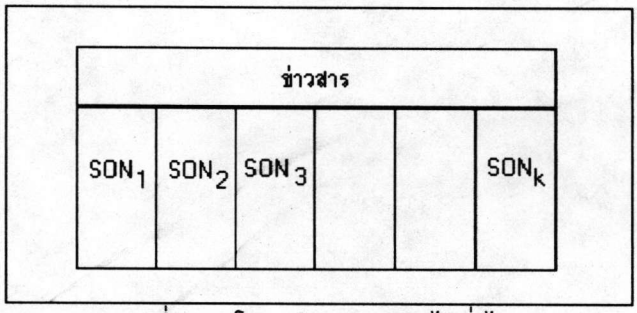
บทที่ 4

การออกแบบคอนโทรล

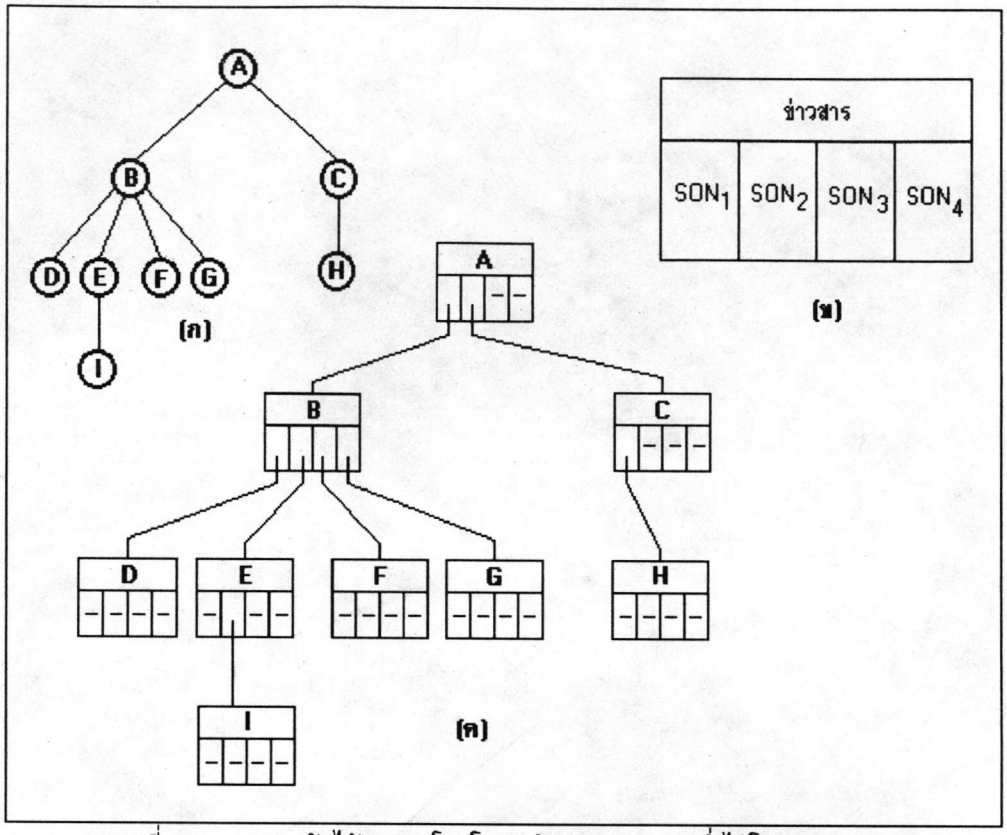
การแทนโครงสร้างข้อมูลแบบต้นไม้ในหน่วยความจำ

จากที่กล่าวถึงในบทที่ 2 หัวข้อโครงสร้างข้อมูลแบบต้นไม้ เราสามารถเก็บในคอมพิวเตอร์โดยให้โครงสร้างของแต่ละบัพเป็นดังรูปที่ 4.1



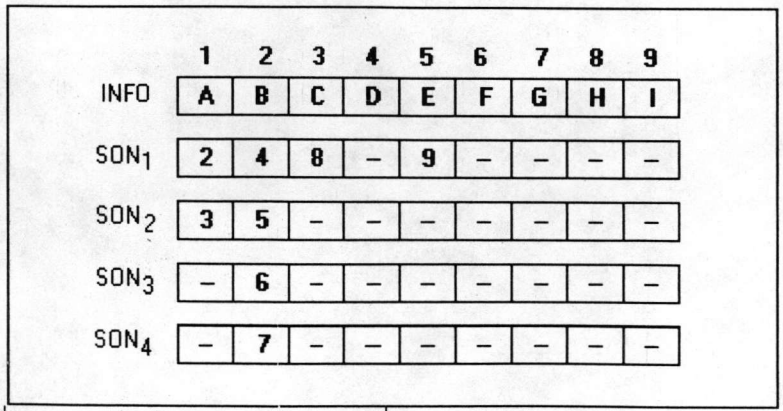
รูปที่ 4.1 โครงสร้างบัพของต้นไม้ทั่วไป

เนื่องจากแต่ละบัพของต้นไม้จะมีบัพลูก (SON) เท่าไรก็ได้ ดังนั้นจึงให้ส่วนเชื่อมโยงของบัพมี k ส่วน ถ้าบัพหนึ่งในต้นไม้นั้นมี 3 บัพลูก ก็จะได้ k = 3 ซึ่งแสดงว่าโครงสร้างของบัพนั้น นอกจากจะมีส่วนข่าวสารแล้วยังมีส่วนเชื่อมโยง 3 ส่วน คือ SON_1 , SON_2 และ SON_3 แต่ถ้าอีกบัพหนึ่งในต้นไม้นั้นมี 5 บัพลูก โครงสร้างของบัพนั้นจะมีส่วนเชื่อมโยง 5 ส่วนคือ SON_1 ถึง SON_5 จะเห็นได้ว่าการแทนบัพลักษณะนี้ขนาดของพื้นที่ความจำที่ใช้สร้างแต่ละบัพ จะมีขนาดไม่คงที่ขึ้นอยู่กับจำนวนบัพลูกของบัพนั้น การจัดการกับโครงสร้างที่มีขนาดไม่เท่ากันตลอดนี้ จะมีความยุ่งยากมากในแง่การจัดสรรและยกเลิกการจัดสรรพื้นที่ความจำให้แก่บัพ วิธีหนึ่งที่จะลดความยุ่งยากนี้ก็คือ ให้จำนวนของบัพลูกคงที่ตลอด ทั้งนี้ให้กำหนดว่า ค่า k คือ ค่าดีกรีสูงสุดของบัพในต้นไม้นั้นมีค่าคงที่ ตามรูปที่ 4.2 (ก) ต้นไม้ A มีดีกรีสูงสุดเท่ากับ 4 (จำนวนบัพลูกของบัพ B) ดังนั้นบัพที่ใช้จะมีโครงสร้างดังรูปที่ 4.2 (ข)
(สุชาย ธนวเสถียร และ วิชัย จิวังกูร, 2535)



รูปที่ 4.2 การแทนต้นไม้รูป (ก) โดยโครงสร้างบัพรูป (ข) ซึ่งได้โครงสร้าง (ค)

โครงสร้างต้นไม้ดังรูปที่ 4.2 อาจจำลองด้วยช่องลำดับ 5 ชุด ชื่อ INFO(9) , SON_1 (1:9) , SON_2 (1:9) , SON_3 (1:9) , SON_4 (1:9) แต่ละชุดจะมีช่องลำดับ 9 ช่อง เนื่องจากต้นไม้นี้มี 9 บัพ รูปที่ 4.3 เป็นลักษณะค่าในแต่ละช่องลำดับ ค่าตัวเลขที่เติมไว้ได้มาจากการมองดูต้นไม้รูปที่ 4.2 (ก) และเลขลำดับของข่าวสารภายในช่องลำดับ ทั้งนี้เราต้องเก็บข่าวสารของบัพไว้ในช่องลำดับชื่อ INFO ดังนั้น INFO (1) จะเก็บค่า A SON_1 (1) จะต้องชี้ไปยัง B ซึ่งคือ 2 ส่วน SON_2 (1) ต้องชี้ไปยังบัพ C ซึ่งอยู่ที่ตำแหน่ง 3 ใน INFO ดังนั้นค่าของ SON_2 (1) จึงเป็น 3 ส่วนค่าของ SON_3 (1) และ



รูปที่ 4.3 การแทนโครงสร้างต้นไม้ตามรูปที่ 4.2 (ก) ด้วยช่องลำดับ 5 ชุด โดยใช้โครงสร้างบัพตามรูปที่ 4.2 (ข) โดยให้ 1 บัพใช้ 1 ช่องของทั้ง 5 ช่องลำดับ

SON_4 (1) จะเป็นศูนย์ (หรือตัวพิเศษอะไรก็ได้ ที่แสดงว่าเป็นส่วนเชื่อมโยงที่เป็นศูนย์หรือว่างเปล่า) ในที่นี้เขียนแทนด้วย - (สุชาย ธนวเสถียร และ วิชัย จิวังกูร, 2535)

จากรูปที่ 4.2 และ 4.3 จะเห็นได้ว่า ช่องส่วนที่ใช้เป็นตัวชี้จะว่างเปล่าและไม่ได้ใช้ประโยชน์อยู่มาก ทั้งนี้เนื่องจากดีกรีของบัพต่างกันนั่นเอง ถ้าทุกบัพในต้นไม้มีดีกรีเท่ากัน การแทนในคอมพิวเตอร์จะไม่มีช่องว่างในส่วนตัวชี้เลย ถ้าดีกรีระหว่างบัพต่างกันมาก ส่วนตัวชี้ก็จะสูญเปล่ามาก เพราะเราให้ขนาดของบัพที่ใช้เท่ากันตลอดดังที่ได้เคยกล่าวไว้แล้ว (สุชาย ธนวเสถียร และ วิชัย จิวังกูร, 2535)

ปกติถ้าต้นไม้มีดีกรี k (k คือค่าดีกรีสูงสุดของบัพในต้นไม้นั้น) และมีจำนวนบัพ n บัพ การแทนโดยโครงสร้างที่กล่าวมาแล้วจะต้องใช้

ส่วนข่าวสาร n ช่อง

ส่วนตัวชี้ทั้งหมด kn ช่อง

เนื่องจากต้นไม้นี้มี n บัพ ดังนั้นจำนวนช่องตัวชี้ที่ใช้จริงเพียง n - 1 ช่อง (ไม่นับบัพราก เพราะไม่มีบัพใดชี้ไปยังบัพราก) ดังนั้น

จำนวนช่องตัวชี้ที่ไม่ได้ใช้ประโยชน์ = kn - (n - 1)

= n (k - 1) + 1

ตามรูปที่ 4.2 และ 4.3 เมื่อ n = 9 และ k = 4 จะเห็นได้ว่า จำนวนช่องที่ไม่ได้ใช้ประโยชน์เท่ากับ 9 (4 - 1) + 1 = 28 ช่อง

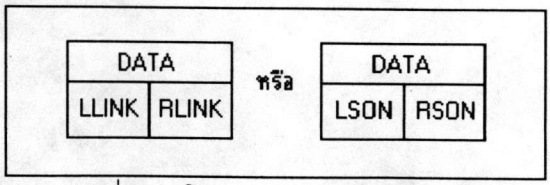
<u>การแทนต้นไม้ไบนารีในหน่วยความจำ</u>

สามารถแทนได้ 2 แบบ คือ

1. <u>การแทนโดยอาศัยตัวชี้</u> (Pointer)

ทำได้โดยให้แต่ละบัพมีโครงสร้างดังรูปที่ 4.4

LLINK หรือ LSON เป็นตัวชี้ไปยังต้นไม้ย่อยทางซ้าย ส่วน RLINK หรือ RSON เป็นตัวชี้ไปยังต้นไม้ย่อยทางขวา ส่วนต้นไม้ไบนารีทั้งต้นจะแทนด้วย T โดยที่ T เป็นตัวชี้ไปยังบัพราก



รูปที่ 4.4 โครงสร้างบัพของต้นไม้ไบนารีโดยอาศัยตัวชี้

ถ้า T เท่ากับ NULL แสดงว่าต้นไม้ไบนารีนั้นว่างเปล่า

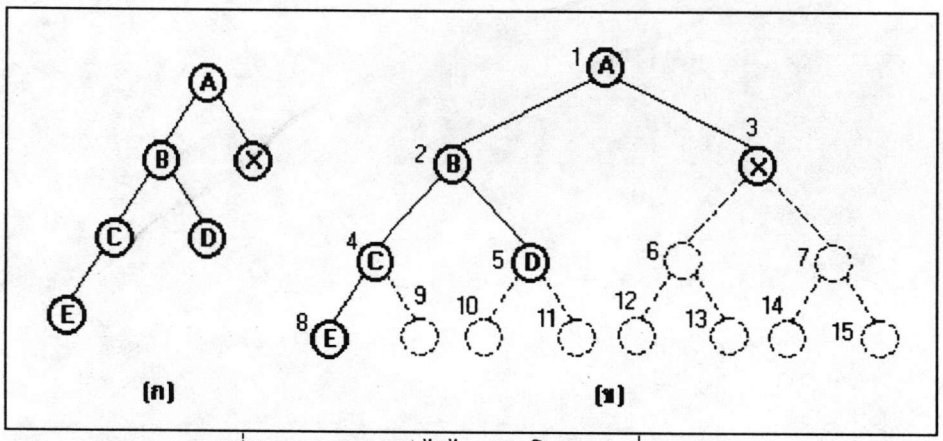
ถ้า T ไม่เท่ากับ NULL แสดงว่าต้นไม้ไบนารีนั้นไม่ว่างเปล่า และ T จะเป็นที่อยู่ในหน่วยความจำของบัพราก

วิธีการแทนลักษณะนี้จะช่วยให้การเพิ่ม และลบลัพที่ส่วนกลางของต้นไม้ง่ายขึ้นมาก แต่เมื่อต้องการหาบัพพ่อของบัพใด ๆ (i) อาจมีข้อยุ่งยากบ้าง แต่ก็สามารถใช้กลไกของกองซ้อนช่วยได้ (สุชาย ธนวเสถียร และ วิชัย จิวังกูร, 2535)

2. <u>การแทนโดยอาศัยที่อยู่ของบัพ</u> (หรือการแทนแบบซีเควนเชียล (Sequential))

วิธีนี้เป็นการแทนต้นไม้ไบนารีด้วยช่องลำดับมิติเดียว 1 ชุด การแทนแบบนี้เหมาะกับโครงสร้างต้นไม้ไบนารีแบบสมบูรณ์ที่สุด

การแทนจะเริ่มต้นด้วยการให้หมายเลขแก่แต่ละบัพ ตั้งแต่ระดับ 1 ระดับ 2 .... ไปเรื่อย ๆ จนถึงระดับ k การให้ตัวเลขในแต่ละระดับจะให้จากซ้ายไปขวา โดยให้บัพรากมีหมายเลข 1 เสมอ (รูปที่ 4.5) การให้ตัวเลขจะต้องถือว่า ต้นไม้ไบนารีเป็นต้นไม้ไบนารีแบบสมบูรณ์ ดังนั้นรูปต้นไม้ 4.5 (ก) จะต้องถูกต่อเติมให้เป็นต้นไม้ไบนารีแบบสมบูรณ์ดังรูปที่ 4.5 (ข) จึงจะให้ตัวเลขที่อยู่แก่บัพได้ (สุชาย ธนวเสถียร และ วิชัย จิวังกูร, 2535)



รูปที่ 4.5 การแทนต้นไม้ไบนารี โดยอาศัยที่อยู่ของบัพ

จากรูปที่ 4.5 จะต้องใช้ช่องลำดับจำนวน 15 ช่อง เพื่อเก็บข้อมูลต้นไม้นี้ ตำแหน่งของข่าวสารในต้นไม้จะตรงกับตำแหน่งในช่องลำดับดังรูปที่ 4.6

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	X	C	D	-	-	E	-	-	-	-	-	-	-

รูปที่ 4.6 การแทนต้นไม้รูปที่ 4.5 ด้วยช่องลำดับ

โดยทั่วไปแล้วต้นไม้ไบนารีที่มีความสูง (ระบุโดยค่าระดับที่มากที่สุด) h จะต้องใช้ช่องลำดับที่มีขนาด $2^{h+1} - 1$ ช่อง และเมื่อแทนด้วยช่องลำดับแล้ว เราจะต้องมีวิธีหาว่า "พ่อ" ของบัพ C คือใคร หรือว่า "ลูก" ของบัพ B คือใคร โดย FATHER , LSON และ RSON ของบัพใด ๆ (i) จะหาได้โดยความสัมพันธ์ดังนี้

- FATHER (i) จะอยู่ที่ตำแหน่ง $\lfloor \frac{i}{2} \rfloor$ ถ้า $i \neq 1$ แต่ถ้า $i = 1$ บัพ i จะเป็นบัพราก ซึ่งจะไม่มี FATHER (i)

- LSON (i) จะอยู่ที่ตำแหน่ง $2i$ ถ้า $2i \leq n$ แต่ถ้า $2i > n$ แล้ว i จะไม่มีลูกทางซ้าย

- RSON (i) จะอยู่ที่ตำแหน่ง $2i + 1$ ถ้า $2i + 1 \leq n$ แต่ถ้า $2i + 1 > n$ แล้ว i จะไม่มีลูกทางขวา

คอนโทรลทั้งสามที่สร้างขึ้นใช้การแทนแบบซีเควนเชียลนี้ โดยได้กำหนดคุณสมบัติเป็นแบบช่องลำดับของอักขระปิดท้ายด้วย " \0 " ไว้ในโครงสร้างของผู้เขียนโปรแกรม จำนวน 1,024 ช่อง คือช่องที่ 0 ถึง 1,023 โดยช่องแรก (ช่องที่มีดัชนีเป็น 0) นั้นไม่ใช้ ทำให้ได้ต้นไม้ที่มีความสูงไม่เกิน 9 ระดับ ($2^{9+1} - 1 = 1,023$)

```
....
#define  MAXNODE   1023
....
....
typedef struct tagBISEARCH                    // Binary Search Tree Control
  {
    ....
    HSZ     Key[MAXNODE + 1];
    SHORT  OccupiedArr[MAXNODE + 1];
    ....
    ....
} BISEARCH;
```

ตัวแปร OccupiedArr ใช้ในการตรวจสอบว่าช่องลำดับนั้นมีข้อมูลอยู่แล้วหรือไม่เท่านั้น (ถูกจับจองแล้วหรือไม่) จึงไม่ต้องกำหนดโครงสร้าง PROPINFO ให้ แต่ต้องกำหนดโครงสร้าง PROPINFO ของตัวแปร Key ไว้เป็น

```
PROPINFO Property_Key =
  {
  "Key",
  DT_HSZ | PF_fPropArray | PF_fGetMsg | PF_fSetMsg | PF_fNoShow ,
  OFFSETIN(BISEARCH, Key),
  0, 0, NULL, 0
};
```

สัญรูปที่ใช้แทนคอนโทรลในหน้าต่างกล่องเครื่องมือ

สัญรูปที่ใช้แทนคอนโทรลในหน้าต่างกล่องเครื่องมือนั้น จะต้องใช้แฟ้มบิตแมพทั้งหมด 4 แฟ้ม 2 แฟ้มสำหรับจอภาพแบบ VGA เพราะต้องใช้สำหรับแสดงตอนที่ไม่ได้มีการคลิกเมาส์ในสัญรูป 1 แฟ้ม และมีการคลิกเมาส์อีก 1 แฟ้ม ส่วนจอ EGA และ Monochrome นั้นใช้อย่างละ 1 แฟ้ม โดยที่เมื่อมีการคลิกเมาส์ที่สัญรูป ก็จะใช้การสลับสีของแต่ละบิตในภาพแทน ส่วนจอภาพแบบ CGA นั้นจะใช้ภาพบิตแมพเดียวกับที่ใช้กับจอ Monochrome

ตารางที่ 4.1 สัญรูปที่ใช้แทนคอนโทรลทั้งสามในหน้าต่างกล่องเครื่องมือ

สัญรูปสำหรับ	ต้นไม้ไบนารี	ต้นไม้ไบนารีค้นหาข้อมูล	ต้นไม้ที่มีความสูงสมดุล
จอภาพ VGA ตอนที่ไม่ได้ถูกคลิกด้วยเมาส์			
จอภาพ VGA ตอนที่ถูกคลิกด้วยเมาส์			
จอภาพ Monochrome			
จอภาพ EGA			

คุณสมบัติของคอนโทรล

1. คุณสมบัติมาตรฐาน (STANDARD PROPERTY) คอนโทรลทั้งสามจะมีคุณสมบัติมาตรฐานดังต่อไปนี้

1.1 BackColor

1.2 BorderStyle

1.3 DragIcon

1.4 DragMode

1.5 FontBold, FontItalic, FontStrikethru, FontUnderLine

1.6 FontName

1.7 FontSize

1.8 HelpContextID

1.9 Height, Width

1.10 hWnd

1.11 Index

1.12 Left

1.13 MousePointer

1.14 Name

1.15 Parent

1.16 Top

1.17 Tag

1.18 Visible

2. คุณสมบัติที่สร้างขึ้นใหม่ (CUSTOM PROPERTY) ตารางที่ 4.2 เป็นการเปรียบเทียบคุณสมบัติที่สร้างขึ้นใหม่ของคอนโทรลทั้งสาม โดยเครื่องหมาย ✓ หมายถึง คอนโทรลมีคุณสมบัตินั้น

ตารางที่ 4.2 คุณสมบัติของคอนโทรลทั้งสาม

คุณสมบัติ	ต้นไม้ไบนารี	ต้นไม้ไบนารีค้นหาข้อมูล	ต้นไม้ที่มีความสูงสมดุล
AnimationSpeed		✓	✓
BranchWidth	✓	✓	✓
DeletedKey		✓	✓
EnableInsDelBox		✓	✓
FoundNode		✓	✓
Insertedkey		✓	✓
InOrder, PreOrder, PostOrder	✓	✓	✓
Key	✓	✓	✓
LeftNode	✓	✓	✓
LevelDistance	✓	✓	✓
LevelLimit	✓	✓	✓
ListOfKeys		✓	✓
LoadNodeShape	✓	✓	✓
NodeDistance	✓	✓	✓
NodeShape	✓	✓	✓

ตารางที่ 4.2 (ต่อ) คุณสมบัติของคอนโทรลทั้งสาม

NumberedNode	✓	✓	✓
ParentNode	✓	✓	✓
ResponseMsg	✓	✓	✓
RightNode	✓	✓	✓
Root	✓	✓	✓
ScrollBar	✓	✓	✓
SearchKey		✓	✓
ShowWeight			✓
TreeLevel	✓	✓	✓
ValueType		✓	✓
ValuePosition	✓	✓	✓

รายละเอียดของทุกคุณสมบัติมีอยู่ในภาคผนวก ค

<u>เหตุการณ์ของคอนโทรล</u>

1. เหตุการณ์มาตรฐาน (STANDARD EVENT) คอนโทรลทั้งสามมีเหตุการณ์มาตรฐานดังนี้
   1.1 Click
   1.2 DblClick
   1.3 DragDrop
   1.4 DragOver
   1.5 MouseDown , MouseUp
   1.6 MouseMove

2. เหตุการณ์ที่สร้างขึ้นใหม่ (CUSTOM EVENT) เป็นดังตารางที่ 4.3 โดยเครื่องหมาย ✓ หมายถึง คอนโทรลมีเหตุการณ์นั้น

ตารางที่ 4.3 เหตุการณ์ของคอนโทรลทั้งสาม

เหตุการณ์	ต้นไม้ไบนารี	ต้นไม้ไบนารีค้นหาข้อมูล	ต้นไม้ที่มีความสูงสมดุล
NodeDecrease		✓	✓
NodeIncrease		✓	✓
TreeChange		✓	✓

รายละเอียดเพิ่มเติมของทุกเหตุการณ์มีอยู่ในภาคผนวก ค

<u>วิธีของคอนโทรล</u>

คอนโทรลทั้งสามมีวิธีดังต่อไปนี้

1. AddItem
2. Clear
3. Drag
4. Move
5. Refresh
6. RemoveItem
7. Z-Order

โดยรายละเอียดของทุกวิธีมีอยู่ในภาคผนวก ค

<u>ลักษณะของคอนโทรลที่เพิ่งถูกสร้างขึ้นในฟอร์ม</u>

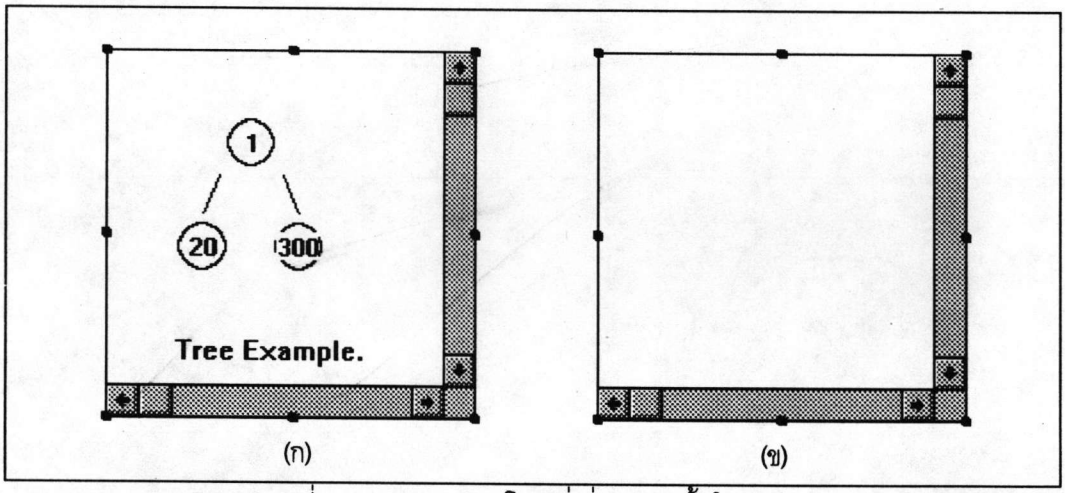
1. <u>คอนโทรลต้นไม้ไบนารี</u>

จะมีลักษณะที่เพิ่งถูกสร้างขึ้นในฟอร์มดังรูปที่ 4.7 (ก) ซึ่งเป็นลักษณะของต้นไม้ตัวอย่างที่มีข้อมูลเป็น 1, 20 และ 300 ในช่วงเวลาการออกแบบไม่สามารถเพิ่มข้อมูลเข้าไปในคอนโทรลนี้ได้ แต่เมื่อทำการเปลี่ยนแปลงค่าของคุณสมบัติต่าง ๆ ของคอนโทรลนี้ในหน้าต่างคุณสมบัติ ต้นไม้ตัวอย่างก็จะมีลักษณะเปลี่ยนแปลงไปตามค่าของคุณสมบัตินั้น ๆ ที่มีผลต่อลักษณะของคอนโทรล สามารถเพิ่มข้อมูลเข้าไปในคอนโทรลได้ด้วยคุณสมบัติ Key หรือใช้วิธี AddItem โดยต้องทำการเขียนโปรแกรมขึ้นเอง

และลบข้อมูลในแต่ละบัพได้ด้วยการให้ค่าอักขระว่าง "" แก่คุณสมบัติ Key หรือใช้วิธี RemoveItem ก็ได้

2. <u>คอนโทรลต้นไม้ไบนารีค้นหาข้อมูลและต้นไม้ที่มีความสูงสมดุล</u>

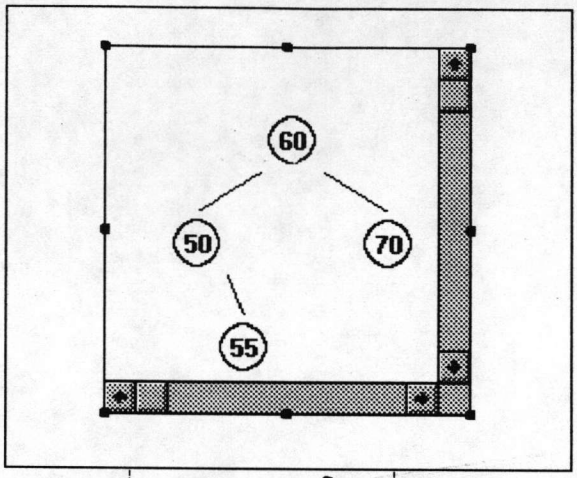
มีลักษณะที่เพิ่งถูกสร้างขึ้นในฟอร์มดังรูปที่ 4.7 (ข) สามารถเพิ่มข้อมูลได้ด้วยคุณสมบัติ InsertedKey ทำการลบข้อมูลด้วยคุณสมบัติ DeletedKey และค้นหาข้อมูลด้วยคุณสมบัติ SearchKey ได้ทั้งในช่วงเวลาการออกแบบและช่วงเวลาดำเนินงาน หรือจะเพิ่มข้อมูลด้วยการกำหนดค่าของคุณสมบัติ Key หรือใช้วิธี AddItem หรือจะลบข้อมูลด้วยวิธี RemoveItem โดยการเขียนโปรแกรมขึ้นเองก็ได้ และเมื่อทำการเปลี่ยนแปลงค่าของคุณสมบัติต่าง ๆ ของคอนโทรลนี้ในหน้าต่างคุณสมบัติ ต้นไม้ก็จะมีลักษณะเปลี่ยนแปลงไปตามค่าของคุณสมบัตินั้น ๆ ที่มีผลต่อลักษณะของคอนโทรล



รูปที่ 4.7 ลักษณะคอนโทรลที่เพิ่งถูกสร้างขึ้นในฟอร์ม

ลักษณะของคอนโทรลเมื่อมีข้อมูล

ลักษณะของคอนโทรลทั้งสามเมื่อมีข้อมูลเป็นดังรูปที่ 4.8



รูปที่ 4.8 ลักษณะคอนโทรลเมื่อมีข้อมูล