บทที่ 5
โครงสร้างลิงค์ลิสต์
(Linked-Lists)

เนื่องจาก โครงสร้างข้อมูลแบบอาร์เรย์ยังไม่ยืดหยุ่นพอ โดยเฉพาะในการเพิ่มและลบข้อมูลในโครงสร้างข้อมูล อีกทั้งการบริหารจัดการเนื้อที่ยังไม่มีประสิทธิภาพเท่าที่ควร โครงสร้างข้อมูลแบบสแต็กและคิว มีหลักการทำงานเฉพาะ เหมาะกับข้อมูลที่มีคุณลักษณะตรงกับรูปแบบของแต่ละโครงสร้างเท่านั้น จึงจำเป็นต้องแสวงหาโครงสร้างข้อมูลที่มีความยืดหยุ่นมากขึ้น สนับสนุนการเพิ่มและการลบข้อมูลได้อย่างสะดวกรวดเร็วมากยิ่งขึ้น สามารถจัดการกับข้อมูลได้อย่างยืดหยุ่นกว่าข้อมูลที่กล่าวไปแล้ว ลิงค์ลิสต์หรือรายการโยงจึงถูกนำมาใช้งาน เนื้อหาบทนี้ นำเสนอความหมาย ประเภท การสร้างโครงสร้างข้อมูล การเพิ่ม การเชื่อมโยง การแทรกข้อมูล และการลบข้อมูล ของลิงค์ลิสต์ทางเดียว การค้นหาข้อมูลในลิงค์ลิสต์ทางเดียว การสร้างลิงค์ลิสต์และดำเนินการกับลิงค์ลิสต์แบบสองทาง รวมถึงแนวทางการประยุกต์ใช้งานลิงค์ลิสต์เบื้องต้น

วัตถุประสงค์

ให้มีความรู้เกี่ยวกับ

1. แนวคิดและความหมายของโครงสร้างลิงค์ลิสต์
2. ประเภทของลิงค์ลิสต์
3. การสร้างและการท่องลิงค์ลิสต์ทางเดียวและลิงค์ลิสต์แบบสองทาง
4. ตัวดำเนินการ เพิ่ม แทรก ลบ และค้นหาข้อมูลในลิงค์ลิสต์แบบทางเดียว
5. แนวคิดของลิงค์ลิสต์แบบสองทาง ลิงค์ลิสต์แบบวงกลม
6. แนวคิดและรูปแบบของลิงค์ลิสต์แบบสองทาง
7. ตัวดำเนินการ เพิ่ม แทรก ลบ และค้นหาข้อมูลในลิงค์ลิสต์แบบทางสองทาง
8. แนวทางการประยุกต์ใช้โครงสร้างข้อมูลลิงค์ลิสต์โดยใช้ภาษาจาวา

5.1 แนวคิดเบื้องต้นของลิงค์ลิสต์

ลิงค์ลิสต์คือโครงสร้างข้อมูลที่ข้อมูลแต่ละรายการโยงกันต่อเนื่องกันไปจากจุดเริ่มต้น เชื่อมโยงข้อมูลรายการแรกหลังจากนั้นจะโยงต่อกันไปแบบรายการต่อรายการ ไปจนถึงรายการสุดท้าย ข้อมูลแต่ละรายการเรียกว่า โนด (Node) การเชื่อมโยงต่อกันใช้ตัวเชื่อมโยงเรียกว่า ลิงค์ (Link) การเชื่อมโยงสามารถเชื่อมโยงได้ทั้งทางเดียวและสองทาง โครงสร้างข้อมูลชนิดนี้ มีจุดเด่นคือสามารถเพิ่ม ลบ ข้อมูลแต่ละโนดในโครงสร้างได้โดยง่าย เพื่อให้สามารถทำความเข้าใจในแนวคิดของการดำเนินการ อัลกอริทึมและตัวแปรที่เกี่ยวข้อง นิยามที่ 5.1 ระบุสิ่งที่ต้องนำมาอธิบายตลอดเนื้อหาเกี่ยวกับลิงค์ลิสต์

นิยามที่ 5.1

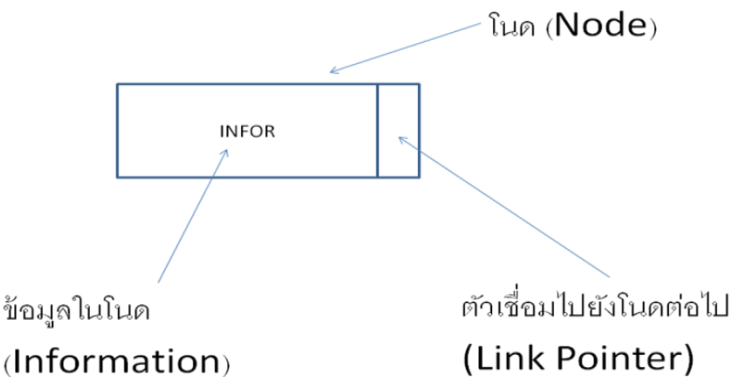
กำหนดให้ NODE คือ หมายถึงข้อมูลรายการลิงค์ลิสต์ 1 รายการที่ประกอบด้วยส่วนข้อมูลและส่วนของการลิงค์ โดยที่

INFOR คือ ข้อมูลที่บรรจุโนด

LINK คือ พอยเตอร์สำหรับเชื่อมโยงไปยัง NODE

START คือ เป็นพอยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์

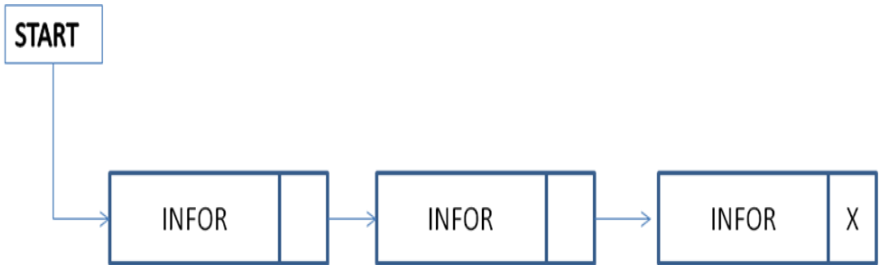
จากนิยามดังกล่าวแสดงเป็นรูปภาพได้ดังนี้



รูปที่ 5.1 แสดงแนวคิดของโนดเดี่ยวแบบมีลิงค์ทางเดียว

ลิงค์ลิสต์จะมีการสร้างโนดขึ้นมาแล้วบรรจุข้อมูลลงไป จากนั้นเชื่อมโยงต่อกันให้เป็น

โครงสร้างที่โยงต่อกันไป แสดงตัวอย่างแนวคิดเมื่อเป็นลิงค์ลิสต์ที่สมบูรณ์ดังรูปที่ 5.2

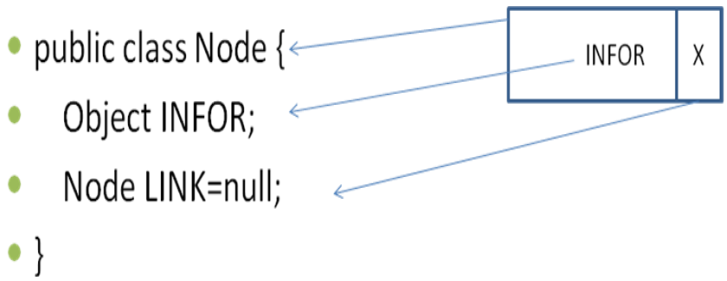


รูปที่ 5.2 แสดงการเชื่อมโยงกันของโนดแบบมีลิงค์ทางเดียว

การเขียนโปรแกรมด้วยภาษาจาวาเพื่อสร้างโนด

```
public class Node {
    Object INFOR;
    Node LINK=null;
}
```

อธิบายรายละเอียดประกอบโครงสร้างโนดที่สร้างขึ้นได้ ดังรูปที่ 5.3

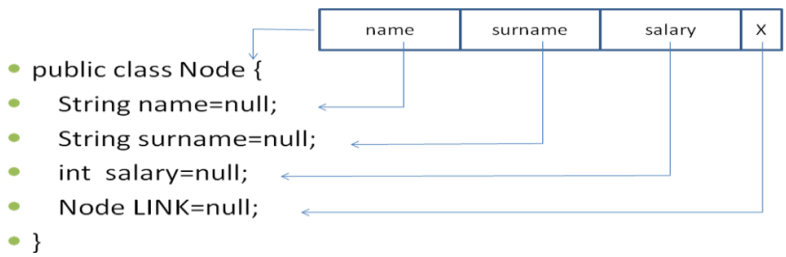


รูปที่ 5.3 แสดงการโครงสร้างข้อมูลโนดที่สร้างจากภาษาจาวา

หากมีข้อมูลที่ต้องการจัดเก็บในส่วนของ INFOR มากกว่าหนึ่งรายการ สามารถแสดงตัวอย่างการสร้างโนดสำหรับจัดเก็บข้อมูล ชื่อ สกุล เงินเดือน ดังนี้

```
public class Node {
    String name=null;
    String surname=null;
    int  salary=null;
    Node LINK=null;
}
```

จากโปรแกรมสามารถแสดงรายละเอียดประกอบภาพ ตามรูปที่ 5.4



รูปที่ 5.4 แสดงการตัวอย่างโครงสร้างที่สร้างจากภาษาจาวา แบบมีลิงค์ทางเดียว

แนวคิดต่อไปนี้แสดงตัวอย่างเบื้องต้นเกี่ยวกับการสร้างโนดและบรรจุข้อมูลเข้าไปในโนดด้วยภาษาจาวา เพื่อจะใช้ประกอบการอธิบายตัวดำเนินการและโปรแกรมในลำดับต่อไป

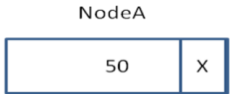
1. การสร้างโนดด้วยภาษาจาวา สามารถดำเนินการด้วยการประกาศอ็อบเจค Node ที่สร้างไว้แล้วก่อนหน้า ดังนี้

   Node NodeA = new Node();

2. เมื่อสร้างโนดขึ้นมาแล้ว การบรรจุข้อมูลเข้าไปในโนด สามารถดำเนินการได้ดังนี้

   NodeA.INFOR = 50;

เมื่อดำเนินการทั้งการสร้างและบรรจุข้อมูลแล้ว จะได้โนดตามแนวคิดดังกล่าวดังรูปที่ 5.5



รูปที่ 5.5 แสดงข้อมูลในโนด A

ตัวอย่างแนวคิดการสร้างลิงค์ลิสต์และการเชื่อมโยงข้อมูลที่ละขั้นตอน เริ่มจากการสร้างโนด บรรจุข้อมูล และทำการเชื่อมโยงโดยให้ START เป็นจุดเริ่มต้นของลิงค์ลิสต์ อธิบายพอสังเขปดังนี้

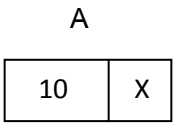
(1) สร้างโนดจากโครงสร้างที่กำหนดไว้ให้

Node A = New Node( ); Node B = New Node( ); จะได้โนดขึ้นมาดังนี้

รูปที่ 5.6 แสดงโนด A และ B เมื่อถูกสร้างขึ้น

(2) บรรจุข้อมูลลงไป โดยระบุ

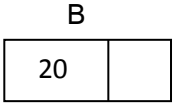
A.INFOR = 10



รูปที่ 5.7 แสดงโนด A เมื่อบรรจุข้อมูล

(3) บรรจุข้อมูลลงไปโดยระบุ

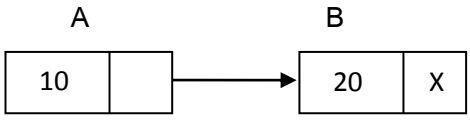
B.INFOR = 20



รูปที่ 5.8 แสดงโนด B เมื่อบรรจุข้อมูล

(4) เชื่อมโยงข้อมูล A->B โดยระบุ

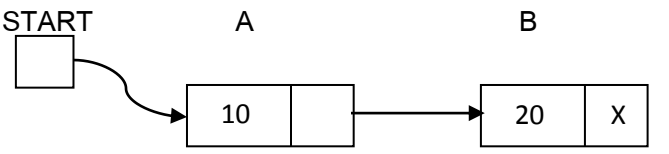
A.LINK = B



รูปที่ 5.9 แสดงโนด A เชื่อมไปยังโนด B

(5) กำหนด START ชี้ไปยังโนดแรก โดยระบุ

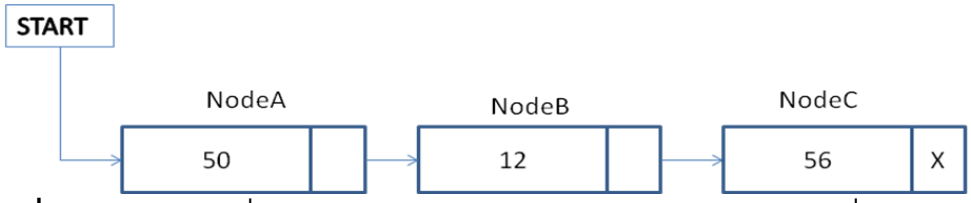
START = A



รูปที่ 5.10 แสดง START เชื่อมไปยังโนด A

<u>ตัวอย่างที่ 5.1</u> จงวาดภาพและแสดงการสร้างโนด จาก NodeA, NodeB และ NodeC นำข้อมูลเข้า 50, 12 และ 56 ตามลำดับ จากนั้นดำเนินการเชื่อมโยงข้อมูลโดย กำหนดให้ START ชี้ไปยัง NodeA เริ่มต้น จากนั้น NodeA-> NodeB->NodeC

เฉลย: สามารถวาดภาพลิงค์ลิต์ดังกล่าวได้ดังรูปที่ 5.11



รูปที่ 5.11 แสดงการเชื่อมโยงกันของโนดแบบมีลิงค์ทางเดียว โดยกำหนดชื่อของโนดอย่างง่าย

การสร้างโนดและเชื่อมโยงด้วยการใช้ภาษาจาวาในการดำเนินการสร้างโนดและเชื่อมโยงข้อมูลแต่ละโนด แสดงดังนี้

```
Node START;
 Node NodeA = new Node();
 NodeA.INFOR = 50;
 Node NodeB = new Node();
 NodeB.INFOR = 12;
 Node NodeC = new Node();
 NodeC.INFOR = 56;
 START.LINK = NodeA;
```

```
NodeA.LINK = NodeB;
NodeB.LINK=NodeC
```

แนวคิดเกี่ยวกับประเภทของลิงค์ลิสต์ โดยทั่วไปแบ่งได้ 2 ประเภทกว้างๆ คือลิงค์ลิสต์แบบทางเดียว และลิงค์ลิสต์แบบสองทาง โดยที่

1) ลิงค์ลิสต์แบบทางเดียว ประกอบด้วยข้อมูลแต่ละโนดคือ ข้อมูล (INFOR) กับ พอยเตอร์(LINK) เพื่อใช้เชื่อมโยงไปด้านหน้า และจะมีพอยเตอร์ระบุข้อมูลรายการแรกสุด คือ START ดังที่ได้อธิบายไปแล้ว

2) ลิงค์ลิสต์แบบสองทาง ประกอบด้วยข้อมูลแต่ละโนด (INFOR) พอยเตอร์เชื่อมโยงไปยังข้อมูลรายการต่อไป (NEXT LINK) และพอยเตอร์เชื่อมโยงย้อนกลับไปยังข้อมูลรายการก่อนหน้า (BACK LINK) รายละเอียดและแนวคิดของลิงค์ลิสต์ประเภทนี้จะได้นำเสนอในส่วนท้ายของบทนี้

นอกจากลิงค์ลิสต์ทั้งสองประเภทแล้ว ยังมีการประยุกต์ลิงค์ลิสต์ให้เป็นลิงค์ลิสต์ที่โนดนำและเป็นแบบลิงค์ลิสต์แบบแบบวงกลมอีกด้วย รายละเอียดจะแสดงรายละเอียดในลำดับต่อไป

ตัวดำเนินการของลิงค์ลิสต์ได้แก่ การท่องลิงค์ลิสต์ (Traversing Linked Lists) การแทรกข้อมูล (Inserting Data into Linked Lists) การค้นหาข้อมูลในลิงค์ลิสต์ (Searching Data in Linked Lists) และการลบข้อมูลในลิงค์ลิสต์ (Deleting Data from Linked Lists) ซึ่งจะได้นำเสนอในหัวข้อถัดๆ ไป

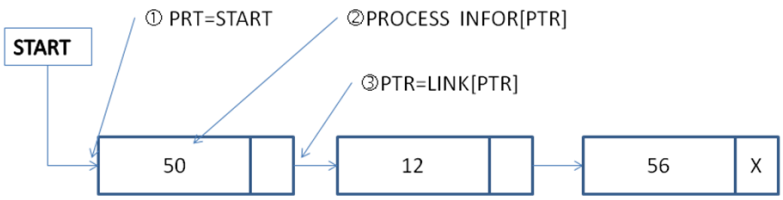
5.2 การท่องลิงค์ลิสต์แบบทางเดียว

5.2.1 หลักการเบื้องต้น

เนื่องจากโครงสร้างลิงค์ลิสต์เป็นข้อมูลที่อยู่กันอย่างต่อเนื่องเชื่อมโยงจากรายแรกไปยังรายการสุดท้ายโดยอาศัยพอยเตอร์ LINK ในการเชื่อมโยง โดยแนวคิดการท่องโครงสร้างจะต้องเริ่มเข้าที่โนดแรกสุด จากตำแหน่งที่ START ชี้อยู่ แล้วดำเนินการกับข้อมูลที่ตำแหน่งนั้น จากนั้นก็ตามพอยเตอร์ LINK ต่อเนื่องกันไป ดังนั้นเพื่อความเข้าใจในการใช้ตัวแปรสำหรับเชื่อมโยง จึงจำเป็นต้องกำหนดนิยามตัวแปรเพิ่มเติมขึ้นมาอีกหนึ่งตัวแปร ดังนิยามที่ 5.2

นิยามที่ 5.2 กำหนดให้
PTR คือ พอยเตอร์ที่ใช้ชี้ไปยังโนดของลิงค์ลิสต์ใดๆ

จากแนวคิดเบื้องต้น และนิยาม 5.2 นำมาแสดงเป็นภาพเบื้องต้น เพื่อใช้ท่องลิงค์ลิสต์ ดังรูปที่ 5.12 โดยดำเนินการนำเอา PTR เชื่อมไปยัง START จากนั้น ดำเนินการกับ ข้อมูลที่ PTR ชี้อยู่ จากนั้น ตามพอยเตอร์ LINK ของ PTR ไปทีละรายการ จนกระทั่ง PTR ชี้ที่ค่า NULL จึงจบการท่องลิงค์ลิสต์



ดำเนินการ ② ③ จนกระทั่ง **LINK[PRT]=NULL**

รูปที่ 5.12 แสดงแนวคิดของการท่องลิงค์ลิสตร์แบบมีลิงค์ทางเดียว

แสดงแนวคิดและอัลกอริทึมดังนี้

แนวคิดการท่องลิงค์ลิสต์แบบทางเดียว โดยต้องมีข้อมูล LIST เป็นลิงค์ลิสต์, PTR, START เพื่ออดำเนินการท่องโนดใน LIST ทั้งหมด	**Algorithm 5.1** Traversing one way LinkedLists **Input** : LIST, PTR, START **Output** : All nodes in LIST are traversed

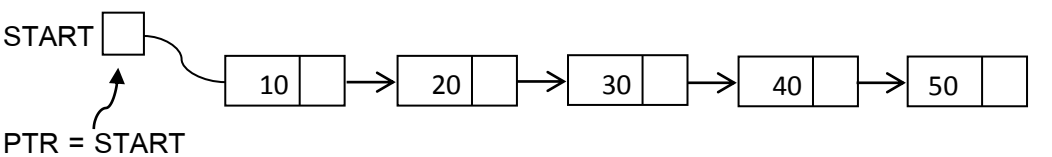
1. นำพอยเตอร์ไปชี้ยัง START เพื่อกำหนดโนดแรกที่จะท่อง 2. ดำเนินการกับข้อมูล INFOR ของโนดที่ PRT ชี้อยู่ INFOR [PTR] 3. ขยับ PTR โดยกำหนด PTR = LINK.PTR เป็นการขยับพอยเตอร์ชี้ไปยังโนดถัดไป 4. หาก PTR ≠ NULL คือจบลิงค์ - วนดำเนินการตามข้อ 2 หากจบลิงค์ จบการทำงาน	1 PTR=START 2 While PTR!=NULL Do 3 Process INFOR[PTR] 4 PTR=LINK[PTR] 5 End While 6 Return

<u>ตัวอย่างที่ 5.2</u> กำหนดลิงค์ลิสต์ ดังนี้ 10, 20, 30, 40, 50 จงอธิบายการท่องตามที่อัลกอริทึม 5.1

<u>แนวทางอธิบาย</u>

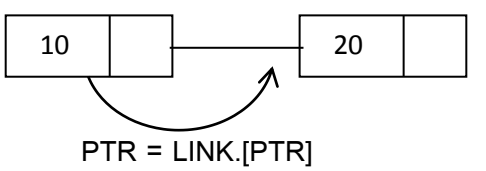
จากโจทย์ สามารถวาดเป็นภาพของลิงค์ลิสต์ได้ดังนี้

เบื้องต้นต้องกำหนดตัวแปรดังนี้



รูปที่ 5.13 แสดงภาพลิงค์ลิสต์เริ่มต้นของตัวอย่างที่ 5.2

1. PROCECSS INFOR.PTR ประมวลผลข้อมูลที่ PTR ชี้อยู่
   ในตัวอย่างนี้คือ ประมวลผล 10 นั่นเอง
2. PTR = LINK.PTR เป็นการเลื่อนตามพอยเตอร์ถัดไปที่โนดแรกชี้อยู่



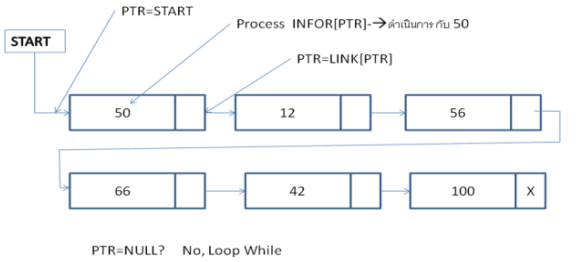
รูปที่ 5.13 แสดงการเลื่อนพอยเตอร์ ไปยังโนดถัดไป

3. ตรวจสอบว่า PTR ≠ NULL ดำเนินการต่อไป

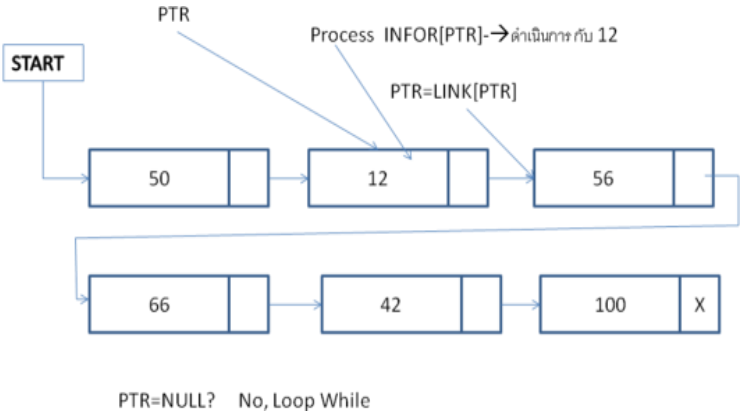
วนไปขั้นตอนที่ 1 ใหม่

หาก PTR = NULL จบการทำงาน

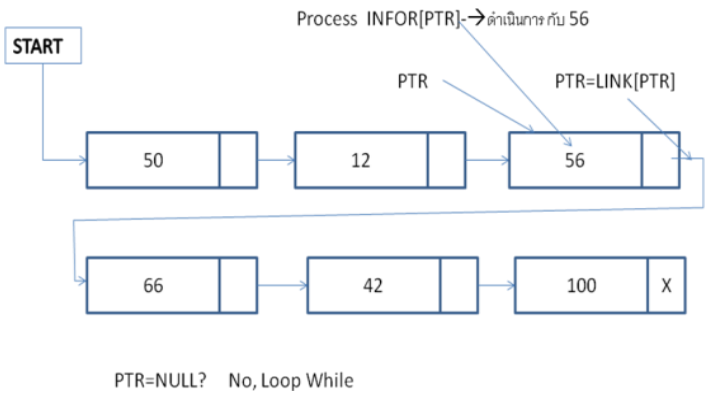
<u>ตัวอย่างที่ 5.3</u> จงแสดงการท่องจากลิงค์ลิสต์ โดยแสดงประกอบภาพการทำงานของอัลกอริทึม มีข้อมูล 50, 12, 56, 66, 42, 100



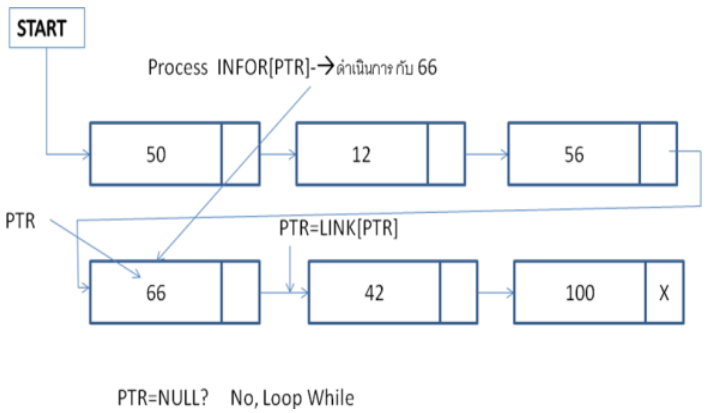
รูปที่ 5.14(a) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดแรก



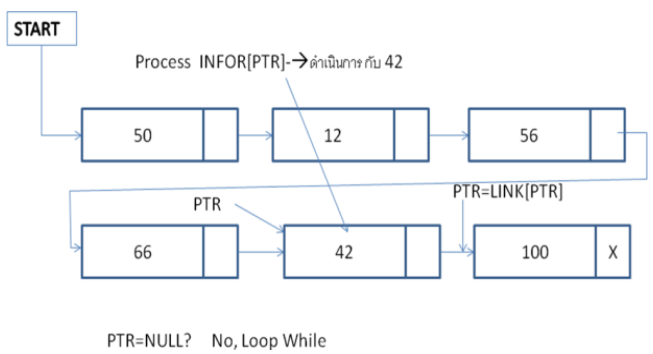
รูปที่ 5.14(b) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดที่ 2



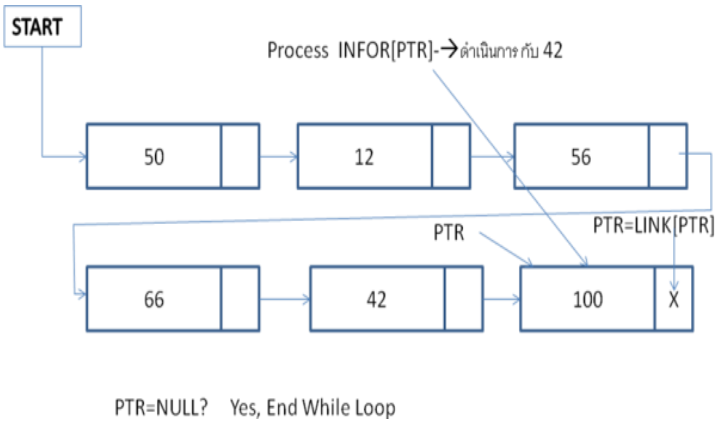
รูปที่ 5.14(c) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดที่ 3



รูปที่ 5.14(d) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดที่ 4



รูปที่ 5.14(e) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดที่ 5



รูปที่ 5.14(f) แสดงภาพการเริ่มท่องลิงค์ลิสต์ตามอัลกอริทึม โนดที่ 6

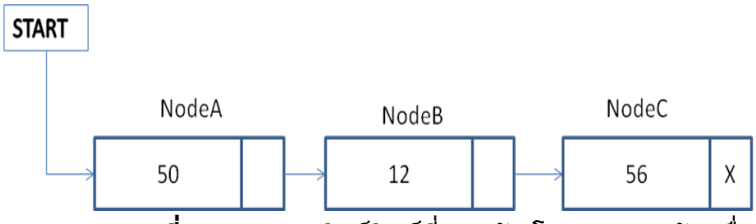
5.2.2 แนวทางการเขียนโปรแกรมสำหรับลิงค์ลิสต์ทางเดียว

การเขียนโปรแกรม สามารถอธิบายเป็นขั้นตอนเบื้องต้นได้ดังนี้

1. สร้างโครงสร้างโนด

```
1   public class Node {
2   Object INFOR;
3   Node LINK=null;
4   }
```

2) สร้างส่วนโนดเก็บข้อมูล ลิงค์ลิสต์ มีพอยเตอร์ PTR , START กำหนดให้ ตามตัวอย่าง



รูปที่ 5.15 แสดงลิงค์ลิสต์ที่ถูกสร้างโดยกำหนดป้ายชื่อ

```
1          Node START=null;
2          Node NodeA = new Node();
3          NodeA.INFOR = 50;
4          Node NodeB = new Node();
5          NodeB.INFOR = 12;
6          Node NodeC = new Node();
7          NodeC.INFOR = 56;
8          START.LINK = NodeA;
9          NodeA.LINK = NodeB;
10         NodeB.LINK=NodeC
11         Node PTR=null;
12         PTR=START;
```

3) เขียนส่วนการท่องตามอัลกอริทึม โดยแสดงดังตัวอย่างนี้

```
1       while(PTR!=null){
2          System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
3          PTR=PTR.LINK;
4       }
```

ตัวอย่างที่ 5.4 จงเขียนโปรแกรมส่วนการสร้างโนดและเชื่อมข้อมูล 50, 12, 56 บรรจุลงในลิงค์ลิสต์ แล้วแสดงการท่องลิงค์ลิสต์ดังกล่าวโดยใช้อัลกอริทึม 5.1

เฉลย: แสดงการสร้างข้อมูลโนด จากคลาส Node และโปรแกรมหลักดังนี้

```
1  public class Algor51TraversingList {
2        public static void main(String[] args) {
3           Node START=null;
4           Node NodeA = new Node();
5           NodeA.INFOR = 50;
6           Node NodeB = new Node();
7           NodeB.INFOR = 12;
8           Node NodeC = new Node();
9           NodeC.INFOR = 56;
10          START.LINK = NodeA;
```

```
11            NodeA.LINK = NodeB;
12            NodeB.LINK=NodeC
13            Node PTR=null;
14            PTR=START;
15          while(PTR!=null){
16             System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
17             PTR=PTR.LINK;
18          }
19      }
20  }
```

ผลการรันโปรแกรม โปรแกรมจะสร้างลิงค์ลิสต์บรรจุข้อมูล 50, 12, 56 แล้วแสดงข้อมูลดังกล่าวออกมาด้วยการทำงานตามอัลกอริทึม 5.1

5.2 การค้นหาข้อมูลในลิงค์ลิสต์ทางเดียว

การค้นหาข้อมูลในลิงค์ลิสต์อาจแบ่งออกเป็นสองลักษณะ คือ การค้นหาลิงค์ลิสต์แบบไม่มีการจัดเรียงข้อมูลในลิงค์ลิสต์ และค้นหาจากลิงค์ลิสต์ที่มีการจัดเรียงแล้ว

5.2.1 การค้นหาลิงค์ลิสต์ที่ยังไม่ได้จัดเรียง

การค้นหาข้อมูลในลิงค์ลิสต์กรณีที่ยังไม่มีการจัดเรียงข้อมูล เริ่มจาก กำหนดค่าเริ่มต้นที่จะค้นหาเปรียบเทียบกับข้อมูลที่ต้องการ นำไปเทียบกับโนดแรก โดย เทียบข้อมูลว่า เท่ากับข้อมูลในโนดหรือไม่ หากเท่ากัน ให้รายงานตำแหน่งโนดนั้น แล้วออกจากการค้นหา แต่หากไม่เท่ากัน ให้เลื่อนพอยเตอร์ ไปยังโนดถัดไป ดำเนินการเทียบกับข้อมูลในตำแหน่งพอยเตอร์ชี้อยู่จนกระทั่งพบหรือพอยเตอร์ที่ชี้ต่อไปนั้น เป็นค่า NULL ซึ่งเป็นข้อมูลตัวสุดท้ายและหากเป็นกรณีนี้แสดงว่าข้อมูลที่ต้องการค้นหานั้นไม่มีในลิงค์ลิสต์ที่ทำการค้นอยู่ เพื่อความเข้าใจสำหรับการใช้งานตัวแปรและอธิบายอัลกอริทึม ได้กำหนดตัวแปรเพิ่มขึ้นในนิยามที่ 5.3

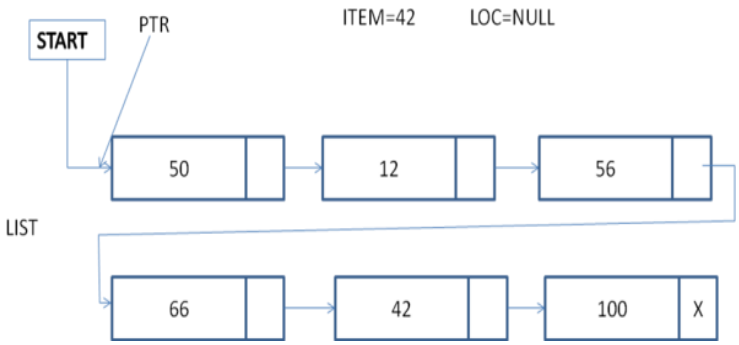
นิยามที่ 5.3 กำหนดให้
LOC คือ พอยเตอร์สำหรับใช้ชี้ไปยังข้อมูลที่ได้จากการค้นหา มีค่าเริ่มต้นคือ NULL
ITEM คือข้อมูลใดๆ ที่ต้องการค้นหา โดยเป็นชนิดเดียวกับข้อมูลที่เก็บอยู่ในลิงค์ลิสต์

แสดงแนวคิดการค้นหาและอัลกอริทึม ดังนี้

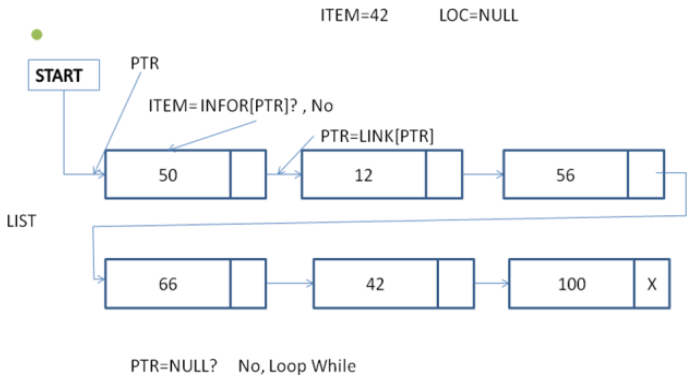
ค้นหาข้อมูล ITEM ในลิงค์ลิสต์ ที่ยังไม่มีการจัดเรียงข้อมูล โดยจะต้องมี LIST, PTR, START, LOC, ITEM เป็นข้อมูลเข้า LOC เป็นข้อมูลออก	**Algorithm 5.2** SearchingNoOrderedItem in Link Lists **Input** : LIST, PTR, START, LOC, ITEM **Output** : LOC
1 กำหนด ITEM คือข้อมูลที่ต้องการค้นหา 2 LOC คือตำแหน่งที่พบ กำหนดค่าเริ่มต้นเป็น NULL 3 กำหนด PTR=START 4 วนดำเนินการ ในขณะที่ PTR!=NULL 5 เทียบ ITEM = INFOR[PTR] หรือไม่ 5.1 หากเท่ากัน LOC=PTR 5.2 หากไม่เท่ากัน PTR=LINK[PTR] 5.3 วนกลับไปขั้นตอน 4	1 PTR=START 2 While PTR!=NULL Do 3 IF ITEM= INFOR[PTR] Then 4 LOC = PTR 5 Return LOC 6 Else 7 PTR=LINK[PTR] 8 End IF 9 LOC=NULL 10 End While 11 Return LOC

<u>ตัวอย่าง 5.5</u> กำหนดให้ข้อมูลในลิงค์ลิสต์ คือ 50, 12, 56, 66, 42, 100 จงแสดงตัวอย่างแสดงการค้นหา ITEM=42 ตามแนวทางของอัลกอริทึมด้วยรูปภาพ

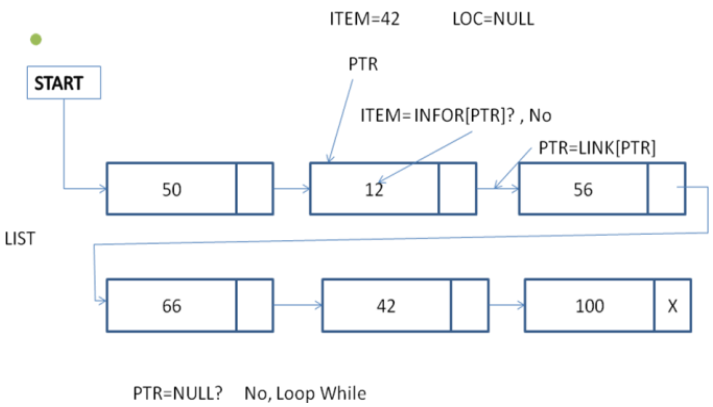
เฉลย: เบื้องต้นต้องกำหนดให้มีการสร้างลิงค์ลิสต์ให้เรียบร้อยก่อน แล้วจึงแสดงการค้นหาดังรูปที่ 5.16



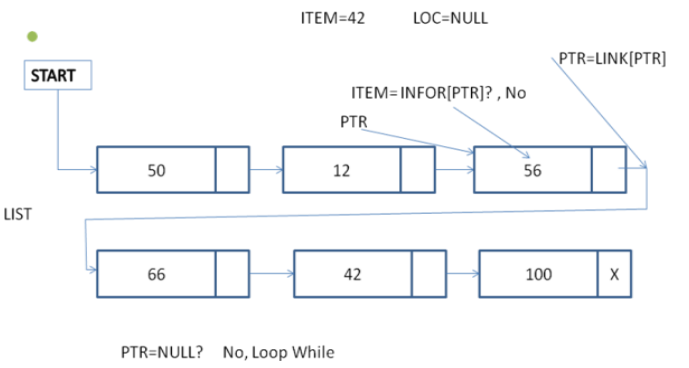
รูปที่ 5.16(a) แสดงการเริ่มต้นการค้นหา



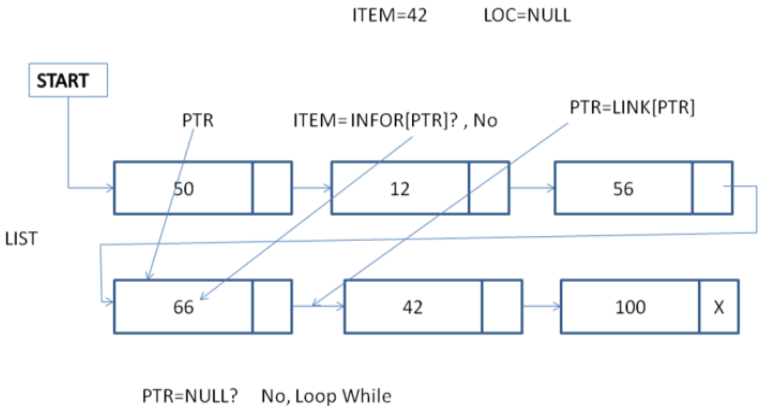
รูปที่ 5.16(b) แสดงการค้นหาโนดที่ 1



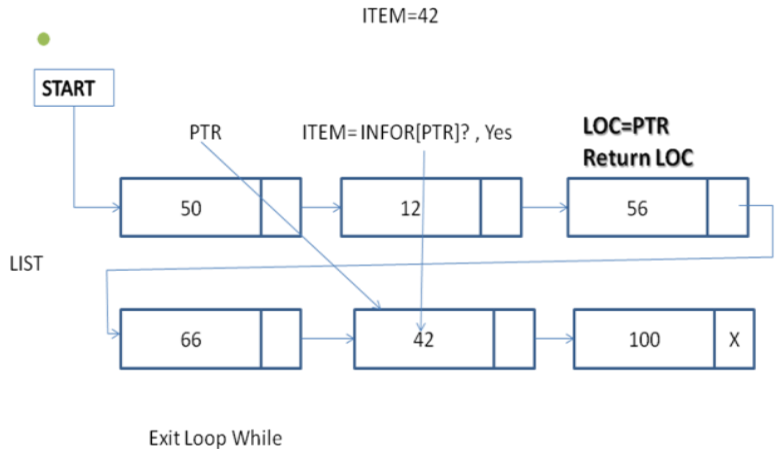
รูปที่ 5.16(c) แสดงการค้นหาโนดที่ 2



รูปที่ 5.16(d) แสดงการค้นหาโนดที่ 3

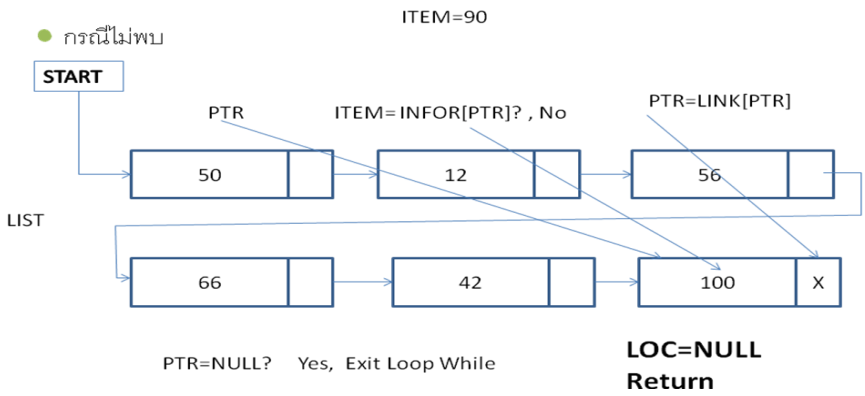


รูปที่ 5.16(e) แสดงการค้นหาโนดที่ 4



รูปที่ 5.16(f) แสดงการค้นหาโนดที่ 5

จากรูปที่แสดงการค้นหาข้อมูล ITEM ดังกล่าว สามารถนำมาอธิบายในกรณีค้นแล้วไม่พบข้อมูลในลิงลิสต์ การค้นจำเป็นต้องค้นต่อไปจนกระทั่งถึงข้อมูลรายการสุดท้าย ซึ่งจะทำให้ได้ค่า LOC = NULL แสดงว่าการค้นหาข้อมูลไม่พบ แสดงดังรูปที่ 5.17



รูปที่ 5.17 แสดงภาพการค้นหาไม่พบ

ตัวอย่างที่ 5.6 กำหนดให้ข้อมูลเก็บในลิงค์ลิสต์ คือ 20, 30, 40, 50, 60, 70, 80, 90, 100 จงเขียนโปรแกรมด้วยภาษาจาวาเพื่อค้นหาข้อมูล ITEM=80 จากลิงค์ลิสต์ดังกล่าว

```
1   package linklistimplementation;
```

```
2   public class Algor52SEARCH {
3       public static void main(String[] args) {
4       Node START;
5       Node NodeA = new Node(); NodeA.INFOR = 20;
6       Node NodeB = new Node(); NodeB.INFOR = 30;
7       Node NodeC = new Node(); NodeC.INFOR = 40;
8       Node NodeD = new Node(); NodeD.INFOR = 50;
9       Node NodeE = new Node(); NodeE.INFOR = 60;
10      Node NodeF = new Node(); NodeF.INFOR = 70;
11      Node NodeG = new Node(); NodeG.INFOR = 80;
12      Node NodeH = new Node(); NodeH.INFOR = 90;
13      Node NodeI = new Node(); NodeI.INFOR = 100;
14      START = NodeA; NodeA.LINK = NodeB; NodeB.LINK = NodeC;
15      NodeC.LINK = NodeD; NodeD.LINK = NodeE; NodeE.LINK = NodeF;
16      NodeF.LINK = NodeG; NodeG.LINK = NodeH; NodeH.LINK = NodeI;
17      //Algorrithm 5.2 SEARCH
18      int ITEM = 80;
19      Node LOC=null;
20      Node PTR=null;
21      PTR=START;
22      while(PTR!=null){
23        if(ITEM==(int)PTR.INFOR) {
24            LOC = PTR;
25            break;
26        }else
27          PTR=PTR.LINK;
28      }
29      //Show Result;
30      if(LOC ==null)
31          System.out.println("Not found.");
32      else
33          System.out.println("Found.");
```

```
34      }
35  }
```

ผลการรันจะแสดงว่า Found จากการค้นหาในลิสต์ดังกล่าว

5.2.2 การค้นหาลิงค์ลิสต์ที่มีการจัดเรียง

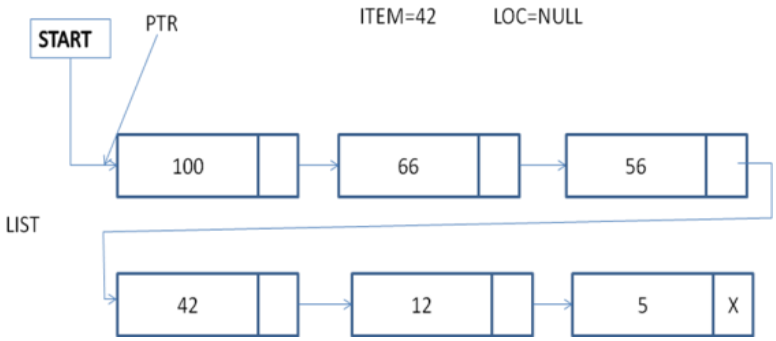
การค้นหาเบื้องต้นใช้หลักการท่องเข้าไปในลิงค์ลิสต์ แล้วเปรียบเทียบเช่นเดียวกันกับแบบไม่มีการจัดเรียง เมื่อพบก็รายงานตำแหน่งที่พบ แล้วจบการค้นหา แต่เมื่อยังไม่พบจะต้องค้นไป จนกระทั่ง พบตัวที่มากกว่า หรือน้อยกว่า ขึ้นอยู่กับว่าการจัดเรียงนั้นเรียงข้อมูลแบบใด หากเรียงแบบน้อยไปมาก เมื่อค้นไปพบข้อมูลที่มากกว่าก็หยุดได้ หรือหากเรียงจากมากไปน้อย เมื่อค้นพบข้อมูลที่น้อยกว่าก็หยุดได้และรายงานว่าไม่พบ อย่างไรก็ตามหากไม่เข้าเงื่อนไขดังกล่าว จะต้องค้นไปจนกระทั่งตัวสุดท้าย แล้วรายงานว่าไม่พบข้อมูล แสดงว่าข้อมูลที่ต้องการค้นนั้นมากกว่าข้อมูลในลิงค์ลิสต์หรือน้อยกว่าในลิงค์ลิสต์

แนวคิดการค้นหา และอัลกอริทึมแสดงดังนี้

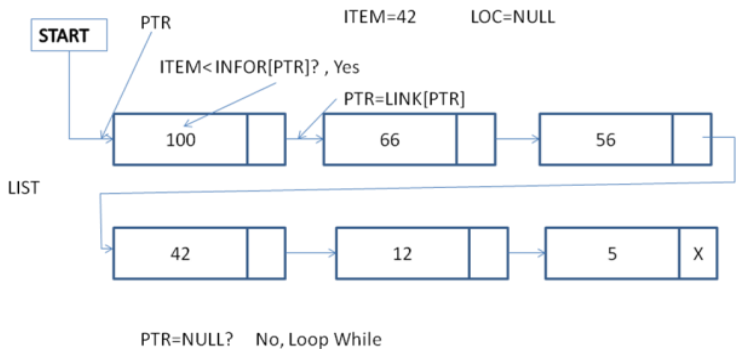
ค้นหาข้อมูล ITEM ในลิงค์ลิสต์ ที่จัดเรียงข้อมูลไว้แล้ว โดยจะต้องมี LIST, PTR, START, LOC, ITEM เป็นข้อมูลเข้า LOC เป็นข้อมูลออก	**Algorithm 5.3** SEARCH Item in Sorted Link Lists **Input** : LIST, PTR, START, LOC, ITEM **Output** : LOC

1 กำหนด PTR=START 2 วนทำงานในขณะที่ PTR!= NULL 2.1 ถ้า ITEM<INFOR[PTR] เลื่อนตัวชี้ต่อไป PTR=LINK[PTR] 2.2 ถ้า ITEM=INFOR[PTR] กำหนด LOC=PTR และจบการค้น 2.3 แต่ถ้ายังไม่ใช่ ให้กำหนด LOC=NULL และจบการค้น เพราะว่า ตัวที่ค้นเกินค่าที่กำหนดไว้แล้ว (น้อยกว่าหรือมากกว่าค่าที่ต้องการค้นแล้ว) 3 หากออกจากลูป แล้วยังไม่เข้ากรณีใดๆ ที่ผ่านมา ให้กำหนด LOC=NULL แสดงว่าการค้นหาค่า ITEM นั้นยังน้อยกว่าหรือมากกว่าค่าใน LIST	1 PTR=START 2 While PTR!=NULL Do 3 IF ITEM < INFOR[PTR] Then 4 PTR=LINK[PTR] 5 Else 6 IF ITEM=INFOR[PTR] Then 7 LOC = PTR 8 Return LOC 9 Else 10 LOC=NULL 11 Return LOC 12 End IF 13 LOC=NULL 14 End While 15 Return LOC

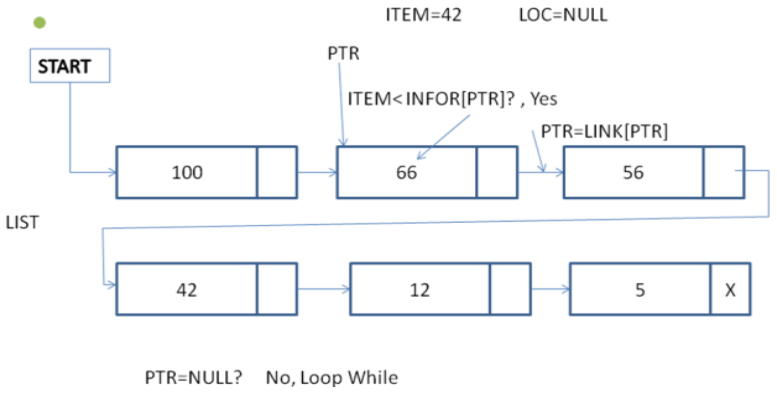
ตัวอย่าง 5.7 แสดงการค้นหาข้อมูลลิงค์ลิสต์ที่มีการจัดเรียงอยู่แล้ว คือ 100 66 56 42 12 5 ตามลำดับ



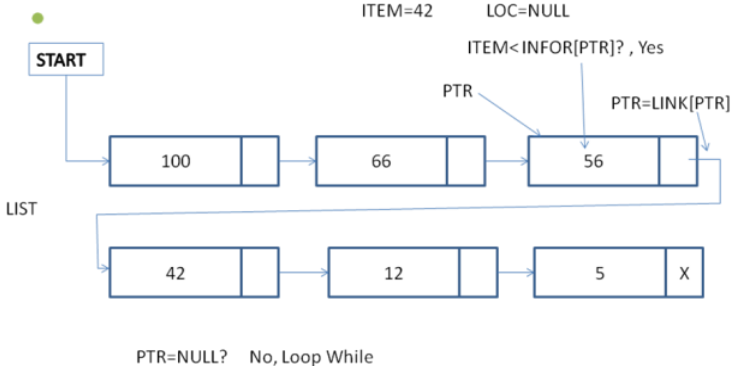
รูปที่ 5.18(a) แสดงการเริ่มต้นค้นหาแบบลิสต์ที่มีการจัดเรียง



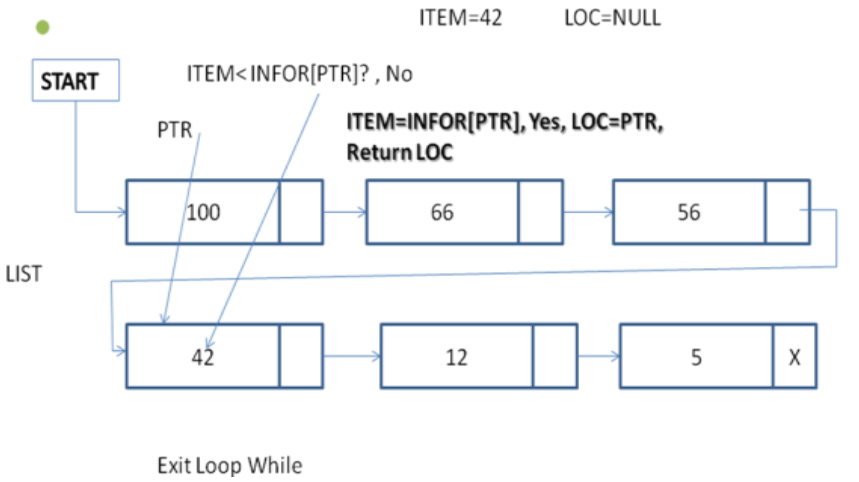
รูปที่ 5.18(b) แสดงการค้นหาแบบลิสต์ที่มีการจัดเรียง รายการแรก



รูปที่ 5.18(c) แสดงการค้นหาแบบลิสต์ที่มีการจัดเรียง รายการที่ 2

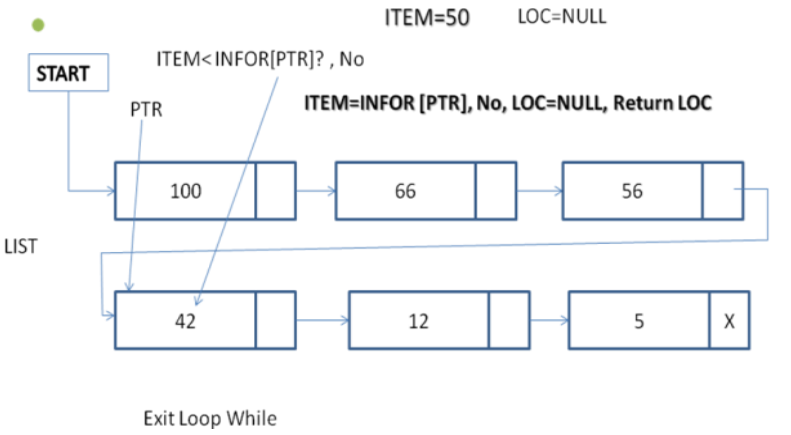


รูปที่ 5.18(d) แสดงการค้นหาแบบลิสต์ที่มีการจัดเรียง รายการที่ 3



รูปที่ 5.18(e) แสดงการค้นหาแบบลิสต์ที่มีการจัดเรียง รายการที่ 4 พบข้อมูล

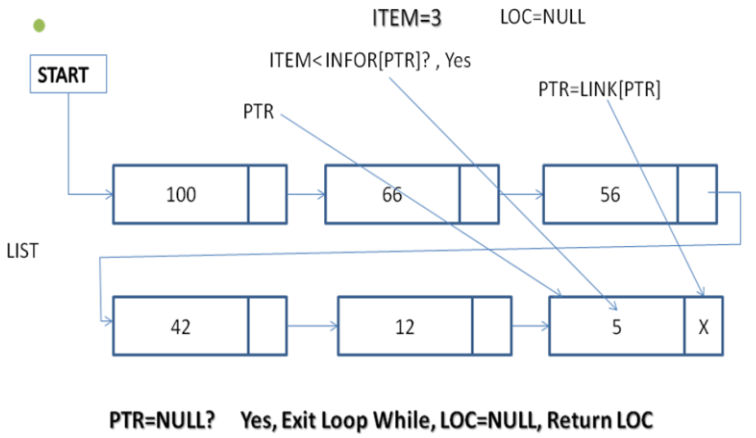
จากรูปที่ 5.18 หากค้นหาข้อมูลในกรณีไม่พบ จำเป็นต้องค้นไปจนกระทั่งพบข้อมูลที่น้อยกว่า หรือค้นไปจนกระทั่งถึงข้อมูลรายการสุดท้ายเช่นเดียวกับข้อมูลที่ยังไม่ได้จัดเรียง จากตัวอย่างดังกล่าว หากต้องการค้นหา ITEM=50 แสดงการหยุดค้นและรายงานผล ดังรูปที่ 5.19



รูปที่ 5.19 แสดงการค้นหา ITEM=50 หยุดเมื่อพบรายการน้อยกว่า

สำหรับการค้นหาข้อมูลที่ไม่มีลิงค์ลิสต์ดังกล่าว จะต้องค้นหาข้อมูลไปจนกระทั่งถึงข้อมูลรายการสุดท้าย เช่นเดียวกับข้อมูลที่ยังไม่ได้จัดเรียง แสดงตัวอย่างดังเช่น รูปที่ 5.20

เมื่อค้นข้อมูล ITEM=3 เป็นต้น



รูปที่ 5.20 แสดงตัวอย่างการค้นหาข้อมูลที่ไม่มีในลิสต์

<u>ตัวอย่างที่ 5.8</u> จงการเขียนโปรแกรมด้วยภาษาจาวา จัดเก็บข้อมูลที่จัดเรียงแล้ว จาก 90,80,70,60,50,40,30,20,10 พร้อมทั้งแสดงการค้นหา ITEM=40

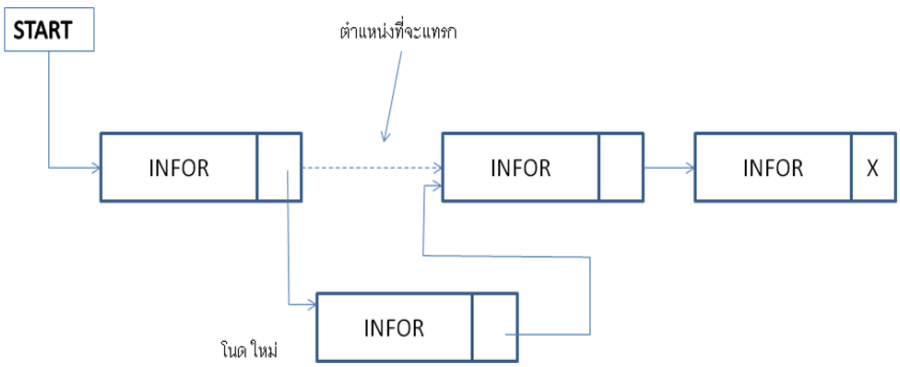
```
1    public class Algor53SRCHSL {
2       public static void main(String[] args) {
3                SEARCH(40);
4        }//end main    //Method of Algorithm 5.2
5       public static void SEARCH(int item) {
6           Node START;
7           Node NodeA = new Node(); NodeA.INFOR = 90;
8           Node NodeB = new Node(); NodeB.INFOR = 80;
9           Node NodeC = new Node(); NodeC.INFOR = 70;
10          Node NodeD = new Node(); NodeD.INFOR = 60;
11          Node NodeE = new Node(); NodeE.INFOR = 50;
12          Node NodeF = new Node(); NodeF.INFOR = 40;
13          Node NodeG = new Node(); NodeG.INFOR = 30;
14          Node NodeH = new Node(); NodeH.INFOR = 20;
15          Node NodeI = new Node(); NodeI.INFOR = 10;
16          START = NodeA; NodeA.LINK = NodeB; NodeB.LINK = NodeC;
```

```
17          NodeC.LINK = NodeD; NodeD.LINK = NodeE; NodeE.LINK = NodeF;
18          NodeF.LINK = NodeG; NodeG.LINK = NodeH; NodeH.LINK = NodeI;
19          //Algorrithm 5.2 SEARCH
20          int ITEM = item;
21          Node LOC=null;
22          Node PTR=null;
23          PTR=START;
24          while(PTR!=null){
25            if(ITEM<(int)PTR.INFOR) {
26                PTR=PTR.LINK;
27            }else
28              if(ITEM==(int)PTR.INFOR){
29                  LOC = PTR;
30                  break;
31              }else{
32                  LOC=null;
33                  break;
34              }
35          }//end while
36          //Show Result;
37          if(LOC ==null)
38              System.out.println("Not found.");
39          else
40              System.out.println("Found.");
41      }
42  }
```

ผลการรันจะแสดงว่าพบข้อมูลด้วยข้อความ Found

5.3 การแทรกข้อมูลลงในลิงค์ลิสต์ทางเดียว

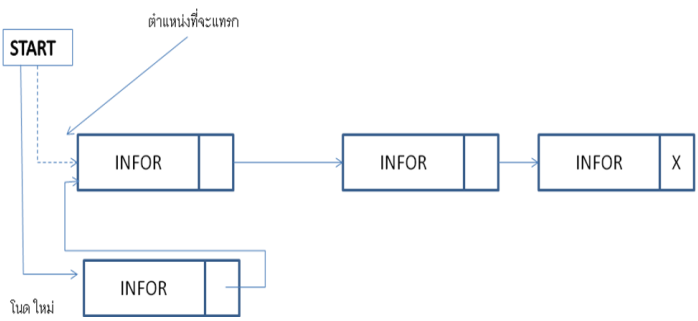
การแทรกหมายถึง การนำโนดใหม่เพิ่มเข้าไปในลิงค์ลิสต์เดิมอยู่แล้วหรือหากยังไม่มีจะต้องทำให้โนดใหม่เป็นโนดแรกสุดของข้อมูลในลิงค์ลิสตร์ การแทรกโนด เริ่มต้นจากสร้างโนดใหม่ แล้วระบุตำแหน่งที่ต้องการแทรก จากนั้นนำพอยเตอร์ของโนดใหม่ชี้ไปยังโนดที่อยู่หลังตำแหน่งหลังการแทรก ด้วยการพิจารณาจากพอยเตอร์โนดที่อยู่ก่อนหน้าตำแหน่งแทรก จากนั้นเปลี่ยนพอยเตอร์ที่เคยชี้ไปยังโนดถัดไปของตำแหน่งก่อนหน้าที่จะแทรกให้มาชี้ยังโนดใหม่ แนวคิดเบื้องต้นเป็นแสดงดังรูปที่ 5.21



รูปที่ 5.21 แสดงแนวคิดในการแทรก

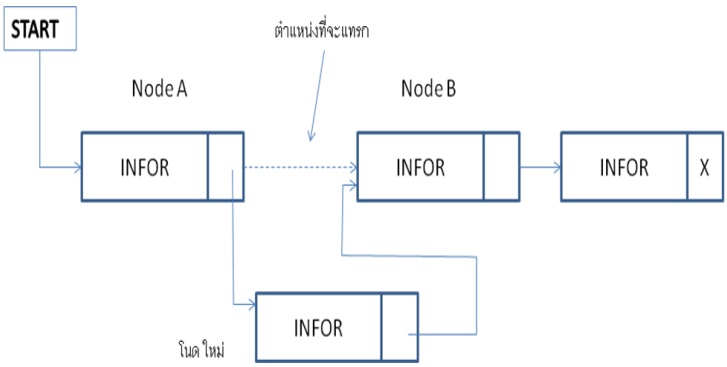
อัลกอริทึมสำหรับการสร้าง การแทรก สามารถอธิบายได้ 2 แบบ คือ แทรกที่ต้นลิสต์ และแทรก ณ ตำแหน่งที่ต้องการ

แทรกที่ต้นลิสต์ แสดงดังรูปที่ 5.22



รูปที่ 5.22 แสดงแนวคิดในการแทรกต้นลิสต์

แทรกยังตำแหน่งที่ที่ต้องการ แสดงดังรูปที่ 5.23



รูปที่ 5.23 แสดงแนวคิดในการแทรกตำแหน่งที่ต้องการในลิสต์

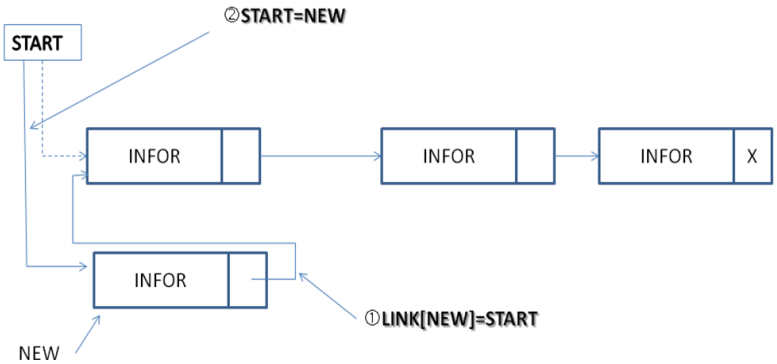
5.3.1 การแทรกข้อมูลลงในลิงค์ลิสต์แบบต้นลิสต์

การแทรกข้อมูลที่ต้นลิสต์ คือ การนำโนดเพิ่มเข้าไปเป็นรายการแรกของลิงค์ลิสต์ ขั้นตอนการดำเนินการไม่มีความซับซ้อน เพราะเพียงแค่สร้างในโนดใหม่ขึ้นมา แล้วเอาไปพอยเตอร์ไปชี้โนดที่ START เคยชี้อยู่ จากนั้นเปลี่ยนพอยเตอร์ของ START มาชี้ที่โนดใหม่ เพื่อให้เข้าใจตัวแปร ที่เกี่ยวข้องมากขึ้น จึงกำหนดนิยาม 5.4 เพื่ออธิบายอัลกอริทึมในลำดับถัดไป

นิยามที่ 5.4 กำหนดให้

NEW คือ โนดใดๆ ที่ถูกสร้างขึ้นมาเพื่อใช้แทรกเข้าไปในลิงค์ลิสต์

แสดงรูปตามแนวคิดของการแทรกโนดใหม่ NEW เข้าไปในต้นลิงค์ลิสต์ ดังรูปที่ 5.24



รูปที่ 5.24 แสดงแนวคิดเริ่มต้นในการแทรก

จากรูปอธิบายได้ว่า เบื้องต้นต้องสร้าง NEW ขึ้นมาและกำหนดข้อมูลที่ใส่ในส่วน INFOR ก่อน หลังจากนี้น นำเอาพอยเตอร์ของโนด NEW ชี้ไปยัง START แล้วเปลี่ยนพอยเตอร์ของ START มาชี้ที่ NEW ตามลำดับ แนวทางการแทรกและอัลกอริทึม ดังนี้

แนวคิดการเพิ่มโนดใหม่ไปที่ต้นลิงค์ลิสต์ โดยต้องตัวแปร LIST, START, NEW, ITEM โดยที่ ITEM คือข้อมูลส่วนของ INFOR ของ NEW ที่จะถูกเพิ่มเข้าไปที่ต้นลิงค์ลิสต์	**Algorithm 5.4** InsertFirst **Input** : LIST, START, NEW, ITEM **Output** : ITEM is inserted into LIST at the first node.
1 NEW คือโนดใหม่ 2 กำหนด INFOR[NEW]=ITEM 3 กำหนด LINK[NEW] = START 4 กำหนด START=NEW	1 INFOR[NEW]=ITEM 2 LINK[NEW]=START 3 START=NEW 4 Return

ตัวอย่าง 5.9 จงเขียนโปรแกรมด้วยภาษาจาวา เพื่อแสดงการเพิ่มข้อมูล 10 เป็นโนดแรก หลังจากนั้น แสดงผลข้อมูลในลิงค์ลิสต์ แล้วเพิ่ม 20, 30, 40 ไปที่ต้นลิสต์ แล้วแสดงผลอีกครั้ง

เฉลย: ออกแบบโปรแกรมโดย สร้างเมธอด

- main() เพื่อใช้รันโปรแกรมหลัก โดยสร้างโนดแรกในส่วนนี้
- สร้างเมธอด INSTFIRST(int item) เพื่อทำงานตามอัลกอริทึม 5.4

- สร้างเมธอด showDatainList() สำหรับการแสดงผล

โปรแกรมทั้งหมดแสดงดังนี้

```
1   public class Algor54INSFIRST {
2   static Node START;
3   public static void main(String[] args) {
4      Node first = new Node();
5      first.INFOR = 10;
6      START = first;
7      System.out.println("Data Befor INSERT:");
8      showDatainList();
9      INSTFIRST(20);
10     INSTFIRST(30);
11     INSTFIRST(40);
12     System.out.println("Data After INSERT:");
13     showDatainList();
14  }
15  public static void INSTFIRST(int item){
16     Node NEW=new Node();
17     int ITEM=item;
18     NEW.INFOR = item;
19     NEW.LINK = START;
20     START=NEW;
21  }
22  public static void showDatainList(){
23     Node PTR=null;
24     PTR=START;
25     while(PTR!=null){
26     System.out.println(PTR.INFOR);
27     PTR=PTR.LINK;
28    } //end while
29  } //end method showData
```

```
30  } //end clsss
```

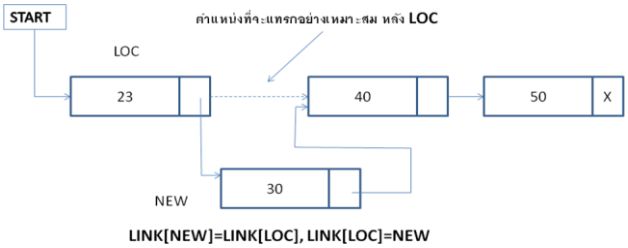
ผลการรันโปรแกรม จะแสดงข้อมูล ก่อนและหลังเพิ่มข้อมูล โดยโนดแรกจะถูกเลื่อนไปเป็นโนดสุดท้ายเพราะมีการแทรกต้นลิสต์

5.3.2 การแทรกข้อมูลลงในลิสต์ ตามตำแหน่งที่ต้องการ

การแทรกโนดที่จะต้องการ โดยการกำหนดตำแหน่งที่จะแทรกด้วยการค้นหาหรือระบุตำแหน่งที่จะแทรก คือ LOC ทั้งนี้ LOC เกิดจากการค้นหา หรือระบุโดยตรงก็ได้ แนวคิดของแทรกก็คือ ต้องสร้างโนดใหม่ขึ้นมาก่อน จากนั้น พิจารณาตำแหน่ง LOC โดยตำแหน่งที่โนดใหม่จะต้องแทรกอยู่หลังโนดที่ LOC ชี้อยู่ จากนั้นเอาพอยเตอร์ของโนดใหม่ชี้ไปยังโนดที่ LOC ชี้อยู่ หลังจากนั้นให้เอาพอยเตอร์โนดที่ LOC ชี้อยู่มาชี้ที่โนดใหม่ ด้วยแนวคิดนี้สามารถแยกแนวคิดย่อยเป็นสองแนวทางคือ การแทรกโดยการค้นหาข้อมูลที่ยังไม่มีการจัดเรียงแล้วแทรก โดยระบุตำแหน่งโดยตรงหลังโนดที่ค้นพบ หรือแทรกในตำแหน่งที่เหมาะสม เมื่อข้อมูลมีการจัดเรียงเอาไว้แล้ว จะต้องค้นหาตำแหน่งที่เหมาะสมแล้วแทรกเข้าไประหว่างตำแหน่งด้านหน้าและด้านหลังโนดที่ค้นพบ

5.3.2.1 การแทรกในตำแหน่งที่ระบุโดยตรง

แนวทางการแทรกคือ ต้องกำหนดโนดที่ต้องการแทรกก่อน จากนั้นกำหนดให้เป็น LOC ชี้ตำแหน่งของโนดที่จะแทรกข้อมูลเข้าไปต่อท้าย แสดงแนวคิดเบื้องต้นเป็นรูปภาพได้ดังนี้



รูปที่ 5.25 แสดงแนวคิดเริ่มต้นในการแทรกตำแหน่งที่ต้องการ

อธิบายแนวคิดเป็นขั้นตอนและอัลกอริทึมแสดงได้ดังนี้

แนวคิด การแทรกโนดใหม่ ที่ ตำแหน่งหลังโนดที่ LOC ชี้อยู่ ต้องมีตัวแปร LIST, START, NEW, ITEM, LOC โดยที ITEM คือข้อมูลที่บรรจุในโนด NEW จะต้องถูกแทรกไปหลังโนดที่ LOC ชี้อยู่	**Algorithm 5.5** Insert Item at LOC in a Link Lists **Input** : LIST, START, NEW, ITEM, LOC **Output** : ITEM is inserted into LIST after LOC node.
1 ตรวจสอบว่า LOC=NULL หรือไม่ หากเป็น NULL 2 แสดงว่า ให้แทรกต้นลิสต์ 2.1 กำหนด LINK[NEW]=START 2.2 START=NEW 3 หากไม่เป็น NULL 4 กำหนด LINK[NEW]=LINK[LOC] 5 LINK[LOC]=NEW	1 INFOR[NEW]=ITEM 2 IF LOC = NULL Then 3 LINK[NEW]=START 4 START=NEW 5 Else 6 LINK[NEW] = LINK[LOC] 7 LINK[LOC] = NEW 8 End IF 9 Return

การเขียนโปรแกรมด้วยภาษาจาวา

<u>ตัวอย่าง 5.10</u> จงเขียนโปรแกรมแสดงการแทรกข้อมูลในลิสต์โดยระบุตำแหน่ง โดยที่ระบุค่าข้อมูลที่จะแทรก คือ 65 หลัง 70 โดยข้อมูลในลิสต์มีข้อมูลดังนี้ 100, 90, 80, 70, 60, 50 โดยแสดงผลข้อมูลก่อนและหลังการแทรก

เฉลย: แนวคิดในการออกแบบโปรแกรม โดยสร้างเมธอดดังนี้

- main() ใช้สำหรับรันโปรแกรมเป็นหลัก และสร้างลิงค์ลิสต์เดิมให้มีอยู่ก่อนแล้วการแทรก
- INSTLOC(int loc, int item) ใช้เป็นเมธอดหลักในการแทรกข้อมูลตามอัลกอริทึมโดยการเรียกใช้เมธอด SEARCH(int ITEM) เพื่อหา LOC ที่จะแทรก

- SEARCH(int ITEM) ใช้เป็นเมธอดในการค้นหาตำแหน่งที่จะแทรก คือ LOC
- showDatainList() ใช้แสดงข้อมูลในลิสต์ทั้งหมด

โปแกรมทั้งหมดแสดงได้ดังนี้

```
1   public class Algor55INLOC {
2       static Node START, first = new Node();
3       public static void main(String[] args) {
4           first.INFOR = 100;
5           START = first;
6           Node A = new Node(); A.INFOR = 90; START.LINK = A;
7           Node B = new Node(); B.INFOR = 80; A.LINK = B;
8           Node C = new Node(); C.INFOR = 70; B.LINK = C;
9           Node D = new Node(); D.INFOR = 60; C.LINK = D;
10          Node E = new Node(); E.INFOR = 50; D.LINK =E;
11          System.out.println("Data Befor INSERT:");
12          showDatainList();
13          INSTLOC(70,65);
14          System.out.println("Data After INSERT:");
15          showDatainList();
16       }
17      public static void INSTLOC(int loc, int item){
18         Node LOC=SEARCH(loc); //ค้นหาข้อมูลที่จะแทรก
19         Node NEW=new Node();   //step. 2
20         int ITEM=item;
21         NEW.INFOR = item;
22         if(LOC==null) {
23             NEW.LINK = START;  //insert first Node
24             START=NEW;
25          }else
26          {
27             NEW.LINK = LOC.LINK;
28             LOC.LINK = NEW;
```

```
29            }
30      }
31      public static Node SEARCH(int ITEM){
32           Node LOC=null;
33           Node PTR=null;
34          PTR=START;
35          while(PTR!=null){
36               if(ITEM==(int)PTR.INFOR) {
37                   LOC = PTR;
38                   break;
39               }else
40                   PTR=PTR.LINK;
41       }
42  //Show Result;
43      if(LOC ==null)
44          System.out.println("Not found.");
45      else
46          System.out.println("Found.");
47      return LOC;
48      }
49     public static void showDatainList(){
50          Node PTR=null;
51          PTR=START;
52          while(PTR!=null){
53              System.out.println(PTR.INFOR);
54              PTR=PTR.LINK;
55          }
56      }
57  }
```

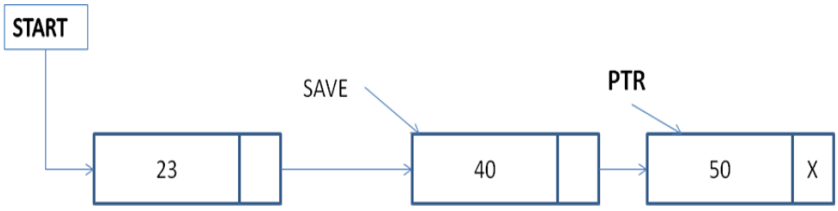
5.3.2.2 การแทรกโนดในลิสต์ที่มีการจัดเรียงแล้ว

แนวทางการแทรก ต้องค้นหาตำแหน่งที่จะแทรกให้พบก่อน LOC สำหรับการค้นหาจะต้องค้นเพื่อให้รู้ว่าตำแหน่งที่จะแทรกอยู่ระหว่างข้อมูลสองโนด ดังนั้นจำเป็นจะต้องใช้พอยเตอร์สองตัวดำเนินการค้นหาไปพร้อมๆ กัน เมื่อพอยเตอร์ตัวหน้าชี้ที่ตำแหน่งใด จะเก็บตำแหน่งที่พอยเตอร์ตัวหลังชี้อยู่เป็นค่า LOC เพื่อใช้งานสำหรับการแทรก ดังนั้นจึงจำเป็นต้องมีตัวแปรสำหรับพอยเตอร์เพิ่มขึ้นอีกหนึ่งตัว ตามนิยามที่ 5.5

นิยามที่ 5.5 กำหนดให้

SAVE คือ พอยเตอร์สำหรับชี้ไปยังโนดของลิงค์ลิสต์ใดๆ

ดังนั้น การค้นหาต้องพอยเตอร์สองตัว PTR และ SAVE ค้นหาในลิสต์ เมื่อค้นที่ตำแหน่งที่ PTR ชี้อยู่ แสดงว่าตำแหน่งที่จะแทรกจะอยู่ระหว่าง SAVE กับ PTR ให้กำหนด LOC=SAVE เพื่อคืนค่า LOC จากนั้นเรียกใช้งานอัลกอริทึมการแทรก ณ ตำแหน่งที่ต้องการ ให้ทำงานค้นหาตำแหน่งที่ต้องการ แสดงตัวอย่าง การค้นโดยใช้พอยเตอร์สองตัวดังรูปที่ 5.26 เมื่อ ITEM=45 ตำแหน่งที่เหมาะสมระหว่าง 40 กับ 50



รูปที่ 5.26 แสดงแนวคิดในการแทรกตำแหน่งที่ต้องการที่มีการจัดเรียง

แนวทางของ อัลกอริทึมมีหลักการสองขั้นตอนคือ

1) เรียกอัลกอริทึมสำหรับการค้นหากตำแหน่งที่มีการจัดเรียงเพื่อหาตำแหน่ง LOC ที่จะแทรก

2) เรียกใช้อัลกอริทึมการแทรก ณ ตำแหน่งที่ต้องการ

อัลกอริทึมการค้นหา LOC โดยใช้พอยเตอร์สองตัว แสดงดังนี้

Algorithm 5.6 Find LOC in a sorted Link Lists

Input : LIST, START, SAVE, ITEM, LOC

Output : LOC

```
1       IF START=NULL Then
2           LOC=NULL
3           Return LOC
4       End IF
5       IF ITEM< INFOR[START] Then
6           LOC=NULL
7           Return LOC
8       End IF
9         SAVE=START,  PTR=LINK[START]
10        While PTR!=NULL  Do
11          IF ITEM< INFOR[PTR] Then
12              LOC = SAVE;
13              Return LOC;
14          End IF
15           SAVE=PTR;
16            PTR=LINK[PTR]
17        End While
18        LOC=SAVE;
19      Return LOC
```

สำหรับอัลกอริทึมหลักอาศัยอัลกอริทึมทั้งสองคือเรียก

1. Call Algorithm 5.6 to find LOC
2. Call Algorithm 5.5 to nsert ITEM at LOC in Link Lists

แสดงตัวอย่างการแทรก ตามแนวทางดังกล่าว ด้วยการเขียนโปรแกรมด้วยภาษาจาวา ดังตัวอย่างที่ 5.11

ตัวอย่างที่ 5.11 จงเขียนโปรแกรมแสดงการแทรกข้อมูล ณ ตำแหน่ง LOC เมื่อข้อมูลในลิสต์ มีดังนี้ 50, 60, 70, 80, 90, 100 โดยแทรกข้อมูล 75 ลงในตำแหน่งที่เหมาะสม

เฉลย: แนวคิดในการออกแบบโปรแกรม โดยสร้างเมธอดดังนี้

- main() ใช้สำหรับรันโปรแกรมเป็นหลัก และสร้างลิงค์ลิสต์เดิมให้มีอยู่ก่อนแล้วการแทรก
- INSTLOC(int loc, int item) ใช้เป็นเมธอดหลักในการแทรกข้อมูลตามอัลกอริทึมโดยการเรียกใช้เมธอด FINDLOC(int ITEM) เพื่อหา LOC ที่จะแทรก
- FINDLOC (int ITEM) ใช้เป็นเมธอดในการค้นหาตำแหน่งที่จะแทรก คือ LOC
- showDatainList() ใช้แสดงข้อมูลในลิสต์ทั้งหมด

```
1   public class Algor56Insearch {
2           static Node START, first = new Node();
3           public static void main(String[] args) {
4           first.INFOR = 50;
5           START = first;
6           Node A = new Node(); A.INFOR = 60; START.LINK = A;
7           Node B = new Node(); B.INFOR = 70; A.LINK = B;
8           Node C = new Node(); C.INFOR = 80; B.LINK = C;
9           Node D = new Node(); D.INFOR = 90; C.LINK = D;
10          Node E = new Node(); E.INFOR = 100; D.LINK =E;
11          System.out.println("Data Befor INSERT:");
12          showDatainList();
13          Node LOC=FINDLOC(70); //ค้นหาข้อมูลที่จะแทรก        //call FINDLOC
14          INSLOC(LOC,75);         //Call INSLOC()...
15          System.out.println("Data After INSERT:");
16          showDatainList();
17          }
```

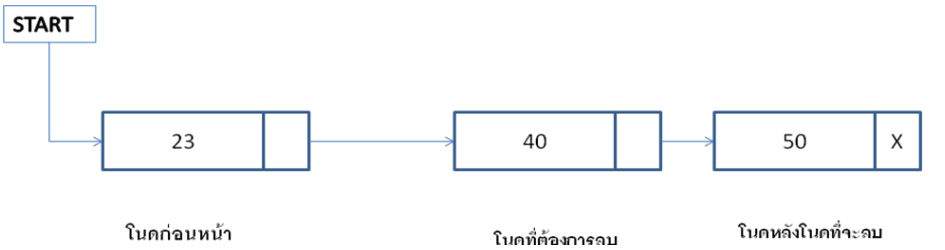
```
18      public static void INSLOC(Node LOC, int item){
19          //Algorrithm 5.5 SEARCH  if AVAIL = NULL step. 1
20          Node NEW=new Node();    //step. 2
21          int ITEM=item;
22          NEW.INFOR = item;
23          if(LOC==null) {
24          NEW.LINK = START;  //insert first Node
25          START=NEW;
26          }else
27          {
28            NEW.LINK = LOC.LINK;
29            LOC.LINK = NEW;
30          }
31      }
32      public static Node FINDLOC(int ITEM){
33          Node LOC=null;
34          Node PTR=null;
35          Node SAVE=null;
36          PTR=START;
37          if(START==null) { LOC=null; return LOC;}
38          if(ITEM<(int)PTR.INFOR) { LOC=null; return LOC; }
39          SAVE=START; PTR=START.LINK;
40          while(PTR!=null){
41            if(ITEM<(int)PTR.INFOR) {
42             LOC = SAVE;
43             return LOC;
44            }
45            SAVE=PTR;
46            PTR=PTR.LINK;
47          }
48          LOC=SAVE;
49          //Show Result;
```

```
50                 if(LOC ==null)
51                    System.out.println("Not found.");
52                 else
53                    System.out.println("Found.");
54       return LOC;
55     }
56    public static void showDatainList(){
57         Node PTR=null;
58         PTR=START;
59         while(PTR!=null){
60             System.out.println(PTR.INFOR);
61             PTR=PTR.LINK;
62         }
63     }
64  }
```

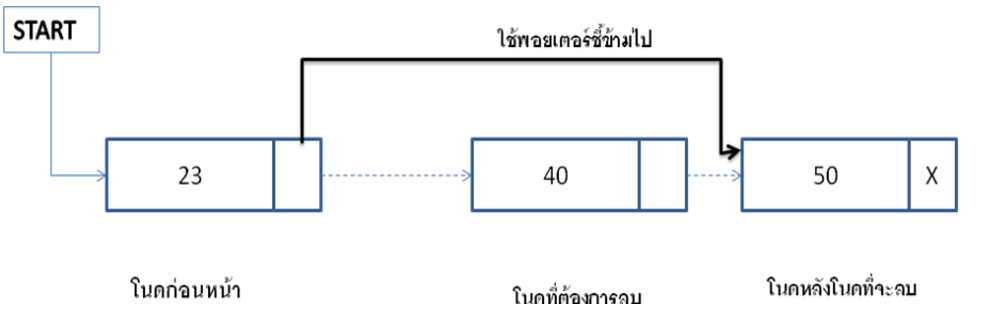
ผลการรันโปรแกรมจะแสดงข้อมูลในลิงค์ลิสต์ก่อนและหลังการแทรก 75 ในตำแหน่งระหว่าง 70 กับ 80

5.4 การลบข้อมูลออกจากลิงค์ลิสต์ทางเดียว

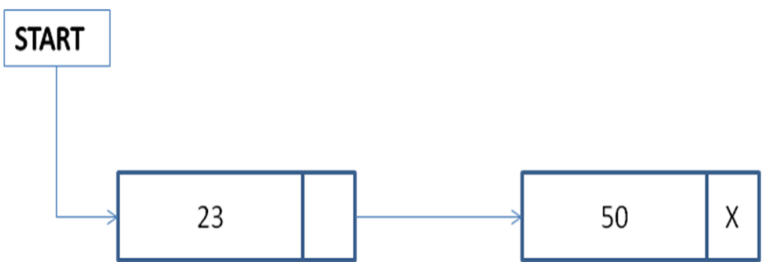
การลบข้อมูลในลิงค์ลิสต์ หมายถึง การลบโนดที่ต้องการออกจากลิงค์ลิสต์ สามารถทำได้โดยง่ายและรวดเร็ว ซึ่งถือว่าเป็นคุณสมบัติเด่นประการหนึ่งของลิงค์ลิสต์ แนวคิดในการลบจะต้องกำหนดโนดทีจะลบ โดยนำโนดก่อนโนดที่จะลบ และโนดหลังจากโนดที่ต้องการจะลบด้วย กระบวนการลบ ทำได้โดยนำพอยเตอร์โนดก่อนหน้าโนดที่จะลบ ชี้ข้ามไปยังตัวหลังโนดที่ลบ จะทำให้โนดที่ต้องการไม่สามารถเข้าถึงได้ เป็นการลบไปโดยปริยาย จากนั้นอาจคืนค่าหน่วยความจำให้กับระบบ เช่น ในภาษาซีพลัสพลัสอาจต้องคืนหน่วยความจำ แต่ในภาษาจาวาไม่ต้องคืนเพราะมีระบบจัดการหน่วยความจำอยู่แล้ว เป็นต้น แสดงแนวคิดการลบ ดังรูป 5.27



รูปที่ 5.27 (a) แสดงลิสต์ก่อนการลบ



รูปที่ 5.27 (b) แสดงลิสต์ขณะการลบ



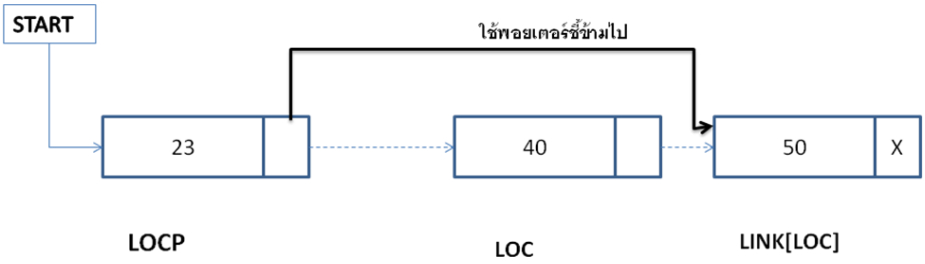
รูปที่ 5.27 (c) แสดงลิสต์หลังการลบ

อย่างไรก็ตามการลบจะต้องมีตัวโนดก่อนหน้าและโนดหลังการลบ ดังนั้นจำเป็นต้องใช้ตัวแปรใช้ตัวแปรเพิ่มเติมอีกเพื่อจดจำตำแหน่งก่อนการลบ ดังนั้นตัวแปรที่จะนำมาใช้เพิ่มเป็นตัวแปรพอยเตอร์ ดังนิยามที่ 5.6

นิยามที่ 5.6 กำหนดให้

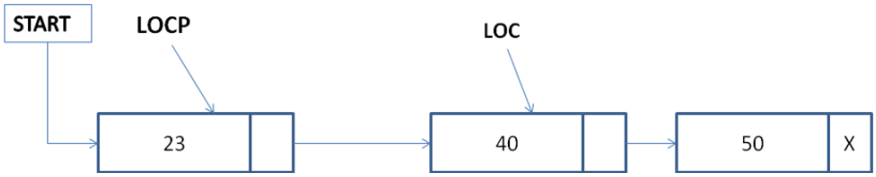
LOCP คือ พอยเตอร์สำหรับชี้ไปยังโนดของลิงค์ลิสต์ใดๆ

ดังที่กล่าวไปแล้วว่า การลบข้อมูลจำเป็นต้องจดจำตำแหน่งโนดก่อนหน้า โนดที่จะลบ และโนดหลังโนดที่จะลบ ต่อไปนี้จะใช้ตัวแปร LOCP แสดงตำแหน่งก่อนหน้าโนดที่จะลบ LOC แทนตำแหน่งของโนดที่จะลบ ขณะที่ LINK[LOC] แทนตำแหน่งหลังจากโนดที่จะลบ แสดงแนวคิดดังกล่าว ดังรูปที่ 5.28



รูปที่ 5.28 แสดงแนวคิดและตัวแปรดำเนินก่อนการลบ

ก่อนการลบจะต้องระบุ LOCP LOC ทั้งนี้อาจได้มาจากการค้นหาโดยใช้อัลกอริทึมสำหรับค้นหาข้อมูลเสียก่อน แสดงตำแหน่งพอยเตอร์ก่อนการลบข้อมูล ดังรูปต่อดังนี้



รูปที่ 5.29 แสดงการระบุตัวแปรก่อนการลบ

แนวคิดของการลบเกิดขึ้น 2 กรณี

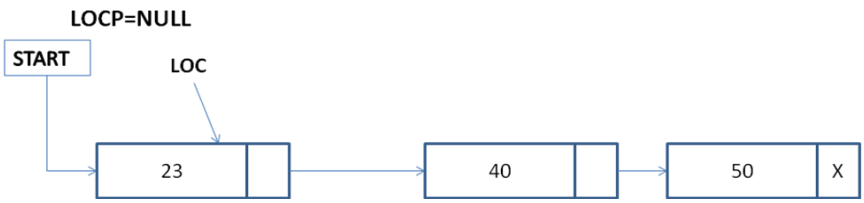
1) ลบโนดแรก ค่าตัวแปรจะเป็นดังนี้

- LOCP = NULL,
- LOC = START

2) ลบโนดอื่นๆ

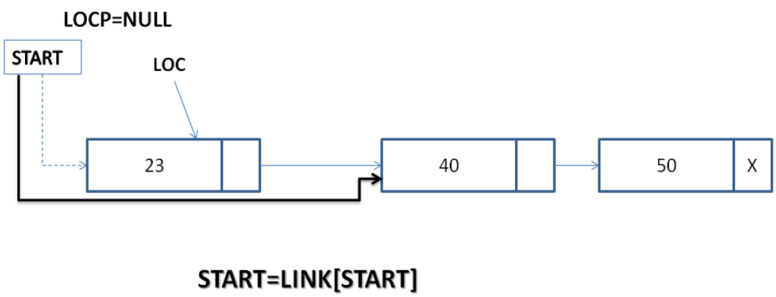
- LOCP,
- LOC ไม่เท่ากับ NULL

การลบโนดแรกสุด LOCP=NULL, LOC=START แสดงภาพก่อน ระหว่าง และหลังลบ ดังรูปที่ 5.30

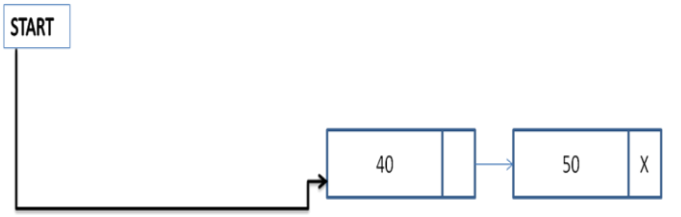


LOCP=NULL, LOC=START (ขณะลบ)

รูปที่ 5.30(a) แสดงการระบุตัวแปรก่อนการลบโนดแรก



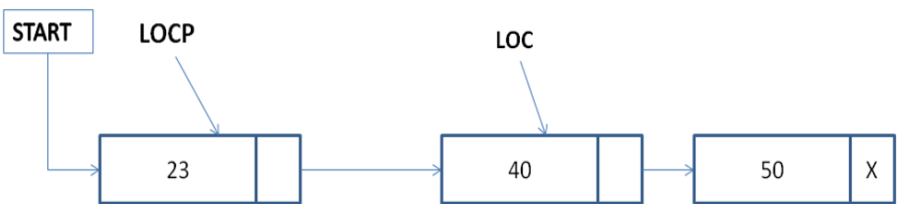
รูปที่ 5.30(b) แสดงการลบโดยกำหนดพอยเตอร์



LOCP=NULL, LOC=START (หลังลบ)

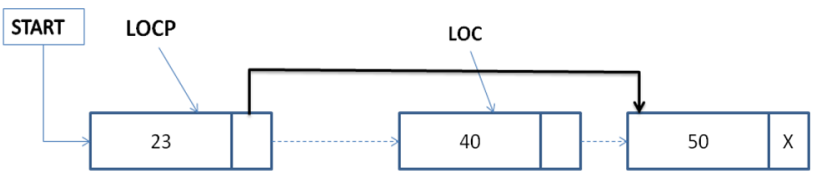
รูปที่ 5.30(c) แสดงพอยเตอร์หลังการลบโนดแรก

การลบโนดอื่นๆ ที่ไม่ใช่โนดแรก เมื่อ LOCP, LOC ไม่เป็น NULL แสดงภาพก่อนระหว่าง และหลังลบ ดังรูปที่ 5.31



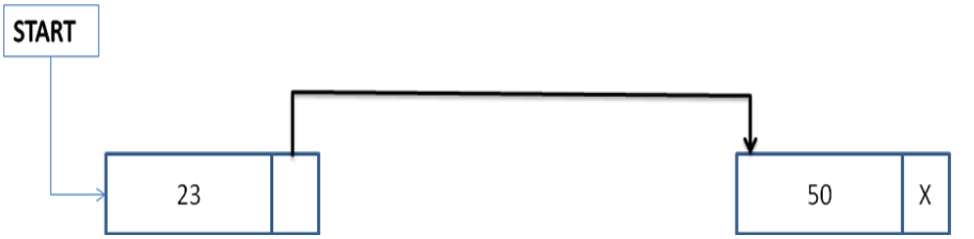
LOCP, LOC ไม่เป็น NULL(ขณะลบ)

รูปที่ 5.31(a) แสดงการกำหนดพอยเตอร์ก่อนการลบโนดที่ต้องการ



LINK[LOCP] = LINK[LOC]

รูปที่ 5.31(b) แสดงการเปลี่ยนพอยเตอร์การลบโนดที่ต้องการ



LOCP, LOC ไม่เป็น NULL(หลังลบ)

รูปที่ 5.31(c) แสดงการข้อมูลหลังการลบโนดที่ต้องการ

ขั้นตอนและอัลกอริทึมแสดงได้ดังนี้

แนวคิดการลบโนดที่ LOC ชี้อยู่ออกจากลิสต์ โดยมีตัวแปร LIST, INFOR, LINK, START, LOCP, LOC โดยที่โนดที่ LOC ชี้อยู่จะถูกลบออกไปจาก LIST	**Algorithm 5.7** : DEL Item in a Link Lists **Input** : LIST, INFOR, LINK, START, LOCP, LOC **Output** : Node LOC is deleted from LIST.
1 ตรวจสอบว่า ค่า LOC เป็น NULL หรือไม่ หากเป็น NULL ให้ลบที่ โนดแรก โดยกำหนด START= LINK[START] 2 หากไม่เป็น LOC ไม่เป็น NULL ให้ดำเนินการลบที่โนด ที่ LOC ชี้อยู่ โดยกำหนด LINK[LOCP] = LINK[LOC]	1 IF LOC = NULL Then 2 START= LINK[START] 3 Else 4 LINK[LOCP] = LINK[LOC] 5 End IF 6. Return

แสดงตัวอย่างการเขียนโปรแกรมตามอัลกอริทึม 5.7 เพื่อลบโนดในลิงค์ลิสต์ โดยระบุ LOCP และ LOC ให้กับเมธอดการทำงานการลบโดยตรง

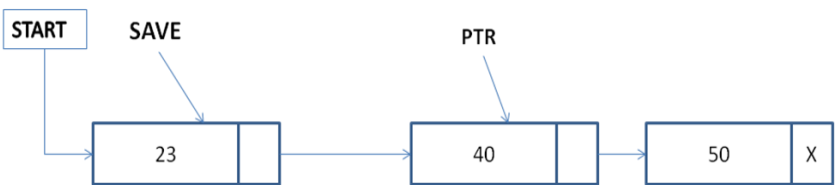
<u>ตัวอย่างที่ 5.12</u> แสดงตัวอย่างการลบโนดในลิงค์ลิสต์ตามอัลกอริทึม 5.7 เมื่อข้อมูลที่บรรจะอยู่ในลิงค์ลิสต์คือ 50, 60, 70, 80, 90, 100 โดยกำหนดให้LOCP ชี้ที่โนด B และ LOC ชี้ที่โนด C

```
1   package linklistimplementation;
2   public class Algor58DEL {
3           static Node START, first = new Node();
4           public static void main(String[] args) {
5           First.INFOR = 50;
6           START = first;
7           Node A = new Node(); A.INFOR = 60; START.LINK = A;
8           Node B = new Node(); B.INFOR = 70; A.LINK = B;
9           Node C = new Node(); C.INFOR = 80; B.LINK = C;
10          Node D = new Node(); D.INFOR = 90; C.LINK = D;
11          Node E = new Node(); E.INFOR = 100; D.LINK =E;
12          Node LOCP=B;
13          Node LOC=C;
14          System.out.println("Data Befor INSERT:");
15          showDatainList();
16          DEL(LOCP,LOC, START);          //Call DEL()...
17          System.out.println("Data After INSERT:");
18          showDatainList();
19          }
20       public static void DEL(Node LOCP, Node LOC, Node START){
21          //Algorrithm 5.8 Delete
22          if(LOCP==null) {
23            START = START.LINK;  //insert first Node
24          }else
25          {
26             LOCP.LINK = LOC.LINK;
27          }
28       }
29     public static void showDatainList(){
30          Node PTR=null;
31          PTR=START;
32          while(PTR!=null){
33              System.out.println(PTR.INFOR);
```

```
34              PTR=PTR.LINK;
35          }
36      }
37  }
```

ผลการรันโปรแกรมจะแสดง ข้อมูลก่อนและหลังลบโนดในตำแหน่ง C ซึ่งข้อมู 80 จะถูกลบไป

การหาตำแหน่ง LOCP, LOC จะใช้ ใช้พอยเตอร์ SAVE เพื่อท่องและคืนค่าเป็น LOC ในขณะที่พอยเตอร์ PTR เพื่อท่องและคืนค่าเป็น LOC แสดงแนวคิดของการค้นหา ดังรูป



รูปที่ 5.32 แสดงการหาตำแหน่งสำหรับการหาข้อมูลที่จะลบ

แนวคิดการค้นหา เพื่อนำ LOCP และ LOC มาใช้สำหรับการลบ และอัลกอริทึมแสดงดังนี้

แนวคิดของการค้นหาตำแหน่งที่จะลบ LOC โดยใช้ตัวแปรคือ LIST, INFOR, LINK, START, LOCP, LOC, SAVE, PTR, ITEM เมื่อค้นหาสำเร็จ จะคืนค่า LOCP, LOC.	**Algorithm 5.8** FINDLOCPandLOC find ITEM in a Link Lists **Input** : LIST, INFOR, LINK, START, LOCP, LOC, SAVE, PTR, ITEM **Output** : LOCP, LOC.
1 ตรวจสอบว่า เป็นลิสต์ว่างหรือไม่ หากเป็นลิสต์ว่า ให้กำหนด LOCP, LOC เป็น NULL แล้วคือค่า จบงาน 2 ตรวจสอบว่า ค่าข้อมูลโนดแรกสุดเท่ากับข้อมูลที่ต้องการค้นหาหรือไม่ หากเท่ากัน	1 IF START = NULL Then 2 LOCP= LOC=NULL 3 Return LOCP, LOC 4 End IF 5 IF INFOR[START]=ITEM Then

ให้กำหนด LOCP เป็น NULL ส่วน LOC ให้เป็น START 3 ถ้าไม่ใช่สองกรณีดังกล่าว ให้กำหนด การท่องโดยใช้พอยเตอร์ SAVE ชี้ที่ START และพอยเตอร์ PTR ชี้ตัวที่ START ชี้อยู่ แล้ววนเปรียบเทียบจนพบข้อมูลตรงกับ ITEM จึงกำหนดให้ LOCP=SAVE LOC=PTR แล้ว คืนค่า LOCP และ LOC	6 LOCP=NULL, 7 LOC = START 8 Return LOCP, LOC 9 End IF 10 SAVE=START, PTR=LINK[START] 11 While PTR!=NULL DO 12 IF INFOR[PTR]=ITEM Then 13 LOCP=SAVE, LOC=PTR 14 Return LOCP, LOC 15 End IF 16 SAVE=PTR, PTR=LINK[PTR] 17 End While 18 End IF 19 Return

การลบข้อมูลแบบเต็มรูปแบบ จะต้องเรียกใช้อัลกอริทึมสำหรับการค้นหาข้อมูล เพื่อได้ LOCP และ LOC ออกมา แล้วจึงดำเนินการลบโดยการปรับพอยเตอร์สำหรับชี้ สำหรับ LOCP แสดงแนวคิดแบบเต็ม และอัลกอริทึมการลบแบบเต็มรูปแบบดังนี้

แนวคิดการลบเต็มรูปแบบ เมื่อต้องการลบโนดที่มี ITEM บรรจุอยู่ ออกจากลิสต์ โดยที่มีตัวแปรคือ LIST, INFOR, LINK, START, LOCP, LOC, ITEM เมื่อลบสำเร็จข้อมูลโนดที่ LOC ชี้อยู่จะถูกลบออกจาก LIST	**Algorithm 5.9:** Full Deletion Item in a Link Lists **Input** : LIST, INFOR, LINK, START, LOCP, LOC, ITEM **Output** : Item at LOC is deleted from LIST.

1 เรียกใช้อัลกอริทึม 5.8 เพื่อให้ได้ค่า LOCP และ LOC 2 ถ้า LOC=NULL แสดงว่าข้อมูลที่จะลบไม่มีอยู่ในลิสต์ให้รายงานว่าไม่พบข้อมูลที่จะลบ แล้วจบงาน 3 ถ้า LOCP=NULL แสดงว่าโนดที่จะลบอยู่โนดแรก ให้ลบโนดแรก โดยกำหนด START=LINK[START] 4 กรณีอื่นๆ ให้ ลบโนด LOC โดยกำหนด LINK[LOCP]=LINK[LOC] แล้วจบงาน	5 Call Algorithm 5.8 FINDLOCPandLOC (LIST, INFOR, LINK, START, LOCP, LOC, SAVE, PTR, ITEM) 6 IF LOC=NULL Then 7 Write ITEM is not exist in LIST 8 Return 9 End IF 10 IF LOCP=NULL Then 11 START=LINK[START] 12 Else 13 LINK[LOCP]=LINK[LOC] 14 End IF 15 Return

แสดงตัวอย่างการเขียนโปรแกรมเพื่อลบข้อมูลออกจากลิงค์ลิสต์ตามแนวทางของอัลกอริทึม 5.8 และ 5.9 ดังตัวอย่างที่ 5.13

ตัวอย่างที่ 5.13 แสดงตัวอย่างโปรแกรมสำหรับการลบโนดในลิงค์ลิสต์ตามอัลกอริทึม 5.8 และ 5.9 เมื่อข้อมูลที่บรรจะอยู่ในลิงค์ลิสต์คือ 50, 60, 70, 80, 90, 100 โดยใช้ LOCP และ LOC จากการค้นหาตามแนวทางของอัลกอริทึม 5.8

```
1   public class Algor59FullDeletion {
2           static Node START, first = new Node();
3           static Node LOCP=null;
4           static Node LOC=null;
```

```
5           public static void main(String[] args) {
6           first.INFOR= 50;
7           START = first;
8           Node A = new Node(); A.INFOR = 60; START.LINK = A;
9           Node B = new Node(); B.INFOR = 70; A.LINK = B;
10          Node C = new Node(); C.INFOR = 80; B.LINK = C;
11          Node D = new Node(); D.INFOR = 90; C.LINK = D;
12          Node E = new Node(); E.INFOR = 100; D.LINK =E;
13          System.out.println("Data Befor INSERT:");
14          showDatainList();
15          DEL(70);          //Call INSLOC()...
16          System.out.println("Data After INSERT:");
17          showDatainList();
18          }
19    public static void DEL(int ITEM){
20          //Algorrithm 5.8 Delete
21          FINDLOCPandLOC(ITEM);
22          if(LOCP==null) {
23            START = START.LINK;  //insert first Node
24          }else
25          {
26             LOCP.LINK = LOC.LINK;
27          }
28     }
29     public static boolean FINDLOCPandLOC(int ITEM){
30          Node PTR=null;
31          Node SAVE=null;
32          PTR=START;
33          if(START==null) { LOC=null; LOCP=null; return true;}
34          if((int)START.INFOR==ITEM) { LOC=START; LOCP=null; return true;}
35          SAVE=START; PTR=START.LINK;
36          while(PTR!=null){
37            if((int)PTR.INFOR==ITEM) {
```

```
38              LOC = PTR; LOCP=SAVE;
39              return true;
40             }
41             SAVE=PTR;
42             PTR=PTR.LINK;
43          }
44          LOC=null;
45          //Show Result;
46                  if(LOC ==null){
47                     System.out.println("Not found.");
48                     return false;
49                  }
50                  else
51                  {
52                      System.out.println("Found.");
53                      return true;
54                  }
55      }
56      public static void showDatainList(){
57          Node PTR=null;
58          PTR=START;
59          while(PTR!=null){
60              System.out.println(PTR.INFOR);
61              PTR=PTR.LINK;
62          }
63      }
64  }
```

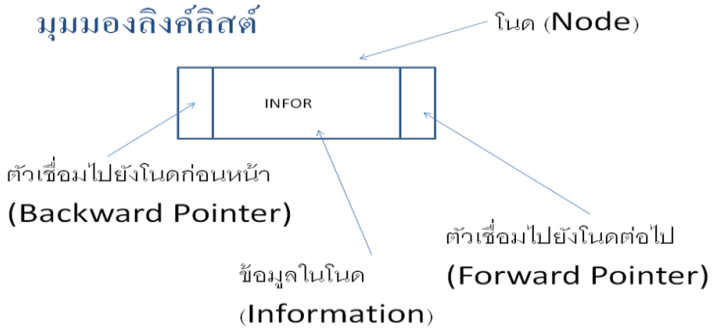
ผลการรันจะแสดงข้อมูลก่อนและหลังการลบโนด 70 ออกจากลิงค์ลิสต์ดังกล่าว

5.5 ลิงค์ลิสต์แบบสองทาง (Two-ways Linked Lists)

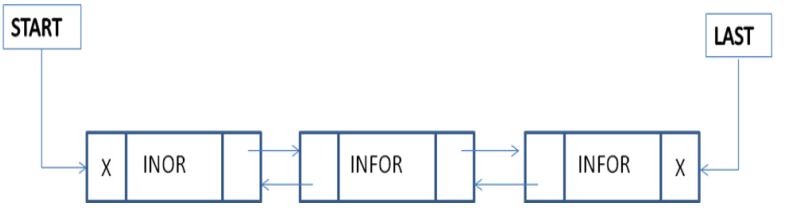
เนื่องจากลิงค์ลิสต์แบบทางเดียวยังมีข้อจำกัดด้านการท่องและมีข้อด้อยบางอย่าง เช่น หากลิงค์เชื่อมโยงไปด้านหน้าเสียหายจะไม่สามารถท่องหรือเข้าถึงข้อมูลได้ ดังนั้นหากมีลิงค์ลิสต์ที่สามารถท่องได้ทั้งสองทาง จะทำให้ลิงค์ลิสต์มีความความเสถียรมากยิ่งขึ้น เป็นต้น ต่อไปนี้จะนำเสนอแนวคิด และตัวดำเนินการต่างๆ ของลิงค์ลิสต์แบบสองทาง

5.5.1 แนวคิดเบื้องต้นเกี่ยวกับลิงค์ลิสต์แบบสองทาง

ลิงค์ลิสต์แบบสองทาง คือ ลิงค์ลิสต์ที่สามารถเชื่อมโยงได้สองทิศทางคือเชื่อมโดยโยงไปยังโนดถัดไปและชี้ย้อนกลับไปยังโนดที่ผ่านมาได้ องค์ประกอบของโนดภายในลิงค์ลิสต์มีองค์ประกอบ 3 ส่วน คือ ข้อมูล(Information--INFOR) พอยเตอร์ชี้ไปข้างหน้า(Forward Pointer –FORW) และพอยเตอร์ชี้กลับ(Backward Pointer—BACK) แสดงดังภาพมุมมองของโนด และลิงค์ลิสต์แบบสองทางดังรูปที่ 33 และรูปที่ 34 ตามลำดับ



รูปที่ 5.33 แสดงมุมมองของโนดลิงค์ลิสต์แบบสองทาง



รูปที่ 5.34 แสดงมุมมองลิงค์ลิสต์แบบสองทางและตัวแปร

ดังนั้น เพื่อให้การอธิบายแนวคิด ตัวแปร และอัลกอริทึม มีความเข้าใจตรงกัน นิยามที่ 5.7

นิยามที่ 5.7 กำหนดให้

DNODE	คือ โนดของลิงค์ลิสต์แบบสองทาง มีองค์ประกอบดังนี้
INFOR	คือ ส่วนที่เก็บข้อมูลในโนด
START	คือ ตัวแปรพอยเตอร์ชี้ไปยังโนดแรก
LAST	คือ พอยเตอร์ที่ชี้ไปยังโนดสุดท้าย
FORW	คือ พอยเตอร์ที่ชี้ไปข้างหน้า
BACK	คือ พอยเตอร์ที่ชี้ไปยังย้อนกลับ

การสร้างโนดลิงค์ลิสต์แบบสองทาง ด้วยภาษาจาวา ดังนี้

```
public class DNode {
    Object INFOR;
    DNode BACK=null;
    DNode FORW=null;
}
```

สำหรับตัวดำเนินการของลิงค์ลิสต์แบบสองทางก็คล้ายกับลิงค์ลิสต์แบบทางเดียว

5.5.2 การท่องลิงค์ลิสต์แบบสองทาง

แนวทางการท่อง สามารถทำได้ 2 แนวทาง คือ

1) ท่องจากโนดแรกที่พอยเตอร์ START ชี้อยู่ ไปยังโนดสุดท้ายที่พอยเตอร์ LAST ชี้อยู่

2) ท่องจากโนดสุดท้ายที่พอยเตอร์ LAST ชี้อยู่ ย้อนมากลับมายังโนดแรกที่ START ชี้อยู่

สำหรับแนวทางการค้นหาเบื้องลึกของอัลกอริทึมนั้น เป็นแบบลิงค์ลิสต์ทางเดียว แสดงแนวคิดและอัลกอริทึมดังนี้

แนวคิดการท่องลิงค์ลิสต์แบบสองทางจากด้านหน้าไปด้านหลัง โดยต้องมีข้อมูล LIST เป็นลิงค์ลิสต์, PTR, START FORW, LAST เพื่ออดำเนินการท่องโนดใน LIST ทั้งหมด	**Algorithm 5.10** Traversing two way LinkedLists **Input** : LIST, PTR, START,LAST, FORW, BACK **Output** : All nodes in LIST are traversed
1. นำพอยเตอร์ไปชี้ยัง START เพื่อกำหนดโนดแรกที่จะท่อง 2. ดำเนินการกับข้อมูล INFOR ของโนดที่ PRT ชี้อยู่ INFOR [PTR] 3. ขยับ PTR โดยกำหนด PTR=FORW.PTR เป็นการขยับพอยเตอร์ชี้ไปยังโนดถัดไป 4. หาก PTR $\neq$ NULL คือจบลิงค์ - วนดำเนินการตามข้อ 2 หากจบลิงค์ จบการทำงาน	1 PTR=START 2 While PTR!=LAST Do 3 Process INFOR[PTR] 4 PTR=FORW[PTR] 5 End While 6 Return

การท่องลิงค์ลิสต์สามารถท่องย้อนกลับได้ โดยเริ่มจาก LAST ย้อนมายัง START

แนวคิดการท่องลิงค์ลิสต์แบบสองทางจากด้านหน้าไปด้านหลัง โดยต้องมีข้อมูล LIST เป็นลิงค์ลิสต์, PTR, START FORW, LAST เพื่ออดำเนินการท่องโนดใน LIST ทั้งหมด	**Algorithm 5.11** Traversing two way LinkedLists from LAST to START **Input** : LIST, PTR, START,LAST, FORW, BACK **Output** : All nodes in LIST are traversed

1. นำพอยเตอร์ไปชี้ยัง LAST เพื่อกำหนดโนดแรกที่จะท่อง 2. ดำเนินการกับข้อมูล INFOR ของโนดที่ PRT ชี้อยู่ INFOR [PTR] 3. ขยับ PTR โดยกำหนด PTR=BACK.PTR เป็นการขยับพอยเตอร์ชี้ไปยังโนดถัดไป 4. หาก PTR ≠ START คือจบลิงค์ - วนดำเนินการตามข้อ 2 หากจบลิงค์ จบการทำงาน	7 PTR=LAST 8 While PTR!=START Do 9 Process INFOR[PTR] 10 PTR=BACK[PTR] 11 End While 12 Return

<u>ตัวอย่างที่ 5.14</u> จงเขียนโปรแกรมแสดงตัวอย่างการท่องลิงค์ลิสต์แบบสองทางจากโนดแรกไปโนดสุดท้าย และจากโนดสุดท้ายย้อนกลับมาหาโนดแรก โดยกำหนดให้มีข้อมูลในลิงค์ลิสต์ คือ 10, 20, 30, 40, 50

เฉลย : เบื้องต้นต้องสร้างโครงสร้างโนด DNode เป็นคลาสเพื่อใช้งานในการเก็บข้อมูล จากนั้นในโปรแกรมที่จะสร้างการท่องโนด กำหนดให้มีเมธอดดังนี้

- main() เป็นเมธอดสำหรับการสร้างลิงค์ลิสต์ และเรียกใช้เมธอดการท่อง
- ForwardTraversingList(); ใช้สำหรับท่องจากโนดแรกไปยังโนดสุดท้าย
- BackwardTraversingList(); ใช้สำหรับท่องจากโนดท้ายย้อนกลับมายังโนดแรก

แสดงโปรแกรมดังนี้

```
1   public class Algor50to511TraversingDoubleList {
2     static DNode START;
3     static DNode LAST;
4     public static void main(String[] args) {
5           DNode NodeA = new DNode(); NodeA.INFOR = 10;
6           DNode NodeB = new DNode(); NodeB.INFOR = 20;
7           DNode NodeC = new DNode(); NodeC.INFOR = 30;
```

```
8           DNode NodeD = new DNode(); NodeD.INFOR = 40;
9           DNode NodeE = new DNode(); NodeE.INFOR = 50;
10          START = NodeA;
11          NodeA.FORW =NodeB; NodeB.BACK = NodeA;
12          NodeB.FORW =NodeC; NodeC.BACK = NodeB;
13          NodeC.FORW =NodeD; NodeD.BACK = NodeC;
14          NodeD.FORW =NodeE; NodeE.BACK = NodeD;
15          NodeE.FORW =null; LAST=NodeE;
16          System.out.println("Forward Traversing:");
17          ForwardTraversingList();
18          System.out.println("Backward Traversing:");
19          BackwardTraversingList();
20          }
21      //end main
22
23    public static void ForwardTraversingList(){
24          DNode PTR=null;
25          PTR=START;
26          while(PTR!=null){
27              System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
28              PTR=PTR.FORW;
29          }
30      }
31  public static void BackwardTraversingList(){
32          DNode PTR=null;
33          PTR=LAST;
34          while(PTR!=null){
35              System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
36              PTR=PTR.BACK;
37          }
38      }
39  }
```

5.5.3 การค้นหาข้อมูลในลิงค์ลิสต์แบบสองทาง

แนวทางการค้นหาข้อมูลในลิงค์ลิสต์แบบสองทาง มีลักษณะคล้ายกับการท่องลิงค์ลิสต์ที่ผ่านมา โดยสามารถนำเอาอัลกอริทึมการค้นหาข้อมูลในลิสต์ที่ผ่านมาปรับใช้งานเพียงแต่อาจต้องปรับพอยเตอร์เป็น FORW และ BACK ในการท่องแต่ละโนดเพื่อเปรียบเทียบ โดยที่ เริ่มค้นโดยเอาพอยเตอร์ PTR ชี้ที่ START และท่องเปรียบเทียบ หรือหากต้องการค้นย้อนกลับจาก LAST ก็สามารถดำเนินการได้

ตัวอย่างที่ 5.15 จงเขียนโปรแกรมค้นหาข้อมูลในลิงค์ลิสต์แบบสองทางที่บรรจุข้อมูล 20, 30, 40, 50, 60, 70, 80, 90, 100 โดย ITEM ที่จะค้นหาคือ 30 โดยประยุกต์ใช้อัลกอริทึมสำหรับการค้นหาที่ผ่านมาในแบบของลิงค์ลิสต์ทางเดียว

เฉลย: การออกแบบและเมธอด SEARCH() เป็นเมธอดหลักในการค้นหาโดยประยุกต์ใช้จากอัลกอริทึมค้นหาที่ผ่านมา แล้วปรับใช้ DNode, START, FORW เป็นพอยเตอร์สำหรับการเดินหน้าท่องลิงค์ลิสต์เพื่อเปรียบเทียบ การออกแบบเมธอดของโปรแกรมดังนี้

- main() เป็นเมธอดสำหรับการสร้างลิงค์ลิสต์ และเรียกใช้เมธอดการค้นหาข้อมูล
- SEARCH(int ITEM) ใช้สำหรับค้นหาจากโนดโนดแรกเป็นต้นไป

แสดงโปรแกรมดังนี้

```
1   public class AlgorSEARCHDoubleList {
2       static DNode START, LAST;
3       public static void main(String[] args) {
4       DNode NodeA = new DNode(); NodeA.INFOR = 20;
5       DNode NodeB = new DNode(); NodeB.INFOR = 30;
6       DNode NodeC = new DNode(); NodeC.INFOR = 40;
7       DNode NodeD = new DNode(); NodeD.INFOR = 50;
8       DNode NodeE = new DNode(); NodeE.INFOR = 60;
9       DNode NodeF = new DNode(); NodeF.INFOR = 70;
10      DNode NodeG = new DNode(); NodeG.INFOR = 80;
11      DNode NodeH = new DNode(); NodeH.INFOR = 90;
```

```
12          DNode NodeI = new DNode(); NodeI.INFOR = 100;
13          START = NodeA;
14          NodeA.FORW =NodeB; NodeB.BACK = NodeA;
15          NodeB.FORW =NodeC; NodeC.BACK = NodeB;
16          NodeC.FORW =NodeD; NodeD.BACK = NodeC;
17          NodeD.FORW =NodeE; NodeE.BACK = NodeD;
18          NodeE.FORW =NodeF; NodeF.BACK = NodeE;
19          NodeF.FORW =NodeG; NodeG.BACK = NodeF;
20          NodeG.FORW =NodeH; NodeH.BACK = NodeG;
21          NodeH.FORW =NodeI; NodeI.BACK = NodeH;
22          NodeI.FORW =null;  LAST = NodeH;
23          //Search Data
24          SEARCH(30);
25          }
26      public static void SEARCH(int ITEM){
27          //Algorrithm 5.2 SEARCH
28          DNode LOC=null;
29          DNode PTR=null;
30          PTR=START;
31          while(PTR!=null){
32            if(ITEM==(int)PTR.INFOR) {
33                LOC = PTR;
34                break;
35            }else
36              PTR=PTR.FORW;
37          }
38          //Show Result;
39          if(LOC ==null)
40              System.out.println("Not found.");
41          else
42              System.out.println("Found.");
43      }
44  }
```

ผลการรันโปรแกรม ค้นหาข้อมูล 30 ว่าพบข้อมูล

5.6.4 การแทรกโนดในลิสต์แบบสองทาง

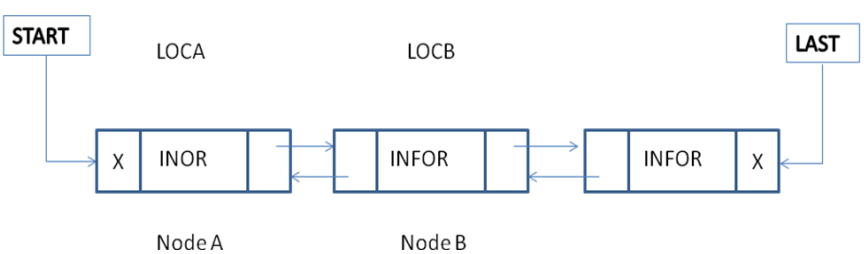
แนวคิดการแทรกโนดของลิงค์ลิสต์แบบสองทางเบื้องต้น คือเพิ่มโนดเข้าไปที่โนดแรก สามารถกระทำได้โดยอัลกอริทึมสำหรับการแทรกแบบที่ได้กล่าวไปแล้วเพียงแต่ปรับพอยเตอร์ให้เป็นแบบสองทาง แต่หากต้องการแทรกระหว่างสองโนด จะมีขั้นตอนเพิ่มขึ้นมากกว่าการแทรกแบบทางเดียว เพื่อให้เข้าในตรงกันในการอธิบายตัวแปรต่างๆ จึงกำหนดนิยาม 5.7 เพื่อประกอบการการอธิบายต่อไป

นิยามที่ 5.7 กำหนดให้

LOCA คือ พอยเตอร์สำหรับชี้ไปยังโนดของลิงค์ลิสต์แบบสองทางใดๆ

LOCB คือ พอยเตอร์สำหรับชี้ไปยังโนดของลิงค์ลิสต์แบบสองทางใดๆ

การแทรกโนดใหม่ในลิสต์แบบสองทาง การแทรกจะต้องคำนึงพ้อยเตอร์ที่ชี้ไปข้างหน้าและย้อนกลับ โดยกำหนดให้โนดก่อนจะตำแหน่งแทรก เป็นโนด A และ หลังการแทรกคือโนด B เมื่อตำแหน่งที่จะแทรกคือตำแหน่งระหว่างโนด A พอยเตอร์ชี้คือ LOCA และโนด B พอยเตอร์ชี้คือ LOCB แสดงแนวคิดของขั้นตอนการดำเนินการก่อนการแทรก เมื่อ NEW เป็นโนดใหม่ที่จะแทรกเข้าไปในลิสต์ ดังรูป



รูปที่ 5.35 แสดงมุมมองลิงค์ลิสต์ก่อนการแทรก

ขณะดำเนินการแทรก มีขั้นตอนดังนี้

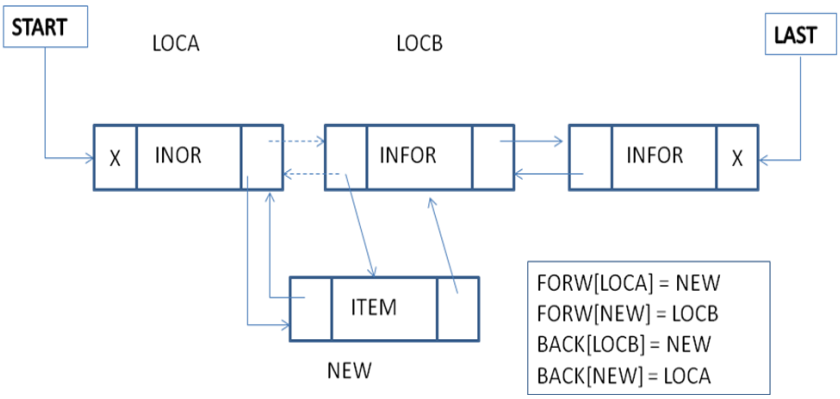
1 FORW[LOCA] = NEW

2 FORW[NEW] = LOCB

3 BACK[LOCB] = NEW

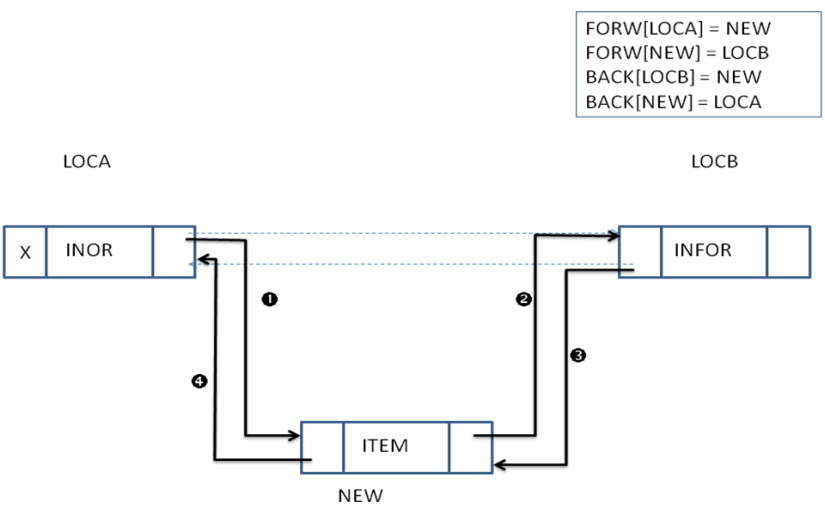
4 BACK[NEW] = LOCA

แสดงภาพขณะที่แทรก ดังนี้



รูปที่ 5.36 แสดงมุมมองลิงค์ลิสต์ขณะการแทรก

แสดงแนวคิดเมื่อพิจารณา เฉพาะระหว่างโนด และขั้นตอนในการแทรก



รูปที่ 5.37 แสดงมุมมองลิงค์ลิสต์ลำดับการแทรก

แนวคิดการแทรกโนด NEW เข้าไปในลิงค์ลิสต์แบบสองทาง โดยต้องมีข้อมูล LIST เป็นลิงค์ลิสต์, PTR, START FORW, LAST, FORW, BACK, NEW โดย NEW แทรกระหว่าง LOCA, LOCB	**Algorithm 5.12** Insert Data into two way LinkedLists **Input** : LIST, PTR, START,LAST, FORW, BACK, NEW, LOCA, LOCB **Output** : NEW is inserted into in LIST
1. นำพอยเตอร์ FORW ของโนดก่อนหน้า ชี้ยังโนดด้านหน้า ชี้ไปยังโนดใหม่ NEW 2. นำพอยเตอร์ FORW ของโนดใหม่ NEW ชี้ไปยัง โนดหลัง LOCB 3. นำพอยเตอร์ย้อนกลับ BACK ของโนดหลัง LOCB ชี้โนดใหม่ NEW 4. นำพอยเตอร์ย้อนกลับ BACK ของ NEW ชี้ไปยังโนด ด้านหน้า LOCA	1 FORW[LOCA] = NEW 2 FORW[NEW] = LOCB 3 BACK[LOCB] = NEW 4 BACK[NEW] = LOCA 5 Return

<u>ตัวอย่างที่ 5.16</u> จงเขียนโปรแกรมแสดง การแทรกตามอัลกอริทึม

```
1   public class Algor516INSTWL {
2         static DNode START;
3     static DNode LAST;
4     public static void main(String[] args) {
5           DNode NodeA = new DNode(); NodeA.INFOR = 10;
6           DNode NodeB = new DNode(); NodeB.INFOR = 20;
7           DNode NodeC = new DNode(); NodeC.INFOR = 30;
8           DNode NodeD = new DNode(); NodeD.INFOR = 40;
9           DNode NodeE = new DNode(); NodeE.INFOR = 50;
10          DNode NodeF = new DNode(); NodeF.INFOR = 35; //New Node to be inserted.
11          START = NodeA;
12          NodeA.FORW =NodeB; NodeB.BACK = NodeA;
13          NodeB.FORW =NodeC; NodeC.BACK = NodeB;
14          NodeC.FORW =NodeD; NodeD.BACK = NodeC;
```

```
15          NodeD.FORW =NodeE; NodeE.BACK = NodeD;
16          NodeE.FORW =null; LAST=NodeE;
17          System.out.println("Before Inserting:");
18          ForwardTraversingList();
19          INSTWL(NodeC,NodeD, NodeF);
20          System.out.println("After Inserting:");
21          ForwardTraversingList();
22          }
23      //end main
24
25      public static void INSTWL(DNode LOCA,DNode LOCB, DNode NEW){
26         LOCA.FORW = NEW; NEW.FORW = LOCB;
27         LOCB.BACK = NEW; NEW.BACK = LOCA;
28      }
29
30      public static void ForwardTraversingList(){
31          DNode PTR=null;
32          PTR=START;
33          while(PTR!=null){
34              System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
35              PTR=PTR.FORW;
36          }
37      }
38  }
```

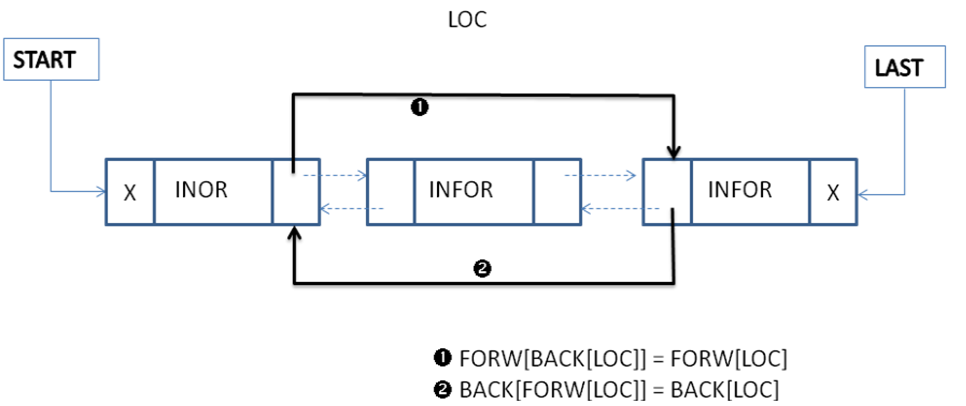
ผลการรันโปรแกรมจะแสดงข้อมูลก่อนและหลังการแทรก

5.5.5 การลบโนดของลิสต์แบบสองทาง

การลบโดยแนวคิดโดยทั่วไปคล้ายลิสต์แบบทางเดียว กล่าวคือจะให้พอยเตอร์โนดก่อนหน้าที่จะลบ ชี้ข้ามโนดที่จะลบไป จะทำให้โนดเป้าหมายถูกลบไป แต่เมื่อมีพอยเตอร์แบบสองทาง จะต้องนำเอาพอยเตอร์โนดที่อยู่หลังลบโนดที่จะลบชี้ข้ามกลับไปยังโนดที่อยู่ก่อนหน้าโนดที่จะลบ คือ

- FORW[BACK[LOC]] = FORW[LOC]
- BACK[FORW[LOC]] = BACK[LOC]

แสดงเป็นภาพได้ดังนี้



รูปที่ 5.34 แสดงแนวคิดการลบลิงค์ลิสต์แบบสองทาง

ขั้นตอนแนวคิดและอัลกอริทึมแสดงดังนี้

แนวคิดการแทรกโนด NEW เข้าไปในลิงค์ลิสต์แบบสองทาง โดยต้องมีข้อมูล LIST เป็นลิงค์ลิสต์, PTR, START FORW, LAST, FORW, BACK, NEW โดย NEW แทรกระหว่าง LOCA, LOCB	**Algorithm 5.13** Delete Data into two way LinkedLists **Input** : LIST, PTR, START,LAST, FORW, BACK, NEW, LOC **Output** : NEW is inserted into in LIST
1. นำพอยเตอร์ FORW ของโนดก่อนหน้า ชี้ยังโนดถัดไปจากโนดที่จะลบ 2. นำพอยเตอร์ BACK ของโนดหลังของโนดที่จะลบ ชี้ย้อนกลับไปยังโนดก่อนหน้าโนดที่จะลบ	1 FORW[BACK[LOC]] = FORW[LOC] 2 BACK[FORW[LOC]] = BACK[LOC] 3 Return

ตัวอย่างที่ 5.17 จงแสดงการลบโนดของลิงค์ลิสต์แบบสองทางโดยใช้อัลกอริทึมที่ 5.13 ลบโนด

```
1   public class Algor513DELTWL {
2         static DNode START;
3    static DNode LAST;
4    public static void main(String[] args) {
5          DNode NodeA = new DNode(); NodeA.INFOR = 10;
6          DNode NodeB = new DNode(); NodeB.INFOR = 20;
7          DNode NodeC = new DNode(); NodeC.INFOR = 30;
8          DNode NodeD = new DNode(); NodeD.INFOR = 40;
9          DNode NodeE = new DNode(); NodeE.INFOR = 50;
10         START = NodeA;
11         NodeA.FORW =NodeB; NodeB.BACK = NodeA;
12         NodeB.FORW =NodeC; NodeC.BACK = NodeB;
13         NodeC.FORW =NodeD; NodeD.BACK = NodeC;
14         NodeD.FORW =NodeE; NodeE.BACK = NodeD;
15         NodeE.FORW =null; LAST=NodeE;
16         System.out.println("Before Deleting:");
17         ForwardTraversingList();
18         DELTWL(NodeC);
19         System.out.println("After Deleting:");
20         ForwardTraversingList();
21         }
22    //end main
23    public static void DELTWL(DNode LOC){
24       LOC.BACK.FORW = LOC.FORW;
25       LOC.FORW.BACK = LOC.BACK;
26    }
27    public static void ForwardTraversingList(){
28         DNode PTR=null;
29         PTR=START;
```

```
30          while(PTR!=null){
31              System.out.println(PTR.INFOR);  //Apply PROCESS to INFOR[PTR]
32              PTR=PTR.FORW;
33          }
34      }
35  }
```

ผลการรันโปรแกรม จะแสดงข้อมูลก่อนและหลังการลบโดยอาศัยอัลกอริทึม 5.13

5.7 การใช้งานโครงสร้างลิงค์ลิสต์ด้วยภาษาจาวา

จากภาษาจาวามีแพคเกจ java.util.LinkedList ให้ใช้งาน เมื่อประกาศเป็นอ็อบเจกต์แบบลิงค์ลิสต์แล้ว จะมีเมธอดใช้เป็นตัวดำเนินการดังนี

- .add() สำหรับเพิ่มข้อมูลโนดใหม่
- .remove() สำหรับลบข้อมูล
- .getFirst(), .getLast() สำหรับนำเอาข้อมูลโนดแรกและโนดสุดท้ายออกมาจากลิงค์ลิสต์
- .indexOf() สำหรับค้นหาข้อมูลในลิงค์ลิสต์

นอกจากนั้นหากต้องการจัดเรียงข้อมูลยังสามารถใช้ทำงานร่วมกันกับ แพคเกจ java.util.Collections เพื่อจัดเรียงข้อมูลในลิงค์ลิสต์ได้อีกด้วย ต่อไปนี้จะแสดงตัวอย่างการใช้งานเมธอดที่กล่าวไว้พอสังเขป

ตัวอย่างที่ 5.18 จงเขียนโปรแกรมแสดงการเพิ่มข้อมูลตัวเลขจำนวนเต็มด้วย .add และนำข้อมูลออกมาแสดงด้วย .getFirst() และ .peek() โดยใช้ java.util.LinkedList

```
1   package JavaADT;
2   import java.util.LinkedList;
3   public class SimpleJavaLinkedListExample {
```

```
4       public static void main(String[] args) {
5         LinkedList myList = new LinkedList();
6         //add elements to LinkedList
7         myList.add(1);
8         myList.add(2);
9         myList.add(3);
10        myList.add(4);
11        myList.add(5);
12        System.out.println("LinkedList contains : " + myList);
13        System.out.println(myList.getFirst());
14        System.out.println(myList.peek());
15      }
16    }
17
```

ผลการรันโปรแกรมจะแสดงข้อมูลที่เพิ่มเข้าไปทั้งหมด และแสดงโนดแรกและโนดบนสุดซึ่งเป็นโนดเดียวกัน

ตัวอย่างที่ 5.19 จงเขียนโปรแกรมแสดงการเพิ่มข้อมูลตัวเลขจำนวนเต็ม และจัดเรียงข้อมูลด้วย java.util.Collections

```
1   package JavaADT;
2   import java.util.LinkedList;
3   import java.util.Collections;
4   public class SortingLinkedList {
5     public static void main(String[] args) {
6       LinkedList myList = new LinkedList();
7       myList.add(4);
8       myList.add(2);
9       myList.add(9);
10      myList.add(1);
```

```
11      myList.add(5);
12      Collections.sort(myList);
13      System.out.println(myList);
14    }
15  }
```

ผลการรันจะแสดงข้อมูลที่จัดเรียงแล้ว ออกมา

สรุป

ลิงค์ลิสต์เป็นโครงสร้างข้อมูลแบบต่อเนื่อง โดยมีข้อมูลแต่ละรายการเรียกว่าโนด เชื่อมโยงต่อกันไป ข้อมูลแต่ละโนดประกอบด้วยส่วนของข้อมูลและส่วนที่ใช้ในการเชื่อมโยงไปยังโนดอื่นๆ ประเภทของลิงค์ลิสต์ที่สำคัญ ได้แก่ ลิงค์ลิสต์แบบทางเดียว ลิงค์ลิสต์แบบสองทาง ตัวดำเนินการของลิงค์ลิสต์ ได้แก่ การท่อง การเพิ่มข้อมูล การลบ และการค้นหาข้อมูล การท่องลิงค์ลิสต์แบบทางเดียวจะเริ่มต้นที่โนดแรก ดำเนินการกับข้อมูลที่พอยเตอร์ชี้อยู่ จากนั้นขยับพอยเตอร์ไปยังโนดถัดไป ดำเนินการไปจนกระทั่งถึงโนดสุดท้าย สำหรับการท่องลิงค์ลิสต์แบบสองทาง สามารถเข้าที่โนดแรกหรือโนดสุดท้ายดำเนินการกับข้อมูลแล้วท่องโดยตามพอยเตอร์ชี้ไปทั้งหน้าหรือตามพอยเตอร์ชี้กลับก็ได้

การแทรกเพื่อเพิ่มข้อมูล สามารถเพิ่มต้นลิสต์โดยให้พอยเตอร์ตัวที่จะเพิ่มชี้ไปยังข้อมูลที่พอยเตอร์ตัวแรกชี้อยู่ จากนั้นให้เปลี่ยนพอยเตอร์ตัวแรกมาชี้ที่ข้อมูลที่ดำเนินการแทรก หากเพิ่มที่ตัวที่กำหนด ให้พอยเตอร์ข้อมูลใหม่ชี้ไปยังตัวเดิมที่ข้อมูลที่กำหนดชี้อยู่ เปลี่ยนพอยเตอร์ที่กำหนดมาชี้ยังข้อมูลโนดใหม่ กรณีการดำเนินการกับลิงค์ลิสต์แบบสองทางก็มีลักษณะคล้ายกันเพียงแต่ต้องดำเนินการกับพอยเตอร์ที่ชี้กลับด้วย ส่วนการลบข้อมูลเพียงดำเนินการให้พอยเตอร์โนดก่อนจะลบชี้ข้ามไปยังโนดหลังจากที่จะลบ กรณีลิงค์ลิสต์แบบสองทางต้องดำเนินการกับพอยเตอร์ที่ทำการชี้กลับด้วย สำหรับการค้นหาข้อมูลที่ลักษณะคล้ายการท่องคือต้องท่องจากต้นลิสต์แล้วเทียบข้อมูลทีละรายการ หากเป็นลิสต์ที่เรียงแล้วสามารถท่องเพียงให้พบหรือท่องไปจนถึงรายการที่มากหรือน้อยกว่าข้อมูลที่จะค้นหาก็เพียงพอ

แบบฝึกหัด

ทฤษฎี

1. จงบอกความหมายและลักษณะของลิงค์ลิสต์
2. โนดคืออะไร มีลักษณะเป็นอย่างไร จงอธิบายและยกตัวอย่าง
3. ลิงค์ลิสต์แบบทางเดียวมีลักษณะอย่างไร จงอธิบายพร้อมยกตัวอย่าง
4. ลิงค์ลิสต์แบบสองทางมีลักษณะอย่างไร จงอธิบายพร้อมยกตัวอย่าง
5. จงสรุปขั้นตอนของการท่องลิงค์ลิสต์
6. จงอธิบายและยกตัวอย่างการเพิ่มข้อมูลแบบต้นลิสต์ ของลิงค์แบบทางเดียว
7. จงอธิบายและยกตัวอย่างการเพิ่มข้อมูลแบบต้นลิสต์ ของลิงค์แบบสองทาง
8. จงอธิบายอัลกอริทึมการแทรกข้อมูลในตำแหน่งที่ต้องการของลิงค์ลิสต์แบบทางเดียว
9. จงอธิบายอัลกอริทึมการแทรกข้อมูลในตำแหน่งที่ต้องการของลิงค์ลิสต์แบบสองทาง
10. จงอธิบายอัลกอริทึมการค้นหาข้อมูลในลิงค์ลิสต์แบบทางเดียว โดยที่ข้อมูลยังไม่ได้จัดเรียง
11. จงอธิบายอัลกอริทึมการค้นหาข้อมูลในลิงค์ลิสต์แบบทางเดียว โดยที่ข้อมูลจัดเรียงแล้ว
12. จงอธิบายอัลกอริทึมการค้นหาข้อมูลในลิงค์ลิสต์แบบสองทาง โดยที่ข้อมูลยังไม่ได้จัดเรียง
13. จงอธิบายอัลกอริทึมการค้นหาข้อมูลในลิงค์ลิสต์แบบสองทาง โดยที่ข้อมูลจัดเรียงแล้ว
14. จงอธิบายการท่องลิงค์ลิสต์แบบวงกลม และลิสต์ที่โนดนำ
15. จงเปรียบเทียบข้อดีและข้อเสียของลิงค์ลิสต์แบบทางเดียว สองทาง มีโนดนำ และแบบวงกลม

ปฏิบัติ

1. จงเขียนรหัสเทียมของโนด แบบทางเดียว
2. จงเขียนโปรแกรมสร้างโนด แบบทางเดียวและสองทาง
3. จงเขียนอัลกอริทึมการท่องลิงค์ลิสต์แบบทางเดียว จากนั้นเขียนโปรแกรมคอมพิวเตอร์แสดงการท่อง
4. กำหนดให้โนด 15 , 20 , 30 , 40 , 50 คือข้อมูลแต่ละโนดของลิงค์ลิสต์แบบทางเดียว จงเขียนโปรแกรมท่องลิงค์ลิสต์ดังกล่าว
5. จากข้อ 4 จงเขียนโปรแกรมแทรก 10 ลงไปที่ต้นลิสต์ และแสดงข้อมูลหลังเพิ่ม
6. จากข้อ 4 จงเขียนโปรแกรมเพิ่มโนดใหม่ที่ตำแหน่งระหว่าง 30 กับ 40 เพิ่ม 35
7. กำหนดให้ 100 , 300 , 500 , 700 , 900 , 1000 เป็นโนดของลิงค์ลิสต์แบบสองทาง จงเขียนโปรแกรมท่องลิงค์ลิสต์ดังกล่าว
8. จงเขียนโปรแกรมแทรกลิงค์ลิสต์ในข้อ 7 ด้วยตัวเลข 800

9. จากข้อ 4 จงเขียนโปรแกรมลบ 30 หลังจากนั้นแสดงข้อมูลที่เหลือ
10. จากข้อ 7 จงเขียนโปรแกรมลบ 700 ออกจาลิสต์ จากนั้นแสดงข้อมูลที่ยังเหลือ
11. จากลิสต์ในข้อ 4 จงเขียนโปรแกรมเพื่อค้นหาข้อมูล 45
12. จากลิสต์ข้อ 7 จงเขียนโปรแกรมค้นหาข้อมูล 300 โดยเริ่มค้นหาจากท้ายลิสต์

แนวข้อสอบ

1. กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นอาร์เรย์ที่เก็บข้อมูลของโนดในลิงค์ลิสต์

LINK เป็นอาร์เรย์ที่เก็บพ้อยเตอร์ที่ชี้ไปยังโนดถัดไปของลิงค์ลิสต์

START เป็นพอยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์

กำหนดข้อมูลที่มีอยู่ในลิสต์ตามลำดับแล้ว ดังนี้ 10,20,30,40,50

1.1. ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์ทางเดียวของข้อมูลดังกล่าว กำหนดให้ START ชี้ไปยังโนดแรก (2 คะแนน)

1.2. จงเขียนอัลกอริทึมสำหรับการนำข้อมูลในลิสต์ดังกล่าวยกกำลังสองแล้วเก็บไว้ที่เดิม พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึมดังกล่าวประกอบ (7 คะแนน)

1.3. จงเขียนอัลกอริทึม และวาดภาพ การแทรกข้อมูล 5 และ 45 ใหม่เข้าไปในลิสต์ดังกล่าว โดยข้อมูลยังจัดเรียงดังเดิม (ใช้ข้อมูลเดิมก่อนการยกกำลังสอง) (5 คะแนน)

1.4. หากปรับลิสต์ดังกล่าวเป็นลิสต์แบบสองทาง จงแสดงและเขียนอัลกอริทึมการลบข้อมูล 40 ออกจากลิสต์ พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึมดังกล่าวประกอบ (5 คะแนน)

2. กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นอาร์เรย์ที่เก็บข้อมูลของโนดในลิงค์ลิสต์

LINK เป็นอาร์เรย์ที่เก็บพ้อยเตอร์ที่ชี้ไปยังโนดถัดไปของลิงค์ลิสต์

START เป็นพอยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์

กำหนดข้อมูลที่มีอยู่ในลิสต์ตามลำดับแล้ว ดังนี้ 10, 20, 30, 40, 50

2.1. ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์ทางเดียวของข้อมูลดังกล่าว กำหนดให้ START ชี้ไปยังโนดแรก **(5 คะแนน)**

2.2. จงเขียนอัลกอริทึมสำหรับการท่องแสดงข้อมูลในลิสต์ดังกล่าว พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึมดังกล่าวประกอบ **(5 คะแนน)**

2.3. จงปรับลิสต์ดังกล่าวเป็นลิสต์แบบสองทาง จากนั้นวาดภาพและเขียนอัลกอริทึมการลบข้อมูล 40 ออกจากลิสต์ พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึมดังกล่าวประกอบ **(10 คะแนน)**

3. กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นที่เก็บข้อมูลของโนดในลิงค์ลิสต์; **LINK** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดถัดไป; **START** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์

กำหนดข้อมูลที่มีอยู่ในลิสต์ถูกเรียงตามลำดับแล้ว ดังนี้ 50, 40, 30, 20, 10

3.1 ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์ทางเดียวของข้อมูลดังกล่าว กำหนดให้ START ชี้ไปยังโนดแรก **(3 คะแนน)**

3.2 จงเขียนอัลกอริทึมสำหรับการค้นหา 25 ในลิสต์ดังกล่าว พร้อมทั้งอธิบายผลการค้นหา และวิเคราะห์ความซับซ้อนของอัลกอริทึม **(7 คะแนน)**

4. กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นที่เก็บข้อมูลของโนดในลิงค์ลิสต์; **LINK** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดถัดไป; **START** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์; **LAST** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดสุดท้าย

กำหนดข้อมูลที่มีอยู่ในลิสต์ถูกเรียงตามลำดับแล้ว ดังนี้ 10, 20, 30, 40, 50

4.1 ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์สองทางของข้อมูลดังกล่าว กำหนดให้ START ชี้ไปยังโนดแรก LAST ชี้ไปยังโนดสุดท้าย **(3 คะแนน)**

4.2 จงเขียนอัลกอริทึมสำหรับการแทรกโนด 35 ลงไปในลิสต์ดังกล่าว พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึม **(7 คะแนน)**

5. กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นที่เก็บข้อมูลของโนดในลิงค์ลิสต์; **LINK** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดถัดไป; **START** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์; **LAST** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดสุดท้าย

กำหนดข้อมูลที่มีอยู่ในลิสต์ถูกเรียงตามลำดับแล้ว ดังนี้ 60, 50, 40, 30, 20, 10

5.1 ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์สองทางของข้อมูลดังกล่าว กำหนดให้ START ชี้ไปยังโนดแรก LAST ชี้ไปยังโนดสุดท้าย **(3 คะแนน)**

5.2 จงเขียนอัลกอริทึมสำหรับการลบโนด 10 ออกจากลิสต์สองทางดังกล่าว พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึม **(7 คะแนน)**

6 กำหนดตัวแปรเกี่ยวกับลิงค์ลิสต์ให้ดังนี้

INFOR เป็นที่เก็บข้อมูลของโนดในลิงค์ลิสต์; **LINK** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดถัดไป; **START** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดแรกของลิงค์ลิสต์ และ **LAST** เป็นพ้อยเตอร์ที่ชี้ไปยังโนดสุดท้ายของลิงค์ลิสต์

กำหนดข้อมูลที่มีอยู่ในลิสต์ถูกเรียงตามลำดับแล้ว ดังนี้ 60, 50, 40, 30, 20, 10

6.1 ให้วาดภาพแสดงการเชื่อมโยงของมูลในลิสต์ที่เป็นลิสต์แบบสองทางของข้อมูลดังกล่าว **(5 คะแนน)**

6.2 จงเขียนอัลกอริทึมการลบโนด 40 ออกจากลิสต์ พร้อมทั้งวิเคราะห์ความซับซ้อนของอัลกอริทึม **(10 คะแนน)**

เอกสารอ้างอิง

เชาวลิต ขันคำ. (2550). **โปรแกรมภาษาจาวาด้วยเน็ตบีน**. ฉะเชิงเทรา : มหาวิทยาลัยราชภัฏราชนครินทร์.

เชาวลิต ขันคำ. (2554). **การเขียนโปรแกรมภาษาคอมพิวเตอร์**. ฉะเชิงเทรา : มหาวิทยาลัยราชภัฏราชนครินทร์.

โอภาส เอี่ยมสิริวงศ์. **โครงสร้างข้อมูลเพื่อการออกแบบโปรแกรมคอมพิวเตอร์** (Data Structures). กรุงเทพฯ: ซีเอ็ดยูเคชัน จำกัด (มหาชน), 2552. (ใช้สำหรับเรียนรู้ทฤษฎี)

Seymour Lipschutz:แปลโดย อุดม จีนประดับ และ ดร.สมคิด เรืองธนะสกุลไทย. **ทฤษฎีและตัวอย่างโจทย์ โครงสร้างข้อมูล---Theory and problem of Data Structures**. กรุงเทพฯ : แมคกรอ-ฮิล อินเตอร์เนชั่นแนล เอ็นเตอร์ไพรส์, อิงค์, (2540). (ใช้สำหรับเรียนรู้ทฤษฎี)

Robert Lafore. **Data Structure & Algorithms in Java**. (2003). 2nd Edition. USA:SAMS publishing. (ใช้ประกอบสำหรับการเขียนโปรแกรมสู่การประยุกต์ใช้งาน)

Michael T. Goodrich and Roberto Tamassia. **Data Structure & Algorithms in Java**. 4th Edition.