MITSUBISHI ELECTRIC
Changes for the Better

PLC
พื้นฐานการเขียนโปรแกรม (Structured Text)

การอบรมนี้กล่าวถึงวิธีการสร้างโปรแกรมพื้นฐานในการควบคุม PLC ของ Mitsubishi
โดยการอบรมนี้จะใช้รูปแบบ Structure text (ST) ในการออกแบบโปรแกรม

บทนำ วัตถุประสงค์ของหลักสูตร

การอบรมนี้อธิบายถึงวิธีการสร้างโปรแกรม PLC ของ Mitsubishi ด้วยรูปแบบ Structure text (ST)

ผู้เรียนจะต้องผ่านหลักสูตรต่อไปนี้หรือมีความรู้เทียบเท่าก่อนทำการเรียนหลักสูตรนี้:

การออกแบบโปรแกรมพื้นฐาน

การมีความรู้หรือประสบการณ์กับ C หรือพื้นฐาน จะสามารถช่วยเหลือในการทำความเข้าใจเนื้อหาของหลักสูตรนี้

บทนำ โครงสร้างของหลักสูตร

เนื้อหาของหลักสูตรนี้มีดังนี้

บทที่ 1 - ข้อมูลทั่วไปของการเขียนโปรแกรมด้วยภาษา Structure text

บทนี้อธิบายถึงลักษณะพิเศษและการใช้งานรูปแบบ Structure text (ST) ที่เหมาะสม

บทที่ 2 - กฎพื้นฐานในการเขียนโปรแกรม ST

บทนี้อธิบายถึงกฎพื้นฐานที่ใช้ในการสร้างโปรแกรมในรูปแบบ ST

บทที่ 3 - การสร้างโปรแกรมควบคุม I/O

บทนี้อธิบายถึงวิธีการสร้างโปรแกรมควบคุม I/O

บทที่ 4 – การทำงานเชิงคณิตศาสตร์

บทนี้อธิบายถึงวิธีการสร้างโปรแกรมการทำงานเชิงคณิตศาสตร์

บทที่ 5 - การแบ่งเงื่อนไข

บทนี้อธิบายถึงการแบ่งเงื่อนไข

บทที่ 6 - การจัดเก็บและการใช้งานข้อมูล

บทนี้อธิบายถึงวิธีการเขียนโปรแกรมเพื่อจัดเก็บและใช้งานข้อมูลอย่างแม่นยำ

บทที่ 7 – การใช้งานข้อมูล String

บทนี้อธิบายขั้นตอนในการใช้งานข้อมูล String

แบบทดสอบประเมินผล

คะแนนในการผ่านหลักสูตร: 60% หรือมากกว่า

บทนำ วิธีการใช้งานเครื่องมือการเรียนรู้อิเล็กทรอนิกส์นี้

ไปที่หน้าถัดไป		ไปที่หน้าถัดไป
กลับไปยังหน้าที่แล้ว		กลับไปยังหน้าที่แล้ว
เลื่อนไปยังหน้าที่ต้องการ		"สารบัญ" จะแสดงขึ้นมาช่วยนำทางไปยังหน้าที่ต้องการได้
ออกจากการเรียนรู้		ออกจากการเรียนรู้

บทนำ ข้อควรระวังในการใช้งาน

ข้อควรระวังด้านความปลอดภัย

เมื่อคุณเรียนรู้โดยมีพื้นฐานบนการใช้งานผลิตภัณฑ์จริง โปรดอ่านข้อควรระวังต่างๆ ในคู่มือการใช้งานอย่างละเอียดให้เข้าใจ

ข้อควรระวังในหลักสูตรนี้

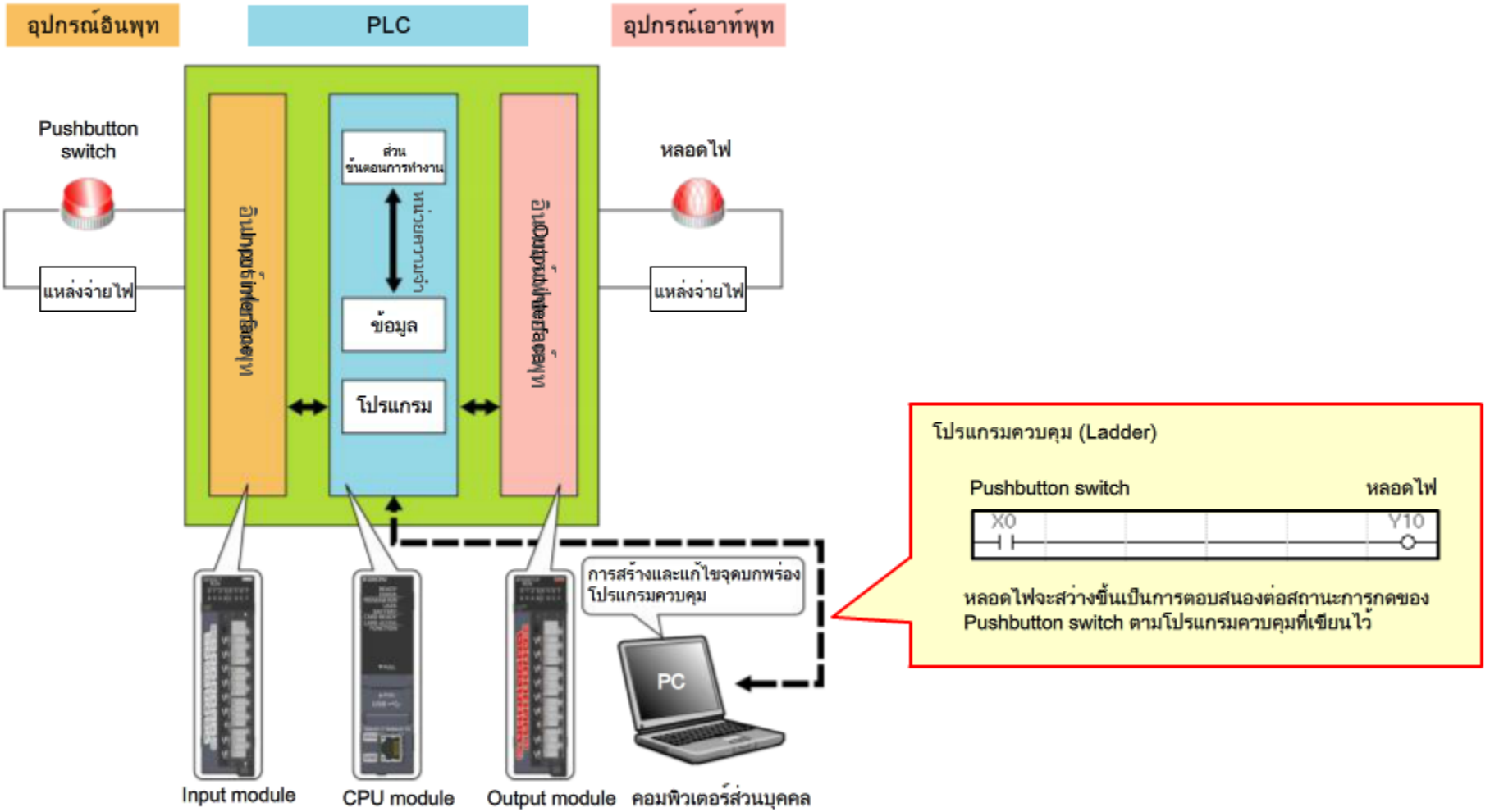
หน้าจอที่ทำการแสดงผลของ MELSOFT engineering software ที่คุณใช้อาจจะแตกต่างจากในหลักสูตรนี้
หลักสูตรนี้ใช้สัญลักษณ์Ladderของ MELSOFT GX Works3 ในการสร้างโปรแกรม

บทที่ 1 ข้อมูลทั่วไปของ structured text

บทนี้อธิบายถึงลักษณะพิเศษและการใช้งานของข้อความที่มีโครงสร้าง (Structured Text, ST) ที่เหมาะสม

1.1 โปรแกรมควบคุม

รูปภาพต่อไปนี้แสดงโครงสร้างของระบบ PLC
PLC ได้ทำงานตามโปรแกรมควบคุมที่กำหนดไว้
การทำงานของ PLC สามารถกำหนดตามที่ต้องการได้โดยการสร้างโปรแกรมควบคุม



ภาษาโปรแกรมสำหรับPLCได้ถูกกำหนดโดยมาตรฐานสากลที่ได้รับการพัฒนาโดย International Electrotechnical Commission (IEC) (คณะกรรมาธิการมาตรฐานอุปกรณ์ไฟฟ้าสากล)

1.2 ลักษณะพิเศษของ ST และการเปรียบเทียบกับภาษาโปรแกรม IEC อื่นๆ

IEC 61131 คือมาตรฐานสากลสำหรับระบบ PLC
ภาษาโปรแกรมสำหรับ PLC นั้นถูกกำหนดมาตรฐานโดย IEC 61131-3 ST ก็เป็นหนึ่งในภาษาโปรแกรม
มาตรฐานนั้น
แต่ละภาษาก็มีลักษณะพิเศษที่มีความแตกต่างกันตามการใช้งานและทักษะของผู้เขียนโปรแกรม

ตารางต่อไปนี้แสดงลักษณะพิเศษของภาษาโปรแกรมของมาตรฐาน IEC 61131-3

ภาษาโปรแกรม	ลักษณะพิเศษ
Ladder Diagram (LD)	• ใช้สัญลักษณ์หน้าสัมผัสและขดลวดในการสร้างโปรแกรมประกอบมาเป็นวงจรไฟฟ้า • การไหลของโปรแกรม ง่ายต่อการติดตามและการทำความเข้าใจ แม้แต่สำหรับผู้เริ่มต้น
Structured Text (ST)	• โปรแกรมถูกเขียนด้วยข้อความ (อักขระ) • ST นั้นง่ายต่อการเรียนรู้สำหรับผู้ที่มีประสบการณ์ในการเขียนโปรแกรมด้วยภาษา C หรือภาษา BASIC • สูตรการคำนวณต่างๆ นั้นเหมือนนิพจน์ทางคณิตศาสตร์ ซึ่งเข้าใจได้ง่าย • ST เหมาะกับการใช้งานข้อมูล
Function Block Diagram (FBD)	• โปรแกรมเขียนโดยการเรียงบล็อคที่มีฟังก์ชันต่างๆ กันและระบุความสัมพันธ์ระหว่างบล็อคต่างๆ • FBD เพิ่มความสามารถในการอ่านได้ดีกว่าภาษาอื่นเนื่องจากการทำงานทั้งหมดสามารถมองเห็นได้โดยง่าย
Sequential Function Chart (SFC)	• เงื่อนไขและการดำเนินการต่าง ๆ ถูกเขียนเป็น Flowcharts • การไหลของโปรแกรม ง่ายต่อการทำความเข้าใจ
Instruction List (IL)	• IL เหมือนกับภาษาเครื่องจักร • IL ไม่ค่อยได้ใช้แล้วในปัจจุบัน

หลักสูตรนี้ อธิบายวิธีการเขียนโปรแกรมควบคุมพื้นฐานโดยใช้ ST

1.3 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- ความสัมพันธ์ระหว่างระบบ PLCและโปรแกรมควบคุม
- มาตรฐานสากลสำหรับโปรแกรมควบคุม
- ลักษณะพิเศษของ ST

จุดสำคัญได้แก่:

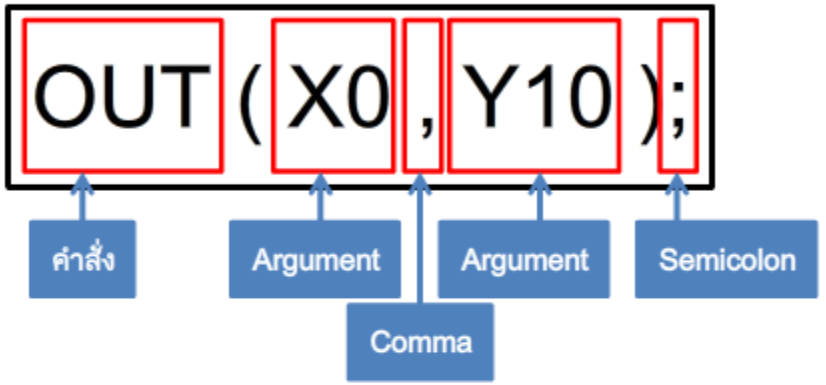
ความสัมพันธ์ระหว่างระบบ PLC และโปรแกรมควบคุม	• ตัวควบคุมแบบตั้งโปรแกรมได้ทำงานตามโปรแกรมควบคุมที่กำหนดไว้ • การทำงานของตัวควบคุมแบบตั้งโปรแกรมได้สามารถกำหนดตามที่ต้องการได้โดยการสร้างโปรแกรมควบคุม
มาตรฐานสากลสำหรับโปรแกรมควบคุม	• ST เป็นหนึ่งในภาษาโปรแกรมของ IEC • ภาษาโปรแกรม IEC อื่นๆ ได้แก่ LD FBD SFC และ IL แต่ละภาษาก็มีคุณลักษณะพิเศษแตกต่างกันตามการใช้งานและทักษะของผู้เขียนโปรแกรม
ลักษณะพิเศษของ ST	• ST นั้นงานในการเรียนรู้สำหรับผู้ที่มีประสบการณ์การเขียนโปรแกรมด้วยภาษา C หรือภาษา BASIC • การคำนวณเช่นการบวกหรือการลบนั้นสามารถเขียนได้เหมือนการเขียนนิพจน์ทางคณิตศาสตร์ปกติ ซึ่งง่ายต่อการเข้าใจ • ST เหมาะกับการใช้งานข้อมูล

บทที่ 2 กฎพื้นฐานในการเขียนโปรแกรม ST

บทนี้อธิบายถึงกฎพื้นฐานที่ใช้ในการสร้างโปรแกรมในภาษา ST

2.1 ตัวอย่างโปรแกรมพื้นฐาน (คำสั่งควบคุม I/O)

ในส่วนเป็นการแสดงตัวอย่างของโปรแกรม ST พื้นฐาน
จากโปรแกรมตัวอย่างต่อไปนี้ เอาท์พุท Y10 ON เมื่ออินพุท X0 ทำงาน และ Y10 OFF เมื่อ X0 ปิดการทำงาน



คำสั่งเป็นตัวกำหนดว่าจะดำเนินการอะไร
ส่วนArgumentนั้นจะใส่ในวงเล็บหลังจากคำสั่ง
โดยArgumentนั้นคือตัวแปร นิพจน์ทางคณิตศาสตร์ และค่าคงที่
PLC ของ Mitsubishi นั้น สามารถใช้ CPU module เป็นตัวแปรได้

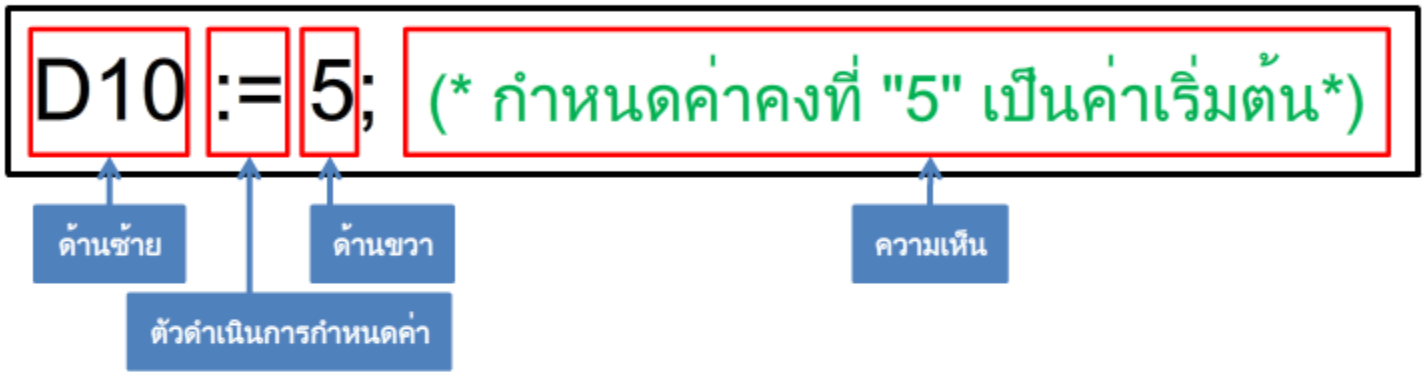
จำนวนArgumentขึ้นอยู่กับคำสั่ง
Argumentหลายตัวแบ่งโดยใช้Comma (,)

บรรทัดด้านบนแสดงคำสั่งหนึ่งคำสั่ง แต่ละคำสั่งจะสิ้นสุดด้วยSemicolon (;)
โปรแกรมหนึ่งโปรแกรมประกอบด้วยการรวมคำสั่งต่างๆ เข้าด้วยกัน

2.2 ตัวอย่างโปรแกรมพื้นฐาน (คำสั่งการกำหนดค่า)

ตัวอย่างต่อไป แสดงโปรแกรมที่ใช้การกำหนดค่า

คำสั่งต่อไปนี้ กำหนดค่าคงที่ เลขฐานสิบ "5" ไปยังตัวแปร "D10"



ตัวดำเนินการกำหนดค่า (:=) ใช้สำหรับคำสั่งการกำหนดค่า หมายเหตุ: เครื่องหมาย colon (:) จะวางทางด้านซ้ายของเครื่องหมายเท่ากับ (=) ตัวดำเนินการการกำหนดค่าจะกำหนดค่าทางด้านขวาไปยังค่าทางด้านซ้าย

การเพิ่ม comment ไปยังโปรแกรม ทำให้การทำงานเข้าได้ใจง่ายมากยิ่งขึ้น กรอก comments เข้าไประหว่างดอกจันทั้งสองด้าน (* *)

2.3 การระบุสัญลักษณ์เชิงตัวเลข

จากโปรแกรมตัวอย่างในหน้าที่ผ่านมา ได้กำหนดค่าเลขฐานสิบไปยังตัวแปร

บางครั้งจำเป็นต้องใช้ค่าที่ไม่ใช่เลขฐานสิบ เช่น เลขฐานสองและเลขฐานสิบหกในการควบคุม
ตารางต่อไปนี้แสดงประเภทของสัญลักษณ์เชิงตัวเลขที่ใช้ใน PLC ของ Mitsubishi

ประเภทของสัญลักษณ์เชิงตัวเลข	วิธีการระบุ	ตัวอย่าง
เลขฐานสอง	เพิ่ม "2#" ที่ด้านหน้า	2#11010
เลขฐานแปด	เพิ่ม "8#" ที่ด้านหน้า	8#32
เลขฐานสิบ	กรอก input โดยตรง	26
	เพิ่ม "K" ที่ด้านหน้า	K26
เลขฐานสิบหก	เพิ่ม "16#" ที่ด้านหน้า	26#1A
	เพิ่ม "H" ที่ด้านหน้า	H1A

ตัวอย่างโปรแกรมด้านล่างแสดงการกำหนดค่าไปยังตัวแปร

```
D10 := 8#32;
D10 := K26;
D10 := H1A;
```

2.3.1 การระบุสัญลักษณ์ bit

Bitsแสดงถึงสถานะ true/false เช่น สถานะ on/off ของสัญญาณ อีกทั้งยังแสดงถึงสถานะการมีอยู่/การไม่มีอยู่ อีกด้วย
ในภาษา ST Bits ไม่สามารถเขียนเป็น "ON" และ "OFF" ได้ แต่ต้องเขียนเป็น "1" (ON) และ "0" (OFF) แทน และยังสามารถเขียนเป็น "TRUE" และ "FALSE" ได้อีกด้วย

ตารางต่อไปนี้แสดงประเภทต่างๆ ของการระบุสัญลักษณ์

สถานะ	ON	OFF
	True	False
สัญลักษณ์ตัวเลข	1	0
สัญลักษณ์ True/false	TRUE	FALSE

ตัวอย่างต่อไปนี้ แสดงการกำหนดค่าไปยังตัวแปรประเภท bit

สัญลักษณ์ตัวเลข

```
X0 := 1;
```

=

สัญลักษณ์ True/false

```
X0 := TRUE;
```

สัญลักษณ์ตัวเลข

```
X0 := 0;
```

=

สัญลักษณ์ True/false

```
X0 := FALSE;
```

2.4 ลำดับการดำเนินการโปรแกรม

คำสั่ง ST ดำเนินการจากบนลงล่าง

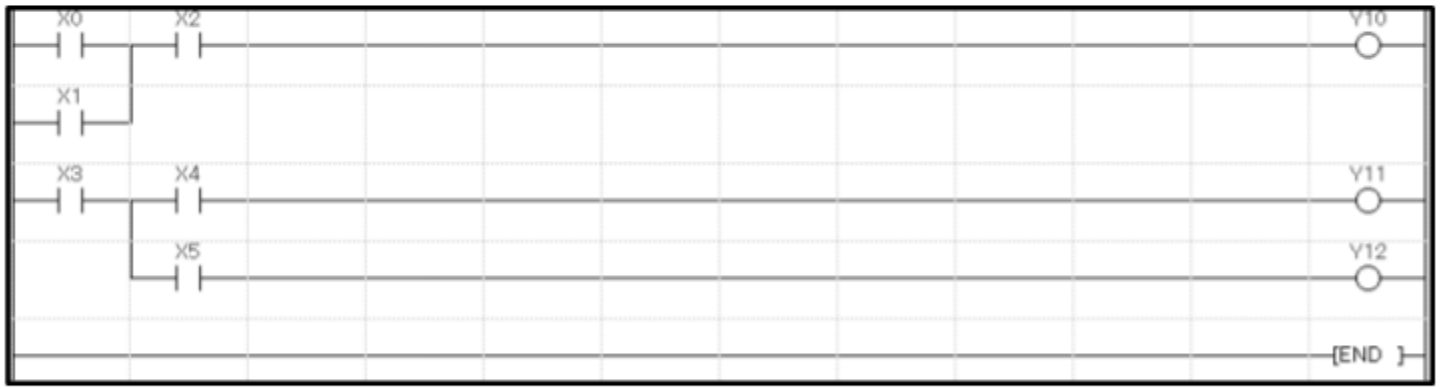
ตัวอย่างโปรแกรม ST

```
Y10 := (X0 OR X1) AND X2;        (* ดำเนินการลำดับแรก *)
Y11 := X3 AND X4;                (* ดำเนินการลำดับที่สอง *)
Y12 := X3 AND X5;    (* ดำเนินการลำดับที่สาม ไม่ต้องมีคำสั่ง END ในการสิ้นสุด*)
```

ดำเนินการซ้ำ

*แม้ว่าคำสั่ง END จะจำเป็นในการสิ้นสุดโปรแกรมในภาษา LD แต่ไม่เป็นจำเป็นในภาษา ST

โปรแกรมLadderต่อไปนี้ แสดงการทำงานแบบเดียวกับตัวอย่างโปรแกรม ST ด้านบน



ดำเนินการซ้ำ

คำสั่งในภาษา ST จะดำเนินการโดยกลับไปยังคำสั่งแรกหลังจากดำเนินการมาถึงคำสั่งสุดท้ายเหมือนภาษา LD

2.5 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- โปรแกรม ST เบื้องต้น
- รูปแบบคำสั่งการกำหนดค่า
- การระบุสัญลักษณ์เชิงตัวเลข
- ลำดับการดำเนินการของโปรแกรม
- ความเห็น

จุดสำคัญได้แก่:

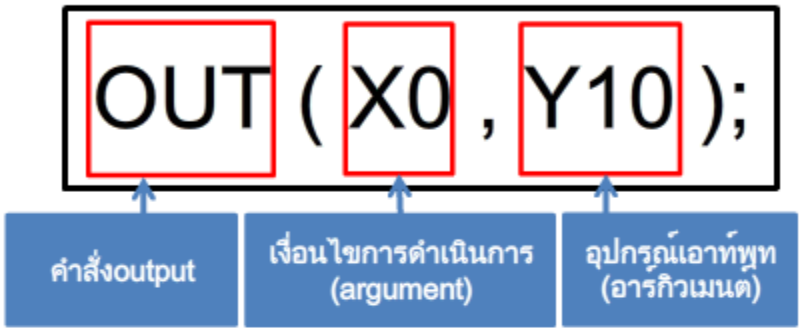
โปรแกรม ST เบื้องต้น	• คำสั่ง คือ องค์ประกอบที่เล็กที่สุดของโปรแกรมในภาษา ST • แต่ละคำสั่งจะสิ้นสุดด้วยเซมิโคลอน (;) • โปรแกรมหนึ่งโปรแกรมประกอบด้วยการรวมคำสั่งต่างๆ เข้าด้วยกัน
รูปแบบคำสั่งการกำหนดค่า	• ตัวดำเนินการกำหนดค่า (:=) ใช้สำหรับการกำหนดค่า
การระบุสัญลักษณ์เชิงตัวเลข	• ประเภทของสัญลักษณ์เชิงตัวเลขในภาษา ST • "1" และ "0" ใช้ในการแสดงค่า bitในภาษา ST แทนสัญลักษณ์ "ON" และ "OFF" • ค่าbitสามารถเขียนเป็น "TRUE" และ "FALSE" ได้ในภาษา ST
ลำดับการดำเนินการโปรแกรม	• โปรแกรมที่สร้างในภาษา ST จะถูกดำเนินการเป็นลำดับจากบนลงล่าง • โปรแกรมในภาษา ST จะดำเนินการคำสั่งซ้ำโดยการย้อนกลับมาที่คำสั่งเริ่มต้นของโปรแกรม เมื่อดำเนินการไปถึงคำสั่งสุดท้าย
ความเห็น	• การเพิ่มความเห็นไปยังโปรแกรม ทำให้การทำงานเข้าได้ใจง่ายมากยิ่งขึ้น • Comment จะอยู่ระหว่างดอกจันทั้งสองด้าน (* *)

บทที่ 3 การสร้างโปรแกรมควบคุม I/O

บทนี้อธิบายถึงวิธีการสร้างโปรแกรมควบคุม I/O ใน ภาษา ST

3.1 โปรแกรมควบคุม I/O

ตัวอย่างต่อไปนี้ เป็นโปรแกรมควบคุม I/O ของPLC



"OUT" คือคำสั่งoutput ส่วน argument จะกำหนดเงื่อนไขการดำเนินการและอุปกรณ์ที่จะส่งoutput ออกไป เมื่อเงื่อนไขการดำเนินการ X0 ถูกต้อง จะเป็นการสั่งให้อุปกรณ์ Y10 ทำงาน

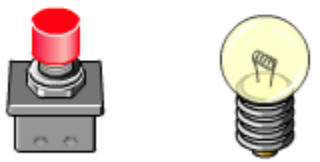
คลิก input switch ที่แสดงด้านล่าง จะเป็นการ ON switch X0

- เมื่อ ON switch X0 หลอดไฟ Y10 จะทำงาน
- เมื่อ OFF switch X0 หลอดไฟ Y10 ก็จะหยุดการทำงานด้วย

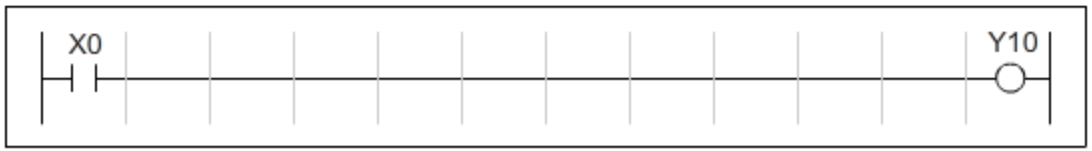
ตัวอย่างโปรแกรมควบคุม I/O ในภาษา ST

```
OUT(X0,Y10);
```

สวิตช์อินพุท X0 หลอดไฟเอาท์พุท Y10



โปรแกรมเดียวกับที่เขียนในภาษา LD



3.1 โปรแกรมควบคุม I/O

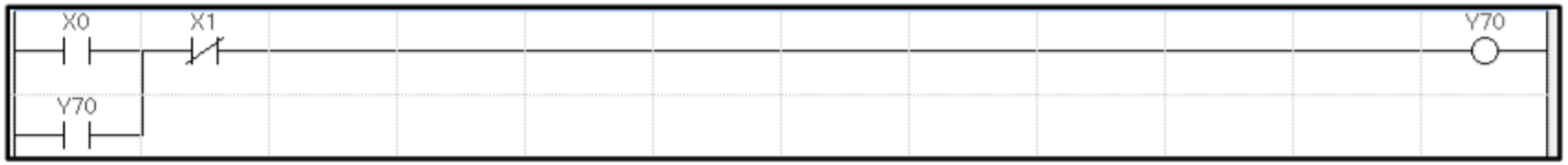
ยังมีคำสั่งอื่นๆ นอกเหนือจากคำสั่ง OUT เช่น คำสั่งควบคุม I/O และคำสั่งประมวลผลข้อมูล เช่นเดียวกับภาษา LD โปรดอ้างอิงตามคู่มือการใช้งานการเขียนโปรแกรมของตัวควบคุมแบบตั้งโปรแกรมได้ของคุณสำหรับข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งที่ใช้งานได้ในภาษา ST

หมายเหตุ: การเขียน "OUT(X0, Y10);" นั้นให้ผลลัพธ์เช่นเดียวกันกับ "Y10 := X0;" writing "OUT(X0, Y10);"

```
Y10 := X0;  (* ผลลัพธ์เช่นเดียวกันกับ "OUT(X0, Y10);" *)
```

3.2 การรวมเงื่อนไขหลายเงื่อนไข

โปรแกรมแลดเดอร์ที่แสดงดังต่อไปนี้วงจรเก็บค่าได้



โปรแกรมดังกล่าว สามารถเขียนในภาษา ST ได้ดังต่อไปนี้

```
Y70 := (X0 OR Y70) AND NOT X1;
```

ตัวดำเนินการเชิงตรรกะ

ตัวดำเนินการเชิงตรรกะ ใช้ในการรวมเงื่อนไขหลายเงื่อนไขในภาษา ST

ตารางต่อไปนี้ แสดงรายการของตัวดำเนินการเชิงตรรกะ

ตัวดำเนินการ	ความหมาย
OR	หรือ
AND	และ
NOT	นิเสธ
XOR	เอกซ์คลูซีฟ หรือ

3.3 การกำหนดนิยามตัวแปร

การใช้งานภาษา ST ของ MELSEC PLC นั้น ทั้งอุปกรณ์และลาเบลสามารถกำหนดชื่อเป็นตัวแปรได้
ผู้ใช้สามารถใช้งานลาเบลตามการใช้งานได้

เมื่อกำหนดลาเบลให้เกี่ยวข้องกับการใช้งาน การทำงานต่างๆ จะสามารถเข้าใจได้โดยง่าย

```
Y10 := (X0 OR X1) AND X2;     (* เขียน โดยใช้ ชื่อ อุปกรณ์ *)
```

```
Lamp := (Switch0 OR Switch1) AND Switch2;     (* เขียนโดยใช้ลาเบล *)
```

สามารถตั้งชื่อลาเบลได้ซอฟต์แวร์ MELSOFT

ตัวอย่างโปรแกรมหลังจากนี้ ในหลักสูตรจะอธิบายโดยใช้ลาเบล

3.4 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

ตัวอย่างโปรแกรมควบคุม I/O

- การทำงานตามลอจิกสามารถทำงานร่วมกับ ST ได้หลายเงื่อนไขพร้อมกัน
- ชื่ออุปกรณ์และลาเบล สามารถใช้เป็นชื่อตัวแปรได้

จุดสำคัญได้แก่:

การรวมเงื่อนไขหลายเงื่อนไข	• การทำงานตามลอจิกสามารถทำงานร่วมกับ ST ได้หลายเงื่อนไขพร้อมกัน
การกำหนดนิยามตัวแปร	• เมื่อกำหนดลาเบลให้เกี่ยวข้องกับการใช้งาน การทำงานต่างๆ จะสามารถเข้าใจได้โดยง่าย

บทที่ 4 การดำเนินการเชิงคณิตศาสตร์

บทนี้อธิบายถึงวิธีการสร้างโปรแกรมการทำงานเชิงคณิตศาสตร์

- การอธิบายการดำเนินการเชิงคณิตศาสตร์
- การกำหนดประเภทข้อมูลตามช่วงจำนวน
- การกำหนดชื่อตัวแปรเพื่อหลีกเลี่ยงความไม่สอดคล้องกันของประเภทข้อมูล

4.1 การดำเนินการเชิงคณิตศาสตร์พื้นฐาน

4.2 ประเภทข้อมูลของตัวแปร

4.3 ชื่อตัวแปรที่แสดงถึงประเภทข้อมูล

4.1 การดำเนินการเชิงคณิตศาสตร์พื้นฐาน

โปรแกรมตัวอย่างต่อไปนี้รวมปริมาณการผลิตของสายการผลิตสองสาย
ด้านขวาของสมการเป็นการดำเนินการทางคณิตศาสตร์ประกอบด้วยตัวแปรและตัวดำเนินการทางคณิตศาสตร์

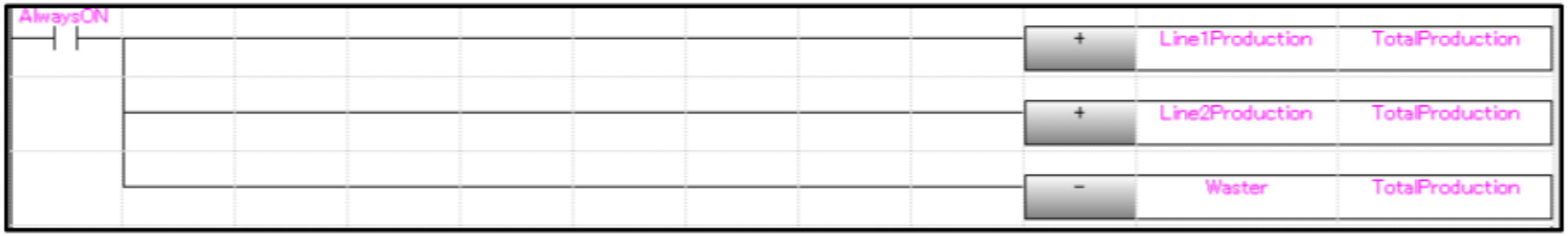
ตัวอย่างโปรแกรมเชิงคณิตศาสตร์ในภาษา ST

เครื่องหมายบวก

เครื่องหมายลบ

```
TotalProduction := Line1Production + Line2Production - Waster;
(* ผลรวมปริมาณการผลิตของสายการผลิตสองสาย ลบด้วยจำนวนของผลิตภัณฑ์ที่มี
ข้อบกพร่องจากการรวม และกำหนดค่าที่ได้รับ *)
```

โปรแกรมดังกล่าว สามารถเขียนได้ในภาษา LD ดังแสดงด้านล่าง



โปรแกรมนี้ต้องใช้ถึง 3 บรรทัดในการเขียนด้วยแลดเดอร์ แต่ใช้เพียง 1 บรรทัดในภาษา ST ดังแสดงด้านบน

ตารางต่อไปนี้ แสดงรายการตัวดำเนินการทางคณิตศาสตร์พื้นฐาน

ตัวดำเนินการ	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร

4.2 ประเภทข้อมูลของตัวแปร

ประเภทข้อมูลจะต้องกำหนดให้แต่ละตัวแปรในการระบุช่วงของค่าในการใช้งาน
ประเภทข้อมูลตัวเลขที่ใช้งานในภาษา ST ได้แก่ บิต จำนวนเต็ม และจำนวนจริง

ภาษา ST มีประเภทข้อมูลหลายชนิด ตารางด้านล่างแสดงรายการประเภทข้อมูลที่ใช้งานในหลักสูตรนี้

ประเภทข้อมูล		ช่วงข้อมูล
บิต		สถานะ ON/OFF (เปิด/ปิด) ของอุปกรณ์บิตและสถานะ true/false (จริง/เท็จ) ของผลลัพธ์การดำเนินการ
จำนวนเต็ม	เวิร์ด (ไม่มีเครื่องหมาย)	0 - 65,535
	เวิร์ด (มีเครื่องหมาย)	-32,768 - 32,767
	ดับเบิล (ไม่มีเครื่องหมาย)	0 - 4,294,967,295
	ดับเบิล (มีเครื่องหมาย)	-2,147,483,648 - 2,147,483,647

เมื่อใช้ตัวแปรประเภทจำนวนเต็ม โปรดเลือกประเภทเป็นเวิร์ดหรือดับเบิลตามช่วงของข้อมูลที่ต้องการใช้ และเลือกว่าจะมีเครื่องหมายหรือไม่ตามความจำเป็นในการใช้งานจำนวนติดลบ
ระบุประเภทขอมูลของตัวแปรขณะตั้งชื่อลาเบลโดยใชซอฟต์แวร์ MELSOFT

4.3 ชื่อตัวแปรที่แสดงถึงประเภทข้อมูล

การใช้งานประเภทข้อมูลต่างกันทางด้านซ้ายและขวาของสมการ อาจก่อให้เกิดข้อผิดพลาดหรือผลลัพธ์ที่ไม่คาดคิดได้ ด้านล่างเป็นตัวอย่างของกรณีดังกล่าว

```
ValueA := ValueB;      (* ValueA: จำนวนเต็มประเภทเวิร์ด  ValueB: จำนวนเต็มประเภทดับเบิล *)
```

จำนวนเต็มประเภทดับเบิล ไม่สามารถกำหนดค่าไปยังจำนวนเต็มประเภทเวิร์ดได้ อย่างไรก็ตาม ในกรณีนี้ประเภทข้อมูลจะไม่สามารถกำหนดได้

การเพิ่มตัวอักษรด้านหน้าชื่อตัวแปรเพื่อระบุประเภทข้อมูลจะทำให้ระบุประเภทข้อมูลได้ง่ายยิ่งขึ้น วิธีการระบุชื่อตัวแปรแบบนี้เรียกว่าการระบุสัญลักษณ์แบบฮังกาเรียน

ประเภทข้อมูล		ช่วงข้อมูล	ตัวอักษรด้านหน้า	ค่าเต็มของตัวอักษรด้านหน้า
บิต		สถานะ ON/OFF (เปิด/ปิด) ของอุปกรณ์บิต และสถานะ true/false (จริง/เท็จ) ของผลลัพธ์การดำเนินการ	b	Bit (บิต)
จำนวนเต็ม	เวิร์ด (ไม่มีเครื่องหมาย)	0 - 65,535	u	unsigned word (เวิร์ดแบบไม่มีเครื่องหมาย)
	เวิร์ด (มีเครื่องหมาย)	-32,768 - 32,767	w	signed word (เวิร์ดแบบมีเครื่องหมาย)
	ดับเบิล (ไม่มีเครื่องหมาย)	0 - 4,294,967,295	ud	unsigned double-word (ดับเบิลแบบไม่มีเครื่องหมาย)
	ดับเบิล (มีเครื่องหมาย)	-2,147,483,648 - 2,147,483,647	d	signed double-word (ดับเบิลแบบมีเครื่องหมาย)

โปรแกรมตัวอย่างด้านบนสามารถเขียนในรูปแบบฮังกาเรียนได้ดังต่อไปนี้:

```
wValueA := dValueB;  (*ตัวแปรดับเบิลไม่สามารถกำหนดค่าไปยังตัวแปรเวิร์ดได้ *)
```

การใช้รูปแบบฮังกาเรียนนั้นทำให้สามารถระบุความไม่สอดคล้องกันของประเภทข้อมูลได้ระหว่างการเขียนโปรแกรม

ตั้งแต่ส่วนนี้เป็นต้นไป การระบุชื่อตัวแปรในหลักสูตรนี้จะเขียนในรูปแบบฮังกาเรียน

4.4 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- การอธิบายการดำเนินการเชิงคณิตศาสตร์
- การกำหนดประเภทข้อมูลตามช่วงจำนวน
- การเพิ่มชื่อตัวแปรที่ระบุประเภทข้อมูลได้

จุดสำคัญได้แก่:

การดำเนินการเชิงคณิตศาสตร์พื้นฐาน	• ตัวดำเนินการที่ใช้ในภาษาโปรแกรมทั่วไปสามารถใช้ในภาษา ST ในการคำนวณได้
ประเภทข้อมูลของตัวแปร	• ประเภทข้อมูลจะต้องกำหนดให้แต่ละตัวแปรในการระบุช่วงของค่าในการใช้งาน
การเพิ่มชื่อตัวแปรที่ระบุประเภทข้อมูลได้	• การระบุชื่อตัวแปรแบบฮังกาเรียน ทำให้ สามารถระบุความไม่สอดคล้องกันของประเภทข้อมูลได้ระหว่างการเขียนโปรแกรม

บทที่ 5 การแยกเงื่อนไข

โปรแกรมควบคุมจะมีส่วนของโค้ดซึ่งจะเปลี่ยนรูปแบบการทำงานตามเงื่อนไขที่กำหนด
บทนี้อธิบายถึงการแตกสาขาแบบมีเงื่อนไข

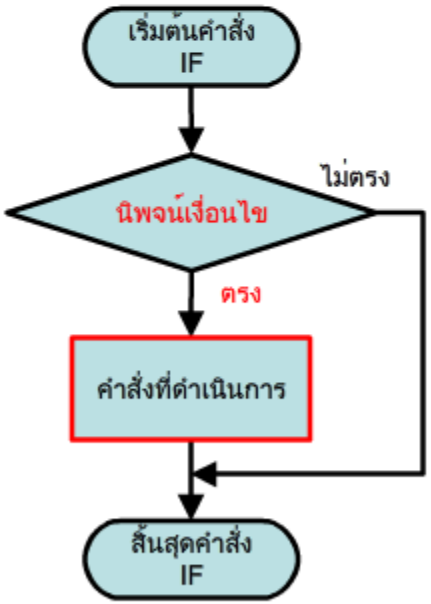
5.1 การแยกเงื่อนไข (IF)

คำสั่ง IF ใช้ในการแยกเงื่อนไข วิธีการใช้งานคำสั่ง IF เป็นดังนี้

```
IF conditional expression THEN
Execution statement;                (* คำสั่งจะดำเนินการหากนิพจน์เงื่อนไขถูกต้อง *)
END_IF;                             (* END_IF; ต้องถูกวางในการสิ้นสุดคำสั่ง IF *)
```

ในโปรแกรมตัวอย่างนี้คำสั่งจะทำงานเมื่อเงื่อนไขถูกต้อง แต่หากเงื่อนไขไม่ถูกต้องคำสั่งจะไม่ทำงาน

รูปภาพต่อไปนี้แสดงขั้นตอนการทำงาน
ในโปรแกรมตัวอย่าง

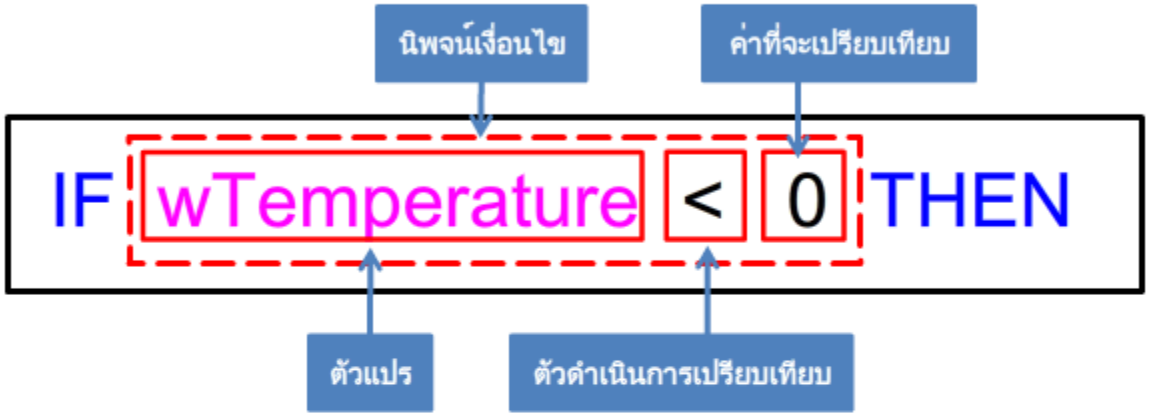


ตัวอย่างต่อไปนี้แสดงการแยกของโปรแกรมโดยการเปรียบเทียบค่าของตัวแปร
ในโปรแกรมตัวอย่าง เครื่องทำความร้อนจะเปิดเมื่ออุณหภูมิที่แผงควบคุมต่ำกว่า 0 องศา

```
IF wTemperature < 0 THEN
  bHeater := 1; (* เครื่องทำความร้อนจะเปิดเมื่ออุณหภูมิในแผงควบคุมต่ำกว่า 0 องศา *)
END_IF;
```

5.1.1 การเขียนนิพจน์เงื่อนไข

หน้าที่ผ่านมานั้น ใช้นิพจน์เงื่อนไข "wTemperature < 0" ซึ่งหมายถึง "เมื่อค่าของตัวแปร wTemperature น้อยกว่า 0"
นิพจน์แบบมีเงื่อนไขใช้ตัวดำเนินการเปรียบเทียบในการแสดงความสัมพันธ์ระหว่างตัวแปรและค่าต่างๆ ในการเปรียบเทียบ



ที่ด้านซ้ายและขวามือของตัวดำเนินการเปรียบเทียบ ค่าจะเขียนเป็นตัวแปรหรือค่าคงที่ในการเปรียบเทียบ

นอกจากจะใช้ในการเปรียบเทียบตัวแปรและค่าคงที่แล้ว นิพจน์เงื่อนไขยังสามารถใช้เปรียบเทียบผลลัพธ์ของการดำเนินการเชิงตรรกะหรือตัวแปรประเภทบิตได้อีกด้วย

เปรียบเทียบตัวแปร

- uValue1 <= uValue2

การประมวลผลทางลอจิกสำหรับเปรียบเทียบสองผลลัพธ์

- (10 < uValue) AND (uValue <= 50)

การประมวลผลทางลอจิกสำหรับตัวแปรประเภทบิตสองตัวแปร

- bSwitch0 OR bSwitch1

ตารางต่อไปนี้แสดงรายการ ประเภทของตัวดำเนินการเปรียบเทียบ

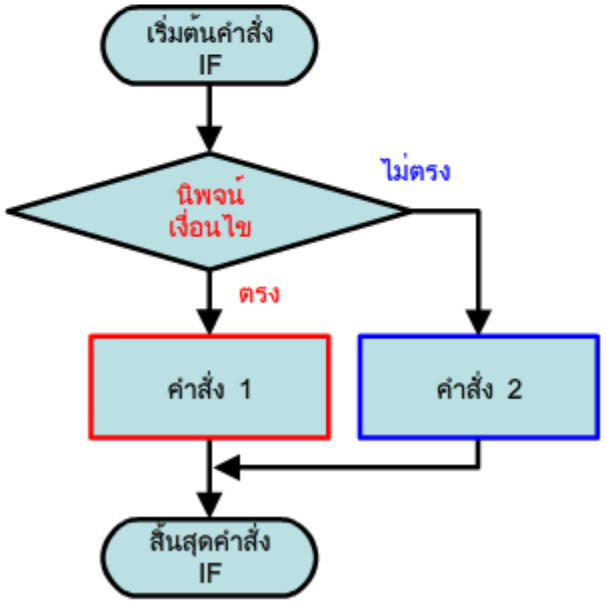
ตัวดำเนินการ	ความหมาย
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
=	เท่ากับ
<>	ไม่เท่ากับ

5.1.2 การแยกข้อยกเว้นสำหรับคำสั่ง IF (ELSE)

คำสั่ง IF โดยทั่วไปแล้ว (โปรดดู 5.1) ใช้ในการดำเนินการคำสั่งเมื่อนิพจน์เงื่อนไขถูกต้อง
ในการดำเนินการคำสั่งอื่นๆ เมื่อนิพจน์เงื่อนไขไม่ถูกต้อง จะใช้คำสั่ง ELSE

```
IF conditional expression THEN
  Execution statement 1;          (* คำสั่ง 1 จะดำเนินการหากนิพจน์เงื่อนไขถูกต้อง *)
ELSE
  Execution statement 2;          (* คำสั่ง 2 จะดำเนินการหากนิพจน์เงื่อนไขไม่ถูกต้อง *)
END_IF;
```

รูปภาพต่อไปนี้แสดงขั้นตอนการทำงาน
เมื่อใช้งานคำสั่ง ELSE



โปรแกรมตัวอย่างต่อไปนี้ ดำเนินการคำสั่งที่แตกต่างกันขึ้นอยู่กับว่าเงื่อนไขถูกต้องหรือไม่

โปรแกรมตัวอย่างใน 5.1 นั้นมีข้อเสียคือเครื่องทำความร้อนจะยังคงเพิ่มอุณหภูมิไปเรื่อยๆ แม้ว่าจะเกิน 0 องศาไปแล้วก็ตาม โปรแกรมต่อไปนี้จะปิดเครื่องทำความร้อนเมื่อ "wTemperature" มีค่ามากกว่า 0 องศา

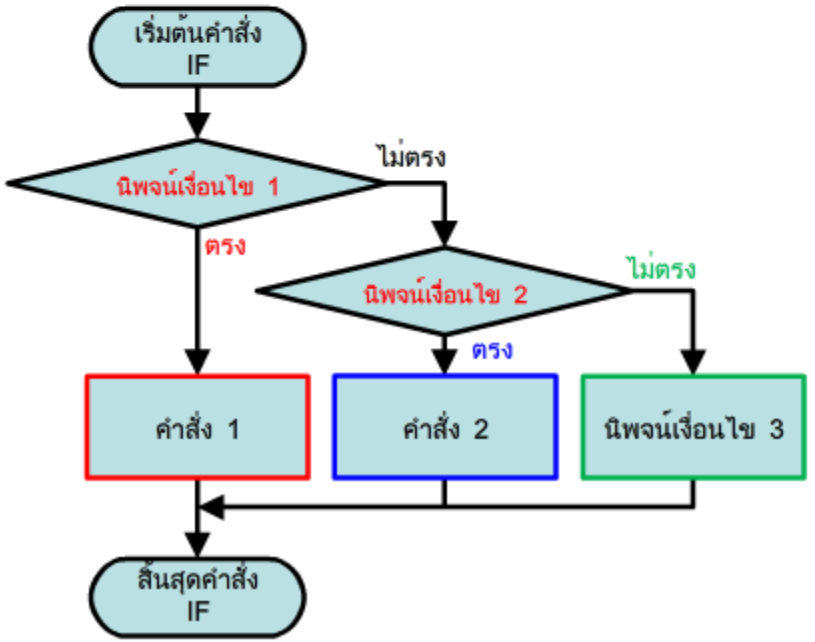
```
IF wTemperature < 0 THEN
  bHeater := 1;  (* เปิดเครื่องทำความร้อนเมื่ออุณหภูมิต่ำกว่า  0 องศา *)
  ELSE
  bHeater := 0;  (* ปิดเครื่องทำความร้อนเมื่ออุณหภูมิเท่ากับหรือสูงกว่า 0 องศา *)
END_IF;
```

5.1.3 การแยกเงื่อนไขเพิ่มเติมสำหรับคำสั่ง IF (ELSIF)

คำสั่ง ELSE ใช้ในการดำเนินการคำสั่งอื่นเมื่อนิพจน์เงื่อนไขไม่ถูกต้อง
โดยจะสามารถเพิ่มเงื่อนไขได้อีกโดยใช้คำสั่ง ELSIF ซึ่งหมายถึงหากนิพจน์เงื่อนไขก่อนหน้าไม่ถูกต้อง จะตรวจสอบนิพจน์เงื่อนไขต่อไป

```
IF Conditional expression 1 THEN
    Execution statement 1;      (* คำสั่ง 1 จะดำเนินการหากนิพจน์เงื่อนไข 1 ถูกต้อง *)
ELSIF Conditional expression 2 THEN
    Execution statement 2;      (* คำสั่ง 2 จะดำเนินการหากนิพจน์เงื่อนไข 1 ไม่ถูกต้องและนิพจน์เงื่อนไข 2 ถูกต้อง *)
ELSE
    Execution statement 3;      (* คำสั่ง 3 จะดำเนินการหากนิพจน์เงื่อนไข 1 และ 2 ไม่ถูกต้อง *)
END_IF;
```

รูปภาพต่อไปนี้แสดงขั้นตอนการทำงานเมื่อใช้งาน
คำสั่ง ELSEIF



คำสั่ง ELSIF ได้เพิ่มไปยังตัวอย่างโปรแกรมดังแสดงใน 5.1.2 เพื่อให้ครอบคลุม
กับกรณีที่อุณหภูมิสูงกว่า 40 องศา

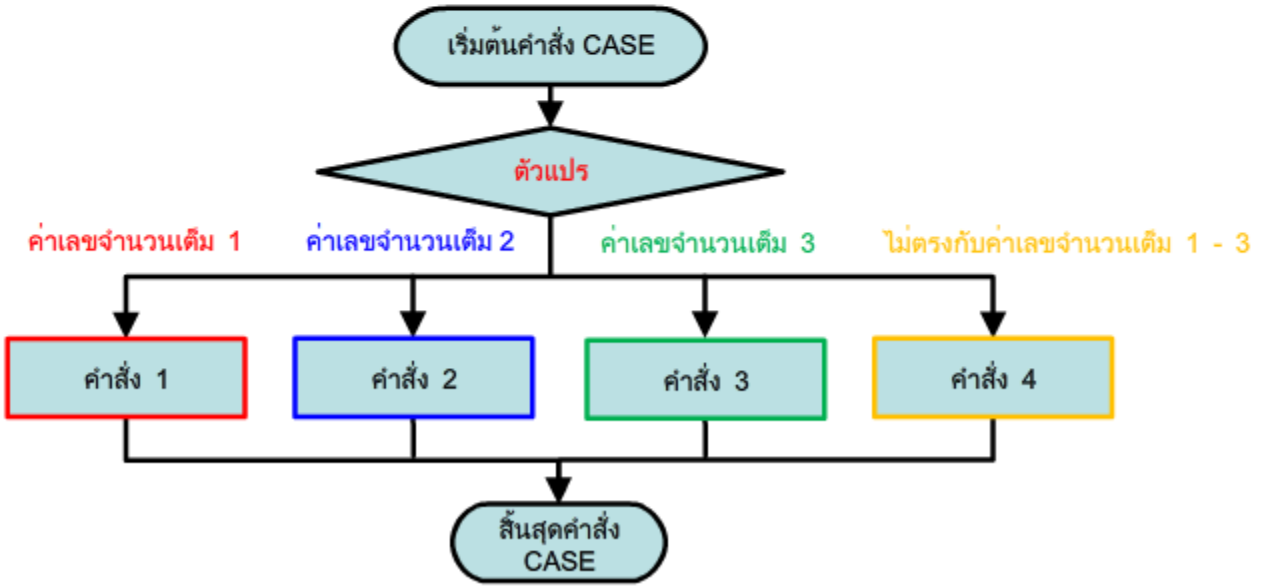
```
IF wTemperature < 0 THEN
   bHeater := 1; (* เปิดเครื่องทำความร้อนเมื่ออุณหภูมิต่ำกว่า  0 องศา *)
   bCooler := 0; (*  ปิดเครื่องทำความเย็นเมื่ออุณหภูมิต่ำกว่า 0 องศา *)
   ELSIF 40 < wTemperature THEN
   bHeater := 0; (* ปิดเครื่องทำความร้อนเมื่ออุณหภูมิสูงกว่า 40 องศา *)
   bCooler := 1; (* เปิดเครื่องทำความเย็นเมื่ออุณหภูมิสูงกว่า 40 องศา *)
   ELSE
   bHeater := 0; (*  ปิดเครื่องทำความร้อนหากไม่มีเงื่อนไขก่อนหน้าถูกต้อง *)
   bCooler := 0; (*  ปิดเครื่องทำความเย็นหากไม่มีเงื่อนไขก่อนหน้าถูกต้อง *)
END_IF;
```

5.2 การแยกเงื่อนไขตามค่าเลขจำนวนเต็ม (CASE)

คำสั่ง IF ใช้ในการแยกเงื่อนไขขึ้นอยู่กับว่านิพจน์เงื่อนไขนั้นถูกต้องหรือไม่
คำสั่ง CASE จะใช้ในการแยกเงื่อนไขตามค่าเลขจำนวนเต็ม
รูปภาพต่อไปนี้แสดงวิธีการใช้งานคำสั่ง CASE

```
CASE Variable OF
    Integer value 1: Execution statement 1;     (* คำสั่ง 1 จะดำเนินการเมื่อตัวแปรตรงกับค่าจำนวนเต็ม 1 *)
    Integer value 2: Execution statement 2;     (* คำสั่ง 2 จะดำเนินการเมื่อตัวแปรตรงกับค่าจำนวนเต็ม 2 *)
    Integer value 3: Execution statement 3;     (* คำสั่ง 3 จะดำเนินการเมื่อตัวแปรตรงกับค่าจำนวนเต็ม 3 *)
    ELSE Execution statement 4;                 (* คำสั่ง 4 จะดำเนินการหาก ตัวแปรไม่ตรงกับค่าจำนวนเต็ม r ใดๆ *)
END_CASE;                                       (* "END_CASE;" ต้องถูกวางในการสิ้นสุดคำสั่ง CASE *)
```

รูปภาพต่อไปนี้แสดงขั้นตอนการทำงานเมื่อใช้งานคำสั่ง CASE



5.2.1 โปรแกรมตัวอย่างสำหรับคำสั่ง CASE

การดำเนินการคำสั่ง CASE นั้นอธิบายได้โดยการดำเนินการโปรแกรมตัวอย่าง

คลิกที่ เพื่อไปยังหน้าถัดไป
ดูภาพเคลื่อนไหวอีกครั้งด้วยการคลิกที่ปุ่ม "Play"

Play


```
CASE wWeight OF
  0..20:    uSize := 1;
  21..30:   uSize := 2;
  31..40:   uSize := 3;
  ELSE      uSize := 4;
END_CASE;
```

น้ำหนัก	uSize	ไซส์
0 ถึง 20 กก.	1	M
21 ถึง 30 กก.	2	L
31 ถึง 40 กก.	3	XL
41 กก. และมากกว่า	4	Oversize

5.3 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- การแยกเงื่อนไขด้วยคำสั่ง IF
- การเขียนการแสดงเงื่อนไข
- การแยกเงื่อนไขตามค่าเลขจำนวนเต็ม (คำสั่ง CASE)

จุดสำคัญได้แก่:

คำสั่ง IF	• โปรแกรมจะถูกแยกเงื่อนไขด้วยคำสั่ง IF เมื่อการแสดงเงื่อนไขถูกต้อง • คำสั่ง ELSE ใช้ในการแยกเงื่อนไขเมื่อการแสดงเงื่อนไขไม่ถูกต้อง • คำสั่ง ELSIF ใช้ในการเพิ่มการแยกเงื่อนไขเมื่อการแสดงเงื่อนไขในคำสั่ง IF ไม่ถูกต้อง
การแสดงเงื่อนไข	• การแสดงเงื่อนไขแสดงความสัมพันธ์ระหว่างตัวแปรและค่าในการเปรียบเทียบโดยใช้ตัวดำเนินการเปรียบเทียบ
คำสั่ง CASE	• คำสั่ง CASE จะใช้ในการแยกเงื่อนไขตามค่าเลขจำนวนเต็ม

บทที่ 6 การจัดเก็บและการใช้งานข้อมูล

ตัวควบคุมแบบตั้งโปรแกรมได้นั้น นอกจากจะใช้ในงานควบคุม I/O ก็ใช้ในการประมวลผลข้อมูลปริมาณมากเป็นแกนของระบบการผลิต
ในการประมวลผลข้อมูลปริมาณมาก ข้อมูลจำเป็นจะต้องถูกเก็บและนำมาอ่าน
บทนี้จะอธิบายถึงวิธีการเขียนโปรแกรมเพื่อจัดเก็บและประมวลผลข้อมูลอย่างแม่นยำ

- Arrays ใช้ในการจัดลำดับและจัดการตัวแปร
- โครงสร้างข้อมูลใช้ในการจัดการตัวแปรที่เกี่ยวข้อง
- การใช้งานลูปทำให้ประมวลผล Arrays ได้อย่างมีประสิทธิภาพมากขึ้นโดยใช้คำสั่ง FOR

โปรแกรมที่จัดเก็บและจัดการข้อมูลได้อย่างแม่นยำสามารถทำได้โดยใช้ Arrays โครงสร้างข้อมูล และคำสั่ง FOR

6.1 การจัดลำดับและการจัดเก็บข้อมูล (Array)

ค่าหลายค่าสามารถจัดการได้ในตัวแปรเดียวโดยใช้ Arrays
ในตัวอย่างต่อไปนี้ ข้อมูลปริมาณการผลิตของโรงงานผลิตรถยนต์ถูกเก็บค่าไว้ตามประเทศเป้าหมาย

เป้าหมาย	ประเทศ A	ประเทศ B	ประเทศ C
ปริมาณการผลิต	35	75	65

ปริมาณการผลิตตามประเทศเป้าหมายถูกกำหนดไปยังตัวแปรหนึ่งตัว หากไม่ได้ใช้ Arrays จะต้องสร้างตัวแปรหนึ่งตัวสำหรับแต่ละเป้าหมาย แต่หากใช้ Arrays ปริมาณการผลิตสำหรับหลายเป้าหมายสามารถกำหนดและเก็บไว้ในตัวแปรเดียว

ตัวแปรแต่ละตัวใน Arrays จะระบุได้โดยใช้หมายเลของค์ประกอบ หมายเลของค์ประกอบเริ่มจากศูนย์ [0]

ในตัวอย่างโปรแกรมต่อไปนี้เป็นการกำหนดตัวแปรของปริมาณการผลิตที่วางแผนไว้สำหรับประเทศ A

```
uShowProductionPlan := uProduction[0];
(* กำหนดจำนวนองค์ประกอบสำหรับประเทศ A *)
```

6.1.1 Arrays เมทริกซ์

นอกจากนี้ยังได้เพิ่มข้อมูลสีรถนอกเหนือจากข้อมูลประเทศเป้าหมาย

เป้าหมาย									
สีรถ									
ปริมาณการผลิต	10	5	20	15	40	20	25	30	10
	รวม 35			รวม 75			รวม 65		

ข้อมูลสามารถแบ่งและเก็บไว้ตามสีรถ (column) และตามประเทศเป้าหมาย (แถว) ได้ ดังแสดงในตารางต่อไปนี้

สีรถ (column)

เป้าหมาย (แถว)	แดง	เหลือง	น้ำเงิน
ประเทศ A	[0,0] 10	[0,1] 5	[0,2] 20
ประเทศ B	[1,0] 15	[1,1] 40	[1,2] 20
ประเทศ C	[2,0] 25	[2,1] 30	[2,2] 10

หมายเลของค์ประกอบแสดงประเทศเป้าหมาย

หมายเลของค์ประกอบแสดงสีรถ

Array ที่จัดการข้อมูลเป็นแถวและcolumnแบบนี้เรียกว่า matrix
หมายเลของค์ประกอบที่แสดงแถวและcolumnถูกแบ่งโดย commas

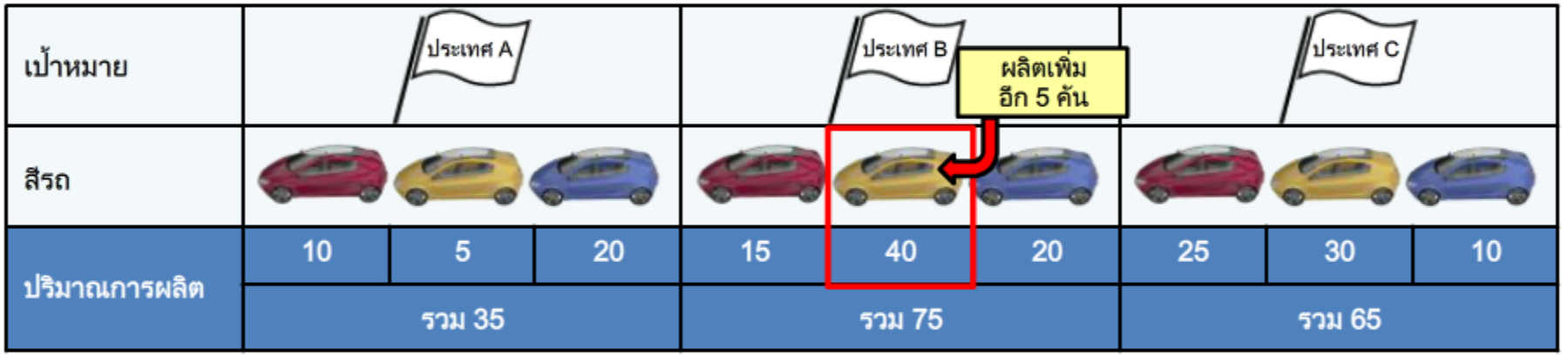
ตัวแปร Array (Array matrix)

```
uProduction[1,1]
```

6.1.2 การกำหนด matrix array

โปรแกรมตัวอย่างต่อไปนี้กำหนดจำนวนรถที่ต้องผลิตเพิ่มเติมจากปริมาณการผลิตที่วางแผนไว้อย่างเร่งด่วนสำหรับรถสีเหลืองที่ส่งไปยังประเทศ B โดยใช้อาร์เรย์ matrix

```
uAdditionalProduction := 5;
uProduction[1,1] := uProduction[1,1] + uAdditionalProduction;
(* เพิ่มจำนวนการผลิตเพิ่มเติม (5 หน่วย) ไปยังปริมาณแผนการผลิตขั้นต้น *)
```



สีรถ (column)

เป้าหมาย (แถว)	แดง	เหลือง	น้ำเงิน
ประเทศ A	[0,0] 10	[0,1] 5	[0,2] 20
ประเทศ B	[1,0] 15	[1,1] 40 -> 45	[1,2] 20
ประเทศ C	[2,0] 25	[2,1] 30	[2,2] 10

6.1.3 การประมวลผลข้อมูลที่จัดเก็บใน matrix array

โปรแกรมตัวอย่างต่อไปนี้คำนวณค่ารวมของปริมาณการผลิตที่วางแผนไว้สำหรับทุก ๆ สีที่ส่งไปยังประเทศ C และกำหนดค่าไปยังตัวแปรโดยใช้อาร์เรย์ matrix

```
uProductionToday := uProduction[2,0] + uProduction[2,1] +uProduction[2,2];
(* คำนวณค่ารวมของแผนปริมาณการผลิตของวันนี้สำหรับทุกสีของประเทศ C และ กำหนดค่าไปยัง
"uProductionToday" *)
```

เป้าหมาย	ประเทศ A			ประเทศ B			ประเทศ C		
สีรถ									
ปริมาณการผลิต	10	5	20	15	45	20	25	30	10
	รวม 35			รวม 80			รวม 65		

สีรถ (คอลัมน์)

เป้าหมาย (แถว)	แดง	เหลือง	น้ำเงิน
ประเทศ A	[0,0] 10	[0,1] 5	[0,2] 20
ประเทศ B	[1,0] 15	[1,1] 45	[1,2] 20
ประเทศ C	[2,0] 25	[2,1] 30	[2,2] 10

6.2 การวนลูป (FOR)

โปรแกรมตัวอย่างในหน้าที่ผ่านมา (การกำหนดค่าปริมาณการผลิตที่วางแผนไว้ของวันนี้) แสดงได้ดังด้านล่าง

```
uProductionToday := uProduction[2,0] + uProduction[2,1] + uProduction[2,2];
```

ตามตัวอย่างนี้ เมื่อจำนวนสิรถเพิ่มขึ้นจะต้องใช้ตัวแปรมาบวกกันเพิ่มมากขึ้นด้วย ดังนั้น การแสดงจะยาวขึ้น ทำให้ยากต่อการอ่าน

```
uProductionToday := uProduction[2,0] + uProduction[2,1] + uProduction[2,2]
                  + uProduction[2,3] + uProduction[2,4] + uProduction[2,5] ...
```

ในกรณีนี้จะใช้คำสั่งลูปในการสร้างโค้ดที่สะอาดมากขึ้น

คำสั่งลูปนั้นได้แก่ คำสั่ง FOR คำสั่ง WHILE และ คำสั่ง REPEAT หลักสูตรใช้คำสั่ง FOR

คำสั่ง FOR ใช้งานได้ดังนี้

```
FOR variable := initial value TO final value BY increments DO
   Execution statement; (* คำสั่งจะดำเนินการในลูปจนกว่า ตัวแปร มีค่าเท่ากับค่าสุดท้าย *)
END_FOR;                (* END_FOR; ต้องถูกวางในการสิ้นสุดคำสั่ง FOR  *)
```

คำสั่งนั้นจะทำซ้ำไปเรื่อยๆ จนกระทั่งถึงค่าสุดท้ายของตัวแปรและถึงคำสั่ง "END_FOR;"

6.2 การวนลูป (FOR)

โปรแกรมตัวอย่างต่อไปนี้เก็บค่าปริมาณการผลิตที่วางแผนไว้สำหรับทุกสีรถที่ส่งไปยังประเทศ C โดยใช้คำสั่ง FOR


```
uProductionToday := 0;                                    (* เริ่มต้นตัวแปร *)
FOR wColor := 0 TO 2 BY 1 DO
    uProductionToday := uProductionToday + uProduction[2,wColor];    (* เพิ่มปริมาณแผนการผลิต *)
END_FOR;
```

ในการใช้คำสั่ง FOR ตัวแปร "wColor" เพิ่มค่าทีละหนึ่งโดยเริ่มจากค่าเริ่มต้นที่เป็นศูนย์ และคำสั่งจะทำซ้ำไปเรื่อยๆ จนกระทั่งตัวแปรเป็นค่าสองตัวแปร "wColor" ถูกกำหนดให้เป็นองค์ประกอบที่สองในอาร์เรย์ "uProduction" ในคำสั่งที่ดำเนินการ
ค่าของตัวแปร "wColor" จะเพิ่มทุกครั้งที่คำสั่งกลับมาทำซ้ำ ปริมาณการผลิตที่วางแผนไว้สำหรับแต่ละสีรถได้นำมาบวกกันในแต่ละครั้งเพื่อให้ได้ค่ารวม

โปรแกรมตัวอย่างนี้ได้ดำเนินการในลูปสามครั้ง (ครั้งแรก: สีแดง [0] => ครั้งที่สอง: สีเหลือง [1] => ครั้งที่สาม: สีน้ำเงิน [2])

การทำงานของโปรแกรมนี้แสดงไว้ที่หน้าต่อไป

6.2 การวนลูป (FOR)

การดำเนินการของคำสั่ง FOR ถูกอธิบายโดยใช้การทำงานของตัวอย่างโปรแกรม

Array ของค่าประมาณการณ์ปริมาณการผลิต

	แดง	เหลือง	น้ำเงิน
ประเทศ A	[0,0] 10	[0,1] 5	[0,2] 20
ประเทศ B	[1,0] 15	[1,1] 45	[1,2] 20
ประเทศ C	[2,0] 25	[2,1] 30	[2,2] 10

คลิกที่ เพื่อไปยังหน้าถัดไป
ดูภาพเคลื่อนไหวอีกครั้งด้วยการคลิกที่ปุ่ม "Play"

Play

Number of repetition of the loop: 3

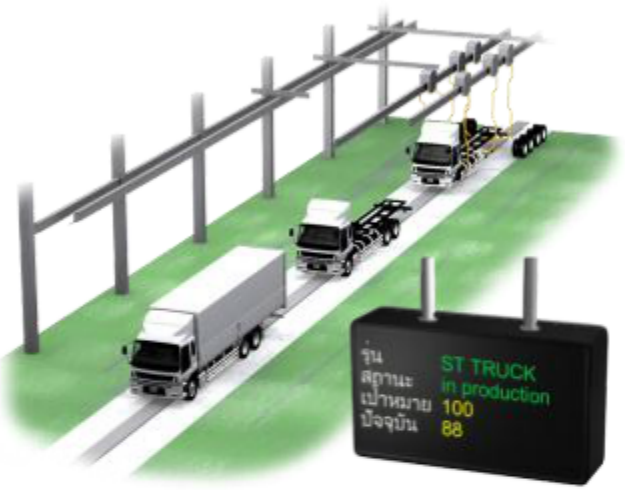
```
uProductionToday := 0;

FOR wColor := 0 TO 2 BY 1 DO
        2
   uProductionToday := uProductionToday + uProduction[2,wColor];
                                          65
END_FOR;
```

6.3 การจัดเก็บข้อมูลที่เกี่ยวข้องกัน (Structure)

โครงสร้างทำให้ชื่อตัวแปรหนึ่งชื่อสามารถแสดงถึงตัวแปรที่เกี่ยวข้องหลายตัวได้
ในตัวอย่างต่อไปนี้ สถานะของสายการผลิตรถยนต์ได้แสดงบน Andon (display board).

ตารางต่อไปนี้แสดงรายการชื่อตัวแปร ค่าตัวแปร และประเภทข้อมูลของตัวแปรที่แสดงรายการ

รายการ	ชื่อตัวแปร	ค่า	ประเภทข้อมูลตัวแปร
Model	sModel	'ST TRUCK'	Text string
Status	bStatus	'in production'	ประเภทBit
เป้าหมายปริมาณการผลิตสำหรับวันนี้	uPlanQty	'100'	Integer ประเภท Word (unsigned)
ปัจจุบันปริมาณการผลิต	uActualQty	'88'	Integer ประเภท Word (unsigned)



หากไม่ได้ใช้ structure ชื่อตัวแปรจะต้องเปลี่ยนไปตามแต่ละสายการผลิตหากมีหลายสาย
ตัวอย่างชื่อตัวแปรตามสายการผลิตแสดงได้ดังต่อไปนี้

Line การผลิตที่ 1

- s1stLineModel
- b1stLineStatus
- u1stLinePlanQty
- u1stLineActualQty

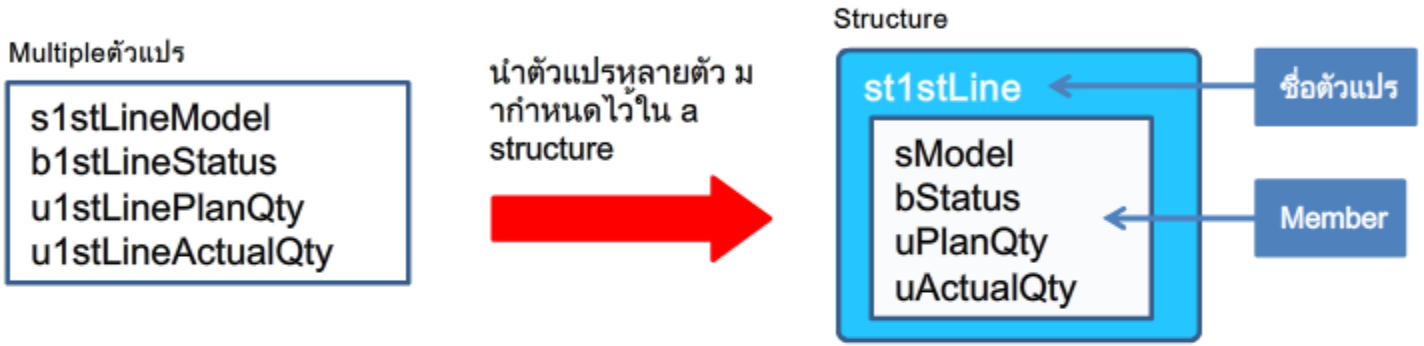
Line การผลิตที่ 2

- s2ndLineModel
- b2ndLineStatus
- u2ndLinePlanQty
- u2ndLineActualQty

เมื่อจำนวนสายการผลิตเพิ่มขึ้น จำนวนตัวแปรที่ต้องใช้งานก็จะเพิ่มขึ้นด้วยเช่นกัน ดังนั้นโปรแกรมจะยาวขึ้นและยากต่อการอ่านมากขึ้น

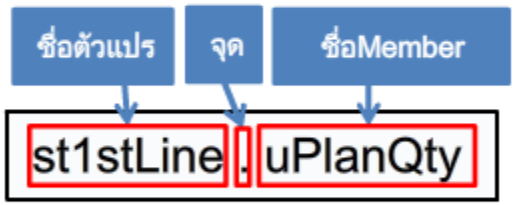
6.3 การจัดเก็บข้อมูลที่เกี่ยวข้องกัน (Structure)

การใช้งาน Structures ทำให้ชื่อตัวแปรหนึ่งชื่อสามารถแสดงถึงตัวแปรหลายตัวที่เกี่ยวข้องกับสายการผลิตเดียวได้ Structures จึงนำมาใช้ในการจัดการ จัดเก็บ และใช้งานข้อมูลเป็นกลุ่มสำหรับวัตถุที่มีเงื่อนไขและข้อมูลจำเพาะเดียวกัน เช่น อุปกรณ์ เครื่องมือ และชิ้นงาน



The structure variable (ตัวแปรโครงสร้าง) ประกอบด้วยตัวอักษรด้านหน้า "st" ในการแสดงว่า นี่คือ a structure ตัวแปรที่กำหนดไว้ใน structure เรียกว่า Member ประเภทข้อมูลของแต่ละMemberอาจแตกต่างกันไป

แต่ละสมาชิกของstructure arraysสามารถกำหนดได้หลังจำนวนองค์ประกอบของ arrays โดยใช้จุดก่อนชื่อMember

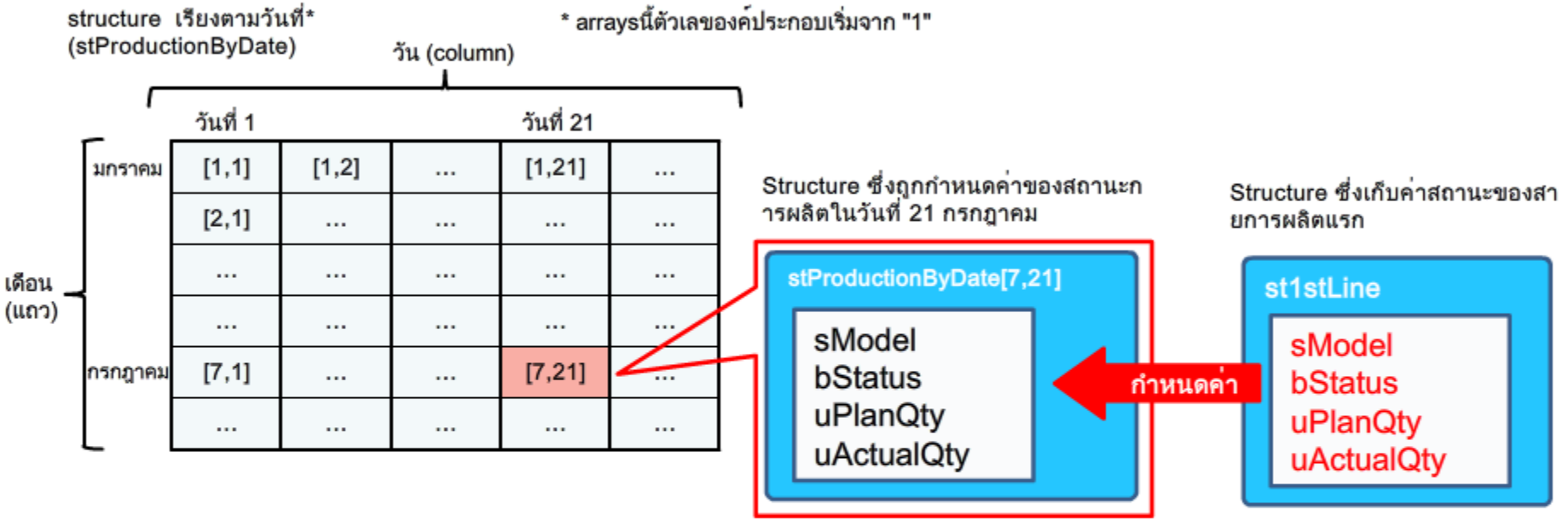


ในตัวอย่างต่อไปนี้ เป็นการกำหนดค่าคงที่ไปยังMemberของตัวแปรstructure สำหรับสายการผลิตแรก

```
st1stLine.uPlanQty := 150;
 (* ตั้งค่า เป้าหมายการผลิตของวันนี้สำหรับสายการผลิตแรกเป็น 150 *)
```

6.3.1 การจัดเก็บ structure arrays

structure หลาย structure สามารถสร้างเป็นarraysได้
ในตัวอย่างต่อไปนี้ สถานะการผลิตถูกเก็บตามวันที่

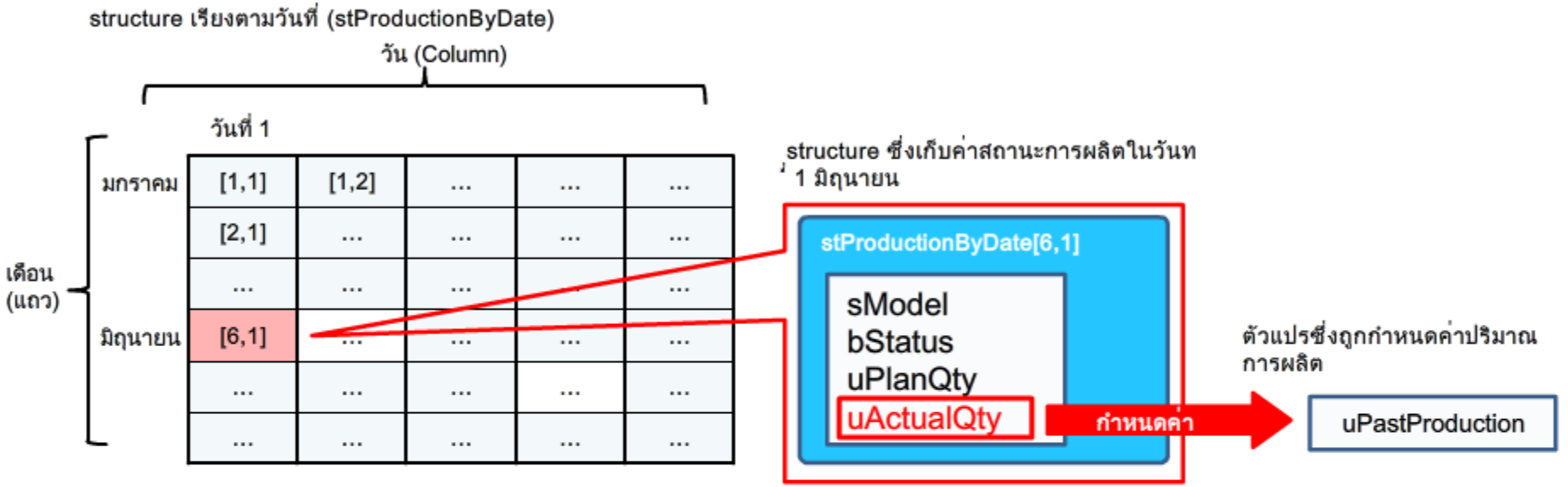



```
stProductionByDate[7,21] := st1stLine;
 (* สถานะการผลิตของวันที่ 21 กรกฎาคมจัดเก็บใน Structure เรียงตามวันที่
(stProductionByDate) *)
```

ด้วยการกำหนดค่าลักษณะนี้ memberจึงไม่จำเป็นต้องกำหนดค่าแยกแต่ละตัวสำหรับการกำหนดค่า Structure

6.3.2 การอ่าน structure arrays

ในตัวอย่างต่อไปนี้ เป็นการอ่านปริมาณการผลิตจาก structure ที่เรียงตามวันที่และกำหนดค่าไปยังตัวแปร




```
uPastProduction := stProductionByDate[6,1].uActualQty;
(* กำหนดปริมาณการผลิตของวันที่ 1 มิถุนายน ไปยังตัวแปร uPastProduction *)
```

แต่ละสมาชิกของ structure arrays สามารถระบุได้โดยเพิ่มจุด (.) และชื่อMemberไปยังหมายเลของค์ประกอบของ arrays

6.4 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- รายละเอียดโดยรวมและการใช้งาน arrays
- การใช้งานลูปโดยใช้คำสั่ง FOR
- รายละเอียดโดยรวมและการใช้งาน structures

จุดสำคัญได้แก่:

Array	• ค่าหลายค่าสามารถใช้งานในตัวแปรเดียวได้โดยใช้ arrays • ตัวแปรแต่ละตัวในarraysถูกกำหนดโดยหมายเลข element ที่เพิ่มที่ด้านหลังของชื่อตัวแปร
คำสั่ง FOR	• คำสั่งลูปจะใช้ในโปรแกรมเมื่อต้องการดำเนินการซ้ำ • คำสั่ง FOR จะใช้เพื่อดำเนินการซ้ำจนกระทั่งครบตามจำนวนรอบที่ตั้งเงื่อนไขไว้ คำสั่งก่อนคำสั่ง "END_FOR;" จะถูกดำเนินการซ้ำๆ
Structure	• Structures ทำให้ชื่อตัวแปรหนึ่งชื่อสามารถแสดงถึงตัวแปรที่เกี่ยวข้องหลายตัวได้ Structures สามารถเก็บตัวแปรที่มีประเภทแตกต่างกันได้ • ตัวแปรหรือ Member แต่ละตัวใน Structures ถูกกำหนดค่าโดยเพิ่มจุดและชื่อ Member ด้านหลังของชื่อตัวแปร Structures

บทที่ 7 การใช้งาน string data

ในบางกรณี PLCได้ใช้งาน string data เพื่อส่งคำสั่งหรือรับผลตอบรับจากอุปกรณ์ที่เชื่อมต่อด้วย เช่น เครื่องอ่านบาร์โคด ตัวควบคุมอุณหภูมิ หรือสเกลอิเล็กทรอนิกส์ ด้วยเหตุนี้จึงจำเป็นต้องทำการเชื่อมหรือดึงค่า string data ตามต้องการได้

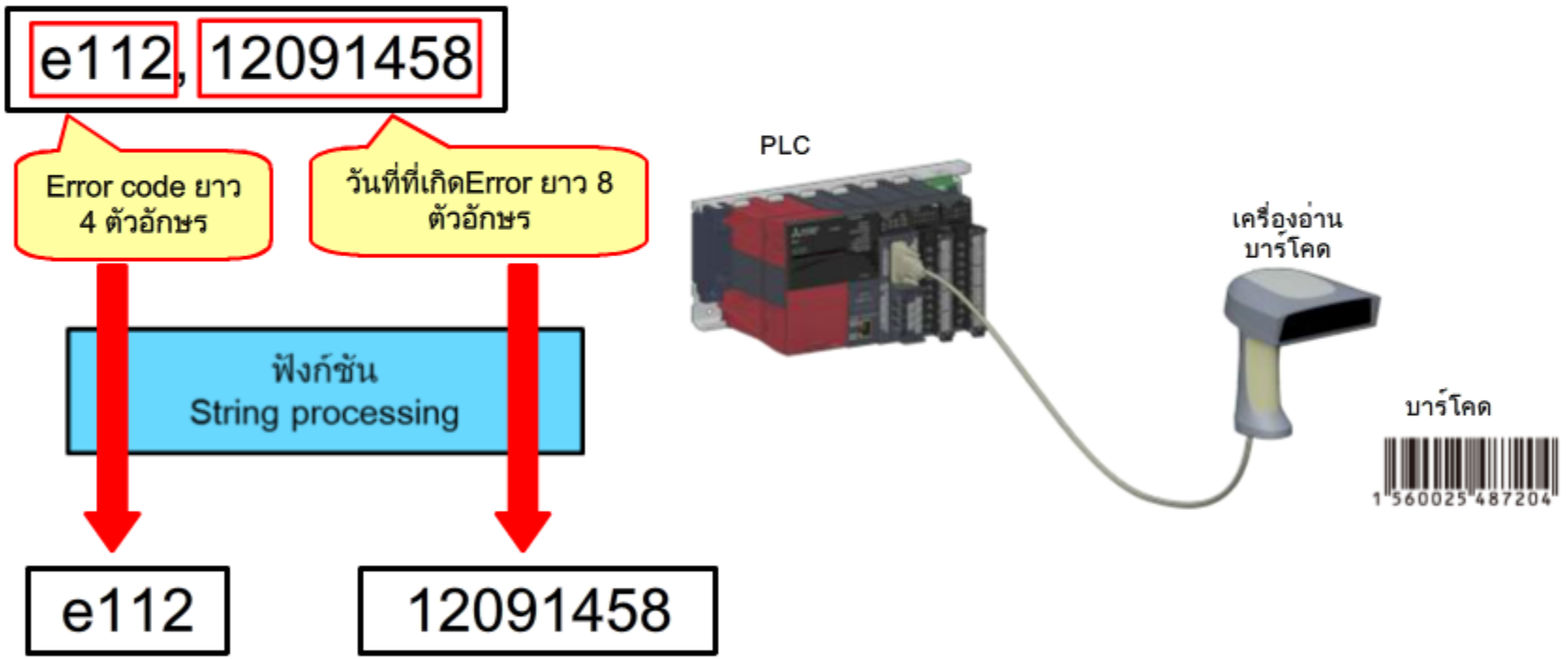
บทนี้อธิบายวิธีการในการใช้งาน string data

7.1 ตัวอย่างการใช้งานstring data

ตัวอย่างในการประมวลผลstring data แสดงโดยใช้กรณีการอ่านข้อมูลจากเครื่องอ่านบาร์โคด
ฟังก์ชัน (ประเภทหนึ่งของคำสั่ง) ได้ใช้ในการประมวลผลstrings

strings ที่อ่านจากเครื่องอ่านบาร์โคดประกอบด้วย Error code ยาว 4 ตัวอักษร และข้อมูลเดือน วันที่ เวลา และนาทียาว 8 ตัวอักษร ดังแสดงด้านล่าง
ตัวอย่างโปรแกรมประมวลผล strings จะอธิบายโดยใช้ระบบนี้

ตัวอย่างการอ่านข้อมูล strings จากเครื่องอ่านบาร์โคด



Error code ที่ดึงออกมา
7.3 การแยก strings (LEFT)

วันที่และเวลาที่เกิด error (14:58, เดือนธันวาคม วันที่ 9) ที่ดึงออกมา
7.4 การแยก Strings (MID)

7.2 การกำหนด strings

ก่อนจะอธิบายเกี่ยวกับวิธีการการแยก Strings ในส่วนนี้จะทำการอธิบายประเภทข้อมูลสำหรับ Strings

ตารางต่อไปนี้แสดงรายการประเภทข้อมูลสำหรับ Strings ที่ใช้ได้กับ PLC

ประเภทข้อมูล	ประเภทตัวอักษรที่ประมวลผลได้	ตัวอักษรนำหน้าในรูปแบบ Hungarian	คำเต็มของตัวอักษรด้านหน้า
String	สตริงของตัวอักษรและตัวเลข (ASCII) หรือ ตัวอักษรภาษาญี่ปุ่น (Shift-JIS)	s	string (สตริง)
String [Unicode]	Strings ของภาษาอื่น ๆ และสัญลักษณ์ต่าง ๆ	ws	wide string (ไวด์สตริง)

ประเภทของ Strings ที่ใช้ขึ้นอยู่กับอุปกรณ์ที่เชื่อมต่อกับPLCหรือภาษาที่เกี่ยวข้อง
บทนี้อธิบายประเภทของ text string แบบต่าง ๆ

เมื่อกำหนดประเภท string ไปยังตัวแปร string ให้ครอบ string ไว้ในเครื่องหมายคำพูดเดี่ยว (')

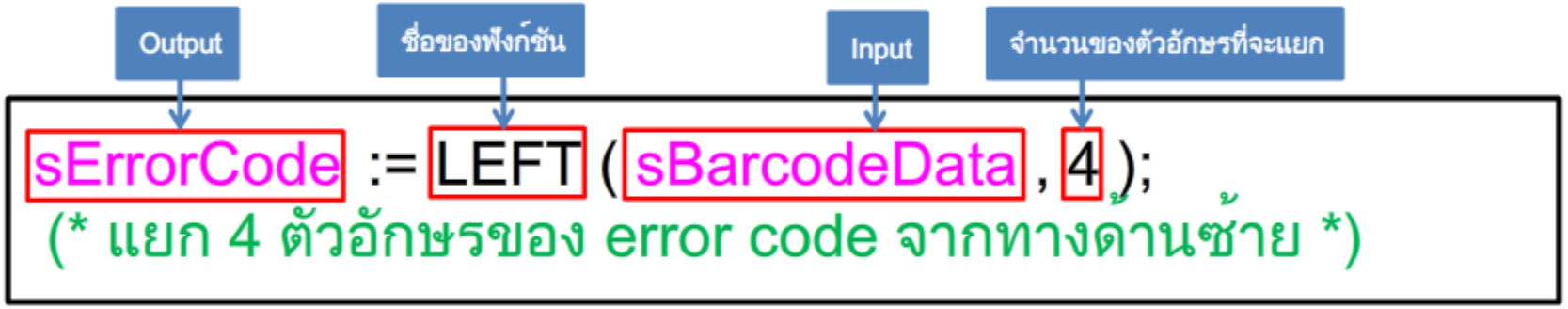
```
sDefault := 'e112,12091458';  (* String assignment *)
```

7.3 การแยก strings (LEFT)

error code "e112" ถูกดึงออกมาจากตัวแปร strings "sBarcodeData" ซึ่งเก็บstrings "e112,12091458"

ชื่อตัวแปร	string ที่เก็บไว้
sBarcodeData	e112, 12091458

ฟังก์ชัน LEFT ดึงเฉพาะตัวอักษรตามจำนวนที่กำหนด โดยเริ่มจากทางด้านซ้ายของ input string
โปรแกรมตัวอย่างแสดงดังต่อไปนี้



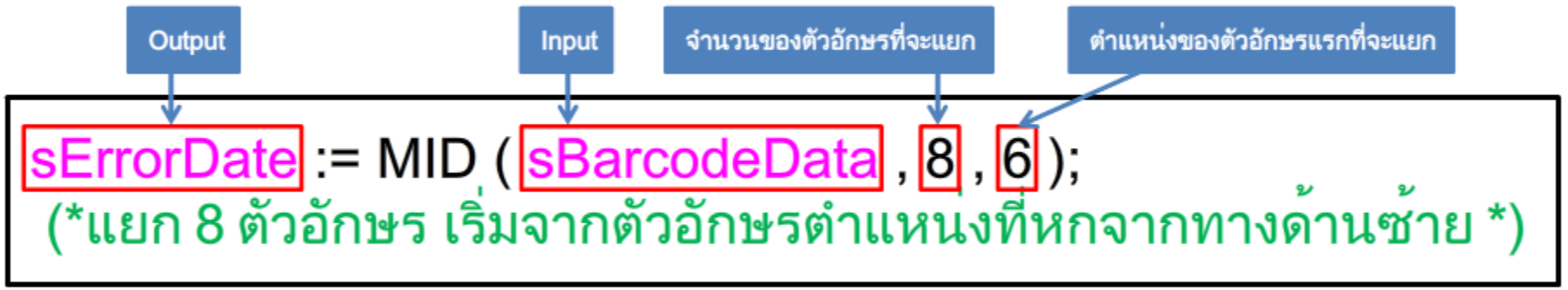
ตัวอักษรสี่ตัวถูกแยกออกจากทางด้านซ้าย ค่า "e112" ซึ่งเป็น string ที่แสดงถึง error code ถูกกำหนดไปยังทางด้านซ้าย

7.4 การแยก strings (MID)

เวลาที่เกิด error "12091458" ถูกแยกออกมาจากตัวแปร string"sBarcodeData" ซึ่งเก็บstring "e112,12091458"

ชื่อตัวแปร	stringที่เก็บไว้
sBarcodeData	e112,12091458

ฟังก์ชัน MID แยกตัวอักษรตามจำนวนที่กำหนดจากตำแหน่งเริ่มต้นที่กำหนดใน input string
โปรแกรมตัวอย่าง แสดงดังต่อไปนี้



ในตัวอย่างนี้ string ยาว 8 ตัวอักษรถูกแยกออกมาโดยเริ่มจากตัวอักษรตำแหน่งที่หก ค่า "12091458" ซึ่งเป็น string ที่แสดงถึงเวลาที่เกิด error ถูกกำหนดไปยังทางด้านซ้าย

7.5 สรุป

เนื้อหาในบทนี้ประกอบด้วย:

- วิธีการกำหนดstringsไปยังตัวแปรstrings
- ฟังก์ชันในการแยกstrings (LEFT และ MID)

จุดสำคัญได้แก่:

String assignment	• ในการ assign a string ไปยังตัวแปร string ให้ครอบ string ไว้ในเครื่องหมายคำพูดเดี่ยว (') • ใช้ประเภทข้อมูล string หรือ string [Unicode] ตามอุปกรณ์ที่เชื่อมต่อกับPLCหรือภาษาที่เกี่ยวข้อง
ฟังก์ชันสำหรับใช้งานstring	• มีหลายฟังก์ชันที่ใช้งานได้ในการใช้งานstring

7.6 สรุปหลักสูตร

หลักสูตรนี้อธิบายถึงพื้นฐานเกี่ยวกับวิธีการสร้างโปรแกรมในภาษา ST
ซึ่งเราก็มาถึงจุดสิ้นสุดของหลักสูตรการเรียนรู้อิเล็กทรอนิกส์นี้กันแล้ว

โปรแกรม ST สร้างได้โดยใช้ซอฟต์แวร์ MELSOFT
สำหรับรายละเอียดในขั้นตอนต่าง ๆ เช่น การกรอกข้อมูล การแก้ไข การบันทึก และการ compileโปรแกรมโดยใช้ซอฟต์แวร์ MELSOFT
โปรดอ้างอิงคู่มือต่อไปนี้

- Mitsubishi FA e-Learning Course "MELSOFT GX Works3 (Structured Text)" (เร็วๆ นี้)
- คู่มือการทำงานของซอฟต์แวร์ MELSOFT

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ ST โปรดอ้างอิงต่อไปนี้

- แนวทางการเขียนโปรแกรมของPLCของคุณ

สำหรับข้อมูลเกี่ยวกับคำสั่งและฟังก์ชันสำหรับการใช้งานของคุณ โปรดอ้างอิงต่อไปนี้

- คู่มือการใช้งานโปรแกรมของPLC ของคุณ

การทดสอบ แบบทดสอบประเมินผล

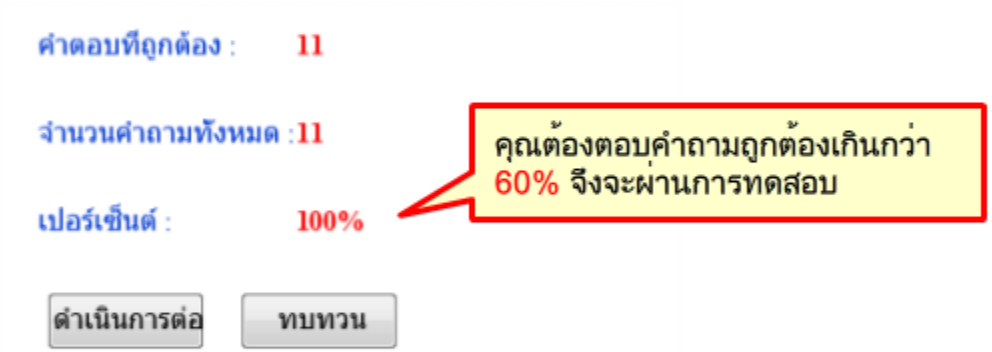
ตอนนี้คุณได้ผ่านหลักสูตรทั้งหมดของ พื้นฐานการเขียนโปรแกรม (StructuredText) แล้ว คุณพร้อมที่จะทำแบบทดสอบประเมินผลแล้ว หากคุณยังไม่มั่นใจเกี่ยวกับหัวข้อต่างๆ ที่จะทดสอบ โปรดทบทวนหัวข้อเหล่านั้น
คำถามในแบบทดสอบประเมินผลนี้มีทั้งหมด 12 ข้อ (20 รายการ)
คุณสามารถทำแบบทดสอบประเมินผลได้หลายครั้งตามต้องการ

วิธีการตอบคำถามในแบบทดสอบ

หลังจากเลือกคำตอบแล้ว ให้คลิกปุ่ม ตอบ คำตอบของคุณจะหายไป ถ้าคุณดำเนินการต่อโดยไม่คลิกปุ่ม ตอบ (โดยจะถือว่าคุณยังไม่ได้ตอบคำถามนั้น)

ผลคะแนน

จำนวนคำตอบที่ถูกต้อง จำนวนคำถาม เปอร์เซ็นต์คำตอบที่ถูกต้อง และผลลัพธ์ที่แสดงว่าผ่าน/ไม่ผ่านจะปรากฏบนหน้าผลคะแนน



- คลิกปุ่ม ดำเนินการต่อ เพื่อออกจากการทดสอบ
- คลิกปุ่ม ทบทวน เพื่อทบทวนการทดสอบ (ตรวจสอบคำตอบที่ถูกต้อง)
- คลิกปุ่ม ลองใหม่ เพื่อทำการทดสอบใหม่อีกครั้ง

การท แบบทดสอบประเมินผล 1

คุณลักษณะของข้อความที่มี (Structured Text, ST)
โปรดเลือกข้อที่อธิบายคุณลักษณะของ ST ผิด

- ST นั้นง่ายในการเรียนรู้สำหรับผู้ที่มีประสบการณ์การเขียนโปรแกรมด้วยภาษา C หรือภาษา BASIC
- การคำนวณเช่นการบวกหรือการลบนั้นสามารถเขียนได้เหมือนการเขียนสมการทางคณิตศาสตร์ปกติ
- ใช้สัญลักษณ์หน้าสัมผัสและขดลวดในการสร้างโปรแกรมประกอบมาเป็นวงจรไฟฟ้า
- ST เหมาะกับการใช้งานข้อมูล

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 2

หลักการพื้นฐานของ ST
โปรดเลือกคำสั่งที่ถูกต้องในภาษา ST

- uProduction = 15
- uProduction := 15:
- uProduction := 15;
- uProduction = 15;

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 3

การอธิบายความเห็น
โปรดเลือกการให้ความเห็นที่ถูกต้องในภาษา ST

- ‘ Assigns a value of 1 to the variable.
- (* Assigns a value of 1 to the variable. *)
- { Assigns a value of 1 to the variable. }
- <!-- Assigns a value of 1 to the variable. -->

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 4

ลำดับการดำเนินการโปรแกรม ST
*ค่าเริ่มต้นของ “uTotalProduction” คือ “100” ค่าของตัวแปร “uTotalProduction” จะเป็น “101”
หลังจากตัวอย่างต่อไปนี้ทำงาน โปรดเลือกสถานะของ “uTotalProduction“
ที่ถูกต้องหลังจากเวลาผ่านไปสองถึงสามวินาที

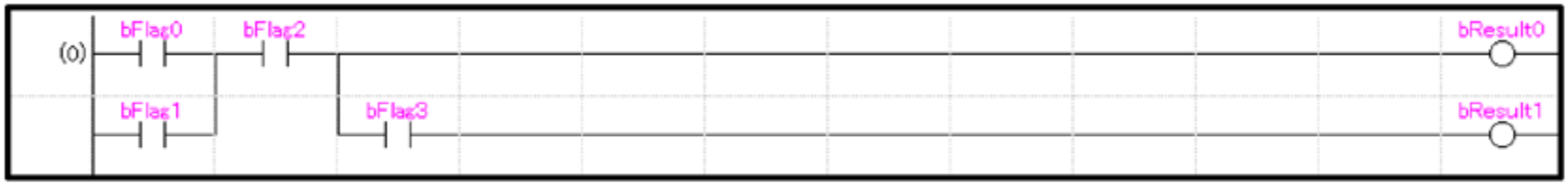
```
uTotalProduction := uTotalProduction + 1;
```

- ค่าจะคงไว้ที่ 101
- ค่าจะเปลี่ยนไปเรื่อย ๆ

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 5

การรวมเงื่อนไขหลายเงื่อนไข
โปรดเลือกตัวอย่างโปรแกรม ST ที่แสดงผลการทำงานแบบเดียวกับตัวอย่างโปรแกรม LD ต่อไปนี้



○
```
bResult0 := (bResult0  OR bFlag1) AND bFlag2;
bResult1 := bResult0 AND bFlag3;
```

○
```
bResult0 := (bFlag0 OR bFlag2) AND bFlag1;
bResult1 := bResult0 AND bFlag3;
```

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 6

คำอธิบายคำสั่ง IF ในภาษา ST
การทำงานต่อไปนี้ถูกดำเนินการโดยตัวอย่างโปรแกรมด้านล่าง
- หากอุณหภูมิลดลงมาถึง 5 องศาหรือต่ำกว่า จะเปิดเครื่องทำความร้อนและปิดเครื่องทำความเย็น
- หากอุณหภูมิขึ้นไปถึง 50 องศาหรือสูงกว่า จะปิดเครื่องทำความร้อนและเปิดเครื่องทำความเย็น
- หากอุณหภูมิไม่ได้อยู่ในช่วงที่กำหนดด้านบน ทั้งเครื่องทำความร้อนและเครื่องทำความเย็นจะปิด

*ชื่อตัวแปร: อุณหภูมิ (wTemperature) เครื่องทำความร้อน (bHeater) และ เครื่องทำความเย็น (bCooler)
โปรดเลือกตัวเลือกที่ถูกต้องสำหรับแต่ละช่องว่างของโปรแกรมตัวอย่าง

```
IF wTemperature [Q1] 5 [Q2]
    bHeater := 1;
    bCooler := 0;
[Q3] 50 [Q4] wTemperature [Q2]
    bHeater := 0;
    bCooler := 1;
[Q5]
    bHeater := 0;
    bCooler := 0;
END_IF;
```

Q1 --Select--

Q2 --Select--

Q3 --Select--

Q4 --Select--

Q5 --Select--

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 7

คำสั่ง CASE

โปรดเลือกตัวเลือกที่ถูกต้องสำหรับแต่ละตัวแปร (Q1 ถึง Q5) คำอธิบายคำสั่ง CASE ต่อไปนี้

คำสั่ง CASE ใช้ในการแยกเงื่อนไขตามค่าของ (Q1)
ในโปรแกรมตัวอย่างต่อไปนี้ เมื่อค่าของ (Q2) เป็น 25 ตัวแปร (Q3) จะถูกกำหนดค่าให้เป็นค่าของ (Q4)
เมื่อค่าของ (Q2) ไม่เท่ากับ 10 หรือ 25 หรือ 8 ตัวแปร (Q3) จะถูกกำหนดค่าให้เป็นค่าของ (Q5)

```
CASE wCode OF
    10:     uLane := 1;
    25:     uLane := 2;
    8:      uLane := 3;
   ELSE     uLane := 4;
END_CASE;
```

Q1 --Select--

Q2 --Select--

Q3 --Select--

Q4 --Select--

Q5 --Select--

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 8

ST arrays และคำสั่งทำซ้ำ
โปรแกรมตัวอย่างต่อไปนี้รวมปริมาณการผลิตที่วางแผนไว้ของทุกรุ่นที่กำหนดไว้สำหรับประเทศ Y
จากนั้นกำหนดค่านี้ไปยังตัวแปร โปรดเลือกส่วนของอาร์เรย์ที่ถูกอ่านหลังจากคำสั่ง FOR ถูกดำเนินการในลูปจำนวน 3 ครั้ง

```
uProductionToday := 0;
FOR wCarModel := 0 TO 3 BY 1 DO
  uProductionToday := uProductionToday + uProduction[1,wCarModel];
END_FOR;
```

Array ใช้ในการเก็บค่าประมาณการณ์ของสินค้าที่ผลิตต่อรุ่นและเป้าหมาย (uProduction)

เป้าหมาย (แถว)	รุ่น (คอลัมน์) รุ่น 1	รุ่น 2	รุ่น 3	รุ่น 4
ประเทศ X	[0,0]	[0,1]	[0,2] C	[0,3]
ประเทศ Y	[1.0]	[1.1] A	[1.2] D	[1.3] E
ประเทศ Z	[2,0]	[2,1] B	[2,2]	[2,3]

- A
- B
- C
- D

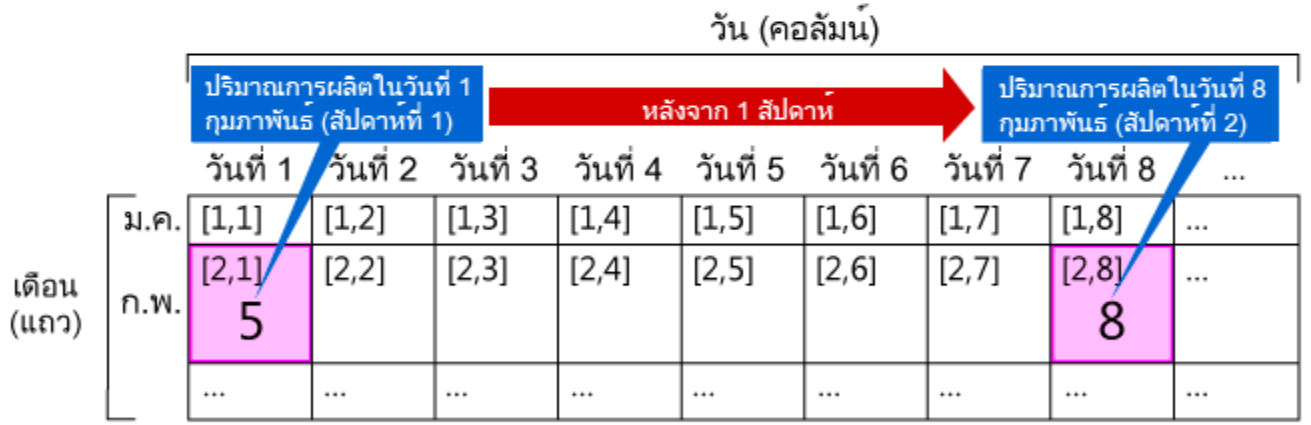
ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 9

ST arrays และคำสั่งทำซ้ำ
ตัวอย่างโปรแกรมต่อไปนี้เก็บปริมาณการผลิตรวมในวันเดียวกันของสัปดาห์ ผลรวมตลอด 4
สัปดาห์นั้นรับค่ามาจากอาร์เรย์ที่เก็บค่าปริมาณการผลิตต่อวัน โปรดเลือกตัวเลขที่ถูกต้องสำหรับโปรแกรมตัวอย่าง

```
uTotalProduction := 0;
FOR wOnceAWeek := 1 TO ■ BY 7 DO
  uTotalProduction := uTotalProduction + uProductionByDate[2,wOnceAWeek];
END_FOR;
(* ดึงค่าและหาผลรวมปริมาณการผลิตของวันเดียวกันในสัปดาห์ตลอด 4 สัปดาห์ เริ่มต้นจากวันที่ 1 กุมภาพันธ์ *)
```

Array ที่เก็บปริมาณการผลิตต่อวัน (uProductionByDate)



- 22
- 21
- 4
- 28

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 10

คุณลักษณะของโครงสร้างใน ST
โปรดเลือกคำอธิบายที่ไม่ถูกต้องของโครงสร้าง

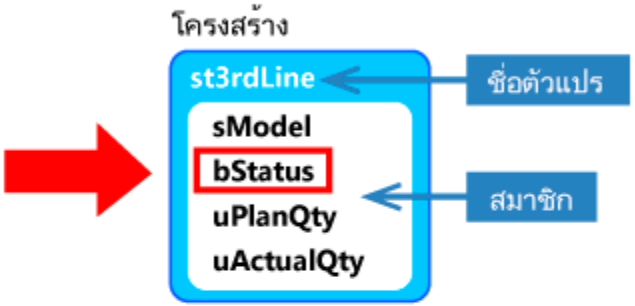
- โครงสร้างใช้สำหรับจัดการและจัดเก็บข้อมูลในอุปกรณ์โดยแยกตามเงื่อนไข เช่น สถานะ และข้อมูลจำเพาะ
- โปรแกรมที่ประมวลผลข้อมูลจำนวนมากสามารถเขียนได้อย่างแม่นยำได้โดยใช้โครงสร้าง
- สมาชิกในโครงสร้างทั้งหมดต้องเป็นข้อมูลประเภทเดียวกัน
- ค่าต่าง ๆ ต้องถูกกำหนดไปยังสมาชิกในโครงสร้างเดียวกันพร้อมกันโดยไม่มีการกำหนดเป็นรายการเฉพาะ

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 11

การกำหนดสมาชิกในโครงสร้างในภาษา ST
โครงสร้างต่อไปนี้จัดการตัวแปรเกี่ยวกับสายการผลิตรถยนต์
โปรดเลือกคำอธิบายที่ถูกต้องสำหรับสมาชิก “bStatus” ในโครงสร้างนี้

พารามิเตอร์	ชื่อตัวแปร
รุ่น	sModel
สถานะ	bStatus
ปริมาณการผลิตเป้าหมายของวันปัจจุบัน	uPlanQty
หมายเลขการผลิตปัจจุบัน	uActualQty

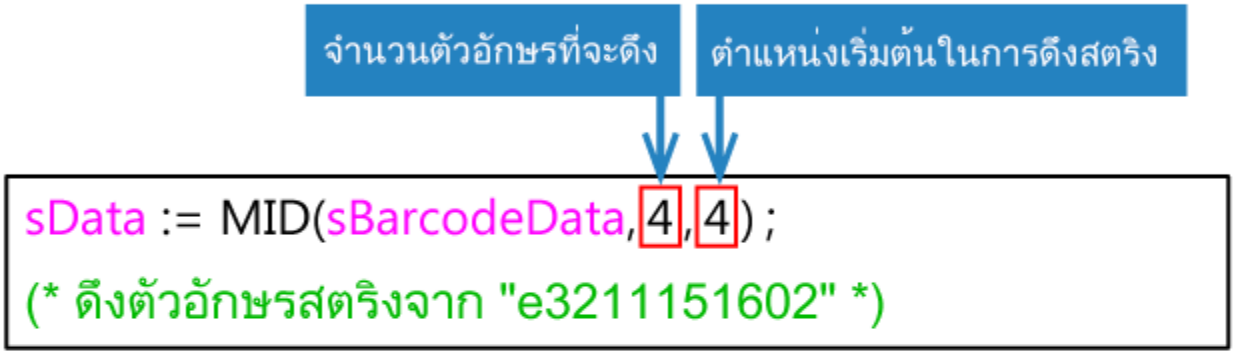


- st3rdLine.bStatus
- st3rdLine->bStatus
- st3rdLine[bStatus]
- st3rdLine[1]

ตอบ ย้อนกลับ

การท แบบทดสอบประเมินผล 12

การใช้งานสตริงในภาษา ST
โปรแกรมตัวอย่างต่อไปนี้ดึง string บางส่วนจาก string "e3211151602" ที่เก็บไว้ในตัวแปร "sBarcodeData" ฟังก์ชัน MID ดึงตัวอักษรตามจำนวนที่กำหนดโดยเริ่มต้นจากตำแหน่งที่กำหนด
โปรดเลือก string ที่ดึงมาให้ถูกต้อง



- 1151
- 1602
- e321
- 1115

ตอบ ย้อนกลับ

การท คะแนนการทดสอบ

คุณทำแบบทดสอบประเมินผลเสร็จสิ้นแล้ว ผลลัพธ์ของคุณมีดังต่อไปนี้
ในการสิ้นสุดแบบทดสอบประเมินผล ให้ไปยังหน้าถัดไป

คำตอบที่ถูกต้อง : 12

คำถามทั้งหมด : 12

เปอร์เซ็นต์ : 100%

ดำเนินการต่อ ทบทวน

ขอแสดงความยินดี คณผ่านการทดสอบ

คุณได้สำเร็จหลักสูตร พื้นฐานการโปรแกรม (StructuredText) แล้ว

ขอขอบคุณสำหรับการเรียนรู้หลักสูตรนี้

เราหวังว่าคุณจะเพลิดเพลินกับบทเรียน และข้อมูลที่คุณได้รับจากหลักสูตรนี้จะเป็นประโยชน์ในอนาคต

คุณสามารถทบทวนหลักสูตรได้หลายครั้งตามต้องการ

ทบทวน ปิด