

เครื่องคอมพิวเตอร์

คอมพิวเตอร์เป็นเครื่องมือช่วยคำนวณ ทำงานได้แต่เฉพาะกับจำนวนเต็มฐานสอง มีองค์ประกอบดังนี้

- | | |
|------------------------------|--|
| หน่วยประมวลผลกลาง | <ul style="list-style-type: none"> - คำนวณทางคณิตศาสตร์ บวก ลบ คูณ หาร - เปรียบเทียบจำนวน - เคลื่อนย้ายข้อมูล <ul style="list-style-type: none"> - หน่วยประมวลผลกลาง $\leftrightarrow$ หน่วยความจำ - หน่วยประมวลผลกลาง $\leftrightarrow$ อุปกรณ์ประกอบ |
| หน่วยความจำ
อุปกรณ์ภายนอก | <ul style="list-style-type: none"> - เก็บข้อมูลที่จำเป็นไว้ระหว่างการทำงาน – ไฟล์ข้อมูลสูญหาย - รับข้อมูลเข้ามาคำนวณ <ul style="list-style-type: none"> - แป้นพิมพ์ รับข้อมูลตัวเลข ตัวอักษร - เมาส์ รับข้อมูลตำแหน่งของ "ตัวชี้" บนหน้าจอ - แสดงผลการทำงาน <ul style="list-style-type: none"> - จอภาพ แสดงผลตัวอักษร ภาพ และภาพเคลื่อนไหว - วงจรเสียง สร้างเสียงจากข้อมูลเสียง - จัดเก็บข้อมูลและโปรแกรม <ul style="list-style-type: none"> - จานแม่เหล็ก อ่านและบันทึกในรูปของอำนาจแม่เหล็ก - จานแสงหรือแผ่นซีดี อ่านและบันทึกในรูปของแสง |

โปรแกรมและข้อมูล

ข้อมูล – สิ่งที่ต้องการประมวลผล อาจเป็น จำนวน ตัวอักษร ข้อความ ภาพ เสียง
ภาพเคลื่อนไหว

โปรแกรม - ขั้นตอนและวิธีการโดยละเอียดในการประมวลผล

+-----+

ข้อมูล $\rightarrow$ | โปรแกรม | $\rightarrow$ ข้อสนเทศ

+-----+

ตัวอย่าง การจัดกลางในโปรแกรมประมวลผลเอกสารเอกสาร

ข้อมูล – ข้อความที่ต้องการจัดกลาง

โปรแกรม

คำนวณความกว้างของตัวอักษรทั้งหมดในข้อความ

คำนวณระยะที่ต้องเว้นจากกริมซ้าย

ระยะ = (ความกว้างทั้งหมดของบรรทัด - ความกว้างของข้อความ/2)

เว้นจากกริมซ้ายเท่ากับระยะที่คำนวณได้ พิมพ์ข้อความออกจอภาพ

การประมวลผลข้อมูลแต่ละประเภทแต่ละแบบมีขั้นตอนวิธีที่ต่างกัน จึงต้องใช้โปรแกรมต่างกัน

$\rightarrow$ ทำให้มีโปรแกรมเป็นจำนวนมาก

ประเภทของโปรแกรม

โปรแกรมสำเร็จรูป – โปรแกรมที่มีผู้ผลิตไว้แล้ว สามารถซื้อใช้งานได้ทันที

โปรแกรมที่สร้างขึ้นเอง – เขียนขึ้นโดยนำคำสั่งพื้นฐานของภาษาสำหรับเขียนโปรแกรม มาจัดเรียงเป็นประโยคคำสั่งตามไวยากรณ์ของภาษานั้น และใช้ตัวแปลภาษาแปลคำสั่งนั้นเป็นคำสั่งที่หน่วยประมวลผลกลางรับรู้และปฏิบัติได้

การทำงานของโปรแกรม

การทำงานของโปรแกรมคือการกำหนดให้หน่วยประมวลผลกลาง ทำงานตามคำสั่งที่แปลได้เป็นลำดับไปจนกว่าจะจบการทำงาน

การทำงานของตัวแปลภาษามีหลายลักษณะ การทำงานอย่างง่ายสำหรับผู้เริ่มต้น ออกแบบให้โต้ตอบกับผู้ใช้ได้ทันทีและโดยตรง มีหน้าจอสำหรับรับคำสั่ง ผู้ใช้ป้อนคำสั่งที่ต้องการ ตัวแปลภาษาแปลเป็นคำสั่งที่หน่วยประมวลผลกลางรู้จัก หน่วยประมวลผลกลางทำงาน และมีหน้าจอสำหรับแสดงผล ตัวอย่างเช่น ภาษาโลโก ซึ่งใช้สำหรับสั่งงานให้ "เต๋า" เดินไปและทิ้งรอยทางเดินไว้บนจอ รอยทางเดินของ "เต๋า" เมื่อนำมาประกอบกันจะได้เป็นภาพลายเส้น เรียกว่า "ภาพวาดของเต๋า"

ภาษาโลโกและตัวแปลภาษาที่ใช้

Logo เป็นภาษาระดับสูงที่มีลักษณะเด่นที่ความง่ายในการเขียนโปรแกรม และความสามารถในการแสดงผลแบบกราฟิกส์ ภาษาโลโกมีรูปแบบไวยากรณ์ที่เข้าใจง่าย แม้แต่สำหรับผู้เริ่มต้น และเป็นภาษาที่เหมาะสมต่อการเรียนรู้ของคนทุกวัย คำสั่งในส่วนของการจัดการแสดงภาพ นิยมเรียกว่า Turtle graphics ซึ่งสามารถแสดงภาพที่ซับซ้อนได้โดยใช้คำสั่งไม่กี่คำสั่ง ภาษาโลโกจึงเป็นภาษาสำหรับเขียนโปรแกรมที่ง่ายที่สุด ซึ่งได้รับการพัฒนาขึ้นโดย Seymour Papert ในปี 1968 เพื่อใช้เป็นเครื่องมือในการเรียนการสอน และได้รับการพัฒนาต่อมาจนเป็นภาษาที่มีประสิทธิภาพในการเขียนโปรแกรมสูง โดยคำสั่งสำหรับดำเนินการต่างๆประมาณ 200 คำสั่ง และถึงแม้ว่าโลโกจะเป็นภาษาสำหรับเขียนโปรแกรมที่สมบูรณ์ แต่การใช้งานส่วนใหญ่ยังคงจำกัดอยู่เฉพาะการทำกราฟิกส์เท่านั้น นับตั้งแต่มีการคิดค้น มีการออกแบบและพัฒนาตัวแปลภาษาโลโกหลายแบบและหลายรุ่น ตัวแปลภาษาโลโกที่ใช้ในบทความนี้ คือ Microsoft Windows Logo เรียกย่อว่า MSWLogo

การเรียกใช้ตัวแปลภาษาโลโก

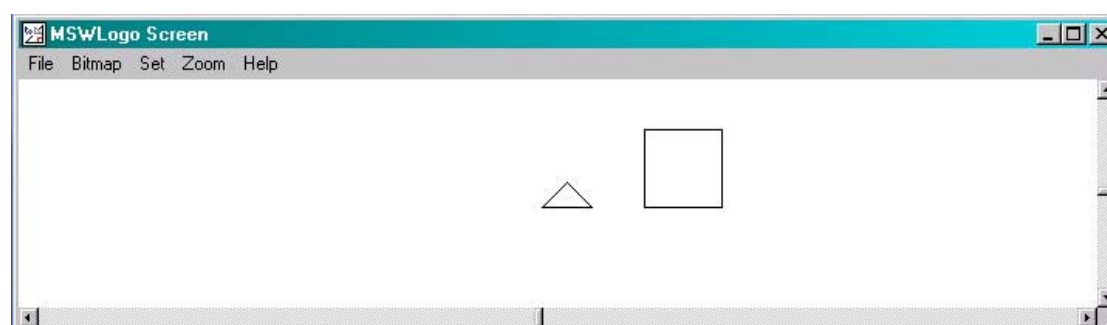
ใช้เมาส์ชี้ที่สัญลักษณ์ของ Microsoft Windows Logo บนจอภาพ และกดปุ่มซ้ายของเมาส์ 2 ครั้งติดกัน รอสักครู่หนึ่งจะได้หน้าจอสองหน้าจอ หน้าจอตอนบนเป็นหน้าจอหรือกระดานสำหรับแสดงภาพ มีข้อความที่มุมบนซ้ายเป็น MSWLogo Screen และมี "เต่า" หรือปลายปากกาสำหรับวาดภาพ อยู่กลางจอภาพ และหน้าจอสำหรับป้อนคำสั่ง หรือ Commander

กระดานสำหรับวาดภาพ

กระดานวาดภาพ ใช้เป็นพื้นที่ในวาดภาพ จอภาพนี้สามารถปรับขนาดได้ และที่ตำแหน่งกึ่งกลางมี "เต่า" สำหรับใช้ในการวาดภาพปรากฏอยู่ เต่านี้มีลักษณะคล้ายปลายปากกา หันหัวไปทางทิศเหนือและหางไปทิศใต้ (ตามทิศที่เรียนมาในวิชาภูมิศาสตร์ โดยเฉพาะการทำแผนที่) ตำแหน่งนี้เรียกว่าตำแหน่งเริ่มต้นของเต่า เรียกทับศัพท์ว่า Home position

ทุกครั้งที่มีการลบจอภาพโดยใช้คำสั่ง CS จะมีการลบ "รอย" ทางเดินของเต่าที่มีอยู่ในขณะนั้นทั้งหมด และวางเต่าไว้ที่ตำแหน่งเริ่มต้น ในกรณีที่ต้องการให้เต่ากลับมาอยู่ที่ตำแหน่งเริ่มต้นโดยไม่ลบภาพ (รอยทางเดินของเต่า) ทำได้โดยใช้คำสั่ง Home

หน้าจอสำหรับวาดภาพมีลักษณะดังนี้



หน้าจอสำหรับป้อนคำสั่ง

หน้าจอสำหรับป้อนคำสั่งแบ่งออกเป็นสองส่วนคือ ช่องสำหรับคำสั่งอยู่ด้านล่าง และพื้นที่ที่แสดงคำสั่งที่เคยมีการใช้ไปแล้ว

คำสั่งที่เคยใช้ไปแล้วนี้สามารถนำกลับมาใช้ใหม่โดยการใช้แป้นลูกศรขึ้นและลง เลื่อนไปยังคำสั่งที่ต้องการ เมื่อใช้แป้นลูกศรเลื่อนไปยังคำสั่งใด คำสั่งนั้นจะปรากฏในช่องสำหรับรับคำสั่ง ซึ่งหากต้องการใช้คำสั่งนั้นซ้ำอีก สามารถกดแป้น <Enter> ได้ทันที ในกรณีที่ต้องการแก้ไขบางส่วนก่อนทำงาน ทำได้โดยการใช้เมาส์คลิกที่คำสั่งในช่องสำหรับรับคำสั่ง ทำการแก้ไข และกดแป้น <Enter> เพื่อให้ทำงาน

หน้าจอสำหรับป้อนคำสั่งมีลักษณะดังนี้



คำสั่งที่จะป้อนเข้าในช่องสำหรับรับคำสั่งต้องเป็นคำสั่งที่ “เต่า” รู้จัก คำสั่งดังกล่าวนี้มีสองแบบ คือ คำสั่งพื้นฐานที่มีมาพร้อมกับตัวแปลภาษา เรียกว่า Primitive และคำสั่งที่กำหนดขึ้นใช้งานเอง เรียกว่ากระบวนการ หรือ Procedure

คำสั่งการเคลื่อนที่พื้นฐานของ “เต่า”

FD <ระยะทาง> เดินตรงไปข้างหน้า (ตามทิศทางของหัวลูกศร หรือปลายปากกา) เป็น <ระยะทาง>ตามที่กำหนด
 BK <ระยะทาง> เดินถอยหลัง (ตามทิศทางของหางลูกศร) เป็น <ระยะทาง> ที่กำหนด
 RT <มุม> เลี้ยวขวา (หันหัวเต่าไปทางขวา) ตาม <มุม> ที่กำหนด
 LT <มุม> เลี้ยวซ้าย (หันหัวเต่าไปทางซ้าย) ตาม <มุม> ที่กำหนด

การเคลื่อนที่ของเต่าโดยทั่วไปต้องใช้คำสั่ง 2 คำสั่งคือ การหันหัวเต่าไปในทิศทางที่ต้องการ โดยใช้ คำสั่ง RT และ LT ตามด้วยมุมในหน่วยองศา เมื่อหันหัวเต่าไปยังทิศที่ต้องการได้แล้ว จึงสั่งให้เต่าเดินไป ข้างหน้าหรือถอยหลังตามที่ต้องการโดยใช้คำสั่ง FD หรือ BK ตามด้วยระยะทางที่ต้องการ โดยระยะทาง 30 หน่วยเท่ากับ 1 เซนติเมตรบนจอภาพโดยประมาณ

คำสั่งอื่นๆ

PD วางปากกานบนกระดานวาดภาพ เมื่อ “เต่า” เดินจะทิ้งรอยเท้าไว้
 PU ยกปากกาขึ้นจากกระดานวาดภาพ เมื่อ “เต่า” เดินจะไม่ทิ้งรอยเท้าไว้
 ST แสดง “เต่า” หรือ ปลายปากกาสำหรับวาดภาพบนจอ
 HT ซ่อนหรือไม่แสดงปลายปากกาสำหรับวาดภาพบนจอ
 CS ลบกระดานวาดภาพ และกำหนดให้เต่าอยู่ในตำแหน่ง “เริ่มต้น” คือตำแหน่งกลาง จอภาพ หันหัวขึ้นทางทิศเหนือ และทำการวางปากกาลงบนกระดานวาดภาพ
 HOME กำหนดให้เต่ากลับมาอยู่ที่ตำแหน่งเริ่มต้นโดยไม่มีการลบจอภาพและเปลี่ยนสถานะ ของปากกา

การป้อนคำสั่งและข่าวสารแสดงความผิดพลาด (Error Message)

ในกรณีที่ป้อนคำสั่งที่ถูกต้อง จะทำงานให้ทันที ในกรณีที่ป้อนคำสั่งที่ไม่รู้จัก เช่นป้อนคำสั่งเป็น draw จะแสดงมีข้อความดังนี้

I don't know how to draw

ซึ่งหมายความว่า “เต่า” ไม่รู้ว่าจะทำตามคำสั่ง draw ได้อย่างไร

ในกรณีที่คำสั่งนั้นต้องมีข้อมูลประกอบคำสั่ง เช่นคำสั่งให้เต่าเดินตรงไปข้างหน้า คือคำสั่ง FD ต้องมีข้อมูลระยะทางประกอบด้วย หากทำการป้อนเฉพาะคำสั่ง FD เพียงอย่างเดียว จะมีการแสดง ข้อความดังนี้

not enough inputs to forward

ซึ่งหมายความว่า “เต่า” มีข้อมูลไม่พอที่จะใช้ในการทำงานตามคำสั่ง forward ได้อย่างไร

การสั่งให้ “เต่า” วาดภาพรูปทรงเรขาคณิต

การกำหนดให้เครื่องคอมพิวเตอร์ทำงานใดงานหนึ่ง (ผ่านตัวแปลภาษา) จำเป็นต้องทำการ วิเคราะห์ลำดับขั้นตอนที่ใช้ก่อนดังนี้

สมมุติว่าในขณะนี้ “เต่า” อยู่ในตำแหน่ง “เริ่มต้น” หากต้องการวาดรูปสี่เหลี่ยมจัตุรัสที่มีความยาว ด้าน 100 หน่วย สามารถทำได้สี่ตำแหน่งคือ ตำแหน่งบนซ้าย, บนขวา, ล่างซ้าย และล่างขวา ซึ่งมีลำดับ การทำงานของแต่ละแบบดังนี้

บนซ้าย	บนขวา	ล่างซ้าย	ล่างขวา
เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย	เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา
เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา	เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย
เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย	เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา
เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา	เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย
เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย	เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา
เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา	เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย
เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย	เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา
เลี้ยวซ้าย 90 องศา	เลี้ยวขวา 90 องศา	เดินตรงไป 100 หน่วย	เดินตรงไป 100 หน่วย

หากต้องการสร้างภาพสี่เหลี่ยมจัตุรัสในตำแหน่ง “บนขวา” ทำได้โดยการแปลขั้นตอนวิธีเป็นคำสั่งในภาษาโลโกดังนี้

คำสั่งภาษาโลโก	คำอธิบาย
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา

ขอให้ทดลองป้อนคำสั่งเหล่านี้และสังเกตผลที่เกิดขึ้น และขอให้สังเกตตำแหน่งสุดท้ายของเต่าไว้ด้วย เพื่อใช้ในการวาดรูปต่อไป

การระบายสี

รูปทรงใดที่เป็นรูปทรงปิด เช่นรูปทรงเรขาคณิต สามารถระบายสีรูปทรงนั้นได้หนึ่งสีต่อหนึ่งรูปทรง โดยจะระบายสีรูปทรงใดได้ก็ต่อเมื่อ “เต่า” หรือปากกาอยู่ในรูปทรงนั้น ดังนั้นเมื่อต้องการระบายสีรูปทรงใด ต้องดำเนินการต่อไปนี้

การเตรียมการ

การเตรียมการ คือการส่งเต่าเข้าไปอยู่ภายในรูปทรงนั้น การเดินของเต่าเข้าไปภายในรูปทรงส่วนใหญ่ไม่ต้องการลากเส้น จึงต้องใช้คำสั่งยกปากกาขึ้น (คำสั่ง PU) แล้วจึงสั่งให้เต่าเดินเข้าไปที่ตำแหน่งใดภายในรูปทรง จากตัวอย่างการวาดรูปสี่เหลี่ยม ตำแหน่งสุดท้ายของเต่าอยู่ที่ตำแหน่งเริ่มต้นและหันหัวไปทางทิศเหนือ ดังนั้นหากต้องการระบายสีต้องดำเนินการดังนี้

หันหัวเต่าไปทางขวา 45 องศา	; มุมใดก็ได้ระหว่าง 1 – 89 องศา
ยกปากกาขึ้น	
เดินตรงไปเป็นระยะทาง 10 หน่วย	; ระยะทางใดระหว่าง 1 – 99 หน่วย

การกำหนดสีที่จะใช้ระบาย

การกำหนดสีที่จะระบายทำได้โดยใช้คำสั่ง SETFC <หมายเลขสี> ซึ่งสามารถกำหนดสีได้ 16 สีดังมีรายละเอียดตามตารางต่อไปนี้

หมายเลขสี	ชื่อสี	หมายเลขสี	ชื่อสี
0	ดำ (Black)	8	น้ำตาล (Brown)
1	น้ำเงิน (Blue)	9	น้ำตาลอ่อน (Light brown)
2	เขียว (Green)	10	เขียวแก่ (Mid-green)
3	ฟ้า (Cyan)	11	ฟ้าอมเขียว (Blue-green)
4	แดง (Red)	12	ส้มอ่อน (Salmon)
5	ม่วงแดง (Magenta)	13	น้ำเงินอมม่วง (Blue-ish)
6	เหลือง (Yellow)	14	ส้ม (Orange)
7	ขาว (White)	15	เงิน (Silver)

คำสั่งระบายสี

การระบายสีในรูปทรงทำได้โดยใช้คำสั่ง FILL

การกลับตำแหน่งเดิม

เมื่อระบายสีเสร็จแล้วจึงให้เต่าเดินถอยหลังออกมาเพื่อกลับสู่ตำแหน่งเดิม โดยใช้คำสั่ง BK <ระยะทาง> โดยใช้ระยะทางเท่าเดิม แล้วใช้คำสั่งวางปากกา (คำสั่ง PD) เพื่อใช้ในการลากเส้นสำหรับวาดรูปต่อไป จากตัวอย่างทำได้ดังนี้

ถอยหลังกลับตำแหน่งเดิมเป็นระยะทาง 10 หน่วย
หันหัวเต่ากลับตำแหน่งเดิม โดยหันไปทางซ้าย 45 องศา
วางปากกาลง

คำสั่งภาษาโลโกสำหรับระบายสี

จากงานที่วิเคราะห์และอธิบายไว้ในหัวข้อที่แล้ว นำมาเขียนเป็นคำสั่งภาษาโลโกได้ดังนี้

คำสั่งภาษาโลโก	คำอธิบาย
RT 45	; หันหัวเต่าไปทางขวา 45 องศา
PU	; ยกปากกาสำหรับวาดรูปขึ้น
FD 10	; ให้เต่าเดินเข้าไปในรูปทรงที่ต้องการระบายสี 10 หน่วย
SETFC 2	; กำหนดสีที่ใช้ระบายเป็นสีหมายเลข 2 (สีเขียว)
FILL	; ทำการระบายสี
BK 10	; ถอยหลังกลับตำแหน่งเดิม 10 หน่วย
LT 45	; หันหัวเต่ากลับตำแหน่งเดิม
PD	; วางปากกาลง เพื่อใช้ในการวาดภาพต่อไป

การปรับปรุงคำสั่งวาดรูปสี่เหลี่ยมจัตุรัส

จากคำสั่งวาดรูปสี่เหลี่ยมจัตุรัสในตอนที่แล้วคือ

คำสั่งภาษาโลโก	คำอธิบาย
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา
FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา

จะเห็นได้ว่าการสั่งงานชุดนี้ประกอบด้วยคำสั่ง 8 คำสั่ง ซึ่งสามารถแบ่งออกได้เป็นคำสั่งที่ซ้ำกัน 4 ชุด แต่ละชุดมี 2 คำสั่งคือ

FD 100	; เดินตรงไป 100 หน่วย
RT 90	; เลี้ยวขวา 90 องศา

หรืออาจเขียนคำสั่งได้อีกแบบหนึ่งคือ

ทำคำสั่งต่อไปนี้ซ้ำกัน 4 ครั้ง คือ FD 100 และ RT 90

ภาษาโลโกกำหนดให้คำสั่ง REPEAT สำหรับกำหนดการทำงานซ้ำ ตามด้วยจำนวนครั้งที่ต้องการทำ ส่วนคำสั่งที่ต้องทำในแต่ละครั้งซึ่งอาจมีได้หลายคำสั่งให้เขียนเรียงกันไว้ในเครื่องหมายวงเล็บก้ามปู ดังนั้นรูปแบบทั่วไปของคำสั่ง REPEAT จึงเป็นดังนี้

REPEAT <จำนวนครั้ง> [คำสั่งที่ต้องการให้ทำในแต่ละครั้ง]

ดังนั้นคำสั่งวาดรูปสี่เหลี่ยมจัตุรัสที่ปรับปรุงให้ใช้คำสั่ง REPEAT แล้วจึงเป็นดังนี้

REPEAT 4 [FD 100 RT 90]

ซึ่งคำสั่งจะสั้นลงและสั่งงานได้ง่ายขึ้น ขอให้ทดลองสั่งงานดู

การสร้างคำใหม่ในภาษาโลโก

ภาษาโลโกมีคำสั่งสองแบบคือคำสั่งพื้นฐานที่มีมาพร้อมกับแปลภาษาเรียกว่า Primitive เช่นคำสั่ง FD, BK, RT, LT และ REPEAT เป็นต้น ซึ่งผู้ใช้สามารถรวมคำสั่งเหล่านี้มากำหนดให้ "เต่า" วาดรูปใดรูปหนึ่งเช่นรูปสี่เหลี่ยมจัตุรัส ได้แก่คำสั่ง

REPEAT 4 [FD 100 RT 90]

คำสั่งชุดนี้สามารถนำมาสร้างเป็นคำสั่งใหม่สำหรับใช้งานต่อไปได้ โดยการกำหนดเป็นกระบวนการ หรือ Procedure โดยการเรียกหน่วยที่ทำหน้าที่สร้างคำสั่ง เรียกว่า Editor การสร้างกระบวนการต้องกำหนด

ชื่อก่อน เช่นการสร้างกระบวนการสร้างสี่เหลี่ยมจัตุรัส กำหนดชื่อกระบวนการเป็น square และป้อนคำสั่ง

```
EDIT "square
```

หมายเหตุ ขอให้สังเกตว่าชื่อกระบวนการต้องนำหน้าด้วยเครื่องหมายคำพูด

เมื่อป้อนคำสั่งได้ถูกต้อง จะปรากฏหน้าจอสำหรับพิมพ์กระบวนการ ซึ่งมีข้อความที่มุมบนซ้ายเป็น Editor และมีข้อความในหน้าจอนี้ 2 บรรทัดคือ

```
to square
end
```

หมายความว่าตัวแปลภาษาโลโกพร้อมที่จะรับการสร้างกระบวนการ หรือคำสั่งชื่อ square แล้ว ผู้ใช้สามารถแทรกคำสั่ง REPEAT 4 [FD 100 RT 90] ระหว่างข้อความเดิมได้เลย หน้าจอสำหรับพิมพ์กระบวนการที่มีการแทรกคำสั่งแล้วเป็นดังนี้



เมื่อตรวจสอบว่าพิมพ์คำสั่งได้ถูกต้องแล้ว ให้ทำการกำหนดกระบวนการนี้แก่ตัวแปลภาษาโดยใช้เมาส์เลือกการคลิก File แล้วเลือกการ Save and Exit เพื่อจัดเก็บคำสั่งนี้ เมื่อทำการจัดเก็บเรียบร้อยแล้ว หน้าจอสำหรับพิมพ์คำสั่งหรือ Editor จะหายไป ในขณะที่กระบวนการหรือคำสั่ง square ได้รับการจัดเก็บไว้เรียบร้อยแล้ว และพร้อมใช้งาน สามารถทดสอบการทำงานได้โดยป้อนคำสั่ง

```
square
```

ผลลัพธ์ที่ได้เป็นอย่างไร เป็นไปตามที่คาดไว้หรือไม่

การปรับปรุงกระบวนการ square

ในขณะนี้ถึงแม้ว่ากระบวนการ square จะสามารถวาดรูปสี่เหลี่ยมจัตุรัสได้แล้ว แต่ยังคงเป็นสี่เหลี่ยมจัตุรัสที่มีความยาวด้านตายตัวคือ 100 หน่วย ในตอนนี้เราจะปรับปรุงให้กระบวนการนี้มีความอ่อนตัวมากขึ้น โดยให้สามารถทำงานในลักษณะเดียวกับคำสั่ง FD ที่สามารถกำหนดระยะทางได้ คือสามารถความยาวด้านที่ต้องการให้กับกระบวนการ square ได้ และเมื่อป้อนคำสั่ง

```
square 50
```

หมายถึงการกำหนดให้สร้างสี่เหลี่ยมจัตุรัสที่มีความยาวด้านเป็น 50 หน่วย และคำสั่ง

```
square 200
```

หมายถึงการกำหนดให้สร้างสี่เหลี่ยมจัตุรัสที่มีความยาวด้านเป็น 200 หน่วยเป็นต้น

การจะทำเช่นนี้ได้ ต้องเพิ่มที่สำหรับเก็บค่าที่ผู้ใช้ป้อนเข้ามาหลังคำสั่ง square และนำไปใช้กับคำสั่ง FD ในกระบวนการ ที่สำหรับเก็บค่าที่ผู้ใช้ส่งมาหลังคำสั่งเรียกว่า "ตัวแปร" เพราะค่านี้สามารถแปรเปลี่ยนไปได้ตามความต้องการ ตัวแปรนี้ต้องมีชื่อที่นำหน้าเครื่องหมาย : (Colon) เช่น :size กระบวนการที่ปรับปรุงแล้วอยู่ด้านซ้ายมือ และขอให้เปรียบเทียบกับกระบวนการเดิมด้านขวามือดังนี้

กระบวนการใหม่

```
to square :size
  repeat 4 [ fd :size rt 90 ]
end
```

กระบวนการเดิม

```
to square
  repeat 4 [ fd 100 rt 90 ]
end
```

การแก้ไขกระบวนการทำได้โดยใช้คำสั่ง EDIT "square เพื่อเรียกหน้าจอสำหรับพิมพ์กระบวนการ ซึ่งจะแสดงกระบวนการ square เดิม ผู้ใช้เปลี่ยนแปลงส่วนที่ต้องการ จากนั้นจึงเลือกการคลิก File และเลือก Save and Exit เพื่อเก็บแทนกระบวนการเดิม เมื่อจัดเก็บเรียบร้อยแล้ว สามารถเรียกใช้กระบวนการใหม่นี้ได้ แต่คราวนี้ต้องกำหนดความยาวด้านที่ต้องการประกอบด้วย

รูปแบบของกระบวนการหรือคำสั่งที่ผู้ใช้สร้างขึ้น

จากที่ได้กล่าวมาแล้วจะเห็นว่ากระบวนการที่ผู้ใช้สร้างขึ้นมีองค์ประกอบที่สำคัญ 3 ส่วนดังต่อไปนี้คือ

1. ขึ้นต้นด้วยคำว่า TO ตามด้วยชื่อของกระบวนการเป็นคำภาษาอังกฤษคำเดียว
2. รายละเอียดและลำดับการทำงานของกระบวนการประกอบขึ้นด้วยคำสั่งพื้นฐาน และ/หรือกระบวนการที่มีการกำหนดไว้เรียบร้อยแล้ว
3. ปิดท้ายกระบวนการด้วยคำสั่ง END

การสร้างรูปทรงเรขาคณิตแบบอื่นๆ

รูปทรงเรขาคณิตที่มีด้านมากกว่าสองด้านขึ้นไป เรียกว่ารูปหลายเหลี่ยม หรือเรียกทับศัพท์ว่า "โพลีกอน" (Polygon) ความยาวของด้านแต่ละด้านจะเป็นเท่าใดก็ได้ เมื่อด้านสองด้านมาบรรจบกันจะเกิดเป็นมุมขึ้น รูปหลายเหลี่ยมที่มีความยาวด้านทุกด้านเท่ากัน และมุมทุกมุมเท่ากันเรียกว่า รูปหลายเหลี่ยมด้านเท่า (Regular polygon) เช่น สามเหลี่ยมด้านเท่า สี่เหลี่ยมจัตุรัส และหกเหลี่ยมด้านเท่า เป็นต้น

การสร้างรูปหลายเหลี่ยมในภาษาโลโก ทำได้ง่าย เนื่องจากรูปหลายเหลี่ยมเป็นรูปทรงปิด ดังนั้นวิธีการสร้างที่ง่ายที่สุดคือ การกำหนดให้ "เต๋า" เดินไปตามขอบของรูปหลายเหลี่ยม เมื่อเดินไปจนครบระยะทางของด้านใดด้านหนึ่ง ต้องทำการเลี้ยวจำนวนหนึ่ง จนกระทั่งมุมที่เลี้ยวรวมกันได้ 360 องศา "เต๋า" จะกลับคืนมายังจุดเริ่มต้น

เนื่องจาก "เต๋า" ต้องสร้างรูปหลายเหลี่ยมเป็นจำนวน n ด้าน ดังนั้นขนาดของมุมที่เลี้ยวในแต่ละครั้งจึงมีค่าเป็น $360/n$

รูปหลายเหลี่ยม	มุมภายใน	360/จำนวนด้าน
สามเหลี่ยมด้านเท่า	60	$360/3 = 120$
สี่เหลี่ยมจัตุรัส	90	$360/4 = 90$
ห้าเหลี่ยมด้านเท่า	108	$360/5 = 72$
หกเหลี่ยมด้านเท่า	120	$360/6 = 60$
เจ็ดเหลี่ยมด้านเท่า	128	$360/7 \approx 51.43$
แปดเหลี่ยมด้านเท่า	135	$360/8 = 45$

แบบฝึกหัด จงทดลองและเขียนกระบวนการเพื่อสร้างรูปสามเหลี่ยมด้านเท่า ห้าเหลี่ยมด้านเท่า และหกเหลี่ยมด้านเท่า ที่ผู้ใช้สามารถกำหนดความยาวด้านได้ โดยใช้ชื่อกระบวนการเป็น triangle, pentagon และ hexagon ตามลำดับ

กระบวนการที่จัดเตรียมไว้ให้และการประยุกต์ใช้งาน

กระบวนการที่จัดเตรียมไว้ให้มี 2 กระบวนการคือ กระบวนการสร้างรูปสามเหลี่ยมหน้าจั่ว และกระบวนการสร้างรูปสี่เหลี่ยมผืนผ้า ดังนี้

1. กระบวนการสร้างรูปสี่เหลี่ยมผืนผ้า

```
to rectangle :width :length
  repeat 2 [ fd :width rt 90 fd :length rt 90 ]
end
```

การเรียกใช้

```
rectangle <กว้าง> <ยาว>
```

ตัวอย่าง

```
rectangle 20 100
```

คำสั่งนี้ใช้สร้างสี่เหลี่ยมผืนผ้าที่มีด้านกว้าง 20 หน่วย และมีด้านยาวเป็น 100 หน่วย

2. กระบวนการสร้างรูปสามเหลี่ยมหน้าจั่ว

```

to isosceles :base :apex
  fd :base
  rt 180 - (180 - :apex)/2
  fd :base/(2 * cos (180 - :apex)/2)
  rt 180 - :apex
  fd :base/(2 * cos (180 - :apex)/2)
  rt 180 - (180 - :apex)/2
end

```

การเรียกใช้

```
isosceles <ความยาวฐาน> <มุมยอด>
```

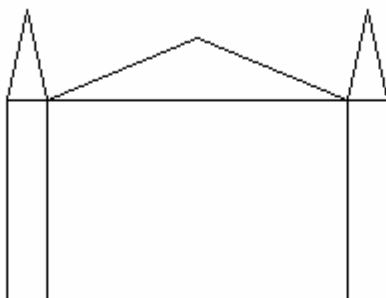
ตัวอย่าง

```
isosceles 200 120
```

เป็นคำสั่งสร้างสามเหลี่ยมหน้าจั่วที่มีความยาวฐาน 200 หน่วย และมีมุมยอดเป็น 120 องศา

การนำกระบวนการหลายกระบวนการมาใช้งานร่วมกัน

เมื่อกำหนดกระบวนการ rectangle สำหรับสร้างรูปสี่เหลี่ยมผืนผ้า และกระบวนการ isosceles สำหรับสร้างสามเหลี่ยมหน้าจั่วได้แล้ว สามารถนำกระบวนการทั้งสองมาใช้ร่วมกันในการสร้างรูปอื่นต่อไป เช่นนำมาใช้สร้างเป็นรูปต่อไปนี้



รูปนี้สร้างจากกระบวนการดังนี้

```

to castle
  rectangle 100 20
  rt 90 fd 20 lt 90 fd 100 lt 90
  isosceles 20 25

  lt 90 fd 100 rt 180

  rectangle 100 150
  fd 100 rt 90 fd 150 rt 180
  isosceles 150 135

  lt 90 fd 100 rt 180

  rectangle 100 20
  rt 90 fd 20 lt 90 fd 100 lt 90
  isosceles 20 25
end

```

กระบวนการ castle มีคำสั่งที่ซ้ำกันหรือไม่ วิธีการในการลดคำสั่งที่ซ้ำกันในลักษณะทำได้โดยการนำคำสั่งที่ซ้ำกัน มาสร้างเป็นกระบวนการต่างหาก แล้วเรียกใช้ในกระบวนการ castle ขอให้ทดลองปฏิบัติ และควรเรียกใช้กระบวนการ rectangle isosceles และกระบวนการสร้างรูปทรงเรขาคณิตที่ทำมาแล้ว ประกอบกับคำสั่งในการเคลื่อนที่สร้างเป็นภาพตามจินตนาการ ภาพของใครจะดูดีที่สุด