บทที่ 6
โครงสร้างการควบคุมโปรแกรมแบบทำซ้ำ
Iteration Control Structure

บทนำ

ในการแก้ปัญหาทางวิทยาการคำนวณ มีหลายสถานการณ์ที่คำสั่งบางอย่างจำเป็นต้องถูกดำเนินซ้ำมากกว่าหนึ่งครั้ง เช่น การแสดงตัวเลขตั้งแต่ 1 ถึง 10 การคำนวณผลรวมของคะแนนหลายค่า การตรวจสอบข้อมูลทีละรายการ หรือการทำงานซ้ำจนกว่าจะเป็นไปตามเงื่อนไขที่กำหนด หากเขียนคำสั่งเหล่านี้แบบเรียงลำดับทีละบรรทัดทั้งหมด จะทำให้โปรแกรมยาว ซ้ำซ้อน และยากต่อการจัดการ ดังนั้นจึงจำเป็นต้องมีโครงสร้างการควบคุมที่ช่วยให้โปรแกรมสามารถดำเนินคำสั่งเดิมซ้ำได้อย่างเป็นระบบ ซึ่งก็คือ “โครงสร้างการควบคุมแบบทำซ้ำ”

โครงสร้างการควบคุมแบบทำซ้ำ (Iteration Structure หรือ Loop Structure) เป็นหนึ่งในโครงสร้างพื้นฐานสำคัญของการเขียนโปรแกรม ร่วมกับโครงสร้างแบบเรียงลำดับและโครงสร้างแบบทางเลือก โครงสร้างนี้ช่วยให้โปรแกรมทำงานซ้ำภายใต้จำนวนรอบหรือเงื่อนไขที่กำหนดไว้อย่างชัดเจน ส่งผลให้การเขียนโปรแกรมมีประสิทธิภาพมากขึ้น ลดความซ้ำซ้อนของคำสั่ง และช่วยให้การแก้ปัญหาที่มีลักษณะเกิดขึ้นซ้ำทำได้อย่างเหมาะสม

สำหรับนักศึกษาครู ความเข้าใจเรื่องการทำซ้ำมีความสำคัญทั้งในเชิงเนื้อหาวิทยาการคำนวณและในเชิงวิธีสอน เพราะการทำซ้ำมิใช่เพียงกลไกของโปรแกรมเท่านั้น แต่ยังเป็นวิธีคิดที่เกี่ยวข้องกับการมองเห็นรูปแบบและความเป็นระบบในกิจกรรมต่าง ๆ ของชีวิตประจำวัน เช่น การเดินขึ้นบันไดทีละขั้น การกวาดห้องจนสะอาด การตอบคำถามหลายข้อ หรือการทบทวนบทเรียนจนเข้าใจ ครูที่เข้าใจหลักการของการทำซ้ำอย่างชัดเจนจะสามารถออกแบบกิจกรรมที่ช่วยให้ผู้เรียนเห็นภาพของการทำงานแบบวนซ้ำได้อย่างเป็นรูปธรรม และสามารถเชื่อมโยงจากสถานการณ์ใกล้ตัวไปสู่ผังงานและการเขียนโปรแกรมได้อย่างมีประสิทธิภาพ

บทนี้มุ่งอธิบายความหมายและลักษณะของโครงสร้างการควบคุมแบบทำซ้ำ ประเภทของการทำซ้ำและการประยุกต์ใช้ในการแก้ปัญหา การออกแบบผังงานแบบทำซ้ำ การใช้โปรแกรม Flowgorithm ในการเขียนและทดสอบผังงานแบบทำซ้ำ ตลอดจนการออกแบบกิจกรรมการเรียนรู้เพื่อ

พัฒนาความเข้าใจเรื่องการทำซ้ำสำหรับผู้เรียนระดับประถมศึกษาและมัธยมศึกษา เพื่อให้นักศึกษาครูสามารถนำความรู้ไปใช้ได้ทั้งในด้านวิชาการและการจัดการเรียนรู้ในอนาคต

6.1 ความหมายและลักษณะของโครงสร้างการควบคุมแบบทำซ้ำ (Iteration Structure)

โครงสร้างการควบคุมแบบทำซ้ำ (Iteration Structure) หมายถึง รูปแบบการทำงานของโปรแกรมที่สั่งให้ชุดคำสั่งหนึ่งหรือหลายคำสั่งถูกดำเนินการซ้ำหลายครั้งตามจำนวนรอบหรือภายใต้เงื่อนไขที่กำหนดไว้ จนกว่าจะครบตามเงื่อนไขของการวนซ้ำหรือสิ้นสุดตามที่ออกแบบไว้ โครงสร้างนี้ช่วยลดความซ้ำซ้อนของการเขียนคำสั่ง และทำให้โปรแกรมสามารถจัดการกับงานที่ต้องทำซ้ำได้อย่างมีประสิทธิภาพ

สาระสำคัญของโครงสร้างแบบทำซ้ำอยู่ที่แนวคิดเรื่อง “การวนกลับ” ไปทำคำสั่งเดิมอีกครั้ง โดยแต่ละรอบของการทำงานจะต้องมีการตรวจสอบเงื่อนไขบางประการ เช่น ทำซ้ำจำนวน 10 ครั้ง ทำซ้ำตราบใดที่ค่าคะแนนยังไม่ครบ หรือทำซ้ำจนกว่าผู้ใช้จะป้อนค่าที่ถูกต้อง โครงสร้างการทำซ้ำจึงมีความสัมพันธ์อย่างใกล้ชิดกับแนวคิดเรื่องเงื่อนไข การตรวจสอบค่า และการควบคุมลำดับการทำงานของโปรแกรม

ลักษณะสำคัญของโครงสร้างแบบทำซ้ำสามารถสรุปได้ดังนี้ ประการแรก คือ มีชุดคำสั่งที่ต้องการให้ทำงานซ้ำ ประการที่สอง คือ มีตัวควบคุมการทำซ้ำ เช่น ตัวแปรนับรอบหรือเงื่อนไขในการตรวจสอบ ประการที่สาม คือ มีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของการวนซ้ำอย่างชัดเจน และประการที่สี่ คือ ต้องมีเงื่อนไขที่ทำให้การวนซ้ำสิ้นสุด มิฉะนั้นอาจเกิดการวนซ้ำไม่รู้จบ ซึ่งเรียกว่า วนซ้ำไม่สิ้นสุด (Infinite Loop)

ในชีวิตประจำวัน แนวคิดของการทำซ้ำสามารถพบได้ในกิจกรรมทั่วไป เช่น การเดินขึ้นบันไดทีละขั้นจนถึงชั้นเรียน การล้างจานทีละใบจนหมด การทบทวนคำศัพท์วันละหลายคำ หรือการวิ่งรอบสนามตามจำนวนรอบที่กำหนด กิจกรรมเหล่านี้ล้วนมีลักษณะร่วมกันคือ มีการทำสิ่งเดิมซ้ำหลายครั้งภายใต้จุดมุ่งหมายหรือเงื่อนไขบางประการ จึงสามารถใช้เป็นพื้นฐานในการอธิบายแนวคิดเรื่องการทำซ้ำในเชิงโปรแกรมได้อย่างดี

ในทางการเขียนโปรแกรม โครงสร้างแบบทำซ้ำมีประโยชน์มากเมื่อโปรแกรมต้องจัดการกับข้อมูลจำนวนมาก หรือเมื่อต้องทำงานลักษณะเดียวกันหลายครั้ง เช่น การแสดงลำดับตัวเลข การหาผลรวมของจำนวนหลายค่า การตรวจคำตอบหลายข้อ หรือการประมวลผลข้อมูลทีละรายการ หากไม่ใช้

โครงสร้างแบบทำซ้ำ ผู้เขียนโปรแกรมอาจต้องเขียนคำสั่งเดิมซ้ำหลายบรรทัด ซึ่งไม่เหมาะสมและเพิ่มโอกาสเกิดข้อผิดพลาด

สำหรับนักศึกษาครู การเข้าใจโครงสร้างแบบทำซ้ำควรครอบคลุมทั้งในมิติของตรรกะการเขียนโปรแกรมและการนำไปออกแบบการเรียนรู้ ครูควรสามารถอธิบายให้ผู้เรียนเห็นว่า การทำซ้ำไม่ได้เป็นเรื่องของโค้ดเพียงอย่างเดียว แต่เป็นรูปแบบการคิดที่เกี่ยวข้องกับกิจกรรมซ้ำ ๆ ในชีวิตจริง และสามารถใช้กิจกรรมเชิงปฏิบัติช่วยสร้างความเข้าใจให้แก่ผู้เรียนก่อนที่จะนำเข้าสู่ผังงานและโปรแกรม

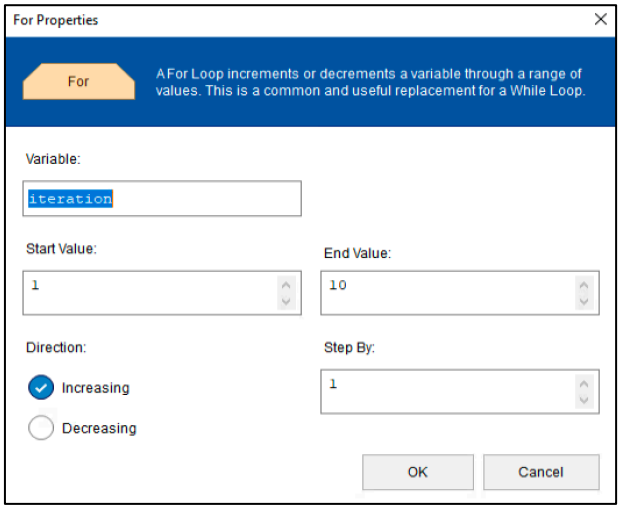
กล่าวโดยสรุป โครงสร้างการควบคุมแบบทำซ้ำเป็นกลไกสำคัญที่ช่วยให้โปรแกรมทำงานซ้ำอย่างเป็นระบบตามจำนวนรอบหรือเงื่อนไขที่กำหนด เป็นพื้นฐานสำคัญของการเขียนโปรแกรมที่มีประสิทธิภาพ และเป็นแนวคิดที่ครูสามารถนำไปเชื่อมโยงกับประสบการณ์ของผู้เรียนได้อย่างกว้างขวาง

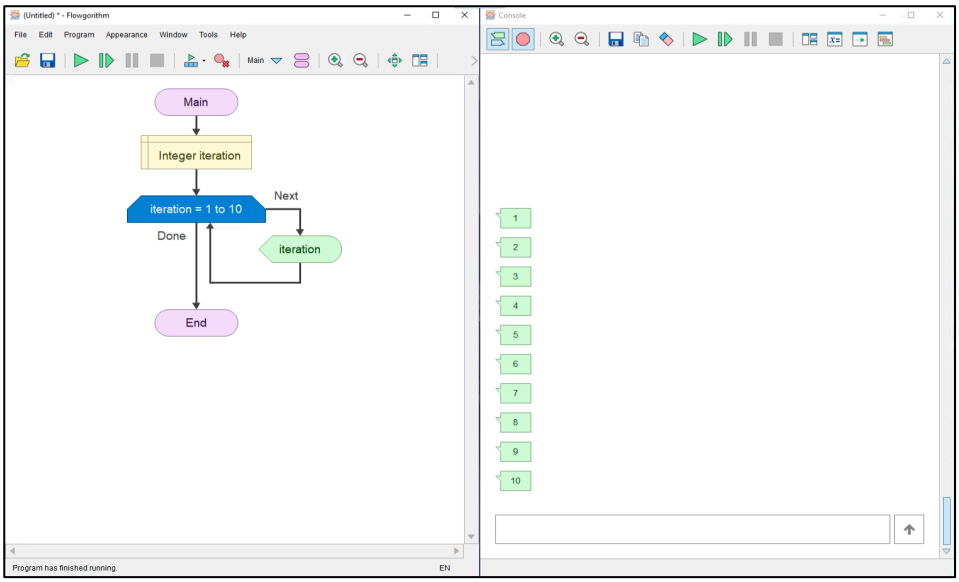
6.2 ประเภทของการทำซ้ำและการประยุกต์ใช้ในการแก้ปัญหา

โครงสร้างการทำซ้ำในทางวิทยาการคำนวณสามารถจำแนกได้หลายลักษณะตามวิธีการควบคุมการวนซ้ำ โดยการเข้าใจประเภทของการทำซ้ำจะช่วยให้ผู้เรียนเลือกใช้แนวทางที่เหมาะสมกับลักษณะของปัญหาและออกแบบอัลกอริทึมได้อย่างมีประสิทธิภาพ โดยทั่วไป ประเภทของการทำซ้ำที่สำคัญสามารถอธิบายได้ดังนี้

6.2.1 การทำซ้ำตามจำนวนรอบที่กำหนด (For)

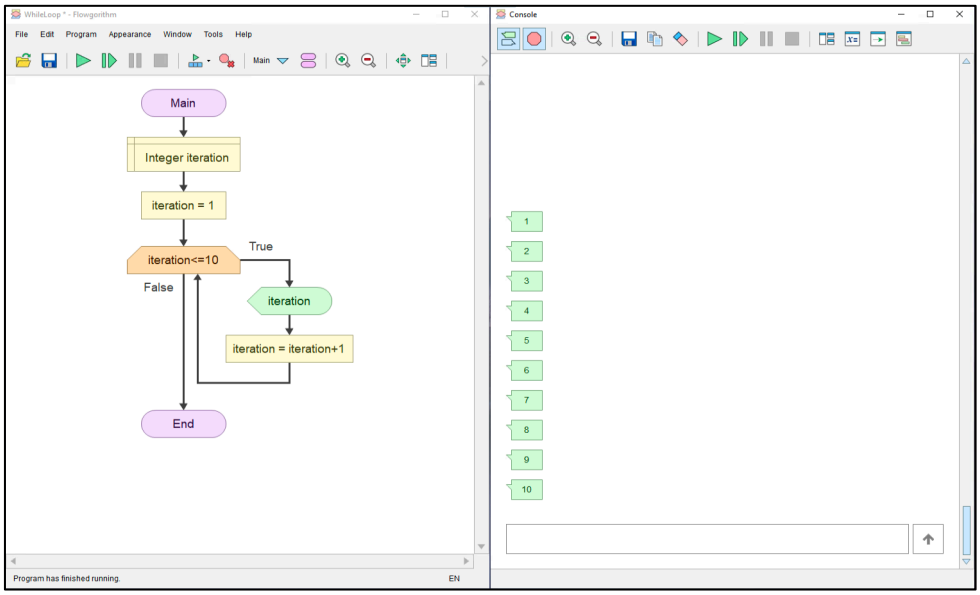
การทำซ้ำประเภทนี้เกิดขึ้นเมื่อผู้เขียนโปรแกรมทราบล่วงหน้าว่าต้องการให้ชุดคำสั่งทำงานกี่ครั้ง เช่น แสดงตัวเลข 1 ถึง 10 พิมพ์ชื่อนักเรียน 5 ครั้ง หรือคำนวณผลรวมของคะแนน 10 ค่า การทำซ้ำลักษณะนี้มักใช้ตัวแปรนับรอบเพื่อควบคุมจำนวนครั้งของการทำงาน และเหมาะกับปัญหาที่มีจำนวนรอบแน่นอน





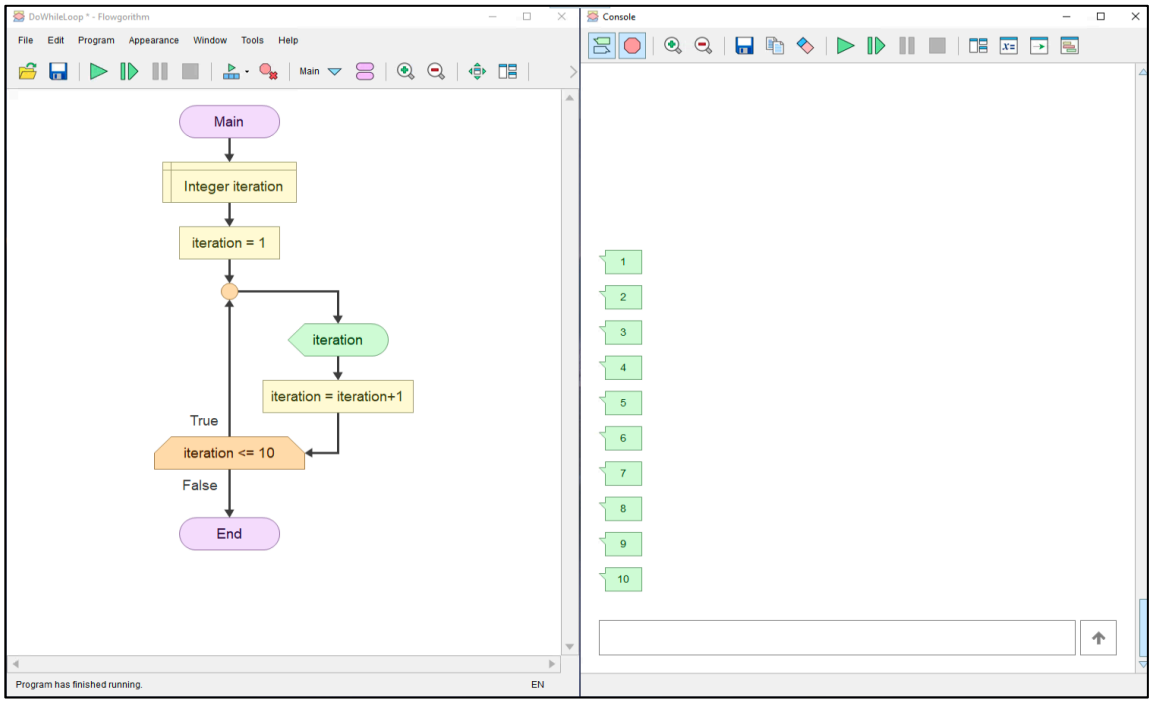
6.2.2 การทำซ้ำตามเงื่อนไขก่อนทำงาน (While)

การทำซ้ำประเภทนี้จะตรวจสอบเงื่อนไขก่อนทุกครั้ง หากเงื่อนไขเป็นจริงจึงจะทำชุดคำสั่งซ้ำต่อไป แต่หากเงื่อนไขเป็นเท็จจะออกจากการวนซ้ำทันที ตัวอย่างเช่น การแสดงตัวเลขตั้งแต่ 1 จนถึง 10 โดยทำซ้ำตราบใดที่ค่าตัวแปรยังไม่เกิน 10 หรือการรับข้อมูลจากผู้ใช้ไปเรื่อย ๆ จนกว่าผู้ใช้จะป้อนค่าที่ถูกต้อง



6.2.3 การทำซ้ำตามเงื่อนไขหลังทำงาน (Do-While)

การทำซ้ำประเภทนี้จะให้ชุดคำสั่งทำงานก่อนหนึ่งรอบ แล้วจึงตรวจสอบเงื่อนไขว่าจะทำซ้ำต่อหรือไม่ ลักษณะนี้เหมาะกับสถานการณ์ที่ต้องการให้โปรแกรมทำงานอย่างน้อยหนึ่งครั้งก่อนเสมอ เช่น การแสดงเมนูให้ผู้ใช้เลือกก่อน แล้วจึงตรวจสอบว่าต้องการทำรายการต่อหรือไม่



การเลือกใช้ประเภทของการทำซ้ำควรพิจารณาจากลักษณะของปัญหา หากทราบจำนวนรอบแน่นอน การใช้การทำซ้ำตามจำนวนรอบ (For) จะเหมาะสมและเข้าใจง่าย แต่หากไม่ทราบจำนวนรอบล่วงหน้าและขึ้นอยู่กับเงื่อนไข เช่น การรับค่าจากผู้ใช้จนกว่าจะถูกต้อง การใช้การทำซ้ำตามเงื่อนไข (While) จะมีความยืดหยุ่นมากกว่า

ในทางปฏิบัติ การทำซ้ำถูกนำไปประยุกต์ใช้อย่างกว้างขวางในการแก้ปัญหาต่าง ๆ เช่น การคำนวณผลรวมและค่าเฉลี่ยของคะแนนหลายค่า การตรวจสอบข้อมูลทีละรายการ การแสดงแม่สูตรคูณ การหาค่ามากที่สุดหรือค่าน้อยที่สุดในชุดข้อมูล การสร้างแบบฝึกหัดที่มีข้อจำนวนมาก หรือการทำงานซ้ำในเกมและโปรแกรมเชิงโต้ตอบ ปัญหาเหล่านี้ล้วนต้องอาศัยการทำงานซ้ำอย่างเป็นระบบเพื่อให้โปรแกรมทำงานได้อย่างครบถ้วน

สำหรับผู้เรียน การทำความเข้าใจประเภทของการทำซ้ำจะช่วยให้มองเห็นความแตกต่างของปัญหา และสามารถเลือกใช้วิธีการที่เหมาะสมในการเขียนผังงานและโปรแกรมได้ นอกจากนี้ ยังช่วยพัฒนาทักษะการวิเคราะห์ปัญหา เพราะผู้เรียนต้องพิจารณาว่าปัญหานั้นควบคุมด้วยจำนวนรอบที่แน่นอน หรือด้วยเงื่อนไขที่อาจเปลี่ยนแปลงได้

สำหรับนักศึกษาครู ควรเชื่อมโยงแนวคิดเรื่องการทำซ้ำกับกิจกรรมในชีวิตประจำวันและการจัดกิจกรรมการเรียนรู้ เช่น การฝึกให้นักเรียนปรบมือ 5 ครั้ง การเดินไปข้างหน้าจนถึงเส้นที่กำหนด การตอบคำถามจนกว่าจะถูก หรือการทำกิจกรรมซ้ำภายใต้เงื่อนไขบางอย่าง กิจกรรมเช่นนี้จะช่วยให้ผู้เรียนเข้าใจประเภทของการทำซ้ำจากประสบการณ์ตรง ก่อนที่จะเรียนรู้ในรูปของผังงานหรือภาษาโปรแกรม

กล่าวโดยสรุป ประเภทของการทำซ้ำมีความแตกต่างกันตามวิธีการควบคุมรอบของการทำงาน การเข้าใจลักษณะเหล่านี้จะช่วยให้ผู้เรียนสามารถออกแบบวิธีแก้ปัญหาได้อย่างเหมาะสม และเป็นพื้นฐานสำคัญของการพัฒนาโปรแกรมที่มีประสิทธิภาพ

6.3 การออกแบบผังงานแบบทำซ้ำ

ผังงานแบบทำซ้ำเป็นการแสดงลำดับขั้นตอนการทำงานของโปรแกรมที่มีการวนกลับไปทำชุดคำสั่งเดิมซ้ำหลายครั้งตามจำนวนรอบหรือเงื่อนไขที่กำหนด การออกแบบผังงานลักษณะนี้ช่วยให้ผู้เรียนเห็นภาพของการวนซ้ำอย่างชัดเจน เข้าใจจุดเริ่มต้น จุดตรวจสอบเงื่อนไข ทิศทางการไหลของงาน และจุดสิ้นสุดของการทำซ้ำได้ดียิ่งขึ้น

สัญลักษณ์ที่ใช้ในผังงานแบบทำซ้ำยังคงอาศัยสัญลักษณ์พื้นฐานเช่นเดียวกับผังงานทั่วไป ได้แก่ สัญลักษณ์เริ่มต้นและสิ้นสุด สัญลักษณ์กระบวนการ สัญลักษณ์ข้อมูลเข้าและข้อมูลออก และสัญลักษณ์การตัดสินใจ อย่างไรก็ตาม สิ่งที่ทำให้ผังงานแบบทำซ้ำแตกต่างจากผังงานแบบเรียงลำดับและแบบทางเลือก คือ การมีเส้นทางย้อนกลับจากชุดคำสั่งไปยังจุดตรวจสอบเงื่อนไข หรือกลับไปยังจุดเริ่มต้นของรอบถัดไป

หลักการออกแบบผังงานแบบทำซ้ำมีประเด็นสำคัญดังนี้ ประการแรก ต้องกำหนดให้ชัดเจนว่าคำสั่งใดคือส่วนที่ต้องทำซ้ำ ประการที่สอง ต้องกำหนดเงื่อนไขหรือจำนวนรอบของการทำซ้ำให้แน่นอน ประการที่สาม ต้องมีการปรับค่าของตัวแปรควบคุมในแต่ละรอบ เช่น การเพิ่มค่า ลดค่า หรือเปลี่ยนสถานะของตัวแปร และประการที่สี่ ต้องออกแบบให้มีเงื่อนไขสิ้นสุดที่ชัดเจนเพื่อป้องกันการวนซ้ำไม่สิ้นสุด

ตัวอย่างผังงานแบบทำซ้ำที่เข้าใจง่าย คือ การแสดงตัวเลข 1 ถึง 5 โดยอาจออกแบบดังนี้

1. เริ่มต้น
2. กำหนดค่า i = 1
3. ตรวจสอบว่า i <= 5 หรือไม่
4. หากจริง ให้แสดงค่า i
5. เพิ่มค่า i = i + 1
6. กลับไปตรวจสอบเงื่อนไขอีกครั้ง
7. หากเท็จ ให้สิ้นสุดการทำงาน

อีกตัวอย่างหนึ่ง คือ การหาผลรวมของตัวเลข 1 ถึง 10 โดยอาจออกแบบให้มีตัวแปร sum สำหรับเก็บผลรวม และตัวแปร i สำหรับนับรอบ จากนั้นให้วนซ้ำเพิ่มค่า sum ด้วยค่า i ในแต่ละรอบ แล้วเพิ่มค่า i ทีละ 1 จนครบเงื่อนไข การออกแบบเช่นนี้ช่วยให้ผู้เรียนเห็นบทบาทของตัวแปรควบคุมและตัวแปรสะสมผลอย่างชัดเจน

การฝึกออกแบบผังงานแบบทำซ้ำมีประโยชน์หลายประการ ได้แก่ ช่วยพัฒนาการคิดอย่างเป็นระบบ ช่วยให้ผู้เรียนเห็นความสัมพันธ์ระหว่างเงื่อนไขกับการทำงานซ้ำ และช่วยเตรียมความพร้อมก่อนการเขียนโปรแกรมจริง นอกจากนี้ การออกแบบผังงานยังช่วยให้ครูตรวจสอบได้ง่ายว่าผู้เรียนเข้าใจจุดที่ต้องวนซ้ำ จุดที่ต้องเปลี่ยนค่าตัวแปร และจุดที่ควรหยุดการทำงานหรือไม่

สำหรับนักศึกษาครู การสอนผังงานแบบทำซ้ำควรค่อย ๆ พัฒนาจากสถานการณ์ที่เป็นรูปธรรม เช่น การเดินรอบสนาม 3 รอบ การปรบมือ 5 ครั้ง หรือการตอบคำถามจนกว่าจะถูก แล้วจึงเชื่อมโยงไปสู่การเขียนผังงานจากโจทย์คำนวณอย่างง่าย ครูอาจใช้กิจกรรมให้ผู้เรียนสวมบทบาทเป็น “ตัวแปร” และ “เงื่อนไข” เพื่อช่วยให้เห็นภาพการทำงานของการวนซ้ำอย่างสนุกและเข้าใจง่าย

กล่าวโดยสรุป การออกแบบผังงานแบบทำซ้ำเป็นกระบวนการสำคัญที่ช่วยให้ผู้เรียนเข้าใจการทำงานแบบวนซ้ำของโปรแกรมอย่างเป็นรูปธรรม และเป็นพื้นฐานสำคัญก่อนการพัฒนาไปสู่การเขียนโปรแกรมจริงด้วยภาษาโปรแกรมต่าง ๆ

6.4 การใช้โปรแกรม Flowgorithm ในการเขียนและทดสอบผังงานแบบทำซ้ำ

Flowgorithm เป็นโปรแกรมที่มีประโยชน์อย่างมากต่อการเรียนรู้โครงสร้างการควบคุมแบบทำซ้ำ เพราะช่วยให้ผู้เรียนสามารถสร้างผังงานที่มีการวนซ้ำ และทดสอบการทำงานของแต่ละรอบได้อย่างชัดเจน ผู้เรียนจึงไม่เพียงเห็นภาพรวมของผังงาน แต่ยังสามารถสังเกตลำดับการทำงานจริงของโปรแกรมและตรวจสอบค่าตัวแปรที่เปลี่ยนแปลงไปในแต่ละรอบได้อีกด้วย

ในการใช้ Flowgorithm เขียนผังงานแบบทำซ้ำ ผู้เรียนควรเริ่มจากการวิเคราะห์โจทย์ก่อนว่าปัญหานั้นต้องการให้ทำซ้ำกี่ครั้ง หรือทำซ้ำจนกว่าเงื่อนไขใดจะเป็นจริง จากนั้นจึงกำหนดตัวแปรที่จำเป็น เช่น ตัวแปรนับรอบ ตัวแปรเก็บผลรวม หรือตัวแปรสำหรับรับข้อมูล แล้วจึงเลือกใช้รูปแบบของการทำซ้ำที่เหมาะสมภายในโปรแกรม

ตัวอย่างเช่น หากต้องการสร้างผังงานเพื่อแสดงตัวเลข 1 ถึง 5 ผู้เรียนอาจกำหนดตัวแปร i และกำหนดค่าเริ่มต้น i = 1 จากนั้นใช้โครงสร้างทำซ้ำที่ตรวจสอบว่า i <= 5 หากเงื่อนไขเป็นจริง ให้แสดงค่า i แล้วเพิ่มค่า i ขึ้นทีละ 1 จากนั้นกลับไปตรวจสอบเงื่อนไขอีกครั้ง จนกระทั่ง i มีค่ามากกว่า 5 จึงสิ้นสุดการทำงาน

อีกตัวอย่างหนึ่ง คือ การหาผลรวมของตัวเลข 1 ถึง 10 ผู้เรียนอาจกำหนดตัวแปร i = 1 และ sum = 0 จากนั้นให้วนซ้ำตราบใดที่ i <= 10 โดยในแต่ละรอบให้คำนวณ sum = sum + i แล้วเพิ่มค่า i = i + 1 เมื่อครบเงื่อนไขแล้วจึงแสดงค่าของ sum ออกมา ตัวอย่างนี้ช่วยให้ผู้เรียนเข้าใจบทบาทของตัวแปรนับรอบและตัวแปรสะสมผลได้อย่างชัดเจน

จุดเด่นของ Flowgorithm คือ ผู้เรียนสามารถ Run ผังงานและสังเกตผลการทำงานทีละขั้นตอน หากผังงานมีข้อผิดพลาด เช่น ลืมเพิ่มค่าตัวแปรนับรอบ เขียนเงื่อนไขผิด หรือกำหนดค่าเริ่มต้นไม่เหมาะสม โปรแกรมจะช่วยให้ผู้เรียนเห็นพฤติกรรมที่ผิดปกติ เช่น วนซ้ำไม่สิ้นสุด หรือได้ผลลัพธ์ไม่ตรงตามคาดหมาย การสังเกตเช่นนี้ช่วยให้ผู้เรียนเรียนรู้จากข้อผิดพลาดได้อย่างเป็นรูปธรรม

นอกจากนี้ Flowgorithm ยังเอื้อต่อการอภิปรายในชั้นเรียน ครูสามารถใช้ผังงานตัวอย่างที่ตั้งใจออกแบบให้มีข้อผิดพลาด แล้วให้ผู้เรียนร่วมกันวิเคราะห์ว่าเหตุใดโปรแกรมจึงทำงานผิด เช่น เหตุใดจึงวนซ้ำไม่หยุด หรือเหตุใดจึงแสดงผลเพียงบางค่า กิจกรรมลักษณะนี้ช่วยพัฒนาทั้งทักษะการคิดวิเคราะห์และการอธิบายเหตุผลของผู้เรียน

สำหรับนักศึกษาครู การใช้ Flowgorithm ควรเชื่อมโยงไปสู่การออกแบบกิจกรรมการเรียนรู้ เช่น การให้ผู้เรียนทำนายผลลัพธ์ก่อน Run ผังงาน การให้เปรียบเทียบผลลัพธ์เมื่อเปลี่ยนค่าเริ่มต้นหรือเปลี่ยนเงื่อนไข หรือการให้ออกแบบผังงานเองจากโจทย์ที่กำหนดแล้วนำมาแลกเปลี่ยนกับเพื่อน กิจกรรมเหล่านี้จะช่วยให้ครูฝึกทั้งการใช้เทคโนโลยีและการจัดการเรียนรู้แบบส่งเสริมการคิด

กล่าวโดยสรุป การใช้โปรแกรม Flowgorithm ในการเขียนและทดสอบผังงานแบบทำซ้ำช่วยให้ผู้เรียนเข้าใจการวนซ้ำอย่างลึกซึ้งมากขึ้น เพราะสามารถเห็นทั้งโครงสร้างของผังงาน ลำดับการทำงานของโปรแกรม และผลจากการเปลี่ยนแปลงค่าตัวแปรในแต่ละรอบได้อย่างชัดเจน

6.5 การใช้ Trace Table เพื่อวิเคราะห์การทำงานของโครงสร้างแบบทำซ้ำ

Trace Table เป็นเครื่องมือสำคัญที่ใช้ในการติดตามและบันทึกค่าของตัวแปรระหว่างการทำงานของอัลกอริทึม ผังงาน หรือโปรแกรมในแต่ละขั้นตอนอย่างเป็นลำดับ โดยเฉพาะอย่างยิ่งในการเรียนรู้เรื่องโครงสร้างการควบคุมแบบทำซ้ำ ซึ่งผู้เรียนมักเกิดความสับสนเกี่ยวกับจำนวนรอบของการทำงาน การเปลี่ยนแปลงค่าของตัวแปร และเงื่อนไขที่ทำให้การทำซ้ำสิ้นสุด การใช้ Trace Table จึงช่วยทำให้กระบวนการที่มีลักษณะเป็นนามธรรมกลายเป็นสิ่งที่มองเห็นและตรวจสอบได้อย่างชัดเจน

ในบริบทของการเรียนวิทยาการคำนวณ Trace Table ทำหน้าที่เสมือนเครื่องมือช่วย “แกะรอย” การทำงานของโปรแกรม โดยแสดงให้เห็นว่าในแต่ละรอบของการทำซ้ำ ตัวแปรมีค่าเริ่มต้นอย่างไร มีการเปลี่ยนแปลงค่าเมื่อใด และส่งผลต่อผลลัพธ์สุดท้ายอย่างไร เครื่องมือนี้จึงมีประโยชน์อย่างมากต่อผู้เรียนที่กำลังเริ่มต้นทำความเข้าใจแนวคิดเรื่องลูป เพราะช่วยลดความซับซ้อนของการคิดหลายขั้นตอนพร้อมกัน และทำให้สามารถติดตามกระบวนการทำงานของโปรแกรมได้ทีละรอบอย่างเป็นระบบ

สำหรับนักศึกษาครู การใช้ Trace Table มีคุณค่าอย่างยิ่งทั้งในฐานะผู้เรียนและในฐานะผู้สอน กล่าวคือ ในฐานะผู้เรียน Trace Table ช่วยเสริมสร้างความเข้าใจเชิงตรรกะเกี่ยวกับการทำซ้ำได้อย่างลึกซึ้ง ส่วนในฐานะผู้สอน Trace Table เป็นสื่อการอธิบายที่มีประสิทธิภาพ สามารถนำไปใช้ช่วยให้นักเรียนระดับประถมศึกษาและมัธยมศึกษาเข้าใจการทำงานของลูปได้ง่ายขึ้น ลดความสับสน และลดภาระทางปัญญาในการประมวลผลข้อมูลหลายส่วนพร้อมกัน

6.5.1 ความหมายและความสำคัญของ Trace Table

Trace Table หมายถึง ตารางที่ใช้บันทึกค่าของตัวแปรที่เกี่ยวข้องกับการทำงานของโปรแกรมในแต่ละขั้นตอนหรือแต่ละรอบของการประมวลผล โดยตารางนี้มักประกอบด้วยคอลัมน์สำหรับลำดับรอบ เงื่อนไขที่ตรวจสอบ ค่าของตัวแปร และผลลัพธ์ที่เกิดขึ้น จุดมุ่งหมายสำคัญของ Trace Table คือเพื่อช่วยให้ผู้เรียนสามารถวิเคราะห์ลำดับการทำงานของโปรแกรมได้อย่างชัดเจนและตรวจสอบได้ว่าผลลัพธ์สุดท้ายเกิดขึ้นจากกระบวนการใด

ความสำคัญของ Trace Table ต่อการเรียนรู้โครงสร้างแบบทำซ้ำมีหลายประการ ประการแรก คือ ช่วยให้ผู้เรียนมองเห็นค่าของตัวแปรในแต่ละรอบ เช่น ตัวแปรนับรอบ ตัวแปรสะสมผล หรือค่าที่รับเข้ามา ประการที่สอง คือ ช่วยให้เห็นความสัมพันธ์ระหว่างเงื่อนไขการทำซ้ำกับการเปลี่ยนแปลงค่าของตัวแปร ประการที่สาม คือ ช่วยให้ตรวจสอบข้อผิดพลาดได้ง่าย เช่น การกำหนดค่าเริ่มต้นผิด การลืมเพิ่มค่าตัวแปรนับรอบ หรือการกำหนดเงื่อนไขหยุดไม่ถูกต้อง และประการที่สี่ คือ ช่วยให้ผู้เรียนเข้าใจผลลัพธ์สุดท้ายของโปรแกรมอย่างมีเหตุผล

ในแง่การจัดการเรียนรู้ Trace Table ยังมีประโยชน์อย่างมากในการลด Cognitive Load หรือภาระทางปัญญาของผู้เรียน เนื่องจากการทำงานของลูปมักต้องอาศัยการติดตามหลายองค์ประกอบพร้อมกัน ได้แก่ ค่าเริ่มต้นของตัวแปร เงื่อนไขของลูป การเปลี่ยนค่าของตัวแปรในแต่ละรอบ และผลลัพธ์สะสม หากผู้เรียนพยายามจินตนาการสิ่งเหล่านี้ทั้งหมดในใจพร้อมกัน อาจทำให้เกิดความสับสนได้ง่าย แต่เมื่อใช้ Trace Table ผู้เรียนจะสามารถติดตามข้อมูลทีละส่วนและทีละรอบอย่างเป็นขั้นตอน ส่งผลให้การทำความเข้าใจเกิดขึ้นได้ง่ายและเป็นระบบมากขึ้น

6.5.2 องค์ประกอบของ Trace Table

แม้ Trace Table จะสามารถออกแบบได้หลากหลายตามลักษณะของปัญหา แต่โดยทั่วไปองค์ประกอบพื้นฐานมักประกอบด้วยส่วนสำคัญดังนี้

1) ลำดับรอบ (Iteration / รอบที่) ใช้แสดงว่าข้อมูลในแถวนั้นเป็นการทำงานรอบใดของลูป เพื่อให้ผู้เรียนติดตามความต่อเนื่องของการทำซ้ำได้อย่างชัดเจน

2) เงื่อนไขการทำซ้ำ ใช้บันทึกผลของการตรวจสอบเงื่อนไขในแต่ละรอบว่าเป็นจริงหรือเท็จ ซึ่งช่วยให้เห็นว่าลูปยังคงทำงานต่อหรือสิ้นสุดลงในรอบใด

3) ค่าของตัวแปรสำคัญ เช่น ตัวแปรนับรอบ ตัวแปรสะสมผล ตัวแปรข้อมูลนำเข้า หรือค่าชั่วคราวต่าง ๆ การบันทึกค่านี้ช่วยให้ผู้เรียนเห็นว่าตัวแปรมีการเปลี่ยนแปลงอย่างไรในแต่ละรอบ

4) ผลลัพธ์หรือการแสดงผล ในบางกรณีอาจมีคอลัมน์สำหรับแสดงข้อความหรือค่าที่โปรแกรมแสดงออกมาในแต่ละรอบ เพื่อให้เห็นความสัมพันธ์ระหว่างกระบวนการภายในกับผลลัพธ์ภายนอก

องค์ประกอบของ Trace Table ควรเลือกให้เหมาะสมกับวัตถุประสงค์ของการวิเคราะห์ หากต้องการเน้นการเปลี่ยนแปลงของตัวแปรสะสมผล ก็ควรมีคอลัมน์สำหรับตัวแปรนั้นอย่างชัดเจน หากต้องการเน้นการตรวจสอบเงื่อนไข ก็ควรมีคอลัมน์แสดงผลการเปรียบเทียบในแต่ละรอบด้วย ทั้งนี้ การออกแบบตารางที่เหมาะสมจะช่วยให้ผู้เรียนมองเห็นกระบวนการทำงานของลูปได้ง่ายขึ้น

6.5.3 การใช้ Trace Table กับการทำซ้ำตามจำนวนรอบ

การทำซ้ำตามจำนวนรอบที่กำหนดเป็นกรณีที่เหมาะสมอย่างยิ่งสำหรับการเริ่มต้นใช้ Trace Table เพราะมีลักษณะตรงไปตรงมาและช่วยให้ผู้เรียนเห็นรูปแบบการเปลี่ยนแปลงของตัวแปรได้ชัดเจน ตัวอย่างเช่น การแสดงตัวเลข 1 ถึง 5 อาจกำหนดขั้นตอนการทำงานดังนี้

1. กำหนดค่าเริ่มต้นให้ i = 1
2. ตรวจสอบว่า i <= 5 หรือไม่
3. หากจริง ให้แสดงค่า i
4. เพิ่มค่า i = i + 1
5. กลับไปตรวจสอบเงื่อนไขอีกครั้ง

Trace Table สำหรับกรณีนี้อาจเขียนได้ดังนี้

รอบที่	i ก่อนตรวจสอบ	เงื่อนไข i <= 5	แสดงผล	i หลังเพิ่มค่า
1	1	จริง	1	2
2	2	จริง	2	3
3	3	จริง	3	4
4	4	จริง	4	5
5	5	จริง	5	6
6	6	เท็จ	-	-

จากตารางจะเห็นได้อย่างชัดเจนว่า ตัวแปร i เริ่มต้นที่ 1 และเพิ่มค่าทีละ 1 ในแต่ละรอบ จนกระทั่งมีค่าเป็น 6 ซึ่งทำให้เงื่อนไข i <= 5 เป็นเท็จ ลูปจึงหยุดทำงาน การใช้ Trace Table ลักษณะนี้ช่วยให้ผู้เรียนเข้าใจว่า แม้โปรแกรมจะแสดงผลเพียง 1 ถึง 5 แต่การตรวจสอบเงื่อนไขยังคงเกิดขึ้นอีกรอบหนึ่งก่อนสิ้นสุดการทำงาน

อีกตัวอย่างหนึ่ง เช่น การหาผลรวมของตัวเลข 1 ถึง 4 โดยมีขั้นตอนดังนี้

1. กำหนดค่า i = 1 และ sum = 0
2. ตรวจสอบว่า i <= 4 หรือไม่
3. หากจริง ให้คำนวณ sum = sum + i
4. เพิ่มค่า i = i + 1
5. กลับไปตรวจสอบเงื่อนไข

Trace Table อาจแสดงได้ดังนี้

รอบที่	i	sum ก่อนบวก	sum หลังบวก	i หลังเพิ่มค่า	เงื่อนไข i <= 4
1	1	0	1	2	จริง
2	2	1	3	3	จริง
3	3	3	6	4	จริง
4	4	6	10	5	จริง
5	5	10	10	-	เท็จ

ตารางนี้ช่วยให้ผู้เรียนมองเห็นบทบาทของตัวแปร 2 ประเภทอย่างชัดเจน คือ ตัวแปร i ที่ทำหน้าที่นับรอบ และตัวแปร sum ที่ทำหน้าที่สะสมผล เมื่อเห็นการเปลี่ยนแปลงค่าทีละรอบ ผู้เรียนจะเข้าใจได้ง่ายขึ้นว่าผลรวมสุดท้ายเกิดขึ้นจากการบวกสะสมอย่างไร

6.5.4 การใช้ Trace Table กับการทำซ้ำตามเงื่อนไข

นอกจากการทำซ้ำตามจำนวนรอบที่แน่นอนแล้ว Trace Table ยังมีประโยชน์อย่างมากกับการทำซ้ำตามเงื่อนไข ซึ่งมักมีความซับซ้อนกว่า เพราะจำนวนรอบอาจไม่สามารถทราบล่วงหน้าได้อย่างแน่นอน ตัวอย่างเช่น การรับค่าคะแนนจากผู้ใช้ไปเรื่อย ๆ จนกว่าจะป้อนค่าที่อยู่ในช่วง 0 ถึง 100

สมมติขั้นตอนการทำงานเป็นดังนี้

1. รับค่าคะแนน
2. ขณะที่คะแนนน้อยกว่า 0 หรือมากกว่า 100 ให้รับค่าคะแนนใหม่
3. เมื่อได้คะแนนที่ถูกต้องแล้ว ให้แสดงข้อความว่า “คะแนนถูกต้อง”

หากผู้ใช้ป้อนค่า 120, -5, 75 ตามลำดับ Trace Table อาจเขียนได้ดังนี้

รอบที่	คะแนนที่รับ	เงื่อนไข คะแนน < 0 หรือ > 100	การทำงาน
1	120	จริง	รับค่าใหม่
2	-5	จริง	รับค่าใหม่
3	75	เท็จ	แสดงผลว่าคะแนนถูกต้อง

ตัวอย่างนี้ช่วยให้ผู้เรียนเข้าใจว่า การทำซ้ำตามเงื่อนไขไม่ได้ขึ้นอยู่กับจำนวนรอบที่ตายตัว แต่ขึ้นอยู่กับผลของข้อมูลที่ได้รับในแต่ละรอบ การใช้ Trace Table จึงช่วยให้มองเห็นจุดสิ้นสุดของลูปได้อย่างเป็นรูปธรรม และช่วยลดความสับสนเกี่ยวกับคำถามที่ว่า “ลูปนี้จะหยุดเมื่อใด”

6.5.5 การใช้ Trace Table เพื่อตรวจสอบข้อผิดพลาดของผังงานและโปรแกรม

ประโยชน์สำคัญอีกประการหนึ่งของ Trace Table คือ การใช้เป็นเครื่องมือวิเคราะห์ข้อผิดพลาดของผังงานและโปรแกรม โดยเฉพาะข้อผิดพลาดที่เกี่ยวข้องกับการทำซ้ำ ซึ่งมักเกิดขึ้นได้บ่อยในผู้เรียนระดับเริ่มต้น เช่น

1. กำหนดค่าเริ่มต้นของตัวแปรไม่ถูกต้อง
2. เขียนเงื่อนไขการทำซ้ำผิด
3. ลืมเพิ่มหรือลดค่าตัวแปรควบคุม
4. บวกสะสมผลผิดลำดับ
5. ทำให้เกิดการวนซ้ำไม่สิ้นสุด

ตัวอย่างเช่น หากผู้เรียนออกแบบลูปแสดงตัวเลข 1 ถึง 5 แต่ลืมเพิ่มค่า i = i + 1 เมื่อสร้าง Trace Table จะพบว่า i มีค่าเป็น 1 ตลอดทุกแถว และเงื่อนไข i <= 5 จะเป็นจริงเสมอ ส่งผลให้โปรแกรมวนซ้ำไม่สิ้นสุด การใช้ตารางจึงช่วยให้ผู้เรียนเห็นข้อผิดพลาดอย่างชัดเจนกว่าการมองจากผังงานหรือโค้ดเพียงอย่างเดียว

ในมุมของครู Trace Table ยังใช้เป็นเครื่องมือวินิจฉัยความเข้าใจของผู้เรียนได้ดี หากผู้เรียนเติมตารางผิด ครูจะสามารถวิเคราะห์ได้ว่าผู้เรียนสับสนที่จุดใด เช่น ไม่เข้าใจว่าต้องตรวจสอบเงื่อนไขก่อนหรือหลังทำงาน ไม่เข้าใจการเปลี่ยนค่าในแต่ละรอบ หรือไม่เข้าใจบทบาทของตัวแปรสะสมผล การวิเคราะห์เช่นนี้จะช่วยให้ครูวางแผนการสอนเสริมได้ตรงจุดมากขึ้น

6.5.6 แนวทางการใช้ Trace Table เพื่อลด Cognitive Load ของผู้เรียน

แนวคิดเรื่อง Cognitive Load หรือภาระทางปัญญา มีความสำคัญอย่างยิ่งต่อการสอนวิทยาการคำนวณ เพราะผู้เรียนมักต้องประมวลผลหลายองค์ประกอบพร้อมกัน เช่น ค่าเริ่มต้นของตัวแปร การเปลี่ยนค่าในแต่ละรอบ เงื่อนไขของลูป และผลลัพธ์ของโปรแกรม หากครูอธิบายทั้งหมดพร้อมกันโดยไม่มีเครื่องมือช่วย ผู้เรียนอาจเกิดความสับสนและไม่สามารถสร้างความเข้าใจที่มั่นคงได้

Trace Table ช่วยลดภาระทางปัญญาได้ด้วยการแยกกระบวนการทำงานของโปรแกรมออกเป็นหน่วยย่อยที่ตรวจสอบได้ทีละรอบ ผู้เรียนไม่จำเป็นต้องจดจำทุกอย่างไว้ในความคิดพร้อมกัน แต่สามารถมองตารางทีละแถวและติดตามการเปลี่ยนแปลงค่าของตัวแปรอย่างเป็นลำดับ วิธีการนี้ช่วยให้ผู้เรียนมุ่งความสนใจไปยังสาระสำคัญของแต่ละช่วงการทำงาน และค่อย ๆ สร้างความเข้าใจภาพรวมจากรายละเอียดทีละส่วน

สำหรับนักศึกษาครู แนวทางการใช้ Trace Table เพื่อลด Cognitive Load อาจดำเนินการได้ดังนี้

1) เริ่มจากตัวอย่างง่ายก่อน ควรเริ่มด้วยลูปที่มีตัวแปรไม่มาก เช่น การแสดงตัวเลข 1 ถึง 5 ก่อน แล้วจึงค่อยเพิ่มความซับซ้อน เช่น การหาผลรวม ค่าเฉลี่ย หรือการรับข้อมูลหลายค่า

2) ใช้ตารางที่มีคอลัมน์เท่าที่จำเป็น ไม่ควรใส่ตัวแปรมากเกินไปในช่วงเริ่มต้น เพราะจะทำให้ผู้เรียนสับสน ควรเลือกเฉพาะตัวแปรที่สำคัญต่อการทำความเข้าใจโจทย์นั้น

3) ให้ผู้เรียนติดตามทีละรอบ ครูควรพาผู้เรียนวิเคราะห์ทีละแถว โดยเน้นคำถาม เช่น “รอบนี้ตัวแปรมีค่าเท่าไร” “เงื่อนไขเป็นจริงหรือเท็จ” “หลังทำคำสั่งแล้วค่าเปลี่ยนเป็นเท่าไร” วิธีนี้ช่วยให้ผู้เรียนไม่ข้ามขั้นตอน

4) เชื่อมโยงกับผังงานและการ Run โปรแกรม หลังจากเติม Trace Table แล้ว ควรให้ผู้เรียนเปรียบเทียบกับผังงานหรือผลจากการ Run ใน Flowgorithm เพื่อให้เห็นว่าตารางไม่ใช่สิ่งแยกขาดจากโปรแกรม แต่เป็นเครื่องมืออธิบายการทำงานของโปรแกรมอย่างเป็นระบบ

5) ใช้ Trace Table เป็นสะพานจากรูปธรรมสู่นามธรรม ในระดับประถมศึกษาและมัธยมศึกษาตอนต้น ครูอาจเริ่มจากกิจกรรมที่เป็นรูปธรรม เช่น การเดินทีละก้าวหรือการนับลูกบอล แล้วจึงสร้างตารางติดตามค่าหรือจำนวนในแต่ละรอบ เพื่อค่อย ๆ เชื่อมโยงไปสู่แนวคิดเรื่องตัวแปรและลูป

กล่าวโดยสรุป Trace Table เป็นเครื่องมือที่มีคุณค่าอย่างยิ่งในการสอนโครงสร้างการควบคุมแบบทำซ้ำ เพราะช่วยให้ผู้เรียนติดตามการเปลี่ยนแปลงค่าของตัวแปรในแต่ละรอบได้อย่างชัดเจน ช่วยตรวจสอบข้อผิดพลาดของผังงานและโปรแกรม และช่วยลด Cognitive Load ในกระบวนการเรียนรู้ สำหรับนักศึกษาครู การเข้าใจและสามารถใช้ Trace Table ได้อย่างเหมาะสม จะเป็นพื้นฐานสำคัญในการอธิบายเรื่องการทำซ้ำให้แก่ผู้เรียนได้ง่ายขึ้น เป็นระบบขึ้น และมีประสิทธิภาพมากขึ้น

6.6 การออกแบบกิจกรรมการเรียนรู้เพื่อพัฒนาความเข้าใจเรื่องการทำซ้ำสำหรับผู้เรียนระดับประถมศึกษาและมัธยมศึกษา

การจัดการเรียนรู้เรื่องโครงสร้างแบบทำซ้ำสำหรับผู้เรียนระดับประถมศึกษาและมัธยมศึกษาควรยึดหลักการสำคัญคือ ทำให้ผู้เรียนเห็นว่าการทำซ้ำเป็นสิ่งที่เกิดขึ้นได้ในชีวิตจริง และสามารถเข้าใจได้ผ่านกิจกรรมที่เป็นรูปธรรม ก่อนที่จะพัฒนาไปสู่การใช้ผังงานและโปรแกรม การออกแบบกิจกรรมจึงควรเน้นความใกล้ตัว ความชัดเจน และการมีส่วนร่วมของผู้เรียน

สำหรับระดับประถมศึกษา กิจกรรมควรมุ่งให้ผู้เรียนสังเกตและปฏิบัติการทำซ้ำจากสถานการณ์ง่าย ๆ เช่น การปรบมือ 5 ครั้ง การกระโดด 3 ครั้ง การเดินตามคำสั่งซ้ำ ๆ หรือการวาดรูปแบบที่ซ้ำกัน ครูอาจใช้บัตรคำสั่ง เกม หรือบทบาทสมมุติเพื่อให้ผู้เรียนเข้าใจว่าคำสั่งเดิมสามารถทำซ้ำได้หลายครั้งโดยไม่ต้องเขียนใหม่ทุกครั้ง จากนั้นจึงเชื่อมโยงไปสู่การวาดภาพลำดับขั้นตอนหรือผังงานอย่างง่าย

สำหรับระดับมัธยมศึกษา ผู้เรียนสามารถเรียนรู้ในระดับที่ซับซ้อนมากขึ้น ครูอาจใช้โจทย์เกี่ยวกับการแสดงลำดับตัวเลข การคำนวณผลรวม การสร้างแม่สูตรคูณ หรือการประมวลผลข้อมูลหลายค่า เพื่อให้ผู้เรียนฝึกวิเคราะห์ว่าปัญหาใดเหมาะกับการใช้โครงสร้างแบบทำซ้ำ จากนั้นให้เขียนอัลกอริทึม วาดผังงาน และทดลองสร้างผังงานใน Flowgorithm

แนวทางการออกแบบกิจกรรมที่เหมาะสมอาจประกอบด้วย

1. การเชื่อมโยงกับสถานการณ์จริง เช่น การวิ่งรอบสนาม การเก็บลูกบอลใส่กล่องทีละลูก หรือการนับจำนวนวัตถุ
2. การใช้กิจกรรม Unplugged เช่น เกมลำดับคำสั่ง เกมการ์ด หรือบทบาทสมมุติเพื่อสาธิตการวนซ้ำ
3. การใช้ผังงานและภาพประกอบเพื่อช่วยให้ผู้เรียนเห็นโครงสร้างของการทำซ้ำ
4. การใช้ Flowgorithm เพื่อให้ผู้เรียนทดลองสร้างและ Run ผังงานด้วยตนเอง
5. การอภิปรายข้อผิดพลาดจากการออกแบบการวนซ้ำ เช่น ลืมปรับค่าตัวแปรหรือตั้งเงื่อนไขผิด

ในการจัดกิจกรรม ครูควรใช้คำถามกระตุ้นการคิด เช่น “เราจะรู้ได้อย่างไรว่าควรหยุดทำซ้ำเมื่อใด” “ถ้าไม่เพิ่มค่าตัวแปรในแต่ละรอบจะเกิดอะไรขึ้น” หรือ “สถานการณ์นี้ควรใช้การทำซ้ำตามจำนวนรอบหรือการทำซ้ำตามเงื่อนไข” คำถามลักษณะนี้จะช่วยให้ผู้เรียนไม่เพียงทำตามตัวอย่าง แต่สามารถเข้าใจเหตุผลของโครงสร้างการทำซ้ำได้อย่างแท้จริง

ในด้านการประเมินผล ครูควรประเมินทั้งความเข้าใจเชิงแนวคิดและทักษะการปฏิบัติ เช่น การตรวจผังงาน การสังเกตการมีส่วนร่วมในกิจกรรม การประเมินผลจากการ Run ผังงานใน Flowgorithm การให้ผู้เรียนอธิบายลำดับการทำงานของโปรแกรม หรือการให้สร้างชิ้นงานหรือใบงานที่สะท้อนความเข้าใจเรื่องการทำซ้ำอย่างเป็นระบบ

สำหรับนักศึกษาครู การออกแบบกิจกรรมเรื่องการทำซ้ำควรเชื่อมโยงไปสู่การจัดทำแผนการจัดการเรียนรู้ โดยกำหนดจุดประสงค์การเรียนรู้ให้ชัดเจน เลือกสื่อและกิจกรรมที่เหมาะกับวัย วาง

ขั้นตอนการจัดการเรียนรู้ให้เป็นลำดับ และกำหนดการประเมินผลที่ครอบคลุมทั้งด้านความรู้ ทักษะ และเจตคติ ทั้งนี้เพื่อให้ผู้เรียนเกิดความเข้าใจเรื่องการทำซ้ำอย่างลึกซึ้งและสามารถนำไปใช้ได้จริง

กล่าวโดยสรุป การออกแบบกิจกรรมการเรียนรู้เพื่อพัฒนาความเข้าใจเรื่องการทำซ้ำควรเริ่มจากประสบการณ์ใกล้ตัว ใช้กิจกรรมที่ผู้เรียนได้ลงมือปฏิบัติ และค่อย ๆ เชื่อมโยงไปสู่การใช้ผังงานและโปรแกรม เพื่อให้ผู้เรียนมองเห็นว่าการทำซ้ำเป็นทั้งแนวคิดพื้นฐานของวิทยาการคำนวณและเครื่องมือสำคัญในการแก้ปัญหาอย่างเป็นระบบ

บทสรุปประจำบทที่ 6

โครงสร้างการควบคุมโปรแกรมแบบทำซ้ำเป็นหนึ่งในโครงสร้างพื้นฐานสำคัญของวิทยาการคำนวณที่ช่วยให้โปรแกรมสามารถดำเนินคำสั่งเดิมซ้ำหลายครั้งตามจำนวนรอบหรือเงื่อนไขที่กำหนดได้อย่างมีประสิทธิภาพ การเข้าใจความหมาย ลักษณะ และประเภทของการทำซ้ำจึงเป็นพื้นฐานสำคัญในการพัฒนาทักษะการคิดเชิงตรรกะ การมองเห็นรูปแบบของปัญหา และการออกแบบวิธีการแก้ปัญหาอย่างเป็นระบบ

การออกแบบผังงานแบบทำซ้ำช่วยให้ผู้เรียนเห็นภาพการทำงานของการวนซ้ำได้อย่างชัดเจน โดยเฉพาะเมื่อใช้ร่วมกับโปรแกรม Flowgorithm ซึ่งช่วยให้ผู้เรียนสามารถสร้าง ทดสอบ และปรับปรุงผังงานได้อย่างเป็นรูปธรรม นอกจากนี้ สำหรับนักศึกษาครู ความเข้าใจเรื่องการทำซ้ำควรเชื่อมโยงไปสู่การออกแบบกิจกรรมการเรียนรู้ที่เหมาะสมกับระดับของผู้เรียน โดยเริ่มจากสถานการณ์ใกล้ตัว กิจกรรมเชิงปฏิบัติ และการใช้สื่อที่ช่วยให้ผู้เรียนค่อย ๆ พัฒนาไปสู่การเข้าใจผังงานและโปรแกรมอย่างมีความหมาย

คำถามท้ายบท

1. จงอธิบายความหมายของโครงสร้างการควบคุมแบบทำซ้ำ และเหตุใดจึงมีความสำคัญต่อการเขียนโปรแกรม
2. โครงสร้างแบบทำซ้ำมีลักษณะสำคัญอะไรบ้าง
3. ประเภทของการทำซ้ำมีอะไรบ้าง และแต่ละประเภทเหมาะกับปัญหาแบบใด
4. เพราะเหตุใดการกำหนดเงื่อนไขสิ้นสุดของการทำซ้ำจึงมีความสำคัญ
5. วนซ้ำไม่สิ้นสุดคืออะไร และอาจเกิดจากสาเหตุใดบ้าง
6. จงอธิบายหลักการออกแบบผังงานแบบทำซ้ำจากโจทย์อย่างง่าย

7. โปรแกรม Flowgorithm มีประโยชน์อย่างไรต่อการเรียนรู้เรื่องการทำซ้ำ
8. หากต้องการออกแบบผังงานเพื่อแสดงตัวเลข 1 ถึง 10 ท่านจะกำหนดตัวแปรและเงื่อนไขอย่างไร
9. หากท่านเป็นครูระดับประถมศึกษา ท่านจะจัดกิจกรรมอย่างไรเพื่อให้ผู้เรียนเข้าใจแนวคิดเรื่องการทำซ้ำ
10. หากท่านเป็นครูระดับมัธยมศึกษา ท่านจะใช้ Flowgorithm เพื่อส่งเสริมความเข้าใจเรื่องการทำซ้ำของผู้เรียนอย่างไร

เอกสารอ้างอิง

กระทรวงศึกษาธิการ. (2560). *ตัวชี้วัดและสาระการเรียนรู้แกนกลาง กลุ่มสาระการเรียนรู้วิทยาศาสตร์ (ฉบับปรับปรุง พ.ศ. 2560) ตามหลักสูตรแกนกลางการศึกษาขั้นพื้นฐาน พุทธศักราช 2551*. กรุงเทพมหานคร: ชุมนุมสหกรณ์การเกษตรแห่งประเทศไทย.

Gaddis, T. (2018). *Starting out with programming logic and design* (5th ed.). Boston, MA: Pearson.

Joyce, D. (2020). *Programming logic and design: Introductory* (9th ed.). Boston, MA: Cengage Learning.

Flowgorithm. (2024). *Flowgorithm documentation*. Retrieved from the official Flowgorithm documentation resources.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.