. ส่วนเร่งรัด-หนี้สิน ทำหน้าที่ ตรวจสอบการโฆษณา ออกใบแจ้งหนี้ ใบเสร็จรับเงินและวางบิลล์

4. ส่วนธุรการ ทำหน้าที่ให้บริการเกี่ยวกับเรื่องการเบิกจ่าย วัสดุสำนักงาน

การดำเนินงานเริ่มจากฝ่ายเร่งรัดหนี้สิน ออกใบเสร็จที่เก็บจากลูกค้า ฝ่ายบัญชีจะทำการบันทึกเป็นลูกหนี้ และเรียกเก็บจากลูกหนี้ ฝ่ายการเงินจะนำเงินเข้าบัญชี และทำการแจ้งให้ทางบัญชีทราบ เพื่อตัดออกจากลูกหนี้



โครงสร้างของการเงินและธุรการแสดงดังรูปที่ 3



รูปที่ 3 โครงสร้างการบริหารงานของกองการเงิน และธุรการ

3.2. จำนวนพนักงานและความต้องการเนื้อที่ แสดงดังตารางที่ 3

	จำนวนพนักงาน	เนื้อที่/คน(ตรม.)	รวมเนื้อที่(ตรม.)
1.ผู้อำนวยการฝ่ายการบัญชีการเงิน	1	25	25
2.ผู้จัดการกองการเงิน	1	16	16
3.สมุห์บัญชี	1	16	16
4.ส่วนบัญชี	3	4	12
5.ส่วนการเงิน	3	4	12
6.ส่วนเร่งรัด-หนี้สิน	11	4	44
7.ส่วนธุรการ	2	4	8
8.ห้องเก็บหนังสือ	1 ห้อง	20	20
9.ห้องสโตร์	2 "	20	40
10.พื้นที่วางอุปกรณ์สำนักงาน , โต๊ะนิมพ์ดีด และ เครื่องถ่ายเอกสาร			10
11. โต๊ะยาวสำหรับพนักงานส่งเอกสาร			12
12. ส่วนเก็บเอกสาร			15
รวม	22 คน		230

ตารางที่ 3 จำนวนพนักงานและความต้องการเนื้อที่

4. กองกิจการทั่วไป (กองส่งเสริมและพัฒนา)

รับผิดชอบจัดการธุรกิจทุกอย่างที่เกี่ยวกับบริษัท จัดกิจกรรมพิเศษต่าง ๆ ในบริษัท อาทิการจัดแข่งขันกีฬาภายใน เป็นต้น

4.1. โครงสร้างการบริหาร

แบ่งออกเป็น 5 สายงาน ดังต่อไปนี้

สายงานที่ 1 แผนกจัดซื้อ

จัดหาอุปกรณ์สำนักงานให้กับฝ่ายต่าง ๆ ตามจดหมายอนุมัติจากท่านกรรมการผู้อำนวยการในลักษณะจัดซื้อด้วยเงินสด หรือทำสัญญาบาร์เตอร์แลกเปลี่ยนด้านโฆษณาสินค้าในหนังสือพิมพ์ฐานเศรษฐกิจ

หมายเหตุ การจัดซื้อเฉพาะสินค้าจำพวกอุปกรณ์สำนักงาน เช่น เครื่องใช้ไฟฟ้า, โต๊ะ , เก้าอี้, เป็นต้น

ยกเว้น การจัดซื้อสินค้าจำพวก เครื่องเขียน

สายงานที่ 2. แผนกงานช่าง

มีหน้าที่รับผิดชอบ งานซ่อมแซมหรือเพิ่มเติม ส่วนต่าง ๆ ตามสภาพ ตามความเหมาะสม เพื่ออำนวยความสะดวก ในการปฏิบัติงานของฝ่ายต่าง ๆ โดยแยกออกเป็น 4 ส่วน ดังต่อไปนี้

2.1 งานอาคารสำนักงานฐานเศรษฐกิจ ซ่อมแซมส่วนที่ทรุดโทรมให้คงสภาพเดิมใช้การได้ดี เช่น รอยแตกรอยร้าวของตัวอาคาร การทาสีใหม่ ซ่อมประตู หน้าต่าง เป็นต้น

2.2 งานเครื่องอุปกรณ์ไฟฟ้า ประกอบด้วย เครื่องปรับอากาศ เครื่องพิมพ์ดีดไฟฟ้า เครื่องโทรศัพท์ เครื่องคิดเลข หลอดไฟฟ้า เหล่านี้เป็นต้น ต้องคอยดูแลรักษาให้ใช้งานได้ดีตลอดเวลา

2.3 งานน้ำประปา ประกอบด้วย ท่อน้ำใช้ ท่อน้ำทิ้ง อ่างล้างหน้า โถน้ำ และอุปกรณ์ต่าง ๆ ที่ประกอบการใช้เกี่ยวกับใช้น้ำประปา ตลอดจน เครื่องปั๊มน้ำ ต้องหมั่นดูแลรักษาให้ใช้ได้ดีเสมอ

2.4 งานอุปกรณ์สำนักงาน ประกอบด้วย โต๊ะ เก้าอี้ ตู้เก็บเอกสาร เป็นต้น ต้องคอยดูแลรักษา

หมายเหตุ พนักงานแผนกช่าง ทางกองยังมีความต้องการเพิ่ม 2 อัตราประจำแผนก คือ

1. พนักงาน ช่างไฟฟ้า โทรศัพท์
2. พนักงาน ช่างก่อสร้าง

เพื่อเป็นการเตรียมตัวล่วงหน้า ให้เรียนรู้ระบบต่าง ๆ ที่เกี่ยวข้อง ของอาคารหลังใหม่ที่กำลังอยู่ในระหว่างการก่อสร้าง

สายงานที่ 3. แผนกแม่บ้าน

รับผิดชอบงานดูแลรักษาความสะอาด ตัวอาคารฐานเศรษฐกิจ และบริเวณรอบ ๆ อาคาร พนักงานแม่บ้านทางกองส่งเสริมและพัฒนามีความเห็นว่าสมควร หาสถานที่จัดออกไปฝึกงานแม่บ้านและการเสริฟ อาหารและเครื่องดื่ม กับโรงแรมที่มีมาตราฐาน เช่น โรงแรมไฮแอทเซ็นทรัล เป็นต้น เตรียมไว้สำหรับอาคารหลังใหม่

สายงานที่ 4. แผนกโอเปเรเตอร์ และการประชาสัมพันธ์

พนักงานแผนกนี้มีความสำคัญ ต่อรูปบริษัทฯ เป็นอย่างยิ่ง เปรียบเสมือนเป็นหน้าด่านของบริษัทกับบุคคลที่จะขอเข้าติดต่อทำธุระกับบริษัท ไม่ว่าจะเป็นการติดต่อด้วยทางโทรศัพท์หรือมาพบด้วยตนเอง ฉนั้นทางแผนก มีความเห็นว่า อาคารหลังใหม่ควรจะเปลี่ยนพนักงานแผนกนี้เสียใหม่ ควรจะมีคุณสมบัติครบถ้วนของพนักงานประชาสัมพันธ์ เช่น ด้านการศึกษา บุคคลิกที่ดี ผู้พบเห็นมีความรู้สึกอยากติดต่อด้วย แผนกโอเปเรเตอร์ มีหน้าที่รับผิดชอบ การโอนสายโทรศัพท์ งานแจกจ่ายเอกสารที่มาจากภายนอกไป ยังปลายทางที่เป็นเจ้าของเรื่อง และ งานประชาสัมพันธ์ ต้อนรับชี้แจงถึงปัญหาภายในบริษัทให้แก่ผู้ที่มาติดต่อให้เข้าใจ ทรัพย์สินของบริษัทตลอด 24 ชั่วโมง โดยแบ่งความรับผิดชอบออกเป็น 2 ระยะ คือ เวลา 06.00 - 18.00 น. และ 18.00 - 06.00 น.

หมายเหตุ เมื่อเปิดที่ทำการอาคารใหม่ควรมีพนักงานของบริษัท ฐานเศรษฐกิจ จำกัด 1 ท่าน ที่ได้รับการฝึกอบรม ด้านทหารหรือพนักงานรักษาความปลอดภัย คอยควบคมดแลความมีระเบียบวินัยที่ดีของพนักงานรักษาความปลอดภัย เพื่อประสิทธิภาพของคนและงาน

4.2. จำนวนพนักงาน และความต้องการพื้นที่ แสดงดังตารางที่ 4

	จำนวนพนักงาน	เนื้อที่/คน(ตรม.)	รวมเนื้อที่(ตรม.)
1.ผู้จัดการกองกิจการทั่วไป	1	16	16
2.ช่าง	2	4	8
3.เลขา	1	4	4
4.ที่เก็บของและที่วางโต๊ะนิมพ์ดีด			5
5.ที่เก็บสิ่งของ เบิกของ			5
6.แม่บ้าน			0
7.คนขับรถ			0
8.พนักงานรักษาความปลอดภัย			0
รวม		24	38

ตารางที่ 4 จำนวนพนักงานและความต้องการเนื้อที่ ของกองกิจการทั่วไป

5. <u>กองการผลิต</u>

กุองการผลิต รับผิดชอบเกี่ยวกับการควบคมการผลิตของหนังสือพิมพ์รายเศรษฐกิจและสิ่งพิมพ์ในเครือทั้งหมด

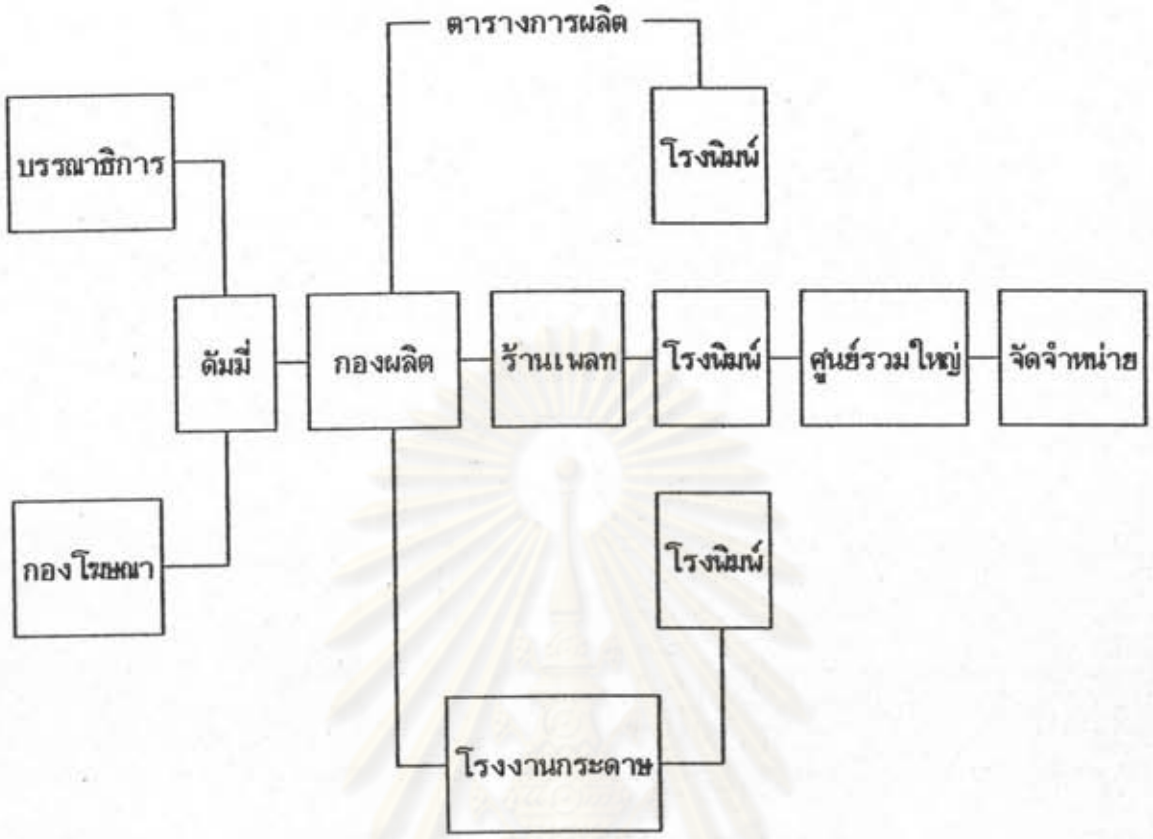
ความผิดชอบของฝ่ายผลิต มี 3 ส่วน คือ

1. หนังสือพิมพ์
2. ฉบับพิเศษ
3. ฉบับพิเศษของบริษัทในเครือ เช่น ฐานการตลาด , MDR และ Young Executive

<u>ขั้นตอนการทำงาน</u>

1. การทำหนังสือพิมพ์

กองบรรณาธิการและกองโฆษณาจะทำดัมมี่มาให้ กองการผลิตจะจัดทำ traffic (ตารางการพิมพ์) และสั่งกระดาษไปยังโรงพิมพ์ และส่งข่าวไปยังร้านเพลท 3 แห่ง ร้านเพลททำตามดัมมี่ จากนั้นร้านเพลทจะส่งเพลทไปยังโรงพิมพ์ทั้งหมด 12 โรง และส่งไปยังศูนย์รวมใหญ่ เพื่อเข้าเล่ม



รูปที่ 5 การทำหนังสือพิมพ์

5.1. โครงสร้างการบริหาร

กองการผลิต แบ่งออกเป็น 2 ส่วน คือ

1. ส่วนคอมพิมพ์ ทำหน้าที่ทำตัวหนังสือออกมาให้บรรณาธิการประจำวัน
2. ส่วนการผลิต ทำหน้าที่

2.1 ทำ plate ดูความถูกต้องของโฆษณา เนื้อเรื่อง

2.2 ตรวจดูหน้า บนหน้า เข้าศูนย์รวมเพื่อเข้าเล่ม เช็คเวลา

5.2. จำนวนพนักงานและเนื้อที่ แสดงดังตารางที่ 5

	จำนวนพนักงาน	เนื้อที่/คน(ตรม.)	รวมเนื้อที่(ตรม.)
1.ผู้จัดการกองการผลิต	1	16	16
2.พนักงาน			
(ไม่รวม passtime 3 คน)	11	4	44
3.ที่เก็บเอกสาร +เครื่องใช้สำนักงาน		10	10
4.เครื่องคอมพิมพ์		8	8
รวม			78 ตรม.

ตารางที่ 5 จำนวนพนักงานและความต้องการเนื้อที่ ของกองการผลิต

พนักงาน 15 คน พื้นที่รวม 78 ตรม.

6. กองจัดจำหน่าย

กองจัดจำหน่าย รับผิดชอบในการจัดจำหน่ายของหนังสือพิมพ์รายเศรษฐกิจในขั้นต้นก่อนที่จะแกงานต่อไปยังบริษัท ฐานการตลาด และบริษัท ฐานรวมห่อ จำกัด

6.2. จำนวนพนักงานและความต้องการเนื้อที่ แสดงได้ตารางที่ 6

	จำนวนพนักงาน	เนื้อที่/คน(ตรม.)	รวมเนื้อที่(ตรม.)
1.ผู้จัดการกองจัดจำหน่าย	1	16	16
2.ผู้ช่วยผู้จัดการกองจัดจำหน่าย	1	4	4
3.เลขานุการกอง	1	4	4
4.พนักงานทั่วไป	2	4	8
5.พื้นที่ใช้สอยสำนักงาน			10
รวม	5 คน		42 ตร.ม.

ตารางที่ 5 จำนวนพนักงานและความต้องการเนื้อที่ ของกองจัดจำหน่าย

พนักงานรวม 5 คน

เนื้อที่ทั้งหมด 42 ตารางเมตร

7. กองกฎหมาย และ บุคคล

กองกฎหมายและบุคคล ทำหน้าที่เป็นฝ่ายบุคคลบริษัท รับผิดชอบเรื่องที่เกี่ยวกับกฎหมาย ปรึกษาปัญหากฎหมายให้แก่พนักงานทุกคนฟรี จัดการด้านการสรรหาบุคคลเข้าทำงานในบริษัทและการวางแผนงานบุคคล เป็นต้น

7.1. โครงสร้างการบริหาร

แบ่งออกเป็น 2 หน่วยงาน ดังนี้

1. งานกองกฎหมาย มีหน้าที่

1) จดทะเบียนบริษัท/ในเครือ
2) ให้คำปรึกษาด้านกฎหมาย/ว่าความเป็นทนายจำเลยในกรณีถูกฟ้องร้อง
3) เรื่องเกี่ยวกับที่ดินและนิติกรรมต่าง ๆ
4) จัดวาระและรายงานการประชุมของบริษัท/ในเครือ
5) จัดทำ จัดเก็บและเตรียมเอกสารเกี่ยวกับกฎหมายของบริษัท/ในเครือ
6) งานที่ผู้อำนวยการมอบหมายเป็นพิเศษ
7) การทำบัญชีผู้ถือหุ้นและการส่งบัญชีงบดุล งบกำไรขาดทุน ต่อกรมทะเบียนการค้าของบริษัท/ในเครือ

2. งานกองบุคคล มีหน้าที่

1) รับสมัครพนักงานและสัมภาษณ์ขั้นต้น
2) พิมพ์บัตรบันทึกเวลา บริษัท/ในเครือ
3) รายงานการขาด-ลา
4) รายงานการเข้า - ออกงาน
5) ร่วมพิจารณาโบนัส
6) ร่วมพิจารณาเงินเดือน
7) ออกหนังสือรับรอง
8) ร่างและเสนอกฎระเบียบของบริษัทฯ
9) อบรม/ดูแลพนักงานบริษัท/ในเครือ

10) เสนอปรับ/บรรจุ พนักงานที่ครบการทดลองงาน

11) จัดทำบัตรประจำตัวพนักงานของบริษัท/ในเครือ

12) ด้านสวัสดิการต่าง ๆ

- การรักษาพยาบาล
- การประกันอุบัติเหตุ
- การอุปสมบท
- การสมรส
- พิธีงานศพ

โครงสร้างการบริหารงารของกฎหมาย - บุคคล แสดงได้ดังรูปที่ 7

รูปที่ 7 โครงสร้างการบริหารงานของกอง กฎหมาย - บุคคล

7.2. จำนวนพนักงานและความต้องการเนื้อที่ แสดงดังตารางที่ 7

	จำนวนพนักงาน	เนื้อที่/คน(ตรม.)	รวมเนื้อที่(ตรม.)
1.ผู้อำนวยการฝ่ายบริหาร	1	20	20
2.ผู้จัดการกอง	1	16	16
3.ผู้ช่วยผู้จัดการ	2	4	8
4.เลขา	1	4	4
5.ส่วนเก็บเอกสาร	1	4	4
รวม	5 คน		48 ตร.ม.

ตารางที่ 7 จำนวนพนักงานและความต้องการเนื้อที่ ของกองกฎหมาย - บคคล

พนักงานรวม 5 คน

ความต้องการใช้พื้นที่ 48 ตารางเมตร

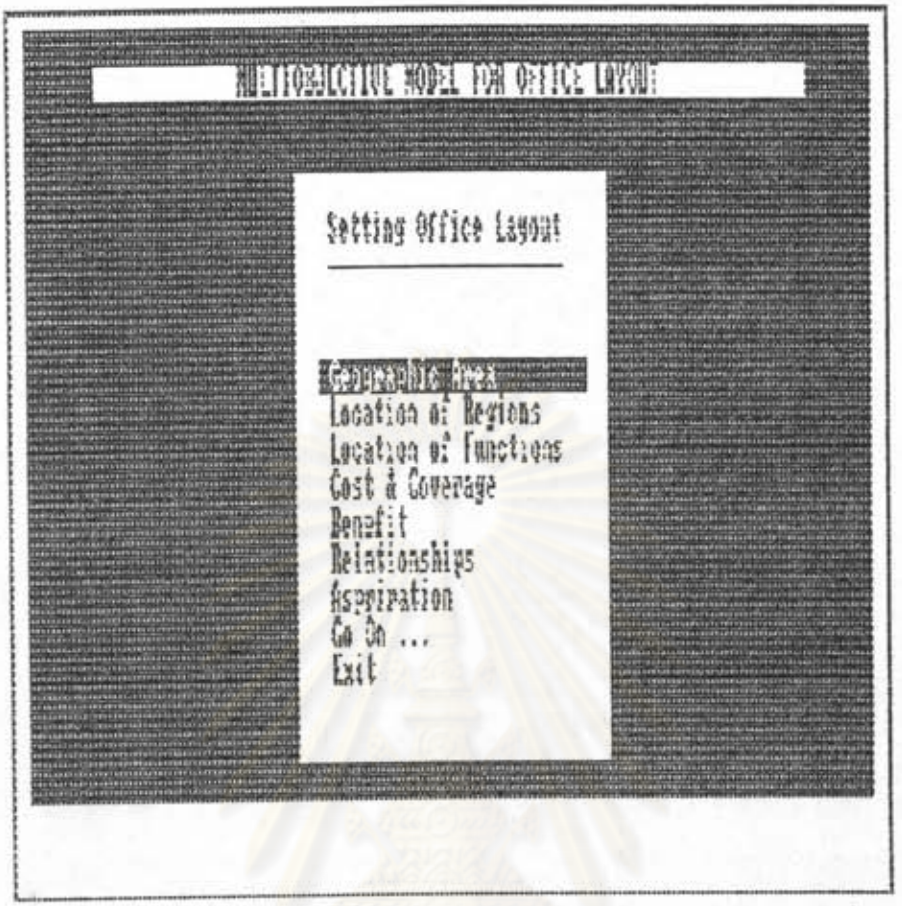
ภาคผนวก ข

ผลลัพธ์ของโปรแกรม แบบจำลองชนิดหลายเป้าหมาย

รูปที่ 1 การเข้าสู่ โปรแกรม

การเข้าสู่โปรแกรม ใช้คำสั่ง A>OFFICE จะเกิดภาพ ดังรูปที่ 1

กด ENTER จะเกิดภาพดังรูปที่ 2



รูปที่ 2 Setting Office Layout

จากรูปที่ 2 ผู้ใช้จะต้อง Set Office Layout ได้แก่

- Geographic Area
- Location of region
- Location of Function
- Cost & coverage
- Benefit
- Relationships
- Aspiration

เมื่อกดเลือก Geographic Area จะเกิด ภาพดังรูปที่ 3

รูปที่ 3 Geographic Area

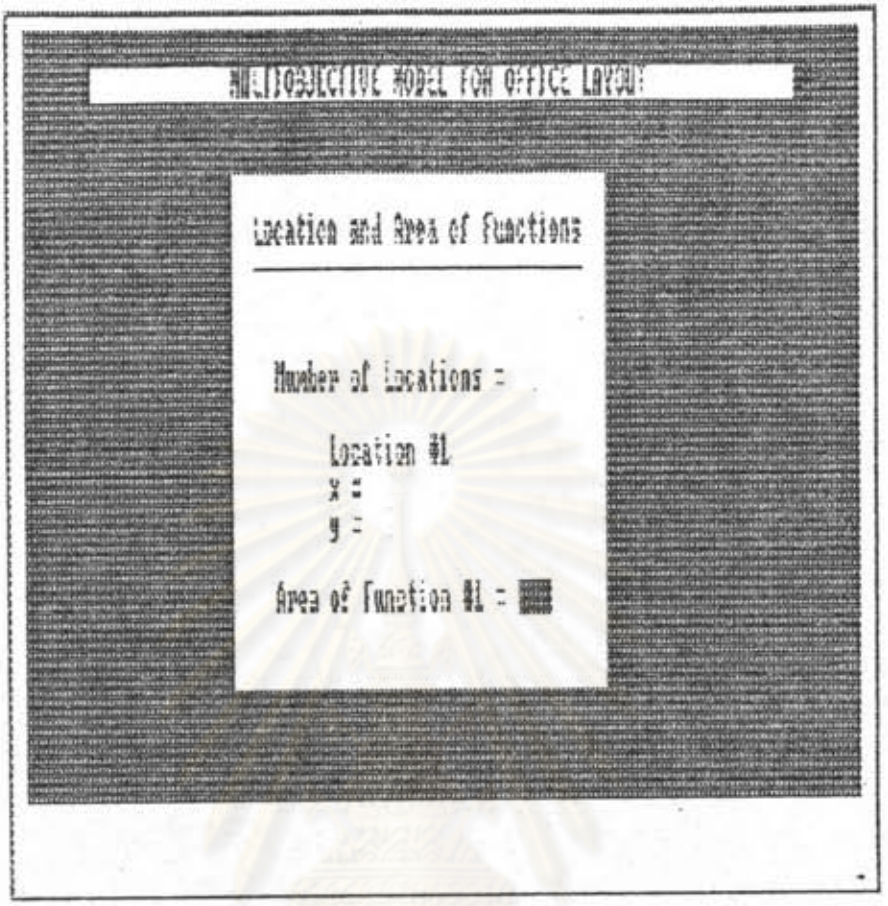
จากรูปที่ 3 จะต้องกำหนด

- จำนวน Region
- จำนวน Function
- ขนาดพื้นที่ทั้งหมด
- จำนวนมุมของพื้นที่ ถ้าเป็นสี่เหลี่ยมจัตุรัส จะมี 4 มุม
- โคออดิเนตของแต่ละมุม

เมื่อกด Location of Region จะเกิดภาพดังรูปที่ 4

รูปที่ 4 Location of Regions

จากรูปที่ 4 กำหนด โคออดิเนตของ Region และ พื้นที่ ของ Region เมื่อกด Location of Function จะเกิดภาพดังรูปที่ 5

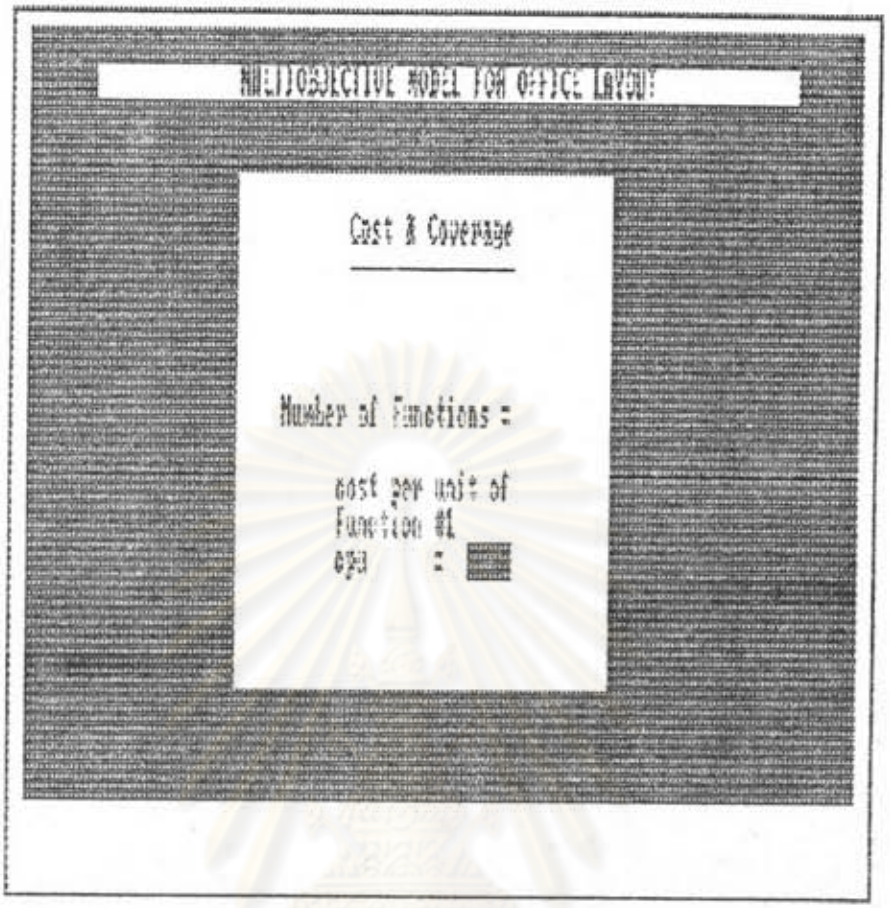


รูปที่ 5 Location of Functions

จากรูปที่ 5 กำหนด - จำนวน Location ที่จะเลือก

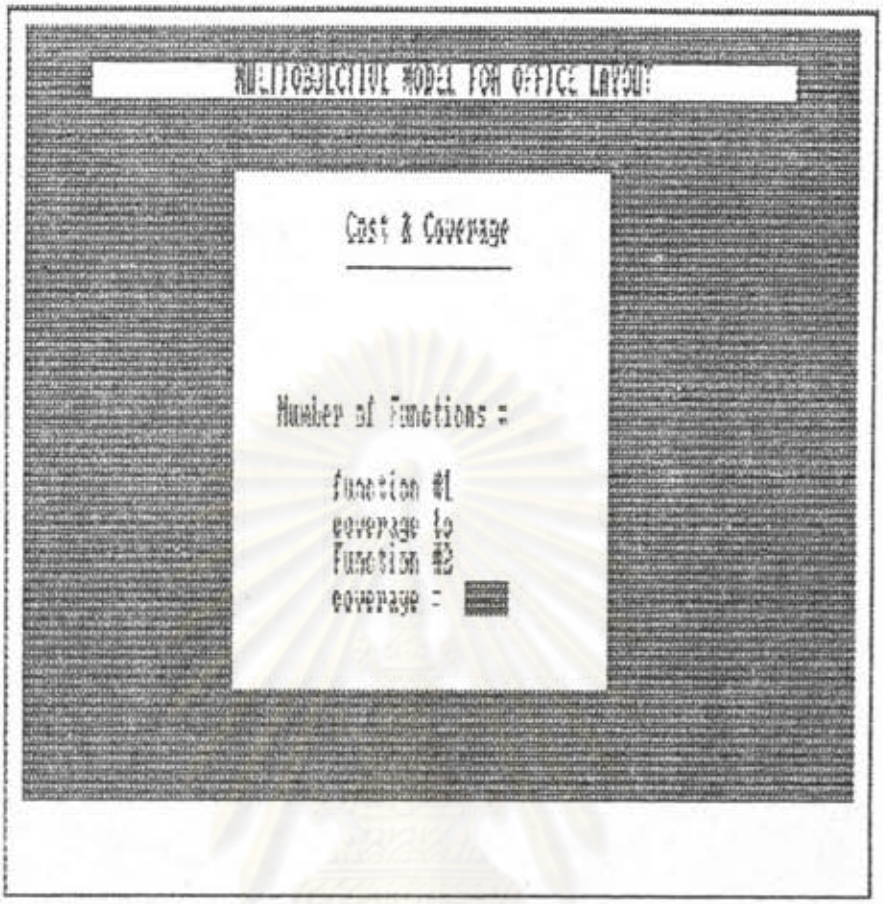
- โคออดิเนตของ Location ต่าง ๆ
- ขนาดพื้นที่ของแต่ละ Function

เมื่อกด Cost & Coverage จะเกิดภาพดังรูปที่ 6 , 7



รูปที่ 6 Cost ของ Function

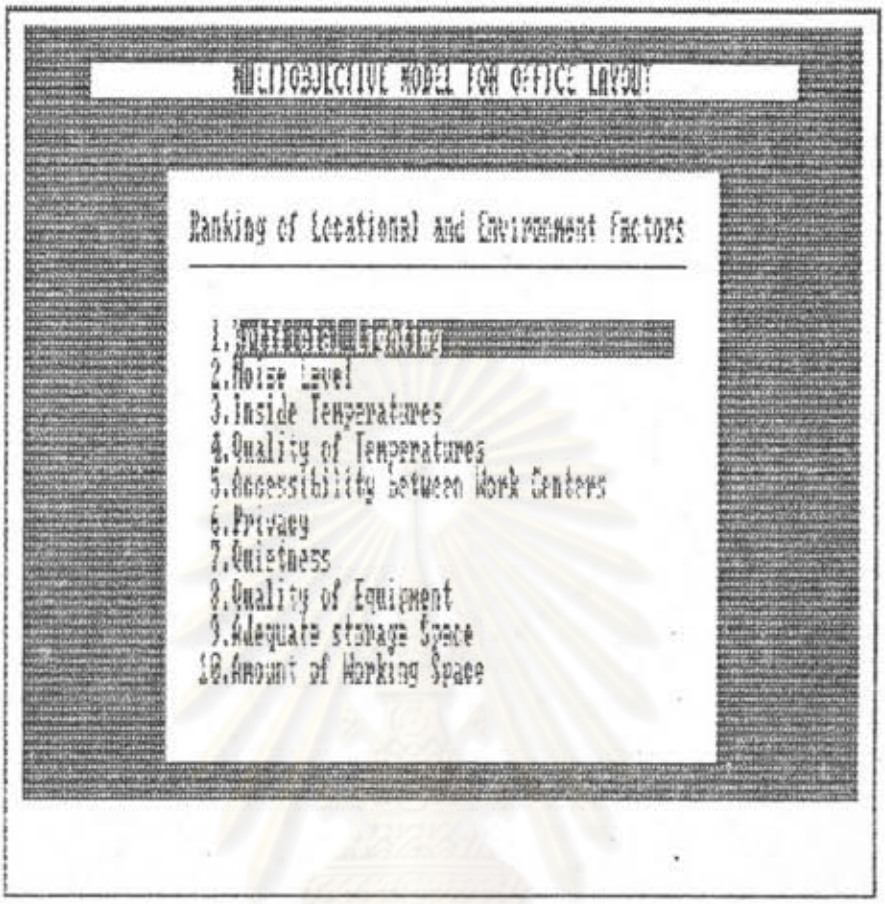
จากรูปที่ 6 กำหนด Cost ของแต่ละ Function



รูปที่ 7 Coverage ระหว่าง Function

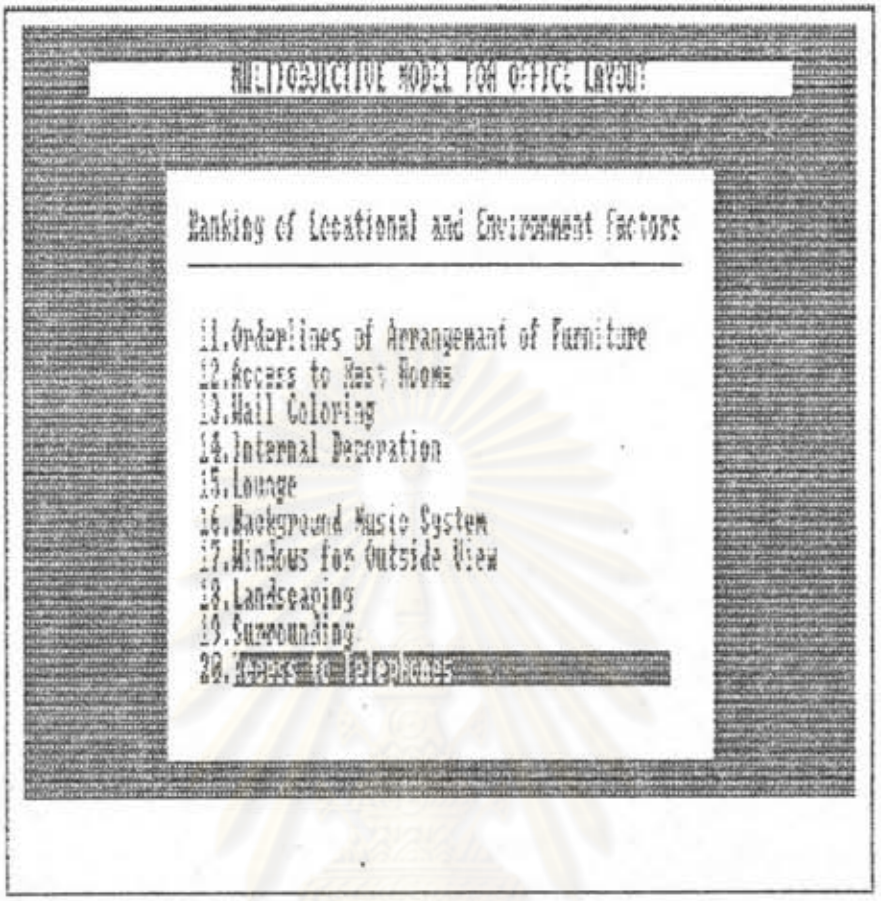
จากรูปที่ 7 กำหนด Coverage ระหว่าง Function

เมื่อกด Benefit จะเกิดภาพดังรูปที่ 8 , 9

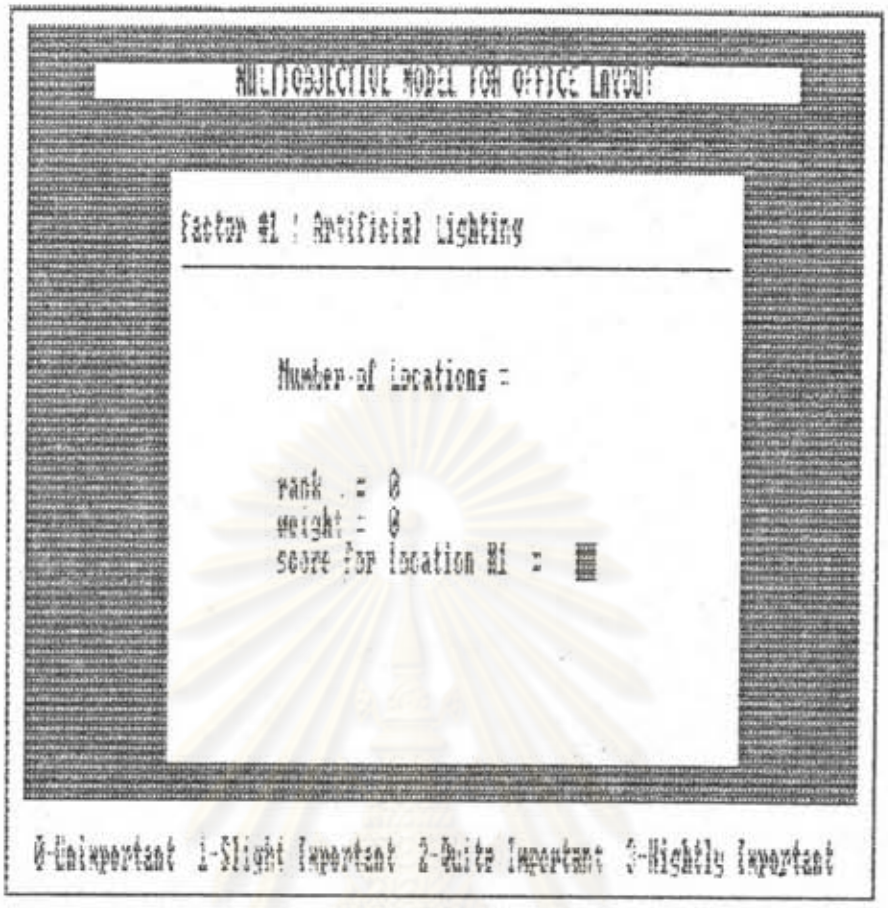


รูปที่ 8 Environment Factor

จากรูปที่ 8 , 9 จะแสดงปัจจัยทางสภาพแวดล้อมต่าง ๆ ในแต่ละปัจจัยจะต้องกำหนด Rank , Weigth ของแต่ละปัจจัย และ Score สำหรับ Location ต่าง ๆ ดังรูปที่ 10 ทำจนครบทุกปัจจัย (อาจจะตัดบางปัจจัยออกก็ได้)

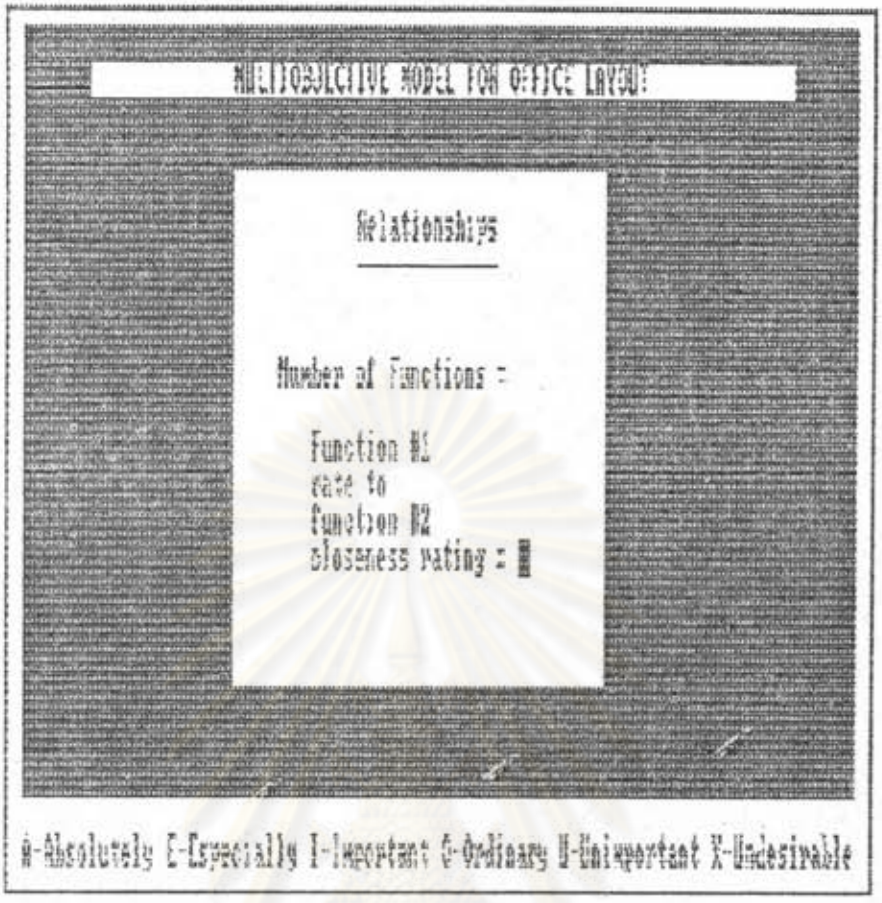


รูปที่ 9 Environment Factor (ต่อ)



รูปที่ 10 การกำหนด Rank , weight , score ของแต่ละปัจจัย

เมื่อกด Relationship จะเกิด ภาพดังรูปที่ 11



รูปที่ 11 ความสัมพันธ์ระหว่าง Function

จากรูปที่ 11 กำหนดความสัมพันธ์ระหว่าง Function โดย

A = สัมพันธ์มากที่สุด

E = สัมพันธ์มาก

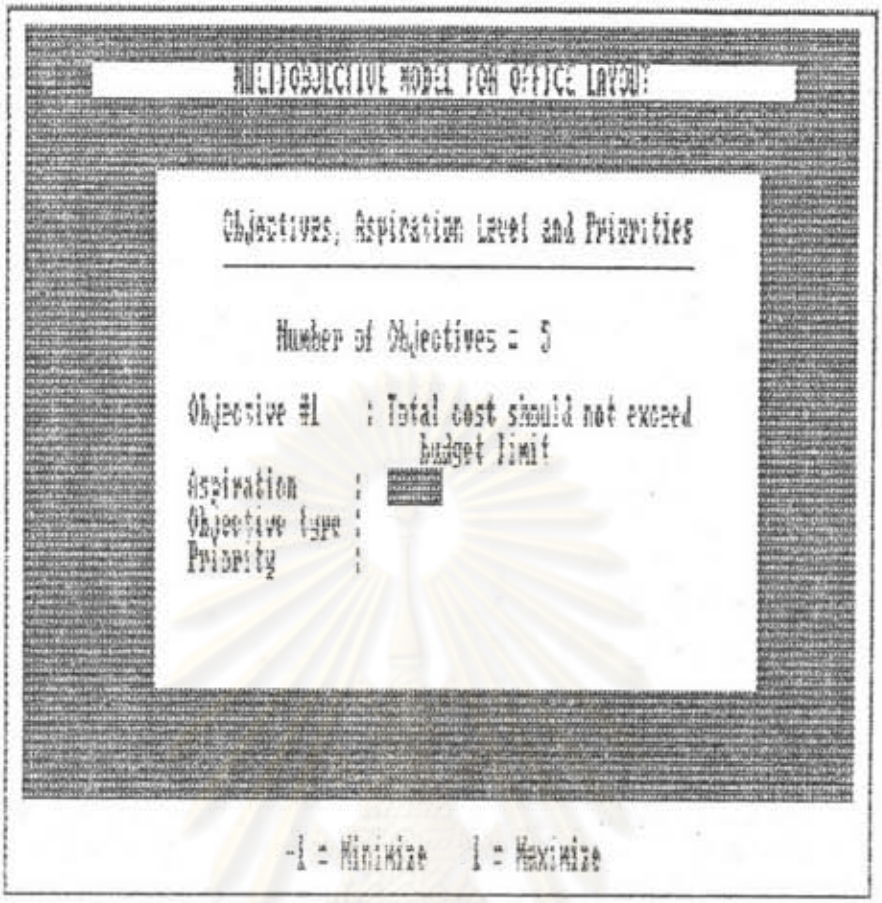
I = สัมพันธ์

O = สัมพันธ์น้อย

U = สัมพันธ์น้อยมาก

X = ไม่สัมพันธ์กันเลย

เมื่อกด Aspiration จะเกิดภาพดังรูป 12 - 15

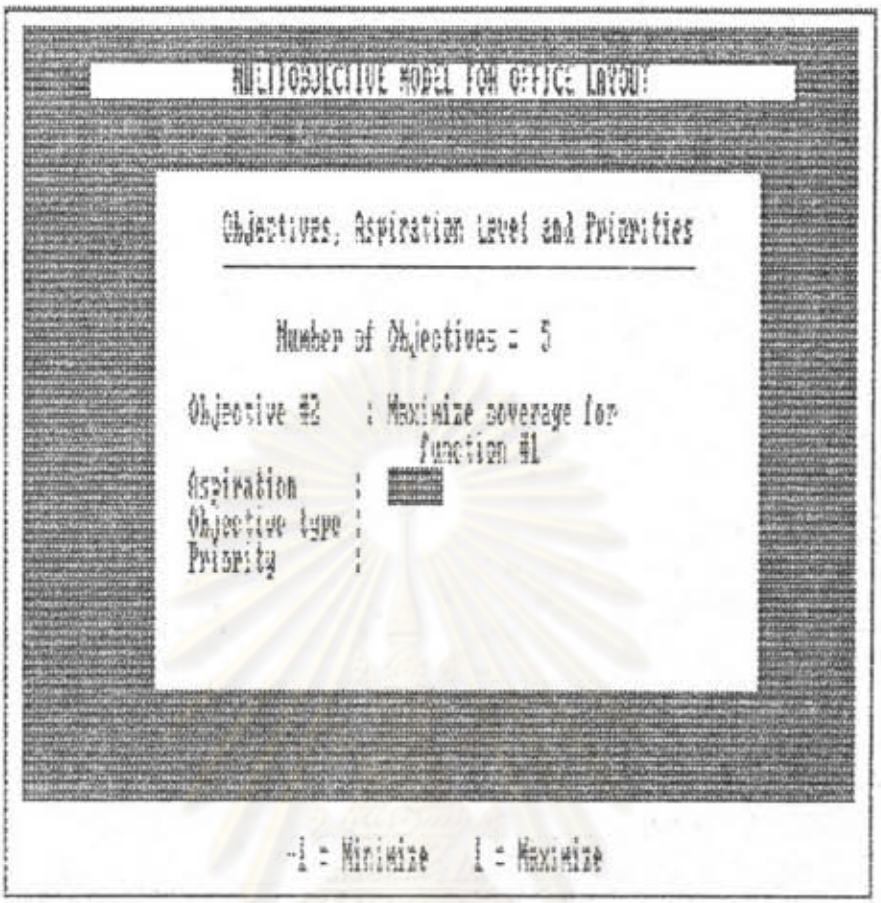


รูปที่ 12 การกำหนดเป้าหมายต่าง ๆ

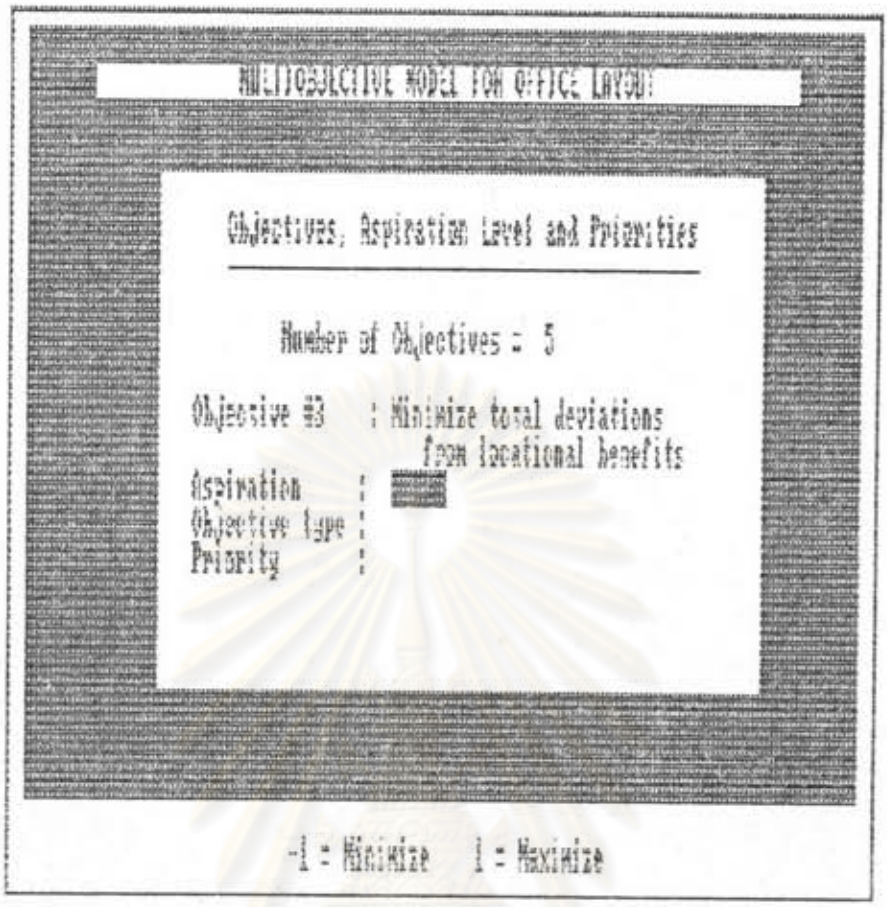
กำหนด - ค่าทางขวามือ (Aspiration)

- ชนิดของเป้าหมาย ถ้าเป็น Maximize ใส่ -1 minimize ใส่ 1

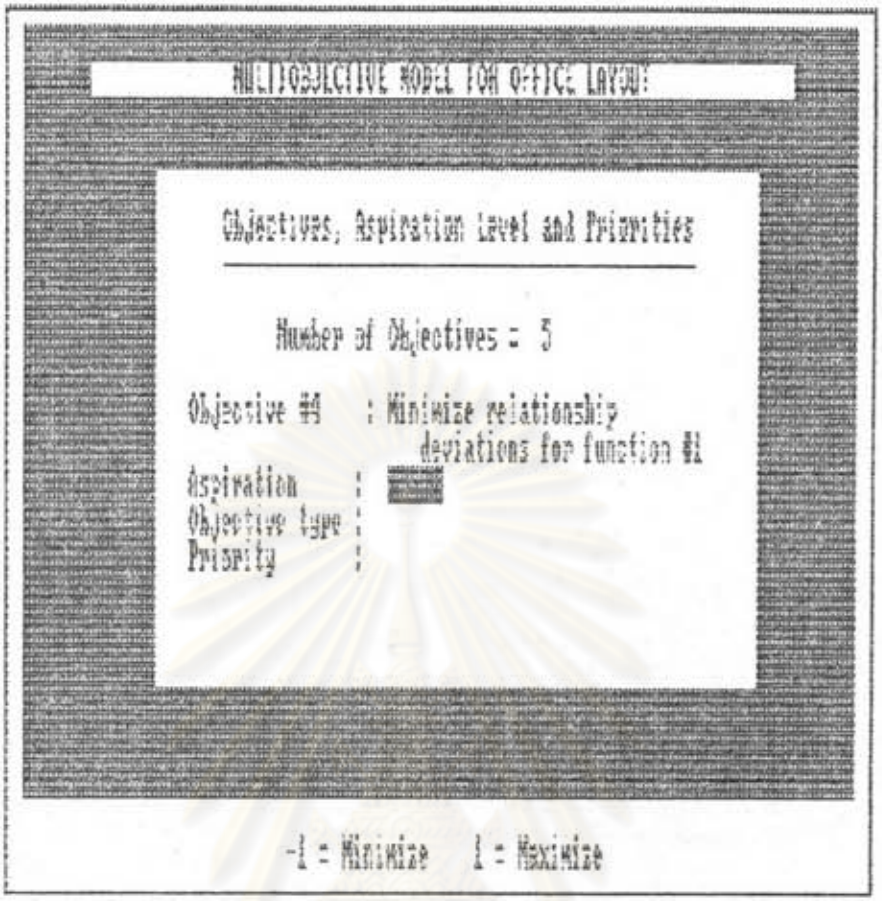
กระทำทุกเป้าหมายจนจบ



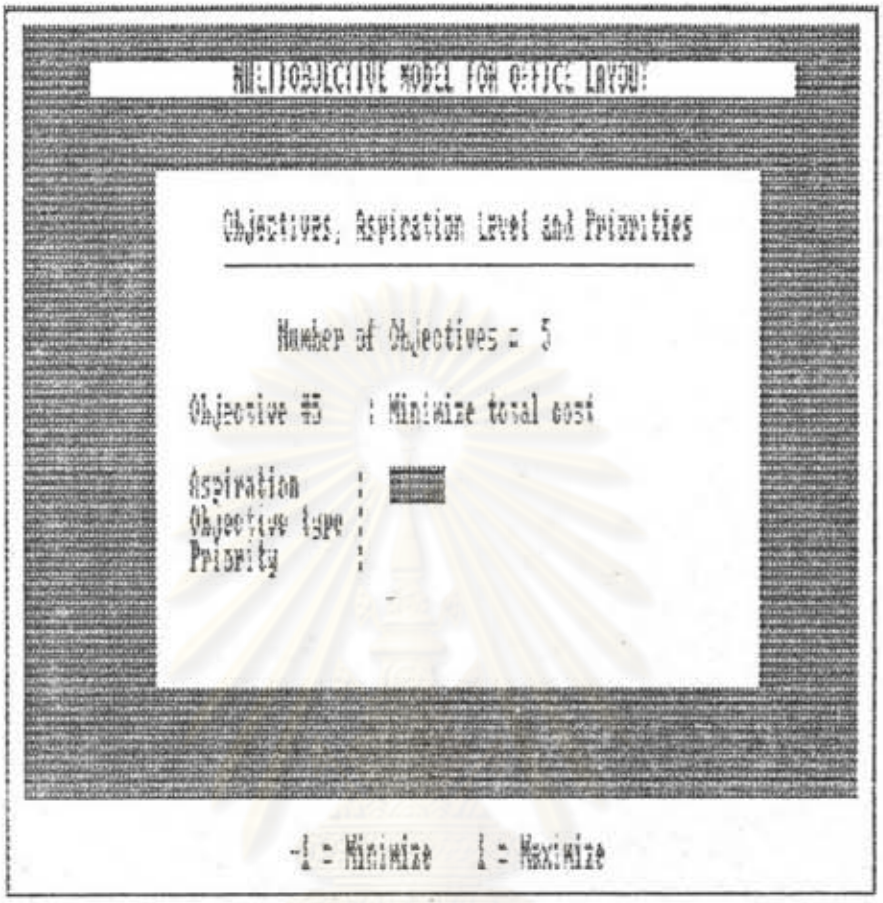
รูปที่ 13 การกำหนดเป้าหมายที่ 2



รูปที่ 14 การกำหนดเป้าหมายที่ 3



รูปที่ 15 การกำหนดเป้าหมายที่ 4



รูปที่ 16 การกำหนดเป้าหมายที่ 5

หลังจากนั้น ให้กด GO ON และกด ESC จะเกิดภาพ LAY OUT

OFFICE.PRJ

```
office
screen
grf
pie
```

COMPUTER PROGRAM

```
void scroll_down(int nline, char attr, int x1, int y1, int x2, int y2);
void changeattribute(int page,int row,int col,int length,int newattrib);
void write_char(int x, int y, char ch, char attrib);
void write_string(int x, int y, char *st, char attrib);
void fastwrite(int page,int row,int col,int attribute,char *string);
void fastwriteva (int page,int row,int col,int attrib,char *msg, ...);
void txtbar(int x1, int y1, int x2, int y2, char ch, char attrib);
void draw_box(int x1, int y1, int x2, int y2, char linestyle, char lineattri,
                char *headwin, char head_attrib, char hstyle);
void goto_xy(int x, int y);
void init_window(void);
void save_screen(int x1, int y1, int x2, int y2,
                        char far *buf_ptr, char bk_attrib);
void restore_screen(int x1, int y1, int x2, int y2, char far *buf_ptr);
int  open_window(int x1, int y1, int x2, int y2, char boxstyle, char attribbox,
                        char bk_attrib, char *headwin, char head_attrib, char hstyle);
void close_window(void);
void errsound(void);
void stradd(char *s, char c, int num);
void writestr(int x, int y, int length, char *s, int attrib);
int  editstr(char *s, char *legal, int x, int y, int maxlength, int attrib);


/*---------------------- get video mode ----------------------*/
int video_mode(void)
{
        union REGS r;

        r.h.ah = 15;
        int86(16, &r, &r);
        return r.h.al;
}

/*------------------------ set video mode --------------------*/
void set_video_mode(void)
{
        int mode;

        mode=video_mode();
        if (mode == 1 || mode == 3)
                videoseg=(char far *)COLOR_BASE;
        else
                videoseg=(char far *)MONO_BASE;
}

/*---------------------- clear the screen --------------------*/
void cls(void)
{
        union REGS r;

        r.h.ah=6;       /* screen scroll code  */
        r.h.al=0;               /* clear screen code   */
        r.h.ch=0;               /* start row           */
        r.h.cl=0;               /* start column    */
        r.h.dh=24;      /* end row             */
```

```
        r.h.dl=79;      /* end column          */
        r.h.bh=7;               /* blank line is black */
        int86(0x10, &r, &r);
}

/*-------------------- read cursor position ---------------------*/
void get_xy(int *x, int *y)
{
        union REGS r;

        r.x.ax = 0x0300;
        r.x.bx = 0;
        int86(16, &r, &r);
        *x = r.h.dl;
        *y = r.h.dh;
}

/*---------------------- set cursor size ----------------------*/
void set_cursor_size(int s)
{
        union REGS r;

        r.x.ax = 0x0100;
        r.x.bx = 0;
        r.x.cx = s;
        int86(16, &r, &r);
}

void cursoroff()
{
        union REGS r;

        r.x.ax = 0x0100;
        r.h.ch = 0x20;
        r.h.cl = 0;
        int86(16, &r, &r);
}
void cursoron()
{
        union REGS r;

        r.x.ax = 0x0100;
        r.x.bx = 0;
        r.h.ch = 12;
        r.h.cl = 13;
        int86(16, &r, &r);
}
/*----------------------- scroll window up ----------------------*/
void scroll_up(int nline, int attr, int x1, int y1, int x2, int y2)
{
        union REGS r;

        r.h.ah = 6;
        r.h.al = nline;
        r.h.bh = attr;
```

```
        r.h.ch = y1;
        r.h.cl = x1;
        r.h.dh = y2;
        r.h.dl = x2;
        int86(16, &r, &r);
}

/*---------------------- scroll window down ----------------------*/
void scroll_down(int nline, char attr, int x1, int y1, int x2, int y2)
{
        union REGS r;

        r.h.ah = 7;
        r.h.al = nline;
        r.h.bh = attr;
        r.h.cl = x1;
        r.h.ch = y1;
        r.h.dl = x2;
        r.h.dh = y2;
        int86(16, &r, &r);
}


/*---------------------- get Shift status ----------------------*/
int  get_shift_status(void)
{
        union REGS r;

        r.x.ax =0x0200;
        int86(16, &r, &r);
        return r.h.al;
}

/*-------------- write a character at (x, y) -----------------*/
void write_char(int x, int y, char ch, char attrib)
{
        char far *v;

        v     = videoseg;
        v    += (y*160) + x*2;
        *v++  = ch;
        *v    = attrib;
}

/*---------- display a string with specified attribute -------*/
void write_string(int x, int y, char *st, char attrib)
{
        register int i;
        char far *v;

        v=videoseg;
        v += (y*160) + x*2;   /* compute the address */
        for (i=x; *st; i++){
                *v++ = *st++;    /* write the character */
                *v++ = attrib;   /* write the attribute */
```

```
        }
}


void fastwrite(int page,int row,int col,int attribute,char *string)
{
        register int offset = (page<<12)+col+col+row*160;
        while(*string!='\0'){
                pokeb(0xB000,offset++,*string++);
                pokeb(0xB000,offset++,(char)attribute);
        }
}

void fastwriteva(int page,int row,int col,int attrib,char *msg, ...)
{
        char buf[256];
        va_list    ap;
        va_start  (ap,msg);
        vsprintf(buf,msg,ap);
        fastwrite(page,row,col,attrib,buf);
}

void changeattribute(int page,int row,int col,int length,int newattrib)
{
        register int offset=(page<<12)+col+col+row*160;
        register int i;
        offset++;
        for(i=0; i<length; i++) {
                pokeb(0xB000,offset,(char)newattrib);
                offset+=2;
        }
}
/*------------------------ txtbar ----------------------------*/
void txtbar(int x1, int y1, int x2, int y2, char ch, char attrib)
{
        int j, i;
        char far *v;
        char far *t;

    v=videoseg;
        t=v;
        for (i=y1; i<=y2; i++){
                v=t;
                v += (i*160) + x1*2;
                for (j=x1+1; j<x2+1; j++){
                        *v++= ch;
                        *v++= attrib;
                }
        }

}

/*----------------------- Draw box ---------------------------*/
void draw_box(int x1, int y1, int x2, int y2, char linestyle, char lineattri,
                        char *headwin, char head_attrib, char hstyle)
```

```
{
        register int i;
        char far *v;
        char far *t;

        if (hstyle == 1){
                txtbar(x1, y1, x2, y1, ' ', head_attrib);
                fastwrite(0, y1, x1+(x2-x1-strlen(headwin))/2, head_attrib, headwin);
                y1 += 1;
        }
        v=videoseg;
        t=v;
        for (i=x1+1; i<x2; i++){
                v  += (y1*160) + i*2;
                *v++= border[linestyle][horiz];
                *v  = lineattri;
                v   = t;
                v  += (y2*160) + i*2;
                *v++= border[linestyle][horiz];
                *v  = lineattri;
                v   = t;
        }
        for (i=y1+1; i<y2; i++){
                v  += (i*160) + x1*2;
                *v++= border[linestyle][verti];
                *v  = lineattri;
                v   = t;
                v  += (i*160) + x2*2;
                *v++= border[linestyle][verti];
                *v  = lineattri;
                v   = t;
        }
        write_char(x1, y1, border[linestyle][upleft],   lineattri);
        write_char(x2, y1, border[linestyle][upright],  lineattri);
        write_char(x1, y2, border[linestyle][lowleft],  lineattri);
        write_char(x2, y2, border[linestyle][lowright], lineattri);
        if (hstyle == 0){
                fastwrite(0, y1, x1+(x2-x1-strlen(headwin))/2, head_attrib, headwin);
        }
}

/*---------------------- cursor position -------------------*/
void goto_xy(int x, int y)
{
        union REGS r;

        r.h.ah = 2;     /* cursor addressing function */
        r.h.dl = x;    /* row coordinate                  */
        r.h.dh = y;    /* column coordinate            */
        r.h.bh = 0;    /* video page                      */
        int86(0x10, &r, &r);
}

/*------------------------ save screen ----------------------*/
void save_screen(int x1, int y1, int x2, int y2,
```

```
                        char far *buf_ptr, char bk_attrib)
{
        register int i, j;
        char far *v;
        char far *t;

        v=videoseg;
        for (i=x1; i<x2+1; i++){
                for (j=y1; j<y2+1; j++){
                        t= v + (j*160) + i*2;   /* compute the address*/
                        *buf_ptr++=*t++;        /* read the character */
                        *buf_ptr++=*t;          /* read the attribute */
                        *(t-1)=' ';             /* clear the window   */
                        *t=bk_attrib;           /* clear attrib       */
                }
        }
}

/*------------------- restore screen -------------------------*/
void restore_screen(int x1, int y1, int x2, int y2, char far *buf_ptr)
{
        register int i,j;
        char far *v;
        char far *t;

        v=videoseg;
        t=v;
        for (i=x1; i<x2+1; i++){
                for (j=y1; j<y2+1; j++){
                        v   =t;
                        v  +=(j*160) + i*2;     /* compute the address */
                        *v++=*buf_ptr++;        /* write the character */
                        *v  =*buf_ptr++;        /* write the attribute */
                }
        }
}

/*--------------------- initial window -------------------*/
void init_window(void)
{
        set_video_mode();
        activewindow=NULL;
        fixedsize=sizeof(WINDOW_BLOCK);
}

/*------------------------- open window ----------------------*/
int open_window(int x1, int y1, int x2, int y2, char boxstyle, char attribbox,
                        char bk_attrib, char *headwin, char head_attrib, char hstyle)
{
        WINDOW_LINK block;
        char far *scrb;
        int  linelength, scrsize;

        linelength= x2 - x1 + 1;
        scrsize   = linelength*(y2 - y1 + 1)*2;
```

```
        if ( (x1<0) || (x2>80) || (y1<0) || (y2>25) )
                return(0);
        else{
                block=(WINDOW_BLOCK *)(malloc(fixedsize));
                scrb=(char far *)(farmalloc(scrsize));
        }
        if ( block && scrb ){
              block->x1=x1;
                block->x2=x2;
                block->y1=y1;
                block->y2=y2;
                block->x =wherex();
                block->y =wherey();
                if (hstyle == 0) block->y +=1;
                block->backlink=activewindow;
                block->screen_contents=scrb;
                activewindow=block;
                windowcount++;
                block->no=windowcount;
                save_screen(x1, y1, x2, y2, block->screen_contents, bk_attrib);
                draw_box(x1, y1, x2, y2, boxstyle, attribbox, headwin, head_attrib, hstyle);
                if (hstyle == 0)
                        window(x1+1, y1+1, x2-1, y2-1);
                else
                        window(x1+1, y1+2, x2-1, y2-1);
                gotoxy(2, 2);
                return(1);
        }
        else{
                printf(" cannot open window \n");
                return(0);
        }
}

/*-------------------- close window ----------------------*/
void close_window(void)
{
        WINDOW_LINK block, tmpblock;

        if (activewindow != NULL){
                block=activewindow;
                windowcount--;
                restore_screen(block->x1, block->y1, block->x2, block->y2, block->screen_contents);
                activewindow=block->backlink;
                if ((tmpblock=activewindow) != NULL) {
                        window(tmpblock->x1+1, tmpblock->y1+1, tmpblock->x2-1, tmpblock->y2-1);

                }
                else {
                        window(0, 0, 79, 24);
                }
                gotoxy(block->x, block->y);
                farfree(block->screen_contents);
                free(block);
                block=NULL;
```

```
        }
}

int  getkey(void)
{
        while(bioskey(1) == 0);
        return bioskey(0);
}

void errsound(void)
{
        int i;

   nosound();
   sound(440);
        for(i=0;i<15000;i++);
   nosound();
}

void stradd(char *s, char c, int num)
{
        int i=strlen(s), n=i+num;

        while (i<n)
                s[i++] = c;
        s[i]=0;
}

void writestr(int x, int y, int length, char *s, int attrib)
{
        char text[256]="";

        strcpy(text,s);
        stradd(text, ' ', length - strlen(s));
        fastwrite(0, y, x, attrib, text);
}
int editstr(char *s, char *legal, int x, int y, int maxlength, int attrib)
{
        int c, len=strlen(s), pos=len, insert=0;
        int exit = FALSE;

        cursoron();
        do {
                writestr(x, y, maxlength, s, attrib);
                goto_xy(pos + x, y);
                c=getkey();
                switch (c) {
                        case HOMEKEY : pos =0;
                                       break;
                        case ENDKEY  : pos = len;
                                       break;
                        case INSKEY  : insert = !insert;
                                       if (insert)
                                              set_cursor_size(0x0D);
                                       else
```

```
                                        set_cursor_size(0x0C0D);
                                    break;
                    case LEKEY   : if (pos > 0)
                                        pos--;
                                    else
                                        errsound();
                                    break;
                    case RIKEY   : if (pos < len)
                                        pos++;
                                     else
                                        errsound();
                                     break;
                    case BSKEY   : if (pos > 0) {
                                        movmem(&s[pos], &s[pos-1], len - pos + 1);
                                        pos--;
                                        len--;
                                    }
                                    else
                                        errsound();
                                    break;
                    case DELKEY  : if (pos < len) {
                                        movmem(&s[pos+1], &s[pos], len - pos);
                                        len--;
                                    }
                                    break;
        case RETKEY  :
        case UPKEY   :
                        case DNKEY   :
                        case PGUPKEY :
                        case PGDNKEY :
                        case CNTRL_N :
                        case CNTRL_Y :
                        case CNTRL_S : exit = TRUE; break;
                    case ESCKEY  : len=0; break;
                    default    : c = c & 0x00FF;
                                    if (((legal[0] == 0) || (strchr(legal, c) != NULL))
                                        && ((c >= ' ') && (c <= '~')) &&
                                        (len < maxlength)) {
                                        if (insert) {
                                            memmove(&s[pos +1], &s[pos], len - pos +1);
                                            len++;
                                        }
                                        else
                                            if (pos >= len )
                                                len++;
                                        s[pos++] = c;
                                    }
                                    else
                                        errsound();
                                    break;
            }
            s[len] = 0;
    }
while (!exit);
    cursoroff();/*set_cursor_size(0x0C0D);*/
```

```
    return c;
}
```

PIE.C

```
#include "c:\pie.h"
#include "c:\const.h"
#include <graphics.h>

#define TRUE        1
#define FALSE       0
#define SINGLE      0
#define NORMAL      0x07
#define REVERSE     0x70
#define BLINKVIDEO  0x0E
#define SPACEBAR    0X3920

void set_video_mode(void);
void cls(void);
void cursoroff(void);
void cursoron(void);
void errsound(void);
void title(int *item, int *nfc);
int  readkey(void);
int  menu_setting(int *item);
int  readstr(int col, int row, char *s, int len, int attrib);
int  editstr(char *s, char *legal, int x, int y, int maxlength, int attrib);
void goto_xy(int x, int y);
void write_string(int x, int y, char *st, char attrib);
void writestr(int x, int y, int length, char *s, char attrib);
void txtbar(int x1, int y1, int x2, int y2, char ch, char attrib);
void draw_box(int x1, int y1, int x2, int y2, char linestyle, char lineattri,
          char *headwin, char head_attrib, char hstyle);
int  set_benefit(void);
void disp_bar(int x, int y, int top, int bottom);
void create_factor_list(void);

int  geographic_area(void);
int  location_of_regions(void);
int  location_of_functions(void);
int  set_cost_coverage(void);
int  set_coverage(void);
int  set_relationship(void);
int  set_aspiration(void);
int  go_on(void);
int  readint_xy(int x, int y, int len);

int rating(char rate);
get_aspi(char *s1, char *s2, int i);
cal_cost(int fcn);
cal_cover(int fcn);
cal_benefit(void);
cal_relation(int i);
init_flg_fcn_reg(void);
arrange_loc(int locate);
pr_function_name(void);

typedef struct node Node, *List;
struct node {
   int  seq;
```

```
        char factor[50];
    int  rank;
    struct node *link;
};

struct objective {
        int aspi;
        int type;
        int priority;
} object[MAXOBJ];

int fcost[MAXLOCA];
int cver[MAXREG][MAXLOCA];

int fixcost[MAXFUNC][MAXLOCA];
int coverage[MAXFUNC][MAXREG][MAXLOCA];
int benefit[MAXLOCA];
int relationship[MAXFUNC][MAXLOCA];


int costing[MXF];
int covering[MXF][MXF];
int rateing[MXF][MXF];

int flg_fcn[MXF];
int regi[MXF];

List L=NULL;
int nf;
int num_reg, num_fun, num_loc;
int num_factor=22;
int num_obj;
int num_prio=MAXPRIO;
int num_edge;
int maxarea;
int edge[MAXPOINT][2];
int loc_reg[MAXREG][3];
int loc_fun[MAXFUNC][3];
int loc[MAXLOCA][2];
int cpu[MAXFUNC][MAXREG];
int cover[MAXFUNC][MAXREG];
int close_rating[MAXFUNC][MAXREG];
int factor_rank[MAXFACTOR], factor_weight[MAXFACTOR];
int scores[MAXFACTOR][MAXLOCA];

insert_singly_linked_list(List *L, int s, char *fact, int r);
delete_singly_linked_list(List *L, int s);
static List search_list(List L, int i);

extern int office_process(int nfunc, int nloc, int nreg, int nobj, int nprio, int first);
extern int grmain(int fcn);
extern int shape2(int fcn);
main()
{
    int item, t;
```

```
        int fcn, nfc, locate;

  set_video_mode();
       title(&item, &nfc);
       txtbar(28,5,52,20, ' ', NORMAL);
       write_string(30,17,"By ...", NORMAL);
       write_string(30,18,"PIPAT TREKITTINURAX",NORMAL);
       write_string(30,19,"I.E. DEPARTMENT CU.",NORMAL);
   getch();t=0;
       create_factor_list();
       do {
              menu_setting(&item);
              switch (item) {
               case 0 : geographic_area();       break;
               case 1 : location_of_regions();   break;
               case 2 : location_of_functions(); break;
               case 3 : set_cost_coverage();     break;
               case 4 : set_benefit();           break;
               case 5 : set_relationship();      break;
               case 6 : set_aspiration();        break;
        case 7 : do {
              fcn = go_on();
              locate = office_process(num_fun, num_loc, num_reg, num_obj, num_prio, t);
              if(num_reg == 1)
              grmain(fcn);
              else
              shape2(fcn);
                  flg_fcn[fcn] = 1;
              regi[nfc++]   = fcn;
              loc_reg[num_reg][2]=loc_fun[fcn][2];
                     arrange_loc(locate);
                     num_reg++;  num_loc--; t++;
               } while(nfc < nf);
               pr_function_name();
               getch();
               closegraph();
               title(&item, &nfc);t=0;
                 break;
               case 8 : /*end_program();*/       break;
              }
       } while (item < 8);
       txtbar(0,0,80,25, ' ', NORMAL);
       cursoron();cls();
       getch();
}

init_flg_fcn_reg()
{
       int i;

       for(i=0; i<MXF; i++) {
              flg_fcn[i] = 0;
              regi[i] = 0;
       }
       flg_fcn[0]=1;
```

```
}

arrange_loc(int locate)
{
        int i;

        for(i=locate; i<num_loc-1; i++) {
              . loc[i][0]=loc[i+1][0];
                loc[i][1]=loc[i+1][1];
        }
}


pr_function_name()
{
        int i;
        char buff[10], c;

        for(i=0; i<num_reg; i++) {
                c = 'A' + regi[i];
                sprintf(buff, "%2d-%c", regi[i]+1, c);
                outtextxy(10+i*30, getmaxy()-10, buff);
        }
}

void title(int *item, int *nfc)
{
        cls();nosound();cursoroff();
        draw_box(0,0,79,24,SINGLE,NORMAL,NULL,NORMAL,2);
        txtbar(2,1,78,21, ' ', REVERSE);
        write_string(8,2,"              MULTIOBJECTIVE MODEL FOR OFFICE LAYOUT              ",NORMAL);
         *item=0; *nfc=1;
    init_flg_fcn_reg();
}

int geographic_area(void)
{
        int i;

        txtbar(27,7,52,18, ' ', NORMAL);
        write_string(32, 8,"Geographic Area", BLINKVIDEO);
        write_string(32, 9,"DDDDDDDDDDDDDD", BLINKVIDEO);
        write_string(31,11,"Regions   = ", NORMAL);
        write_string(31,12,"Functions = ", NORMAL);
        write_string(31,13,"Area      = ", NORMAL);
        write_string(31,14,"Number of edges = ",NORMAL);
        num_reg = readint_xy(43, 11, 3);
        nf      = readint_xy(43, 12, 3);
        num_fun = 1; /* select one function which the best */
        maxarea = readint_xy(43, 13, 4);
        num_edge= readint_xy(49, 14, 2);
        for(i=0; i<num_edge; i++) {
                goto_xy(37,15);printf("X%-2d =     ", i+1);
                goto_xy(37,16);printf("Y%-2d =     ", i+1);
                edge[i][0] = readint_xy(43, 15, 4);
```

```
                edge[i][1] = readint_xy(43, 16, 4);
        }
        txtbar(27,7,51,12, ' ', REVERSE);
}

int location_of_regions(void)
{
        int i;
        int x, y;

        char s[14];

        txtbar(23,5,53,18, ' ', NORMAL);
        write_string(24,6,"Location and Area of Regions", BLINKVIDEO);
        write_string(24,7,"DDDDDDDDDDDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);
        write_string(26,10,"Number of Regions =    ", NORMAL);
        goto_xy(46, 10); printf("%d", num_reg);
        cursoron();
        x = 30; y = 12;
        for(i=0; i<num_reg; i++) {
                sprintf(s, "Region #%d", i+1);
                write_string(x, y, s, NORMAL);
                write_string(x, y+1, "x =      ", NORMAL);
                write_string(x, y+2, "y =      ", NORMAL);
                write_string(x, y+3, "area =      ", NORMAL);
                loc_reg[i][0] = readint_xy(x+5, y+1, 3);
                loc_reg[i][1] = readint_xy(x+5, y+2, 3);
                loc_reg[i][2] = readint_xy(x+8, y+3, 4);
        }
        txtbar(23,5,53,18, ' ', REVERSE);
}

location_of_functions(void)
{
        int i;
        int x, y;

        char s[80];

        txtbar(21,5,55,18, ' ', NORMAL);
        write_string(23,6,"Location and Area of Functions", BLINKVIDEO);
        write_string(23,7,"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);
        write_string(25,10,"Number of Locations =     ", NORMAL);
        cursoron();
        num_loc = readint_xy(47, 10, 3);
        x = 30; y = 12;
        for(i=0; i<num_loc; i++) {
                sprintf(s, "Location #%d", i+1);
                write_string(x, y, s, NORMAL);
                write_string(x, y+1, "x =      ", NORMAL);
                write_string(x, y+2, "y =      ", NORMAL);
                loc[i][0] = readint_xy(x+5, y+1, 3);
                loc[i][1] = readint_xy(x+5, y+2, 3);
        }
```

```
        for(i=0; i<nf; i++) {
                sprintf(s, "Area of Function #%d = ", i+1);
                write_string(x-5, y+4, s, NORMAL);
                loc_fun[i][2] = readint_xy(x+17, y+4, 3);
        }
        txtbar(21,5,55,18, ' ', REVERSE);
}

set_cost_coverage(void)
{
        int i, j;
        int x, y;
        char s[40];

        txtbar(21,5,55,18, ' ', NORMAL);
        write_string(30,6," Cost & Coverage ", BLINKVIDEO);
        write_string(30,7," DDDDDDDDDDDDDD ", BLINKVIDEO);
        write_string(25,11,"Number of Functions =     ", NORMAL);
        goto_xy(47, 11);
        printf("%3d", nf);
        cursoron();
        x = 30; y = 13;

        for(j=0; j<nf; j++) {
               write_string(x, y, "cost per unit of", NORMAL);
               sprintf(s, "Function #%d", j+1);
               write_string(x, y+1, s, NORMAL);
               write_string(x, y+2, "cpu      =      ", NORMAL);
               costing[j] = readint_xy(x+12, y+2, 4);
        }

        txtbar(28,13,50,17, ' ', NORMAL);

        for(i=0; i<nf; i++) {
               sprintf(s, "function #%d  ", i+1);
               write_string(x, y, s, NORMAL);
               write_string(x, y+1, "coverage to ", NORMAL);
               for(j=0; j<nf; j++) {
                      if(j != i) {
                      sprintf(s, "Function #%d  ", j+1);
                      write_string(x, y+2, s, NORMAL);
                      write_string(x, y+3, "coverage =      ", NORMAL);
                      covering[i][j] = readint_xy(x+12, y+3, 4);
                      }
               }
        }
        txtbar(21,5,55,18, ' ', REVERSE);
}

set_relationship(void)
{
        int i, j;
        int x, y;
        char s[40];
```

```
        txtbar(21,5,55,18, ' ', NORMAL);
        write_string(31,6," Relationships ", BLINKVIDEO);
        write_string(31,7," DDDDDDDDDDDD ", BLINKVIDEO);
        write_string(25,10,"Number of Functions =     ", NORMAL);
        goto_xy(47, 10);  printf("%3d", nf);
        write_string(2,23,"A-Absolutely E-Especially I-Important O-Ordinary U-Unimportant X-Undesirable", BLINKVIDEO);
        cursoron();
        x = 28; y = 11;
        for(i=0; i<nf; i++) {
                sprintf(s, "Function #%-2d", i+1);
                write_string(x, y+1, s, NORMAL);
                write_string(x, y+2, "rate to", NORMAL);
                rateing[i][i] = '0';
                for(j=i+1; j<nf; j++) {
                        sprintf(s, "function #%-2d", j+1);
                        write_string(x, y+3, s, NORMAL);
                        write_string(x, y+4, "closeness rating =     ", NORMAL);
                        do {
                                s[0] = '\0';
                                editstr(s, "AEIOUXaeioux", x+19, y+4, 1, REVERSE);
                        } while(s[0] == '\x0');
                        sscanf(s, "%c", &rateing[i][j]);
                        rateing[j][i]=rateing[i][j];
                }
        }
        txtbar(21,5,55,18, ' ', REVERSE);
        write_string(2,23,"                                                                                ", NORMAL);
}

get_aspi(char *s1, char *s2, int i)
{
        int x, y;

        txtbar(14,11,69,18, ' ', NORMAL);
        x = 17; y = 11;
        goto_xy(x,y); printf("Objective #%-2d   : ", i+1);
        write_string(x+18, y, s1, NORMAL);
        write_string(x+18, y+1, s2, NORMAL);
        write_string(x, y+2,"Aspiration     : ",NORMAL);
        write_string(x, y+3,"Objective type : ", NORMAL);
        write_string(x, y+4,"Priority       : ", NORMAL);
        object[i].aspi = readint_xy(x+18, y+2, 5);
        do {
                object[i].type = readint_xy(x+18, y+3, 2);
                if(object[i].type == -1 || object[i].type == 1)
                        break;
                else
                        errsound();
        } while(1);
        object[i].priority = readint_xy(x+18, y+4, 2);
        txtbar(14,11,69,18, ' ', NORMAL);
}
set_aspiration()
{
```

```
        int i, j;
        char s[80];

        txtbar(14,5,69,18, ' ', NORMAL);
        write_string(20,6,"Objectives, Aspiration Level and Priorities", BLINKVIDEO);
        write_string(20,7,"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);
        write_string(2,23,"                          -1 = Minimize    1 = Maximize                          ", BLINKVIDEO);
        write_string(25,9,"Number of Objectives =    ", NORMAL);
        num_obj = 5 + 2*(num_fun-1);
        goto_xy(48, 9); printf("%2d", num_obj);
        cursoron();
        j=0;
        get_aspi("Total cost should not exceed ", "   budget limit", j++);
        for(i=0; i<num_fun; i++) {
                sprintf(s, "   function #%d", i+1);
                get_aspi("Maximize coverage for ", s, j++);
        }
        get_aspi("Minimize total deviations ", "   from locational benefits", j++);
    for(i=0; i<num_fun; i++) {
                sprintf(s, "   deviations for function #%d", i+1);
                get_aspi("Minimize relationship ", s, j++);
        }
        get_aspi("Minimize total cost ", " ", j);
        write_string(2,23,"                                                                                ", BLINKVIDEO);
        txtbar(14,5,69,18, ' ', REVERSE);
}

int tmp_rel[MXF];
int tmp_cost[MXF][MXF];
int tmp_cover[MXF][MXF];
int tmp_fcn[MXF];


relate(void)
{
        int i,j,k;

        k=0;
        for(i=0; i<nf; i++) {
                if(flg_fcn[i] == 0) {
                        tmp_rel[k] = 0;
                        tmp_fcn[k] = i;
                        for(j=0; j<num_reg; j++) {
                         tmp_rel[k] += rating(rateing[i][regi[j]]);
                        }
                        k++;
                }
        }
        return k;
}


cost_cover(int fcn)
{
        int cmax[MXF], cvmax[MXF];
```

```
        int i,j,k;

        for(i=0; i<num_reg; i++) {
                cmax[i] = 0;
                cvmax[i]= 0;
                k = 0;
                for(j=0; j<nf; j++) {
                        if(flg_fcn[j] == 0) {
                tmp_cost[i][k] = costing[regi[i]];
                tmp_cover[i][k]= covering[j][regi[i]];
                                if(cmax[i] < tmp_cost[i][k])
                                        cmax[i] = tmp_cost[i][k];
                                if(cvmax[i]< tmp_cover[i][k])
                                        cvmax[i]= tmp_cover[i][k];
                                k++;
                        }
                }
                cmax[i]  /= 6;
                cvmax[i] /= 6;
        }

        for(i=0; i<num_reg; i++) {
                for(j=0; j<fcn; j++) {
                        tmp_cost[i][j]  /= cmax[i];
                        tmp_cover[i][j] /= cvmax[i];
                }
        }
}

select_fcn(int fcn)
{
        int i,j;
        int s_max, s_fcn;
        int tmax;

        s_max = s_fcn = 0;
        for(i=0; i<fcn; i++) {
                tmax = tmp_rel[i];
                for(j=0; j<num_reg; j++) {
                        tmax += tmp_cost[j][i] + tmp_cover[j][i];
                }
                if(s_max < tmax) {
                        s_max = tmax;
                        s_fcn = tmp_fcn[i];
                }
        }
        return s_fcn;
}



go_on(void)
{
        int fcn, sfcn;
```

```
        fcn = relate();
        cost_cover(fcn);
        sfcn = select_fcn(fcn);
        cal_cost(sfcn);
        cal_cover(sfcn);
        cal_benefit();
        cal_relation(sfcn);
    return sfcn;
}

cal_cost(int fcn)
{
        int j, k;

        for(j=0; j<num_loc; j++) {
                fixcost[0][j] = 0;
                for(k=0; k<num_reg; k++)
                        fixcost[0][j] += (( abs(loc_reg[k][0] - loc[j][0])
                         +abs(loc_reg[k][1] - loc[j][1]) )*costing[fcn]);
        }
}



cal_cover(int fcn)
{
        int j, k;
        int max;

        max=0;
    for(j=0; j<num_reg; j++) {
       for(k=0; k<num_loc; k++) {
          cver[j][k] =  abs(loc_reg[j][0] - loc[k][0])
                        +abs(loc_reg[j][1] - loc[k][1]);
          if(cver[j][k] > max)
             max = cver[j][k];
       }
     }
     for(j=0; j<num_reg; j++)
        for(k=0; k<num_loc; k++)
           cver[j][k] = (max - cver[j][k])*covering[fcn][j];

     for(j=0; j<num_reg; j++)
        for(k=0; k<num_loc; k++)
            coverage[0][j][k] = cver[j][k];
}

cal_benefit(void)
{
        int i, j;
        int max_bene=0;

        for(i=0; i<num_loc; i++) {
                benefit[i] = 0;
```

```
                for(j=0; j<num_factor; j++)
                        benefit[i] += factor_rank[j]*factor_weight[j]*scores[j][i];
                if(benefit[i] > max_bene)
                        max_bene = benefit[i];
        }
        for(i=0; i<num_loc; i++)
                benefit[i] = max_bene - benefit[i];
}

rating(rate)
char rate;
{
        int r=0;

        switch (rate) {
                case 'A' : case 'a' : r=6;break;
                case 'E' : case 'e' : r=5;break;
                case 'I' : case 'i' : r=4;break;
                case 'O' : case 'o' : r=3;break;
                case 'U' : case 'u' : r=2;break;
                case 'X' : case 'x' : r=1;break;
        }
        return r;
}


cal_relation(int i)
{
        int j,k;
        int r, max, maxr=0;
   int d[MAXREG][MAXLOCA];

        max=0;
        for(j=0; j<num_reg; j++) {
                for(k=0; k<num_loc; k++) {
                        d[j][k] =  abs(loc_reg[j][0] - loc[k][0])
                                +abs(loc_reg[j][1] - loc[k][1]);
                        if(d[j][k] > max)
                                max = d[j][k];
                        }
                }
                for(j=0; j<num_reg; j++) {
                        r = rating(rateing[i][j]);
                        for(k=0; k<num_loc; k++)
                                d[j][k] = (max-d[j][k])*r;
                }
                for(j=0; j<num_loc; j++) {
                        relationship[0][j] = 0;
                        for(k=0; k<num_reg; k++)
                                relationship[0][j] += d[k][j];
                        if(relationship[0][j] > maxr)
                                maxr = relationship[0][j];
                }
        for(j=0; j<num_loc; j++)
```

```
                relationship[0][j] = maxr - relationship[0][j];
}

menu_setting(int *item)
{
        char *menu[]={ " Geographic Area          ",
                                                  " Location of Regions    ",
                                                  " Location of Functions  ",
                                                  " Cost & Coverage        ",
                                                  " Benefit                ",
                                                  " Relationships          ",
                                                  " Aspriration            ",
                                                  " Go On ...              ",
                                                  " Exit                   ",
                                                  "" };
        int numofitem=9;
    int exit = FALSE;
        int i, j, key;

        cursoroff();
        txtbar(26,5,54,20, ' ', NORMAL);
        write_string(29,6,"Setting Office Layout", BLINKVIDEO);
        write_string(29,7,"DDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);
    for(i=0; i<=numofitem; i++)
                write_string(28,10+i,menu[i],NORMAL);
        i=(*item);
        write_string(28,10+i,menu[i],REVERSE);
    do {
       key = readkey();
       j=i;
       switch (key) {
                        case UPKEY : --i; if(i<0) i=numofitem-1; break;
                        case SPACEBAR :
         case DNKEY : ++i; if(i>numofitem-1) i=0; break;
         case RETKEY: exit = TRUE; break;
         default    : break;
       }
                write_string(28,10+j,menu[j],NORMAL);
                write_string(28,10+i,menu[i],REVERSE);
        } while (!exit);
        txtbar(26,5,54,20, ' ', REVERSE);
        *item = i;
}
int readkey()
{
   while (bioskey(1) == 0)
         ;
  return bioskey(0);
}


void create_factor_list(void)
{
        int i;
```

```
	char *fact[]={ "Artificial Lighting",
			"Noise Level ",
			"Inside Temperatures",
			"Quality of Temperatures",
			"Accessibility between Work Centers",
			"Privacy",
			"Quietness",
			"Quality of Equipment",
			"Adequate storage Space",
			"Amount of Working Space",
			"Orderlines of Arrangemant of Furniture",
			"Access to Rest Rooms",
			"Wall Coloring",
			"Internal Decoration",
			"Lounge",
			"Background Music System",
			"Windows for Outside View",
			"Landscaping",
			"Surrounding",
			"Access to Telephones",
			"Access to Cafeteria, Lobby, etc.",
			"Access to Main Exit",
						"" };

	for(i=1; i<=num_factor; i++) {
		insert_singly_linked_list(&L, i, fact[i-1], 0);
	}
}


int set_benefit(void)
{
	int n=50; /* number of Environment Factors */
	int i, j, x, y;
	int bottom, top;
	static seq;
	List  p;
	int rank=3;
	char factor[50], s[50];
	int key;


	txtbar(15,5,65,20, ' ', NORMAL);
	write_string(17,6,"Ranking of Locational and Environment Factors", BLINKVIDEO);
	write_string(17,7,"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);

	i=1; j=1; seq = num_factor; top = 1;
	x = 18; y=8;
	if(num_factor > 10)
		bottom = 10;
	else
		bottom = num_factor;
	disp_bar(x, y, top, bottom);
	strcpy(factor, L->factor);
	do {
```

```
goto_xy(x, y+i-top+1);printf("%2d.", i);
key = readstr(x+3,y+(i-top+1), factor, 40, REVERSE);

j = i;
switch (key) {
        case UPKEY : --i;
             if(i<=0) { i=1; errsound(); }
             else
             if(i<top) {
             top--;
             if(seq - top > 9)
             bottom--;
             i=top;
             disp_bar(x, y, top, bottom);
             }
             else
             writestr(x+3, y+(j-top+1), 40, factor, NORMAL);
             break;
        case DNKEY : ++i;
             if(i>seq) { i--; errsound(); }
                    else
                    if(i>bottom) {
                    top++;
                    bottom++; i=bottom;
                    disp_bar(x, y, top, bottom);
                          }
                    else
                    writestr(x+3, y+(j-top+1), 40, factor, NORMAL);
                    break;
        case PGUPKEY:if(seq > 10) {
                    if(top - 10 < 0) {
                    top = 1;
                    bottom = 10;
                    i=top;
                          }
                    else {
                    top -= 10;
                    bottom = top + 9;
                    i -=10;
                          }
                          }
                    else
                    if(bottom - top <= 10) {
                    top = 1; i=top;
                          }
                    disp_bar(x, y, top, bottom);
                    break;
        case PGDNKEY: if(bottom < seq) {
                    top += 10;
                    if(seq - bottom - 10 < 0) {
                    bottom += seq-bottom;
                    i=bottom;
                          }
                    else {
                    bottom += 10;
```

```
                    i += 10;
                          }
                          }
                    else
                    i = bottom;
                    disp_bar(x, y, top, bottom);
                    break;
          case RETKEY: if(i==seq && (strcmp(factor, "") == 0))
                     break;
                    else {
                    ++i;
                    if(i>seq) {
                    seq++; bottom++;
                    if(bottom > top+9) {
                    top++;
                    disp_bar(x, y, top, bottom);
                          }
                          }
                    else
                    if(i>bottom) {
                    top += 1;
                    bottom += 1;
                    i=bottom;
             disp_bar(x, y, top, bottom);
                    }
                    goto_xy(x, y+(j-top+1));printf("%2d.", j);
                    writestr(x+3, y+(j-top+1), 40, factor, NORMAL);
                    }
                    break;
          case CNTRL_N : /* insert new record */
                    if(i<seq) {
                          for(j=seq; j>i; j--) {
                          p=search_list(L, j);
                          strcpy(factor,p->factor);
                          rank=p->rank;
                          (p->seq)++;
                          insert_singly_linked_list(&L, p->seq, factor, rank);
                             }
                          i++; j=i; seq++; bottom++;
                          if(bottom - top > 10) top++;
                          strcpy(factor, "");
                          insert_singly_linked_list(&L, j, factor, rank);
                          disp_bar(x, y, top, bottom);
                          goto_xy(x, y+i-top+1);printf("%2d.", i);
                          key = readstr(x+3,y+(i-top+1), factor, 40, REVERSE);
                          insert_singly_linked_list(&L, j, factor, rank);
                          goto_xy(x, y+(j-top+1));printf("%2d.", j);
                          writestr(x+3, y+(j-top+1), 40, factor, NORMAL);
                          i++;
                    }
                          else
                          errsound();
                          break;

case CNTRL_Y : /* delete a record */
```

```
                        if(i < seq) {
                        delete_singly_linked_list(&L, i);
                        if(bottom == seq) bottom--;
                        seq--;
                        disp_bar(x, y, top, bottom);
                        p=search_list(L, i);
                        strcpy(factor, p->factor);
                                }
                        break;
            }

            if(strlen(factor) == 0 && i == seq && key == RETKEY
                   || key == CNTRL_S)
                        break;
            else {
                   if(strlen(factor) != 0)
                          insert_singly_linked_list(&L, j, factor, rank);
        if((p=search_list(L, i)) != NULL)
            strcpy(factor, p->factor);
        else strcpy(factor, "");
      }
    } while(i<50);
    if(i==seq && (strcmp(factor, "") == 0)) seq--;
    n=seq;num_factor=seq;
    p=L;
    for(i=0; i<num_factor; i++) {
           txtbar(15,5,67,20, ' ', NORMAL);
           write_string(3,23,"0-Unimportant  1-Slight Important  2-Quite Important  3-Hightly Important", BLINKVIDEO);
           sprintf(factor, "factor #%-2d: %-25s", p->seq, p->factor);
           p=p->link;
           write_string(16,6,factor, BLINKVIDEO);
           write_string(16,7,"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD", BLINKVIDEO);
           write_string(25,10,"Number of Locations =     ", NORMAL);
           goto_xy(47, 10); printf("%-3d", num_loc);
           x = 25; y = 12;
           write_string(x, y+1, "rank   =   ", NORMAL);
           write_string(x, y+2, "weight =   ", NORMAL);
           do {
                  factor_rank[i] = readint_xy(x+10, y+1, 1);
                  if(factor_rank[i] >=0 && factor_rank[i]<4)
                         break;
                  else
                         errsound();
           } while(1);
           factor_weight[i] = readint_xy(x+10, y+2, 2);
           for(j=0; j<num_loc; j++) {
     sprintf(s, "score for location #%-2d = ", j+1);
                  write_string(x, y+3, s, NORMAL);
                  scores[i][j] = readint_xy(x+27, y+3, 2);
           }
    }
    write_string(3,23,"                                                                                ", BLINKVIDEO);
    txtbar(15,5,67,20, ' ', REVERSE);
    return n;
}
```

```
readstr(int col, int row, char *s, int len, int attrib)
{
        char legal[100];
        int i;

        for(i=0;i<24; i++) {
          legal[i]='a'+i;
          legal[i+24] = 'A' + i;
        }
        legal[48]=' ';
        legal[49]='\0';
        return editstr(s, legal, col, row, len, attrib);
}

int readint_xy(x, y, len)
int x, y;
int len;
{
        int numer;
        int i, j;
        char s[5];

        s[0]='\0';
        editstr(s, "0123456789+-", x, y, len, REVERSE);
        numer = atoi(s);
        sprintf(s, "%d", numer);
        j = strlen(s);
        for(i=0;i<(len - j); i++)
                s[j+i] = ' ';
        s[j+i] = '\0';
        write_string(x, y, s, NORMAL);
        return numer;
}


insert_singly_linked_list(L, s, fact, r)
List *L;
int s;
char *fact;
int r;
{
        List p;
        List temp = (List) malloc(sizeof(struct node));

        if (temp == NULL) {
                printf("Error! cannot allocate memory...");
                exit(1);
        }
        else {
        temp->seq    = s;
                strcpy(temp->factor, fact);
        temp->rank   = r;
        temp->link   = NULL;
        }
```

```
        p = *L;
        if (p == NULL) {
                *L = temp;
         (*L)->seq    = s;
                      strcpy((*L)->factor, fact);
         (*L)->rank   = r;
         (*L)->link   = NULL;
        }
    else {
                if (temp->seq < p->seq) { /* insert first */
                             temp->link = *L;
                             *L = temp;
                }
                else
                       if (temp->seq == p->seq) {
                                strcpy(p->factor, temp->factor);
                                p->rank = temp->rank;
                                free(temp);
                       }
                       else { /* insert last or other position */
                              while (p->link != NULL ) {
                                     if (temp->seq < p->link->seq) {
                                            temp->link = p->link;
                                            p->link = temp;
                                        break;
                                     }
                                     if (temp->seq == p->link->seq) {
                                            strcpy(p->link->factor, temp->factor);
                                            p->link->rank = temp->rank;
                                            free(temp);
                                            break;
                                     }
                                p = p->link;
                              }
                              if (p->link == NULL) { /* insert last */
                                     p->link = temp;
                              }
                       }
        }
}

delete_singly_linked_list(L, s)
List *L;
int   s;
{
    List p=*L, q;
    int x=0;

    if (p == NULL)
        x = -1;
    else
                if (p->seq == s) {
                       x = p->seq;
                       (*L)=p->link;
```

```
                        free(p);
                        p=p->link;
                        while(p != NULL) {
                                (p->seq)--;
                                p=p->link;
                        }
                }
        else
           while ( p->link != NULL ) {
                                if (p->link->seq == s) {
                q = p->link;
                x = q->seq;
                                        p->link = q->link;
                p = p->link;
                                        free(q);
                                        q = NULL;
                                        while(p != NULL) {
                                                (p->seq)--;
                                                        p=p->link;
                                        }
                                        break;
              }
              else
                 p = p->link;
           }
   return x;
}

static List search_list(List L, int i)
{
        List p;

        p=L;
        while (p != NULL)
                if(p->seq == i)
                        break;
                else
                        p = p->link;
        return p;
}

void disp_bar(x, y, top, bottom)
int x, y;
int top, bottom;
{
        List p;
        char factor[40];

        txtbar(15,8,65,20, ' ', NORMAL);
        p = search_list(L, top);
        while (p != NULL && p->seq <= bottom) {
                strcpy(factor, p->factor);
                goto_xy(x, y+(p->seq - top + 1));
                printf("%2d.", p->seq);
                writestr(x+3, y+(p->seq - top+1), 40, factor, NORMAL);
```

```
            p = p->link;
        }
    }
```

OFFICE.C

```
#include "pie.h"

extern struct objective {
        int aspi;
        int type;
        int priority;
} object[MAXOBJ];
extern int fixcost[MAXFUNC][MAXLOCA];
extern int coverage[MAXFUNC][MAXREG][MAXLOCA];
extern int benefit[MAXLOCA];
extern int relationship[MAXFUNC][MAXLOCA];
extern int loc_fun[MAXFUNC][3];
extern int loc_reg[MAXREG][3];
extern int loc[MAXFUNC][2];

int abar[MAXPRIO][MAXLOCA];
int astar[MAXPRIO];
int obj[MAXOBJ][MAXLOCA];
int cvr[MAXFUNC][MAXLOCA];
int xmpr[MAXFUNC][MAXREG][MAXLOCA];
int ympr[MAXFUNC][MAXREG][MAXLOCA];
int select[MAXLOCA];
int isel[MAXFUNC][MAXREG];
int jsite[MAXLOCA];
int fix[MAXFUNC];
int fax[MAXFUNC][MAXLOCA];
int fay[MAXFUNC][MAXLOCA];
int rel[MAXFUNC];
int nselect;
int bn;

office_process(int nfunc, int nloc, int nreg, int nobj, int nprio, int first);
initialize(int num_fun, int num_loc, int num_reg, int num_prio, int *chosen, int first);
print_input(int nfunc, int nloc, int nreg);
print_cost(int nfunc, int nloc);
print_coverage(int nfunc, int nreg, int nloc);
print_benefit(int nloc);
print_relationship(int nfunc, int nloc);
print_objective(int numobj, int nloc);
print_achievement(int nprio, int nloc);
print_select(int sel, int nprio);
print_result(int nfunc, int nreg);
final_selection(int nfunc, int nreg);
cal_coefficient(int nfunc, int nloc, int nreg);
coeff_fixcost(int nfunc, int nloc, int *nobj);
coeff_coverage(int nfunc, int nreg, int nloc, int *nobj);
coeff_benefit(int nloc, int *nobj);
coeff_relationship(int nfunc, int nloc, int *nobj);
cal_deviation(int nloc, int nprio, int numobj);
select_astar(int nloc, int nprio);
int find_min(int table[][MAXLOCA], int num);
int clear_a_flg(int table[], int nloc);
int mark_flg(int a_flg[], int mark, int num);
int start_flg(int a_flg[], int *start, int num);
int clear_flg(int table_flg[], int end);
```

```
int get_astar(int table[], int num);
is_best_solution(int nprio, int klc);
selection_improve(int nfunc, int nloc, int nreg, int nprio, int klc);
update(int nfunc, int nloc, int nreg, int klc);


#define  YES    0
#define  NO    -1


initialize(num_fun, num_loc, num_reg, num_prio, chosen, first)
int num_fun, num_loc, num_reg, num_prio;
int *chosen;
int first;
{
   int i,j, k;

   for (i=0; i<num_fun; i++) {
      rel[i] = 0;
      for(j=0; j<num_reg; j++) {
         isel[i][j] = 0;
         for(k=0; k<num_loc; k++) {
            xmpr[i][j][k] = 0;
            ympr[i][j][k] = 1;
         }
      }
   }

   for(i=0; i<num_fun; i++) {
      for(j=0; j<num_loc; j++) {
         fax[i][j] = 0;
         fay[i][j] = 1;
      }
   }
   for(i=0; i<num_loc; i++) {
      jsite[i]=0;
      fix[i] = 0;
            select[i] = 0;
   }

   for(i=0; i<num_prio; i++)
            astar[i] = 9999;

   bn = 0;
   (*chosen) = 0;
       nselect = -1;
}

wait()
{
       printf("\n\n press any key to continue....");
       getch();
}

print_result(nfunc, nreg)
```

```
int nfunc, nreg;
{
        int i, j, k;
        int final[MAXFUNC][MAXREG];
        int max;
        int chosen=0;

        for(i=0; i<nfunc; i++) {
                for(j=0; j<nreg; j++) {
                        max = coverage[i][j][jsite[0]+1];
                        final[i][j] = jsite[0]+1;
                        for(k=0; k<=nselect; k++) {
                                if(coverage[i][j][jsite[k]] > max) {
                                        max = coverage[i][j][jsite[k]];
                                        final[i][j] = jsite[k]+1;
                                }
                        }
                }
        }

        chosen = final_selection(nfunc, nreg);
        return chosen;
}

final_selection(nfunc, nreg)
int nfunc, nreg;
{
        int i, j, k;
        int s, res;

        s=0; res=0;
        if(nselect == 0)
                res=jsite[0];
        else {
                for(i=0; i<nfunc; i++) {
                        for(j=0; j<nreg; j++) {
                                for(k=0; k<nselect; k++) {
                                        if(coverage[i][j][jsite[k]] > s) {
                                                res = jsite[k];
                                        }
                                }
                        }
                }
        }
        loc_fun[0][0]=loc[res][0];
        loc_fun[0][1]=loc[res][1];
        loc_reg[nreg][0] = loc[res][0];
        loc_reg[nreg][1] = loc[res][1];
        return res;
}


office_process(nfunc, nloc, nreg, nobj, nprio, first)
int nfunc, nloc, nreg, nobj, nprio, first;
{
```

```
    int chosen;

    initialize(nfunc, nloc, nreg, nprio, &chosen, first);
    do {
                cal_coefficient(nfunc, nloc, nreg);
                cal_deviation(nloc, nprio, nobj);
                chosen=select_astar(nloc, nprio);
                if(is_best_solution(nprio, chosen) == YES) {
                        break;
                }
                else {
                        if(selection_improve(nfunc, nloc, nreg, nprio, chosen) == YES)
                                break;
                }
    } while(1);
        chosen = print_result(nfunc, nreg);
        return chosen;
}

cal_coefficient(nfunc, nloc, nreg)
int nfunc, nloc, nreg;
{
    int  nobj;

    nobj = 0;
    coeff_fixcost(nfunc, nloc, &nobj);
    coeff_coverage(nfunc, nreg, nloc, &nobj);
    coeff_benefit(nloc, &nobj);
        coeff_relationship(nfunc, nloc, &nobj);
    coeff_fixcost(nfunc, nloc, &nobj);
}

coeff_fixcost(nfunc, nloc, nobj)
int nfunc, nloc, *nobj;
{
        int i, j;

    for(i=0; i<nloc; i++) {
       obj[*nobj][i] = 0;
       if(select[i] == 0) {
          for(j=0; j<nfunc; j++) {
             obj[*nobj][i] += fixcost[j][i]*fay[j][i] + fix[j]*fax[j][i];
          }
       }
    }
    (*nobj)++;
}

coeff_coverage(nfunc, nreg, nloc, nobj)
int nfunc, nreg, nloc, *nobj;
{
    int i,j,k;
    int nx;

    for(i=0; i<nloc; i++) {
```

```
      if(select[i] == 0) {
         for(j=0; j<nfunc; j++) {
            nx = j + (*nobj);
            obj[nx][i] = 0;
            for(k=0; k<nreg; k++) {
               obj[nx][i] += coverage[j][k][i]*ympr[j][k][i]
                           + cvr[j][k]*xmpr[j][k][i];
            }
         }
      }
   }
   (*nobj) = nx+1;
}

coeff_benefit(nloc, nobj)
int nloc, *nobj;
{
   int i;

   for(i=0; i<nloc; i++) {
      if(select[i] == 0) {
         obj[*nobj][i] = benefit[i] + bn;
      }
   }
   (*nobj)++;
}

coeff_relationship(nfunc, nloc, nobj)
int nfunc, nloc, *nobj;
{
   int i,j;
   int nx;

   for(i=0; i<nfunc; i++) {
      nx = i + (*nobj);
      for(j=0; j<nloc; j++) {
         if(select[j] == 0) {
            obj[nx][j] = rel[i] + relationship[i][j];
         }
      }
   }
   (*nobj) = nx + 1;
}

cal_deviation(nloc, nprio, numobj)
int nloc, nprio, numobj;
{
   int i,j,k;
   int dev[MAXOBJ][MAXLOCA];

   for(j=0; j<numobj; j++)
      for(k=0; k<nloc; k++)
                  if(select[k] == 0) {
                        dev[j][k] = (obj[j][k] - object[j].aspi)*(object[j].type)*(-1);
                        if(dev[j][k] <= 0 )
```

```
                                        dev[j][k] = 0;
                        }
                else
                   dev[j][k] = 0;

        for(i=0; i<nprio; i++)
           for(j=0; j<nloc; j++)
              abar[i][j] = 0;

        for(i=0; i<nprio; i++)
           for(j=0; j<nloc; j++)
              for(k=0; k<numobj; k++)
                                if(object[k].priority == i+1)
                    abar[i][j] = abar[i][j] + dev[k][j];
}

select_astar(nloc, nprio)
int nloc, nprio;
{
        int i,j,k;
        int min;
     int temp, sel;
     int a_flg[MAXLOCA];

        clear_a_flg(a_flg, nloc);
        min = find_min(abar, nloc);  /* find min of first priority */
        mark_flg(a_flg, min, nloc);
        j=0;
        start_flg(a_flg, &j, nloc);
        for(i=1; i<nprio; i++) {
               start_flg(a_flg, &j, nloc);
               for(k=j; k<nloc; k++) {
                      temp = abar[i][j];
                      if((abar[i][k] < temp) && (a_flg[k] == 1) && (select[k] == 0)) {
                             clear_a_flg(a_flg, k);
                             j = k;
                      }
                      else
                             if(abar[i][k] > temp)
                                    a_flg[k] = 0;
               }
        }
     sel = get_astar(a_flg, nloc);
     select[sel]=1;
     return sel;
}

int find_min(table, num)
int table[][MAXLOCA];
int num;
{
        int i;
        int min;

        min = table[0][0];
```

```
        for(i=0; i<num; i++)
        if((table[0][i] < min) && (select[i] == 0))
                        min = table[0][i];
        return min;
}

int clear_a_flg(table, nloc)
int table[];
{
    int i;

     for(i=0; i<nloc; i++)
        table[i] = 0;
}

int mark_flg(a_flg, mark, num)
int a_flg[];
int mark, num;
{
        int i;

        for(i=0; i<num; i++) {
        if((abar[0][i] == mark) && (select[i] == 0))
            a_flg[i] = 1;
                else
            a_flg[i] = 0;
        }
}

int start_flg(a_flg, start, num)
int a_flg[];
int *start, num;
{
        int i;

    for(i=*start; i<num; i++) {
        if((a_flg[i] == 1) && (select[i] == 0)) {
            *start = i;
                        break;
                }
        }
}

int clear_flg(table_flg, end)
int table_flg[];
int end;
{
        int i;

        for(i=0; i<end; i++)
                table_flg[i] = 0;
}

int get_astar(table, num)
int table[];
```

```
int num;
{
        int i;

        for(i=0; i<num; i++)
        if((table[i] == 1) && (select[i] == 0))
                        return i;
        return 0;
}

is_best_solution(nprio, klc)
int nprio, klc;
{
   int j;

   for(j=0; j<nprio; j++) {
      if(abar[j][klc] > astar[j]) {
         select[klc] = 0;
         return YES;
      }
      else
         if(abar[j][klc] < astar[j])
            return NO;
   }
   return YES;
}

selection_improve(nfunc, nloc, nreg, nprio, klc)
int nfunc, nloc, nreg, nprio, klc;
{
        int j,k;
   int nx, mx;
   int flag = NO;

   nselect++;
   jsite[nselect] = klc;
   for(j=0; j<nprio; j++)
      astar[j] = abar[j][klc];

   nx = 0;
   for(j=0; j<nfunc; j++) {
      mx = 0;
      for(k=0; k<nreg; k++)
         if(ympr[j][k][klc] == 1) {
            isel[j][k] = klc;
            cvr[j][k] = coverage[j][k][klc];
            if(mx == 0) {
               fix[j] += fixcost[j][klc];
               mx++;
               nx++;
            }
         }
   }
   if((nx == 0) || (nx >=nloc))
      flag = YES;
```

```
    else {
       update(nfunc, nloc, nreg, klc);
       flag = NO;
    }
    return flag;
}

update(nfunc, nloc, nreg, klc)
int nfunc, nloc, nreg, klc;
{
        int i,j,k;
        int kx, ky;

    for(i=0; i<nfunc; i++)
       for(j=0; j<nreg; j++)
          for(k=0; k<nloc; k++)
             if(select[k] == 0)
                if(coverage[i][j][k] > cvr[i][j]) {
                   xmpr[i][j][k] = 0;
                   ympr[i][j][k] = 1;
                }
                else {
                   xmpr[i][j][k] = 1;
                   ympr[i][j][k] = 0;
                }

    for(i=0; i<nfunc; i++)
       for(j=0; j<nloc; j++) {
          kx=0; ky=0;
          for(k=0; k<nreg; k++) {
             if(select[j] == 0) {
                fay[i][j] = 1;
                fax[i][j] = 1;
                if(xmpr[i][k][j] == 0)
                   ky++;
                else
                   kx++;
             }
          }
          if(kx == 0) fax[i][j] = 0;
          if(ky == 0) fay[i][j] = 0;
       }

    bn += benefit[klc];
    for(j=0; j<nfunc; j++)
       rel[j] += relationship[j][klc];
}

/*-------------------------------------------------------------------------*/
```

GRF.C

```
#include <graphics.h>
#include "pie.h"
#include "const.h"

#define  TRUE        1
#define  FALSE       0
#define  SPACEBAR    0x3920
#define MAXWIDTH                30
#define  MAXLENGTH   70

extern int num_reg, num_fun, num_loc;
extern int num_edge;
extern int edge[MAXPOINT][2];
extern int loc_reg[MAXREG][3];
extern int loc_fun[MAXFUNC][3];
extern int loc[MAXLOCA][2];
extern int regi[MXF];
int cx, cy;
int unit;
static int ox=360, oy=185;
int marker[MAXWIDTH+5][MAXLENGTH+5];

int ratio(int x, int y);
int findmax(int *l, int *w);
int shape(int fcn);
int line_grid(int l, int w);
int line_shape(int point[][2]);
int mark(int u, int v, int i, int *region, int flg);
int locate_region(int length, int width);
int locate_function(int length, int width, int fcn);
int grmain(int fcn);

int ratio(x, y)
int x, y;
{
        if(x>60 || y>30) return 5;
        else return 10;
}

int findmax(l, w)
int *l, *w;
{
        int i;
        *l =0;  *w=0;
        for(i=0; i<num_edge; i++) {
                if(edge[i][0] > *l)
                        (*l) = edge[i][0];
                if(edge[i][1] > *w)
                        (*w) = edge[i][1];
        }
}

int shape(int fcn)
{
        int gr_edge[MAXPOINT][2];
```

```
        int length, width;
        int i;

        length=0;width=0;
        findmax(&length, &width);
        unit=10;
        cx=(length+1)/2; cy=(width+1)/2;
        for(i=0; i<num_edge; i++) {
            gr_edge[i][0] = ox + (edge[i][0] - cx)*unit;
            gr_edge[i][1] = oy - (edge[i][1] - cy)*unit;
        }
        line_grid(length, width);
        line_shape(gr_edge);
        locate_region(length, width);
        locate_function(length, width, fcn);
}

int line_grid(l, w)
int l, w;
{
        int i;
        int row, col, scale;
        int startx, starty, endx, endy;

        startx = ox-cx*unit; starty = oy-cy*unit;
        endx   = ox+cx*unit; endy   = oy+cy*unit;
        rectangle(startx-8, starty-8, endx+8, endy+8);
        row = ((w+1)/2)*2;  col = ((l+1)/2)*2;
        scale=unit;
        setlinestyle(DOTTED_LINE,0,1);
        for(i=0; i<=col; i++)
            line(startx+scale*i, starty, startx+scale*i, endy);
        for(i=0; i<=row; i++)
            line(startx, starty+scale*i, endx, starty+scale*i);
        setlinestyle(SOLID_LINE,0,1);
}

int line_shape(point)
int point[][2];
{
        int i;

        moveto(point[0][0], point[0][1]);
        for(i=1; i<num_edge; i++) {
            lineto(point[i][0], point[i][1]);
        }
        lineto(point[0][0], point[0][1]);
}

int mark(u, v, i, region, flg)
int u, v, i;
int *region;
int flg;
{
```

```
        int x, y;
        char r[10];

        if(marker[v][u] == 0) {
                        marker[v][u] = num_reg+i+1;
                        x = ox + (u - cx)*unit;
                        y = oy - (v - cy)*unit;
                        if(flg == 0) {
                                r[0] = 'A'+i;
                                r[1]='\0';
                        }
                        else {
                                r[0] = 'A' + i + num_reg;
                                r[1] = '\0';
                        }
                        outtextxy(x+2, y+2, r);
                        (*region)--;
        }
        if ( (*region) <= 0) return TRUE;
        else return FALSE;
}

int boundary(int x, int y, int l, int w)
{
        if(x >= 0 && x <= l && y > 0 && y <= w) return TRUE;
        else return FALSE;
}


int locate_region(length, width)
int length, width;
{
        int i, j, k;
        int reg[MAXREG][3];
        int noarea, flg;
        int u, v, x, y;
        int key;
        void *ptr1, *ptr2, *ptr3;
        unsigned size;

        reg[0][0] = ox + (loc_reg[0][0] - cx)*unit;
        reg[0][1] = oy - (loc_reg[0][1] - cy)*unit;
        reg[0][2] = loc_reg[0][2];

        for(i=1; i<num_reg; i++) {
                reg[i][0] = ox + (loc_reg[i][0] - cx)*unit;
                reg[i][1] = oy - (loc_reg[i][1] - cy)*unit;
                reg[i][2] = loc_reg[i][2];
        }
        /* locate region */
        /* clear marker */
        for(i=0; i<MAXWIDTH+4; i++)
                for(j=0; j<MAXLENGTH+4; j++) {
                        marker[i][j] = 0;
        }
```

```
    /* set internal and external point  */

    bar(2,2,8,7);
    size=imagesize(2,2,8,7);
    ptr1 = (void *)malloc(size);
    ptr2 = (void *)malloc(size);
    ptr3 = (void *)malloc(size);
    getimage(2,2,8,7,ptr1);
    putimage(2,2,ptr1,XOR_PUT);
    getimage(2,2,8,7,ptr3);

    u = 0 ; v = ((width+1)/2)*2;
    x = ox + (u - cx)*unit;
    y = oy - (v - cy)*unit;
    setfillstyle(1,0);
    do {
  x = ox + (u - cx)*unit;
          y = oy - (v - cy)*unit;
          getimage(x+2,y+2,x+8,y+7,ptr2);
          putimage(x+2,y+2,ptr1,COPY_PUT);
  while(bioskey(1) == 0)
          ;
          key = bioskey(0);
          putimage(x+2,y+2,ptr2,COPY_PUT);
          if (marker[v][u] == -1)
                  outtextxy(x+2, y+2, "*");

          switch (key) {
                          case UPKEY : v++; if(v>((width+1)/2)*2) v=1;  break;
                          case DNKEY : v--; if(v<=0) v=(width+1)/2*2;  break;
      case LEKEY : u--; if(u<0) u=length-1; break;
      case RIKEY : u++; if(u>length-1) u=0; break;
                          case SPACEBAR : if(marker[v][u] == -1)
                                                                  marker[v][u] = 0;
                      else
                                                                  marker[v][u] = -1;
                          break;
          }
          x = ox + (u - cx)*unit;
          y = oy - (v - cy)*unit;
          if(key == SPACEBAR) {
                  if (marker[v][u] == -1)
                          outtextxy(x+2, y+2, "*");
                  else
                          putimage(x+2,y+2,ptr3,COPY_PUT);
                  u++;
                  if(u>length-1)
                          u=0;
          }
  } while (key != ESCKEY);

    free(ptr1);
    free(ptr2);
    free(ptr3);
    for(i=0;i<num_reg;i++) {
```

```
            u=loc_reg[i][0];
            v=loc_reg[i][1];
            noarea = FALSE;
            if(marker[v][u] == 0 && boundary(u, v, length, width) && noarea == FALSE) {
                    mark(u, v, i, &reg[i][2], 0);
                    flg=TRUE;
            }
            if(reg[i][2] != 0) {
                    k=1;
                    noarea=FALSE;
                    do {
                            x=u+k;
                            y=v-(k-1);
        flg=FALSE;
                            for(j=0; j<2*k; j++) {
                                    if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE){
                                            noarea = mark(x, y, i, &reg[i][2], 0);
                                            flg=TRUE;
                                    }
                                    y++;
                            }
                            x--; y--;
                            for(j=0; j<2*k; j++) {
                                    if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE) {
                                            noarea = mark(x, y, i, &reg[i][2], 0);
                                            flg=TRUE;
                                    }
                                    x--;
                            }
                            x++; y--;
                            for(j=0; j<2*k; j++) {
                                    if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE){
                                            noarea = mark(x, y, i, &reg[i][2], 0);
                                            flg=TRUE;
                                    }
                                    y--;
                            }
                            x++; y++;
                            for(j=0; j<2*k; j++) {
                                    if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE) {
                                            noarea = mark(x, y, i, &reg[i][2], 0);
                                            flg=TRUE;
                                    }
                                    x++;
                            }
                            k++;
                    } while(noarea == FALSE && flg == TRUE);
            }
    }
}

int locate_function(length, width, fcn)
int length, width, fcn;
{
    int j, k;
```

```
int reg[3];
    int noarea, flg;
    int u, v, x, y;

     reg[0] = ox + (loc_fun[0][0] - cx)*unit;
     reg[1] = oy - (loc_fun[0][1] - cy)*unit;
     reg[2] = loc_fun[fcn][2];
    /* locate function */

     u=loc_fun[0][0];
     v=loc_fun[0][1];
noarea = FALSE;
if(marker[v][u] == 0 && boundary(u, v, length, width) && noarea == FALSE)
    mark(u, v, 0, &reg[2], 1);
if(reg[2] != 0) {
    k=1;
    noarea=FALSE;
    do {
        x=u+k;
        y=v-(k-1);
        flg=FALSE;
        for(j=0; j<2*k; j++) {
            if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE){
                        noarea = mark(x, y, 0, &reg[2], 1);
                        flg=TRUE;
            }
            y++;
        }
        x--; y--;
        for(j=0; j<2*k; j++) {
            if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE) {
                noarea = mark(x, y, 0, &reg[2], 1);
                flg=TRUE;
            }
            x--;
        }
        x++; y--;
        for(j=0; j<2*k; j++) {
            if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE) {
                noarea = mark(x, y, 0, &reg[2], 1);
                flg=TRUE;
            }
            y--;
        }
        x++; y++;
        for(j=0; j<2*k; j++) {
            if(marker[y][x] == 0 && boundary(x, y, length, width) && noarea == FALSE){
                noarea = mark(x, y, 0, &reg[2], 1);
                flg=TRUE;
            }
            x++;
        }
        k++;
    } while(noarea == FALSE && flg == TRUE);
}
```

```
}


grmain(int fcn)
{
        int grdriver, grmode;

        grdriver=DETECT;
        initgraph(&grdriver, &grmode, "");
        rectangle(1,0,718,25);
        outtextxy(90,10,"M U L T I O B J E C T I V E  F O R  O F F I C E  L A Y O U T");
        shape(fcn);
}

int shape2(int fcn)
{
        int gr_edge[MAXPOINT][2];
        int length, width;
        int i;

        length=0;width=0;
        findmax(&length, &width);
        unit=10;
        cx=(length+1)/2; cy=(width+1)/2;
        for(i=0; i<num_edge; i++) {
                gr_edge[i][0] = ox + (edge[i][0] - cx)*unit;
                gr_edge[i][1] = oy - (edge[i][1] - cy)*unit;
        }
        locate_region(length, width);
        locate_function(length, width, fcn);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <mem.h>
#include <alloc.h>
#include <string.h>
#include <stdarg.h>

#define MAXFUNC      12
#define MAXLOCA      25
#define MAXREG       22
#define MAXOBJ       30
#define MAXPRIO       5
#define MAXFACTOR    25
#define MAXPOINT     10
```

PIE.H

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <mem.h>
#include <alloc.h>
#include <string.h>
#include <stdarg.h>

#define MAXFUNC        5
#define MAXLOCA        25
#define MAXREG         22
#define MAXOBJ         30
#define MAXPRIO         5
#define MAXFACTOR      25
#define MAXPOINT       10

#define MXF            25
```

```
/************************************/
/*    key code constant   6/11/87 */
/************************************/

#define    ESCKEY       0x0011B
#define    RETKEY       0x01C0D
#define    BSKEY        0x00E08
#define    TABKEY       0x00F09
#define    CNTRLRET     0x01C0A
#define    CNTRLBS      0x00E7F
#define    SHFTTAB      0x00F00

/************************************/
/*    control key constant          */
/************************************/
#define    CNTRL_A      0x01E01
#define    CNTRL_B      0x03002
#define    CNTRL_C      0x02E03
#define    CNTRL_D      0x02004
#define    CNTRL_E      0x01205
#define    CNTRL_F      0x02106
#define    CNTRL_G      0x02207
#define    CNTRL_H      0x02308
#define    CNTRL_I      0x01709
#define    CNTRL_J      0x0240A
#define    CNTRL_K      0x0250B
#define    CNTRL_L      0x0260C
#define    CNTRL_M      0x0320D
#define    CNTRL_N      0x0310E
#define    CNTRL_O      0x0180F
#define    CNTRL_P      0x01910
#define    CNTRL_Q      0x01011
#define    CNTRL_R      0x01312
#define    CNTRL_S      0x01F13
#define    CNTRL_T      0x01414
#define    CNTRL_U      0x01615
#define    CNTRL_V      0x02F16
#define    CNTRL_W      0x01117
#define    CNTRL_X      0x02D18
#define    CNTRL_Y      0x01519
#define    CNTRL_Z      0x02C1A
#define    CNTRLESC     0x01A1B
#define    CNTRLFS      0x02B1C
#define    CNTRLGS      0x01B1D
#define    CNTRLRS      0x0071E
#define    CNTRLUS      0x00C1F

/************************************/
/*    alternate code constant       */
/************************************/
#define    ALT1         0x07800
#define    ALT2         0x07900
#define    ALT3         0x07A00
#define    ALT4         0x07B00
#define    ALT5         0x07C00
```

```
#define    ALT6         0x07D00
#define    ALT7         0x07E00
#define    ALT8         0x07F00
#define    ALT9         0x08000
#define    ALT0         0x08100
#define    ALTA         0x01E00
#define    ALTB         0x03000
#define    ALTC         0x02E00
#define    ALTD         0x02000
#define    ALTE         0x01200
#define    ALTF         0x02100
#define    ALTG         0x02200
#define    ALTH         0x02300
#define    ALTI         0x01700
#define    ALTJ         0x02400
#define    ALTK         0x02500
#define    ALTL         0x02600
#define    ALTM         0x03200
#define    ALTN         0x03100
#define    ALTO         0x01800
#define    ALTP         0x01900
#define    ALTQ         0x01000
#define    ALTR         0x01300
#define    ALTS         0x01F00
#define    ALTT         0x01400
#define    ALTU         0x01600
#define    ALTV         0x02F00
#define    ALTW         0x01100
#define    ALTX         0x02D00
#define    ALTY         0x01500
#define    ALTZ         0x02C00

/************************************/
/*    function key constant         */
/************************************/
#define    HOMEKEY      0x04700
#define    CHOMEKEY     0x07700
#define    UPKEY        0x04800
#define    PGUPKEY      0x04900
#define    CPGUPKEY     0x08400
#define    LEKEY        0x04B00
#define    CLEKEY       0x07300
#define    RIKEY        0x04D00
#define    CRIKEY       0x07400
#define    ENDKEY       0x04F00
#define    CENDKEY      0x07500
#define    DNKEY        0x05000
#define    PGDNKEY      0x05100
#define    CPGDNKEY     0x07600
#define    INSKEY       0x05200
#define    DELKEY       0x05300

#define    F1KEY        0x03B00
#define    F2KEY        0x03C00
#define    F3KEY        0x03D00
```

```
#define     F4KEY       0x03E00
#define     F5KEY       0x03F00
#define     F6KEY       0x04000
#define     F7KEY       0x04100
#define     F8KEY       0x04200
#define     F9KEY       0x04300
#define     F10KEY      0x04400
#define     UF1KEY      0x05400
#define     UF2KEY      0x05500
#define     UF3KEY      0x05600
#define     UF4KEY      0x05700
#define     UF5KEY      0x05800
#define     UF6KEY      0x05900
#define     UF7KEY      0x05A00
#define     UF8KEY      0x05B00
#define     UF9KEY      0x05C00
#define     UF10KEY     0x05D00

#define     CF1KEY      0x05E00
#define     CF2KEY      0x05F00
#define     CF3KEY      0x06000
#define     CF4KEY      0x06100
#define     CF5KEY      0x06200
#define     CF6KEY      0x06300
#define     CF7KEY      0x06400
#define     CF8KEY      0x06500
#define     CF9KEY      0x06600
#define     CF10KEY     0x06700
#define     AF1KEY      0x06800
#define     AF2KEY      0x06900
#define     AF3KEY      0x06A00
#define     AF4KEY      0x06B00
#define     AF5KEY      0x06C00
#define     AF6KEY      0x06D00
#define     AF7KEY      0x06E00
#define     AF8KEY      0x06F00
#define     AF9KEY      0x07000
#define     AF10KEY     0x07100
```

ประวัติผู้เขียน

นายพินัดน์ ศรีกิดดินุรักษ์ เกิดเมื่อวันที่ 8 กันยายน พ.ศ.2507 การศึกษาระดับมัธยมศึกษาตอนปลาย ณ. โรงเรียนเทพศิรินทร์ สำเร็จการศึกษาระดับปริญญาตรี คณะวิทยาศาสตร์ สาขา เคมี สถาบันเทคโนโลยีพระจอมเกล้า ธนบุรี ในปี 2528 หลังจากนั้นในปี 2529 ศึกษาต่อระดับปริญญาโท คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมอุตสาหการ จุฬาลงกรณ์มหาวิทยาลัย