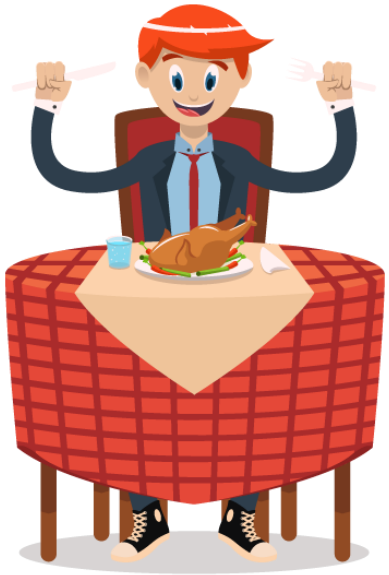
บทที่ 9
การทำซ้ำ

ในศาสตร์ที่เกี่ยวข้องกับคอมพิวเตอร์อย่างการเขียนโปรแกรม การทำซ้ำหรือ loop คือลำดับของชุดคำสั่งที่ทำงานต่อกันอย่างต่อเนื่อง ทำซ้ำแบบเดิมเรื่อย ๆ จนกว่าจะมั่นใจได้ว่าตรงตามเงื่อนไขที่กำหนดไว้แล้ว และเมื่อตรงตามเงื่อนไขที่กำหนดแล้ว เมื่อหยุดทำงานจะคืนค่าผลลัพธ์ที่แตกต่างจากก่อนการเริ่มทำซ้ำด้วย บางครั้งการทำซ้ำจะมีตัวนับจำนวนรอบและหยุดเมื่อตัวนับจำนวนรอบตรงตามที่เงื่อนไขที่กำหนด เช่น ถ้าตัวนับจำนวนรอบเริ่มต้นที่ 1 และเงื่อนไขที่กำหนดคือจำนวน 10 รอบ ตัวนับจำนวนรอบจะเพิ่มขึ้นครั้งละ 1 จนกว่าจะเพิ่มขึ้นถึง 10 จะทำงานทั้งหมด 10 รอบจึงจะหยุดทำงาน และทำงานต่อที่คำสั่งที่นอกเหนือจากส่วนทำซ้ำ และบางครั้งเกิดเหตุการณ์ที่การทำซ้ำทำเรื่อย ๆ ไม่หยุด และใช้พื้นที่หน่วยความจำมากเกินกว่าที่ระบบปฏิบัติการจะรับไหว ระบบปฏิบัติการจะหยุดการทำงานของการทำซ้ำนั้นในที่สุด

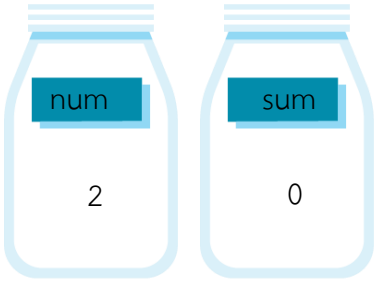
การทำซ้ำที่สามารถพบได้ของโปรแกรมประยุกต์บนอุปกรณ์อิเล็กทรอนิกส์อย่างโทรศัพท์เคลื่อนที่ และคอมพิวเตอร์ เช่น การเปิดเพลงวนซ้ำตามที่ผู้ใช้กำหนด เพียงแค่ผู้ใช้กดปุ่มที่ลักษณ์เป็นลูกศรชี้วนกันเป็นวงกลมหรือสี่เหลี่ยมเท่านั้น หรือการวิ่งเก็บรอบ ซึ่งการวิ่งรอบสนามนับเป็นการทำซ้ำเช่นกัน โดยมีเงื่อนไขที่สามารถหยุดวิ่งได้ก็ต่อเมื่อจำนวนรอบทั้งหมดครบตามจำนวนที่อาจารย์ประจำรายวิชากำหนดนั่นเอง การวิ่งรอบสนามตามที่อาจารย์ประจำรายวิชาสั่งนี้เป็นการทำซ้ำที่มีรอบแน่นอน ตัวอย่างการทำซ้ำที่จำนวนรอบไม่แน่นอนแต่ใช้เงื่อนไขอื่นควบคุมให้หยุดทำงาน เช่น การรับประทานอาหาร การรับประทานอาหารไม่สามารถกำหนดจำนวนคำของอาหารที่ตักเข้าปากได้แน่ชัดว่าจะต้อง 5 คำหรือ 10 คำ แต่เงื่อนไขที่ทำให้หยุดรับประทานอาหารคือ หยุดรับประทานอาหารเมื่ออิ่มเท่านั้น โดยอาจจะนับจำนวนคำของอาหารที่ตักเข้าปากหรือไม่ก็ได้



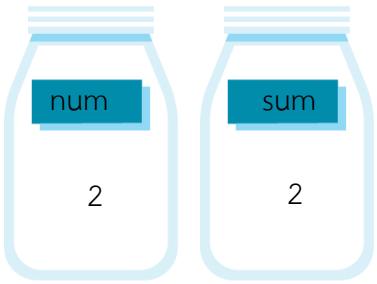
ตัวอย่าง : หาผลรวม

ตัวอย่างนี้จะแสดงการหาผลรวมของตัวเลข 4 จำนวน ประกอบด้วย 2, 4, 90, 52 สามารถบวกได้ครั้งละ 2 จำนวนเท่านั้น และจะต้องเก็บผลลัพธ์ของการบวกแต่ละคู่ลงในตัวแปร sum โดยผู้เขียนจะใช้อุปกรณ์ในการอธิบาย เพื่อง่ายต่อการทำความเข้าใจ ดังนี้

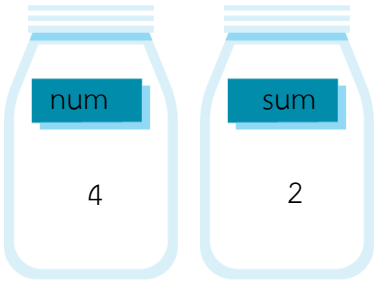
ขวดโหล 2 ใบ ใบที่หนึ่งติดป้ายว่า num ใบที่สองติดป้ายว่า sum ขวดโหลใบที่หนึ่งเก็บตัวเลขไว้หนึ่งจำนวน ส่วนใบที่ติดป้าย sum ใส่เลข 0 ลงไปก่อน ดังภาพ



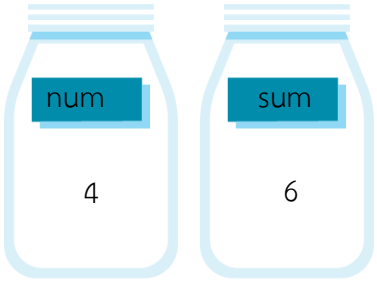
จากนั้น ทำการนำตัวเลขในขวดโหลที่ติดป้าย num มาบวกกับตัวเลขในขวดโหล sum จะได้ผลลัพธ์เป็น 2 นำเลข 0 ในโหล sum ออก และนำเลข 2 ที่เป็นผลลัพธ์เก็บลงในโหล sum แทน



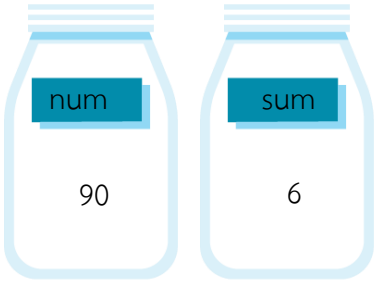
ต่อไปนำเลข 2 จากขวดโหล num ออกและนำเลข 4 ซึ่งเป็นเลขตัวถัดไปมาใส่แทน



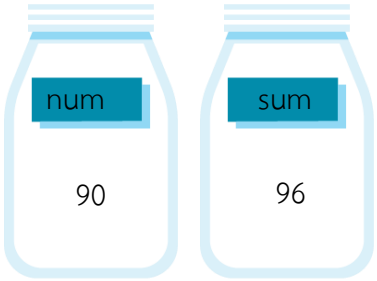
จากนั้นสังเกตเลข 2 ในขวดโหล sum และสังเกตเลข 4 ในขวดโหล num นำทั้งสองจำนวนบวกกันจะได้ผลลัพธ์เป็น 6 นำเลข 6 ซึ่งเป็นผลลัพธ์ใส่ลงในขวดโหล sum แทนที่และนำผลลัพธ์เดิมคือเลข 2 ออกไป



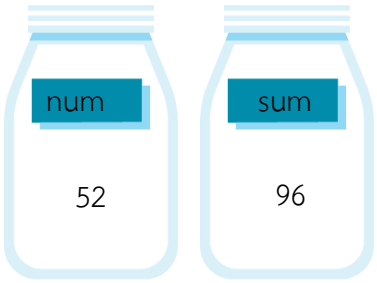
ลำดับต่อมานำเลข 4 ในขวดโหล num ออก และนำเลข 90 ใส่ลงไปแทน



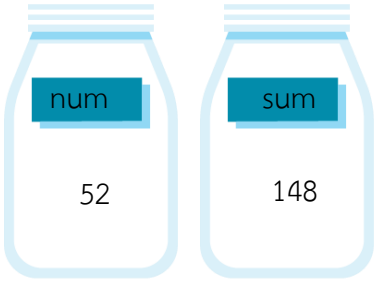
จากนั้นนำเลข 90 ที่อยู่ในขวดโหล num บวกกับเลข 6 ที่อยู่ในขวดโหล sum ได้ผลลัพธ์เป็น 96 และนำเลข 96 ใส่ลงในขวดโหล sum แทนผลลัพธ์เดิมคือ 6



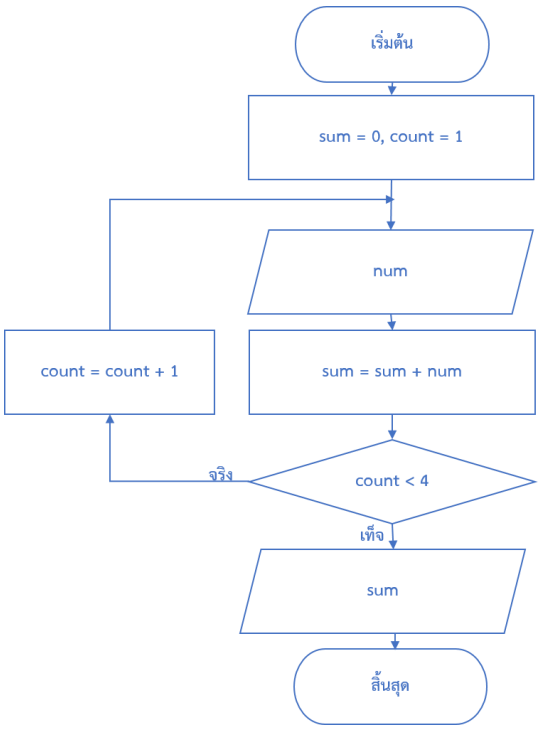
ต่อมานำตัวเลข 52 ใส่ลงในขวดโหล num แทนตัวเลข 90

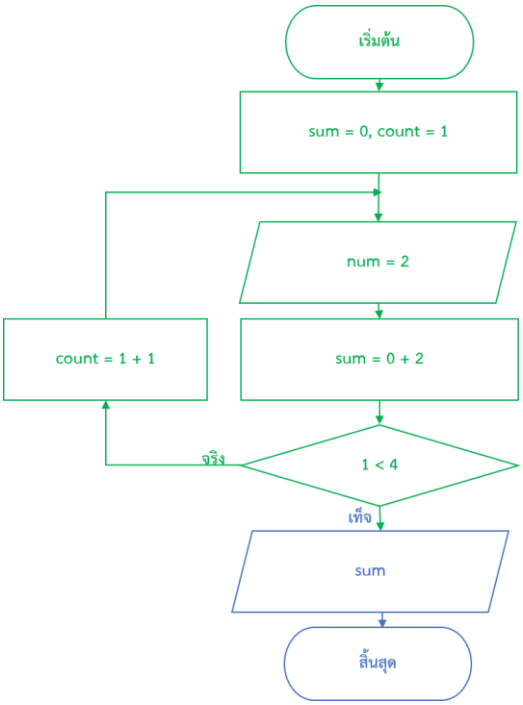


จากนั้น นำตัวเลข 52 ในขวดโหล num บวกกับตัวเลข 96 ในขวดโหล sum จะได้ผลลัพธ์เป็น 148 นำผลลัพธ์เท่ากับ 148 ใส่ลงในขวดโหลด sum แทนที่ผลลัพธ์เดิมคือ 96



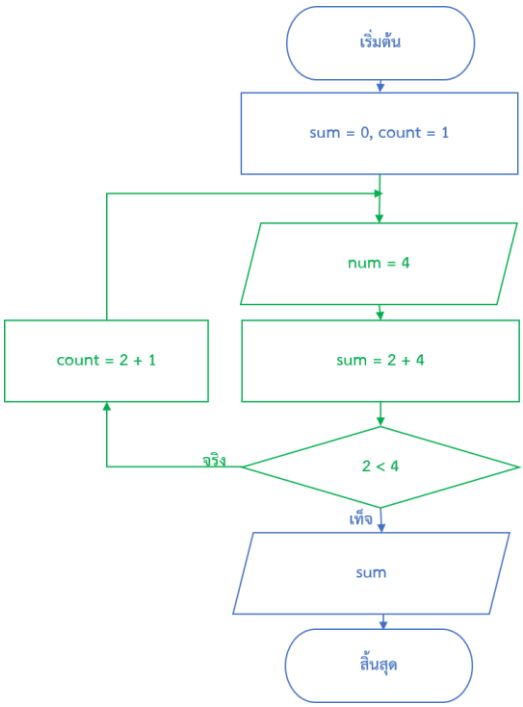
เมื่อจบการบวกเลขทั้ง 4 จำนวนคือ 2, 4, 90, 52 แล้ว ผลลัพธ์คือ 148 ซึ่งจะถูกเก็บอยู่ในขวดโหล sum หรือตัวแปร sum นั่นเอง การหาผลรวมนี้ใช้จำนวนรอบในการทำงานทั้งหมด 4 ครั้งเพราะมีข้อมูลจำนวน 4 จำนวน เป็นการทำงานซ้ำที่สามารถระบุรอบได้อย่างแน่ชัด และการหาผลรวมนี้สามารถแสดงในรูปแบบผังงานได้ดังนี้





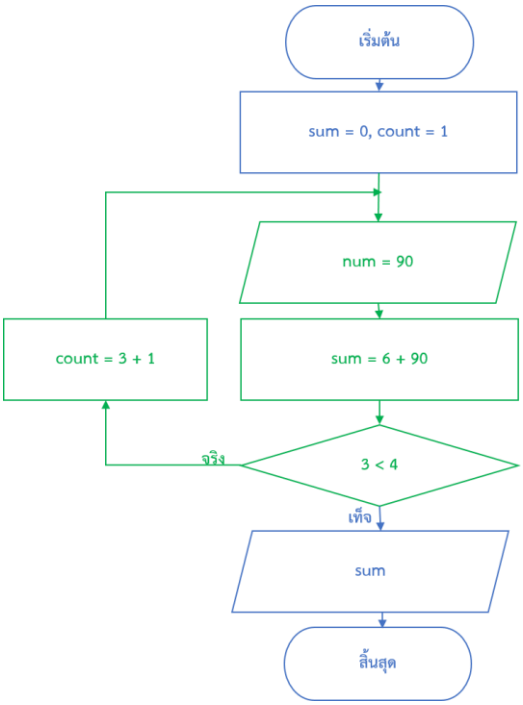
รอบที่ 1

ในรอบที่ 1 นี้ sum จะมีค่าเป็น 0 และมีตัวแปร count สำหรับนับว่าครบรอบ 4 รอบตามจำนวนข้อมูลหรือยัง จากนั้นรับค่าข้อมูลจำนวนแรกคือ 2 เก็บลงในตัวแปร num จากนั้นทำการหาผลรวมโดยนำตัวเลข 2 มาบวกกับค่าเดิมใน sum ทำให้ผลลัพธ์เป็น 2 ต่อด้วยการทำงานที่สัญลักษณ์ Decision เพื่อตรวจสอบว่าจำนวนข้อมูลตอนนี้เป็นตัวที่ 4 หรือยัง ถ้ายังจะเพิ่มค่าในตัวแปร count เพื่อวนไปรับข้อมูลตัวที่ 2 ต่อไป



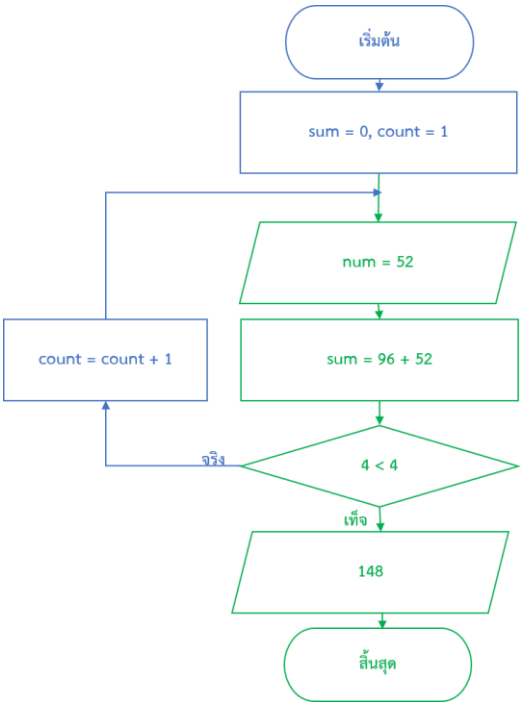
รอบที่ 2

รอบที่ 2 ข้อมูลตัวที่ 2 ที่รับเข้ามาคือ 4 นำเลข 4 บวกกับค่าเดิมจากรอบที่ 1 ในตัวแปร sum คือ 2 นั่นเอง จะได้ผลลัพธ์เป็น 6 และเก็บ 6 ลงในตัวแปร sum จากนั้นที่สัญลักษณ์ Decision ตรวจสอบว่าจำนวนที่รับเข้ามาเป็น 4 จำนวนหรือยัง เมื่อยังจึงทำการบวกค่าตัวแปร count เพิ่มอีก 1 และวนไปรับข้อมูลตัวที่ 3 ต่อไป



รอบที่ 3

รอบที่ 3 จะรับข้อมูลตัวถัดไป คือ 90 และนำ 90 บวกกับค่าในตัวแปร sum จากรอบที่ 2 ที่เท่ากับ 6 ทำให้ได้ผลลัพธ์เป็น 90 และเก็บลงในตัวแปร sum รอบที่ 3 นี้ ต่อไปที่สัญลักษณ์ Decision ตรวจสอบว่าเป็นจำนวนตัวที่ 3 ยังไม่ใช้จำนวนตัวที่ 4 จึงเพิ่มค่าในตัวแปร count อีก 1 และวนไปรับข้อมูลตัวที่ 4 ต่อไป



รอบที่ 4

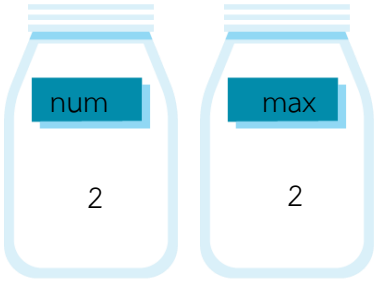
รอบที่ 4 รับข้อมูล 52 ซึ่งเป็นจำนวนสุดท้ายของข้อมูล และนำ 52 บวกกับค่าเดิมในตัวแปร sum จากรอบที่ 3 คือ 96 นั่นเอง ได้ผลลัพธ์เป็น 48 และเก็บลงในตัวแปร sum และทำงานต่อที่สัญลักษณ์ Decision เพื่อตรวจสอบว่าจำนวนข้อมูลที่รับเข้ามาครบ 4 จำนวนหรือยัง เมื่อครบแล้ว จึงลงมาทำงานต่อที่การแสดงผลโดยสัญลักษณ์ Input/Output เพื่อแสดงผลรวมสุดท้ายที่ถูกเก็บในตัวแปร sum คือ 148 และจำนวนรอบทั้งหมดที่ใช้ในการทำงานคือ 4 นั่นเอง

จากการตัวอย่างนี้แสดงให้เห็นถึงการทำงานแบบวนซ้ำที่ทราบจำนวนรอบแน่ชัดอยู่ก่อนแล้ว และทำการวนซ้ำอย่างมีรอบที่จำกัด คือ 4 เงื่อนไขของการหยุดทำงานของการวนซ้ำนี้คือ หยุดเมื่อรอบเท่า 4 นั่นเอง

ตัวอย่าง : หาค่ามากที่สุด

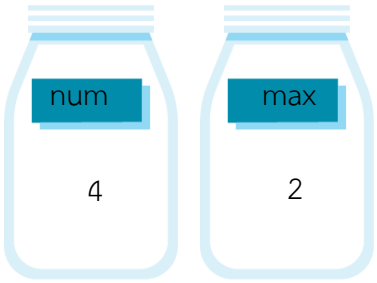
มีข้อมูล 4 จำนวนประกอบด้วย 2, 4, 90, 52 ตามลำดับ จะต้องหาค่าที่มากที่สุด โดยการเปรียบเทียบครั้งละ 2 จำนวน เริ่มต้นด้วยการเก็บเลข 2 ลงตัวแปร num และ max ก่อนในการทำงานรอบที่ 1 จากนั้นตรวจสอบเลขตัวเลขในตัวแปร num และตัวแปร max ในรอบที่ 1 นี้จะมีค่าเท่ากันอยู่ เมื่อการทำงานรอบที่ 2 จะรับค่า 4 มาเป็นข้อมูลตัวถัดไป และเปรียบเทียบกับค่าในตัวแปร max เท่ากับนำ 4 มาเปรียบเทียบกับ 2 ผลลัพธ์ที่มากที่สุดของการเปรียบเทียบรอบที่ 2 นี้คือ 4 มากกว่า 2 จึงเก็บ 4 ลงในตัวแปร max ต่อไปเมื่อทำงานรอบที่ 3 รับข้อมูลตัวที่ 3 คือ 90 ลงในตัวแปร num และเปรียบเทียบค่าในตัวแปร num และตัวแปร max จะได้ผลลัพธ์เป็นค่าที่มากที่สุดของสองค่าในรอบที่ 3 นี้เป็น 90 จึงเก็บ 90 ลงในตัวแปร max และทำการรับข้อมูลตัวสุดท้ายคือ 52 เก็บลงในตัวแปร num ทำการเปรียบเทียบค่าในตัวแปร num และตัวแปร max คือเปรียบเทียบ 52 กับ 90 นั่นเอง ทำให้ผลลัพธ์ของการเปรียบเทียบในรอบที่ 4 นี้มีค่าเท่า 90 และเก็บค่า 90 ลงในตัวแปร max เป็นอันจบการทำงาน และการทำงานนี้สามารถแสดงโดยใช้ขวดโหลที่ติดป้ายได้ ดังนี้

เริ่มต้นในการรับค่ารอบที่ 1 ค่าที่รับคือ 2 เก็บค่า 2 ลงในขวดโหลที่ติดป้าย num และเก็บค่า 2 ลงในขวดโหลที่ติดป้าย max ด้วย

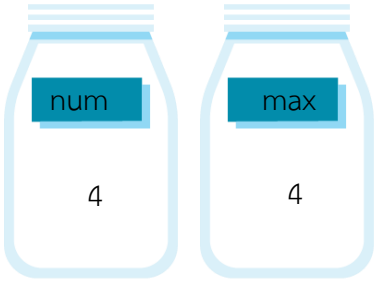


จากนั้นทำการเปรียบเทียบค่าในโหล num และโหล max เพื่อหาค่าที่มากที่สุด ปรากฏว่าในรอบที่ 1 นี้ค่าในทั้งสองขวดโหลเท่ากัน ค่าที่มากที่สุดในรอบที่จึงเป็น 2

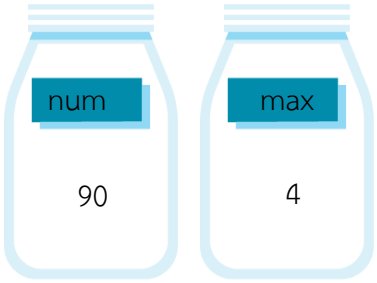
รอบที่ 2 รับข้อมูลจำนวนที่ 2 มีค่าเป็น 4 เก็บค่า 4 ลงในขวดโหลที่ติดป้าย num



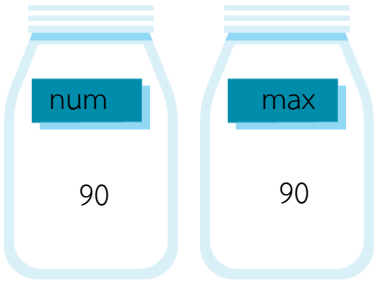
ทำการเปรียบเทียบเพื่อหาจำนวนที่มากที่สุดระหว่างค่าในขวดโหล num และค่าในขวดโหล max คือนำ 4 เปรียบเทียบกับ 2 ผลลัพธ์ที่เป็นจำนวนที่มากที่สุดคือ 4 ดังนั้นจึงเก็บ 4 ลงในขวดโหล max แทนค่าเดิมซึ่งคือ 2



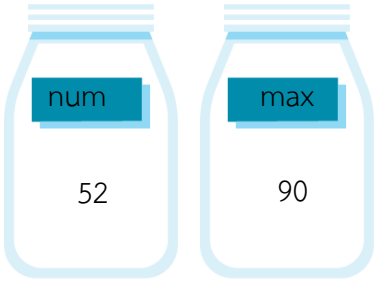
รอบที่ 3 รับข้อมูลจำนวนถัดไปคือ 90 นำ 90 เก็บลงในขวดโหลที่ติดป้าย num



หลังจากนั้นทำการเปรียบเทียบค่าในขวดโหล num และขวดโหล max เพื่อหาค่าที่มากที่สุด นำ 90 เปรียบเทียบกับ 4 ผลลัพธ์ของค่าที่มากที่สุดระหว่าง 2 จำนวนนี้คือ 90 และทำการเก็บค่า 90 ลงในขวดโหล max แทนที่ค่า 4

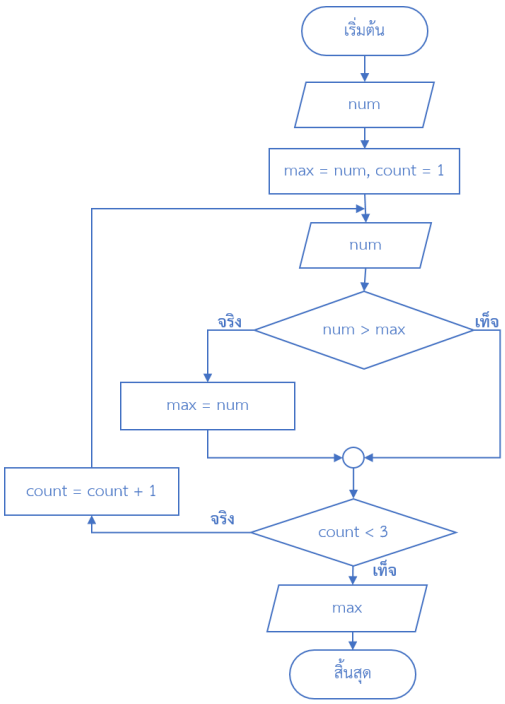


รอบที่ 4 รับข้อมูลจำนวนถัดไป คือ 52 เก็บค่า 52 ลงในขวดโหลที่ติดป้าย num

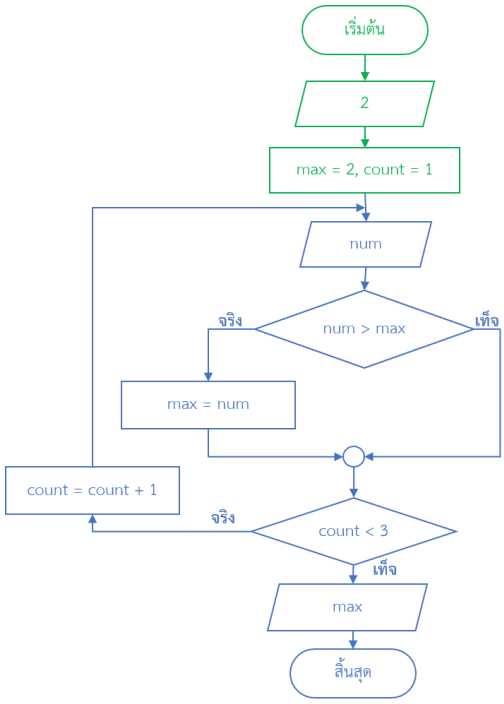


ทำการเปรียบเทียบระหว่างค่าในขวดโหล num และขวดโหล max คือเปรียบเทียบ 52 กับ 90 ทำให้ผลลัพธ์ที่เป็นจำนวนที่มากที่สุดระหว่างสองค่านี้คือ 90 และผลลัพธ์ที่เป็นค่าที่มากที่สุดสำหรับข้อมูลชุดนี้คือ 90 นั่นเอง

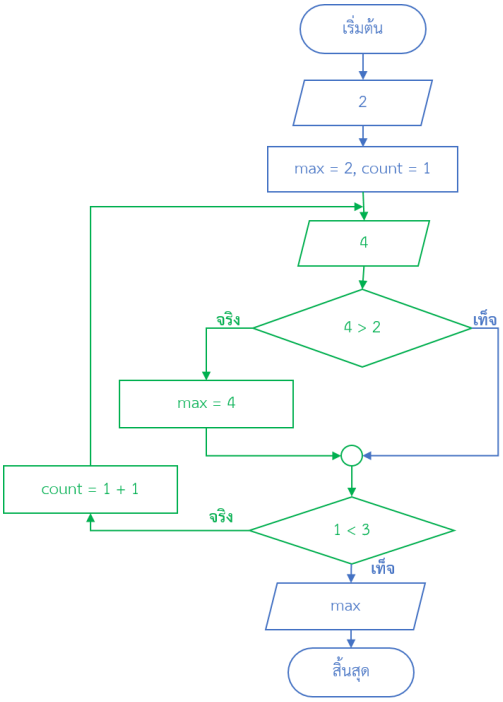
ตัวอย่างนี้มีความแตกต่างจากตัวอย่างการหาผลรวม เพราะมีการตรวจสอบว่าระหว่างการเปรียบเทียบ 2 ค่านั้น ค่าใดเป็นค่าที่มากที่สุด และเมื่อได้ผลลัพธ์ว่าค่าใดเป็นค่าที่มากที่สุดแล้วจึงทำการเก็บค่านั้นลงในตัวแปรสำหรับเก็บค่าที่มากที่สุด ซึ่งก็คือตัวแปร max ตัวอย่างการหาค่าที่มากที่สุดจากข้อมูลทั้ง 4 จำนวนสามารถแสดงในรูปแบบผังงานได้ ดังนี้



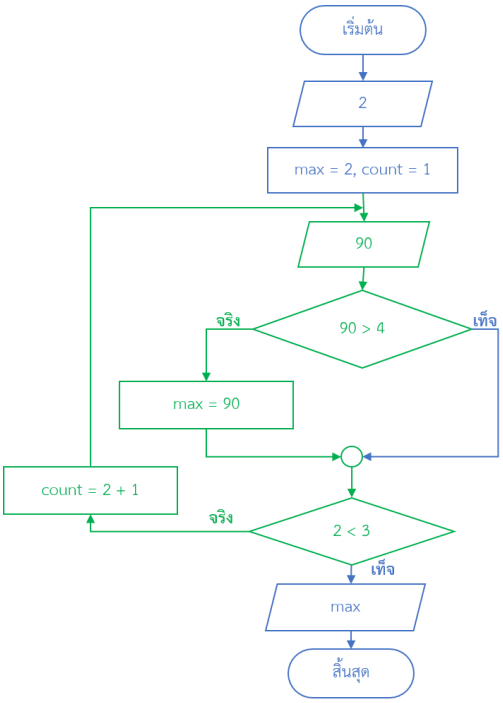
รอบที่ 1 การรับค่าข้อมูลตัวที่ 1 คือ 2 จะยังไม่รับค่าภายใต้การวนซ้ำ เพราะต้องการเก็บค่า 2 ลงในตัวแปร max ก่อน อาจเกิดคำถามขึ้นว่าทำไมจึงไม่เก็บค่า 0 ลงในตัวแปร max เช่นเดียวกับที่เก็บค่า 0 ลงในตัวแปร sum จากตัวอย่างก่อนหน้านี้ คำตอบคือต้องการรองรับตัวเลขที่มีค่าน้อยกว่าศูนย์อย่างตัวเลขติดลบด้วย จึงเกิดการเขียนผังงานในการรับตัวเลขจำนวนที่ 1 ดังนี้



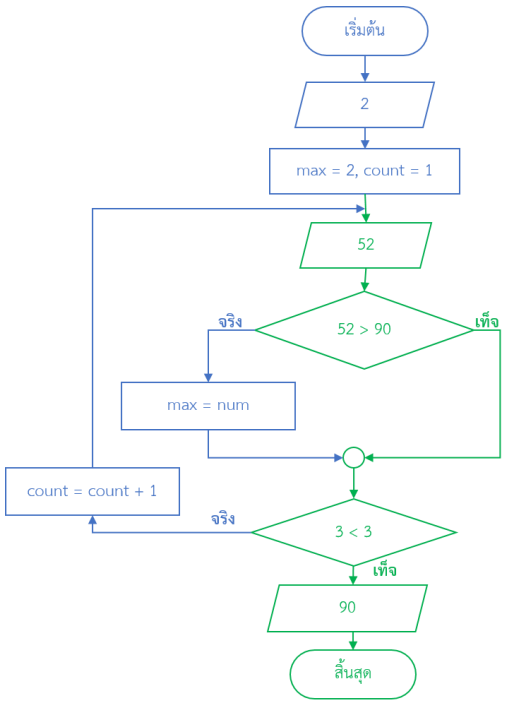
รอบที่ 2 เริ่มทำการรับข้อมูลภายใต้การทำงานแบบวนซ้ำ โดยนำข้อมูลตัวที่ 2 ซึ่งมีค่าเท่ากับ 4 เปรียบเทียบกับข้อมูล ในตัวแปร max มีค่าเท่ากับ 2 ที่สัญลักษณ์ Decision “num > max” ทำให้ผลลัพธ์ที่ได้เป็นจริง และทำงานต่อที่สัญลักษณ์ Process “max = num” เพื่อเก็บค่าผลลัพธ์เท่ากับ 4 ลงในตัวแปร max และตรวจสอบต่อว่าขณะนี้รับข้อมูลเข้ามากี่จำนวนแล้วที่สัญลักษณ์ Decision “count < 3” เหตุที่ใช้ count < 3 เพราะว่าในการรับค่าครั้งแรกคือ 2 ไม่ได้ทำงานภายใต้การวนซ้ำ จึงตัดส่วนนั้นออกไป 1 รอบ เมื่อตรวจสอบแล้ว count ซึ่งในขณะนี้มีค่าเท่ากับ 1 ซึ่งน้อยกว่า 3 อยู่ จึงได้ผลลัพธ์เป็นจริง และทำการเพิ่มค่า count ขึ้นอีก 1 และวนซ้ำเพื่อรับข้อมูลจำนวนถัดไป



รอบที่ 3 ข้อมูลที่รับเข้ามาในรอบที่ 3 นี้คือ 90 นำ 90 เปรียบเทียบกับค่าในตัวแปร max จากรอบที่ 2 คือ 4 ที่สัญลักษณ์ Decision “num > max” หรือ 90 > 4 ทำให้ผลลัพธ์เป็นจริง จึงดำเนินการเก็บค่า 90 ลงในตัวแปร max แทนเลข 4 ซึ่งเป็นค่าเดิม และตรวจสอบที่สัญลักษณ์ Decision “count < 3” จริงหรือไม่ ซึ่งผลลัพธ์ให้ค่าเป็นจริง เพราะขณะนี้ count = 2 จึงทำงานต่อที่การเพิ่มค่า “count = count + 1” และวนรับข้อมูลจำนวนถัดไป

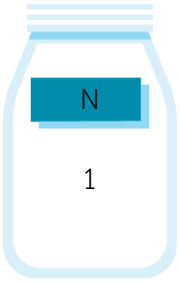


รอบที่ 4 ข้อมูลเข้ามาในรอบนี้คือ 52 เก็บจำนวน 52 ลงในตัวแปร num และทำการเปรียบเทียบระหว่าง 52 กับค่าในตัวแปร sum คือ 90 ปรากฏว่าผลลัพธ์เป็นเท็จ เพราะ 52 มีค่าน้อยกว่า 90 จึงทำการเปรียบเทียบจำนวนรอบที่สัญลักษณ์ Decision “count < 3” เลย ค่า count ในขณะนี้มีค่าเท่า 3 ทำให้การเปรียบเทียบ 3 < 3 เป็นเท็จ และลงมาทำงานต่อที่สัญลักษณ์ Input/Output เพื่อแสดงผลลัพธ์เป็นจำนวนที่มากที่สุดในชุดข้อมูลชุดนี้ เป็นอันจบการทำงาน



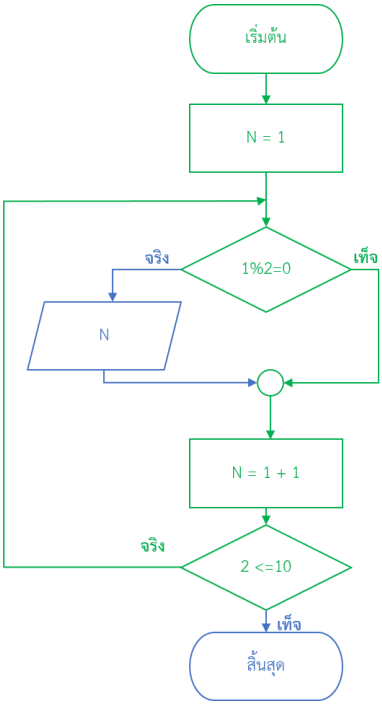
ตัวอย่าง : แสดงจำนวนคู่

ตัวอย่างนี้จะเป็นการแสดงผลภายใต้การวนซ้ำและภายใต้เงื่อนไข โดยกำหนดให้ตัวแปร N เริ่มต้นที่ 1 และเพิ่มค่าขึ้นครั้งละ 1 จนกว่าจะมีค่าเท่ากับ 10 เงื่อนไขของการแสดงผลคือจะต้องเป็นจำนวนคู่เท่านั้น การที่จะทราบได้ว่าจำนวนใด ๆ เป็นจำนวนคู่หรือเป็นจำนวนคี่ สามารถทดสอบได้โดยการ Mod ด้วย 2 ถ้าผลลัพธ์เท่ากับ 0 แปลว่าการหารด้วย 2 นั้นไม่เหลือเศษหรือหารลงตัวพอดี ตัวเลขนั้น ๆ จะเป็นจำนวนคู่

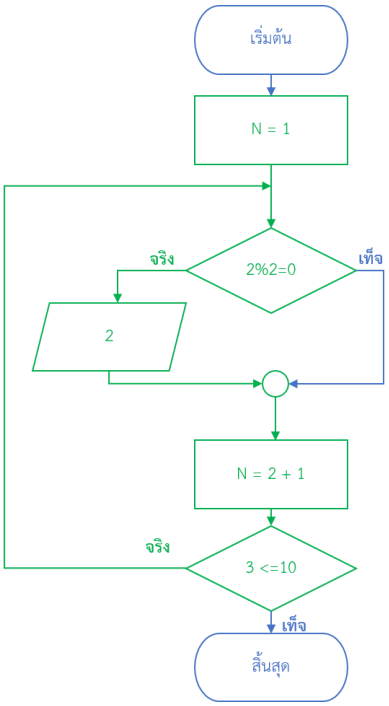


สมมุติเหตุการณ์ให้น้องเหมียวทำการสาธิตหน้าห้องโดยใช้ขวดโหลแทนการใช้ตัวแปร และติดป้ายชื่อไว้ที่ขวดโหลว่า N โดยตัวเลขในขวดโหลคือ 1 น้องเหมียวจะต้องอ่านตัวเลขที่อยู่ในขวดโหลและคำนวณด้วยการ Mod 2 น้องเหมียวจะได้ผลลัพธ์เป็น 1 น้องเหมียวจึงไม่เขียนเลข 1 บนกระดานหน้าห้อง สิ่งที่น้องเหมียวต้องทำต่อไปคือ นำกระดาษในขวดโหล N มาทำการบวก 1 ได้ผลลัพธ์เป็น 2 และใส่เลข 2 ลงไปในขวดโหล

จากนั้น น้องเหมียวก็อ่านหมายเลขในขวดโหล ซึ่งตอนนี้เป็น 2 และคำนวณด้วยหาร Mod 2 ได้ผลลัพธ์เป็น 0 น้องเหมียวจึงทำการเขียนเลข 2 บนกระดานหน้าชั้นเรียน น้องเหมียวอ่านเลขในขวดโหล และคำนวณเช่นนี้ไปเรื่อย ๆ จนเมื่อเลขในขวดโหลเป็น 10 และน้องเหมียวคำนวณด้วยหาร Mod 2 แล้วได้ผลลัพธ์เท่ากับ 0 น้องไปทำการเขียนเลข 10 บนกระดานหน้าชั้นเรียน ดังนั้นตัวเลขที่น้องเหมียวจะเขียนบนกระดานหน้าชั้นเรียนคือ 2, 4, 6, 8, 10 นั่นเองตัวอย่างนี้สามารถแสดงในรูปแบบการเขียนผังงานได้ดังนี้

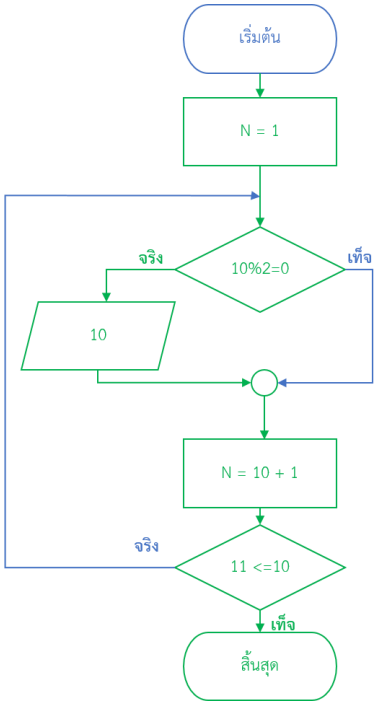


รอบที่ 1 N = 1 นำ N % 2 = 1 จึงไม่แสดงผลตัวเลข และทำงานต่อด้วยการเพิ่มค่า N ขึ้น 1 ทำให้ N = 2 และวนกลับไปทำงานอีกครั้ง



รอบที่ 2 N = 2 ทำการทดสอบว่า N % 2 = 0 จริงหรือไม่ เมื่อผลลัพธ์มีค่าเป็นจริง จึงทำงานต่อที่สัญลักษณ์ Input/Output เพื่อแสดงหมายเลข 2 หรือคือการที่น้องเหมียวเขียนเลข 2 ลงบนกระดานหน้าชั้นเรียนนั่นเอง จากนั้นทำงานต่อโดยการนำค่า N + 1 และเก็บลงในตัวแปร N เช่นเดิม เพื่อวนไปทำงานรอบถัดไป

เมื่อทำการตรวจสอบว่า N % 2 = 0 จริงหรือไม่ ถ้าจริง ให้ทำการแสดงผล และบวกค่า N + 1 วนไปทำงานรอบถัดไปเรื่อย ๆ จนค่า N = 11 ผลลัพธ์ที่สัญลักษณ์ Decision “N <= 10” เป็นเท็จ จึงหยุดการทำงาน ดังแผนผัง



สร้างชิ้นงานด้วยโปรแกรม Scratch

การสร้างชิ้นงานด้วยโปรแกรม Scratch จะสามารถทำให้การทำงานแบบวนซ้ำแสดงผลออกมาให้สามารถเข้าใจได้มากยิ่งขึ้น บล็อกคำสั่งของโปรแกรม Scratch จะประกอบไปด้วยบล็อกคำสั่งสำหรับการวนซ้ำ 3 แบบ บล็อกคำสั่งการวนซ้ำทั้ง 3 แบบจะอยู่ในบล็อกคำสั่งหมวด Control บล็อกคำสั่งแต่ละแบบจะทำงานแตกต่างกัน

บล็อกคำสั่ง repeat..จำนวนรอบ.. บล็อกคำสั่งภายในบล็อกคำสั่งนี้จะทำเท่ากับจำนวนรอบที่ระบุไว้ด้านหลัง

บล็อกคำสั่ง forever บล็อกคำสั่งภายในบล็อกคำสั่งนี้จะทำงานจนกว่าผู้ใช้จะปิดการทำงาน

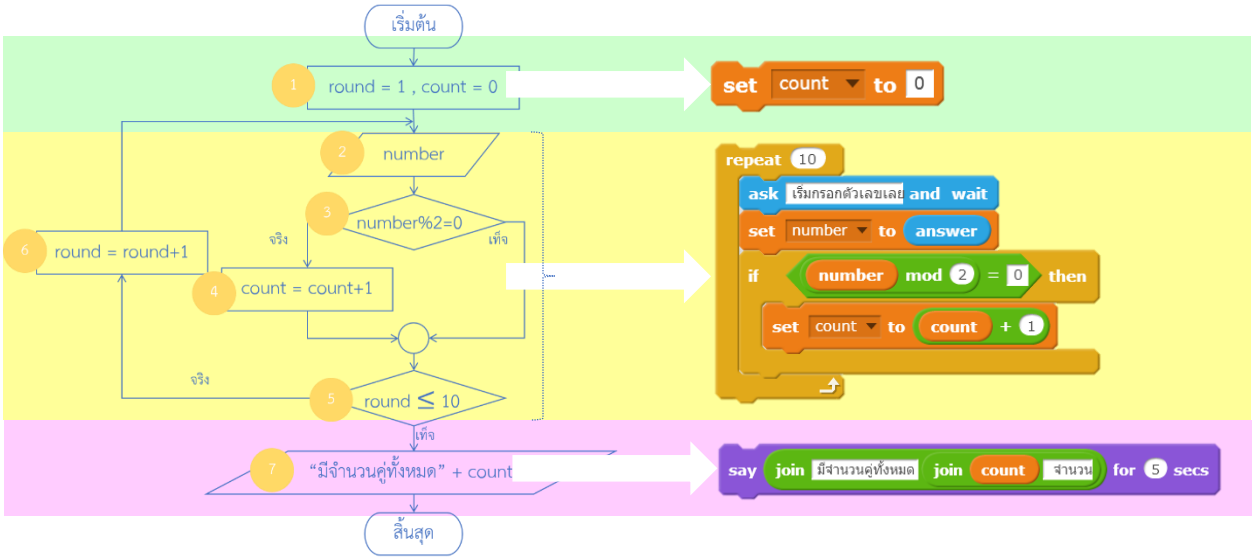
บล็อกคำสั่ง repeat until... เงื่อนไข...

บล็อกคำสั่งภายในบล็อกคำสั่งนี้จะทำงานเมื่อเงื่อนไขด้านหลังเป็นเท็จ และจะหยุดทำงานเมื่อเงื่อนไขด้านหลังเป็นจริง

ตัวอย่างการสร้างชิ้นด้วยโปรแกรม Scratch โดยการใช้บล็อกคำสั่งสำหรับการทำงานแบบวนซ้ำ ตัวอย่างที่ 1 ให้ผู้ใช้ป้อนตัวเลข 10 จำนวน การทำงานคือเก็บจำนวนคู่ทั้งหมดที่ผู้ใช้ป้อน แล้วแสดงผลเป็นผลรวมทั้งหมดของจำนวนคู่ เช่น ผู้ใช้ป้อน 1, 2, 3, 4, 2, 2, 10, 5, 7, 11 ผลลัพธ์สุดท้ายจะเป็น 5 เพราะจากข้อมูลที่ผู้ใช้ป้อนเข้ามาทั้งหมด มีตัวเลขที่เป็นจำนวนคู่ 5 จำนวนคือ 2, 4, 2, 2, 10

ตัวอย่าง : มีเลขคู่กี่ตัว

กำหนดให้รับตัวเลขจากผู้ใช้ 10 จำนวน นับตัวเลขที่เป็นจำนวนคู่ และแสดงผลจำนวนตัวเลขคู่ที่นับ สามารถเขียนผังงานเปรียบเทียบกับบล็อกคำสั่งได้ ดังนี้



จากผังงานและบล็อกคำสั่ง อธิบายได้ดังนี้

(1) round เป็นตัวแปรสำหรับนับรอบที่จะต้องวนรับข้อมูล ส่วนตัวแปร count สำหรับนับจำนวนตัวเลขที่เป็นจำนวนคู่ วางบล็อกคำสั่ง set...count..to..0 สำหรับโปรแกรม Scratch สามารถกำหนดรอบได้ จึงไม่ต้องมีตัวแปรสำหรับนับรอบ จึงสร้างการนับเฉพาะตัวแปร count เท่านั้น

ผังงานบริเวณหมาบเลข (2) ถึง (6) เป็นการทำงานแบบวนซ้ำ

(2) เป็นการรับตัวเลขจากผู้ใช้ โดยการวางบล็อกคำสั่ง ask..เริ่มกรอกตัวเลขเลย...and wait และบล็อกคำสั่ง set..number to answer สำหรับเก็บตัวเลขที่ผู้ใช้กรอกเข้ามาลงในตัวแปร number

(3) ตรวจสอบว่าค่าของ N ตอนนี้ mod 2 แล้วได้ผลลัพธ์เป็น 0 หรือไม่ สำหรับโปรแกรมจะวางบล็อกคำสั่ง if number mod 2 = 0 then และบล็อกคำสั่งภายในบล็อกคำสั่งนี้จะทำงานเมื่อเงื่อนไขนี้เป็นจริงเท่านั้น

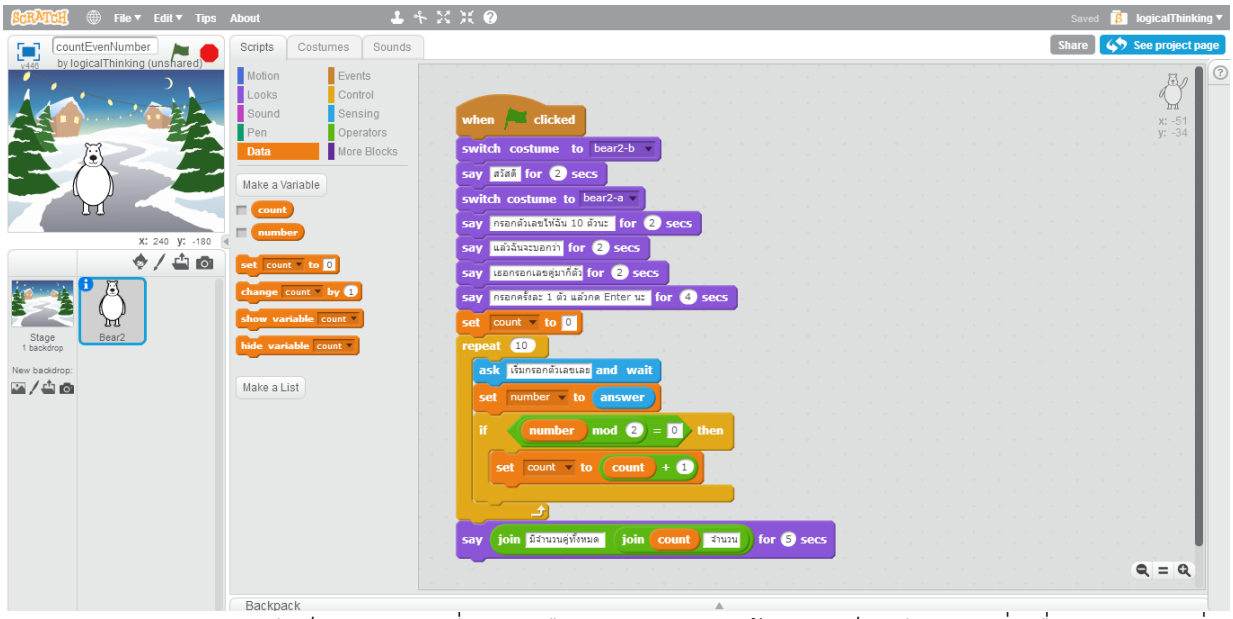
(4) เพิ่มค่าในตัวแปร +1 จะทำเมื่อค่า N นั้น ๆ เป็นเลขคู่เท่านั้น โดยการวางบล็อกคำสั่ง set count to count + 1 ในโปรแกรม Scratch

(5) และ (6) ทำการตรวจสอบว่าตอนนี้ค่า N ครบ 10 หรือยัง เมื่อยังจะทำการเพิ่มค่า count + 1 และวนกลับไปทำงานด้านบนต่อ ส่วนนี้ในการวางบล็อกคำสั่งของโปรแกรมจะไม่มี เพราะในโปรแกรมได้กำหนดรอบไว้เรียบร้อยแล้ว

(7) การทำงานส่วนนี้จะทำงานเมื่อเงื่อนไขที่สัญลักษณ์ Decision “round <= 10” เป็นเท็จ โดยการแสดงผลลัพธ์ของจำนวนคู่ว่ามีกี่จำนวน และจบการทำงาน ในส่วนของการวางบล็อกคำสั่ง จะวางบล็อกคำสั่ง say

join “มีจำนวนคู่ทั้งหมด” join count “จำนวน” for 5 secs เหตุที่ต้องใช้บล็อกคำสั่ง join ร่วมด้วยเพื่อเพิ่มความเข้าใจง่ายสำหรับการแสดงผล หรือจะวางบล็อกคำสั่งเพียง say count เลยก็สามารถทำได้เช่นกัน

สามารถชมตัวอย่างและบล็อกคำสั่งทั้งหมดได้ที่ https://scratch.mit.edu/projects/111933282/

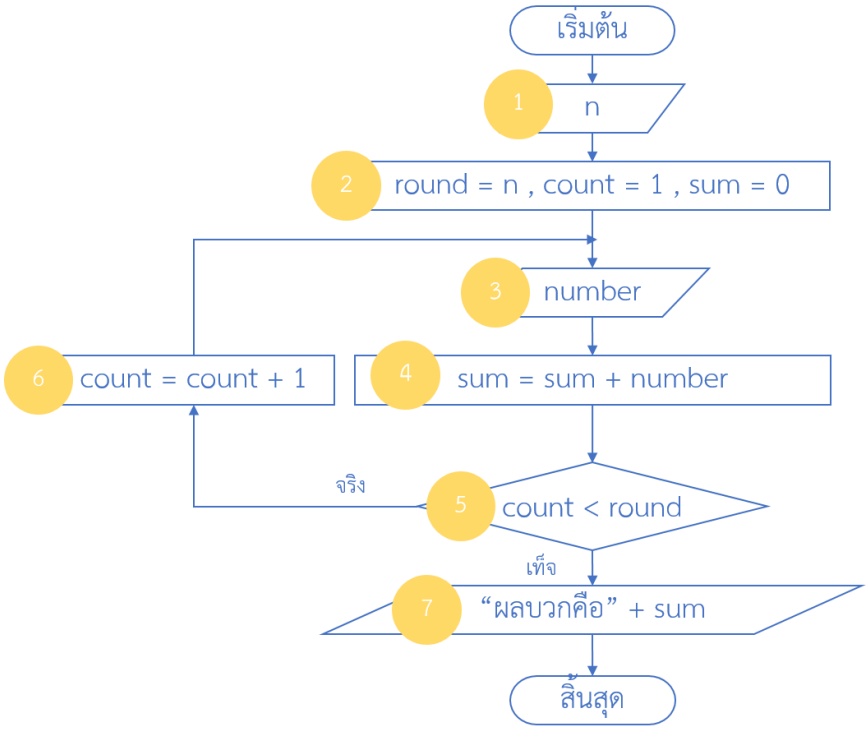


จากตัวอย่างจะมีบล็อกส่วนแรกที่นอกเหนือจากการแนะนำด้านบน เป็นเพียงการเพิ่มเรื่องราวและเกริ่นนำให้กับชิ้นงานเท่านั้น

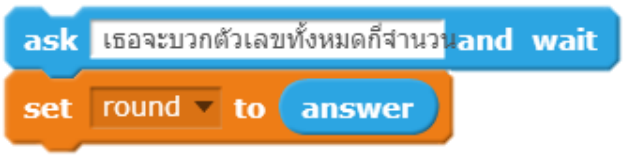
ตัวอย่างถัดไปจะเป็นการแสดงการทำงานแบบวนซ้ำโดยที่ผู้ใช้กำหนดรอบเอง และใช้บล็อกคำสั่งแบบ repeat until

ตัวอย่าง : บวกเลขหลายจำนวน

ตัวอย่างนี้จะให้ผู้ใช้กำหนดรอบเอง โดยตัวเลขตัวแรกที่ผู้ใช้ใส่เข้ามาคือตัวเลขที่บอกว่าจะใส่ตัวเลขต่อไปอีกเท่านี้จำนวน เช่น ผู้ใช้กรอกเลขตัวแรกเป็น 5 หมายความว่าผู้ใช้จะต้องกรอกตัวเลขอีก 5 จำนวนอย่าง 1, 2, 5, 12, 35 เป็นต้น สามารถเขียนในรูปผังงานและวางบล็อกคำสั่งสำหรับโปรแกรม Scratch ได้ ดังนี้



(1) n เป็นตัวแปรสำหรับเก็บตัวเลขของจำนวนทั้งหมดที่ผู้ใช้ต้องการจะกรอก วางบล็อกคำสั่งได้ดังนี้

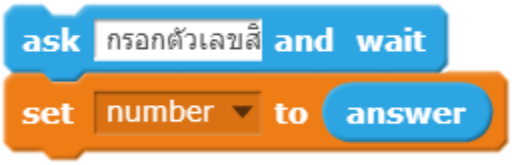


บล็อกคำสั่ง ask.....and wait จะปรากฏข้อความจากตัวละครถามผู้ใช้ และรอจนกว่าผู้ใช้จะกรอกตัวเลขและกดเครื่องหมายถูก หรือกดปุ่ม Enter ที่คีย์บอร์ด ส่วนบล็อกคำสั่ง set round to answer จะเก็บค่าที่ผู้ใช้กรอกเข้ามาลงในตัวแปร round

(2) round = n, count = 1, sum = 0 เป็นการกำหนดว่าจะต้องทำงานวนซ้ำทั้งหมดจำนวน round รอบ และเริ่มนับจำนวนที่ผู้ใช้จะต้องกรอกเริ่มจาก count และตั้งให้ตัวแปร sum มีค่าเริ่มต้นเท่ากับ 0 สามารถวางบล็อกคำสั่งได้ ดังนี้

บล็อกคำสั่ง set round to answer เป็นการเก็บค่าที่ผู้ใช้ใส่เข้ามาลงในตัวแปร round บล็อกคำสั่ง set count to 1 เป็นการกำหนดค่าให้ตัวแปร count มีค่าเท่ากับ 1 และบล็อกคำสั่ง set sum to 0 เป็นการกำหนดค่าให้ตัวแปร sum มีค่าเท่ากับ 0

(3) การรับตัวเลขที่ผู้ใช้ต้องการจะกรอกเพื่อนำมาหาผลบวก ให้วางบล็อกคำสั่งในโปรแกรม Scratch ได้ ดังนี้

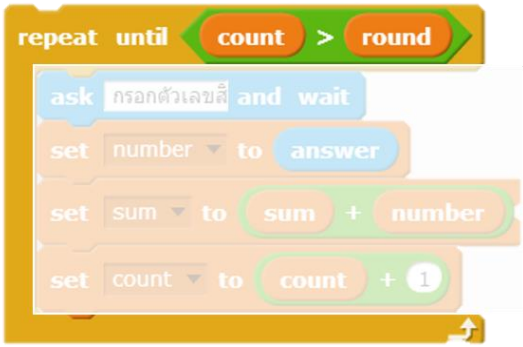


โดยบล็อกคำสั่งนี้จะต้องอยู่ภายในบล็อกคำสั่ง repeat until ..... บล็อกคำสั่ง ask.....and wait จะแสดงผลโดยผ่านการพูดของตัวละคร และบล็อกคำสั่ง set number to answer เป็นการเก็บค่าที่ผู้ใช้กรอกเข้ามาลงในตัวแปร number

(4) คำนวณผลบวกของตัวเลขที่ผู้ใช้กรอกเข้ามาตั้งแต่ตัวเลขที่ count = 1 ถึงตัวเลขที่ count = n โดยนำค่าผลลัพธ์ที่บวกได้ในรอบก่อนหน้ามาบวกกับค่าที่ผู้ใช้กรอกเข้ามา และเก็บลงในตัวแปรเดิม ในรอบที่ 1 ค่าในตัวแปร sum จะเท่ากับ 0 ก่อน สามารถวางบล็อกคำสั่งได้ ดังนี้

บล็อกคำสั่ง set sum to sum + number จะทำการบวกค่าในตัวแปร sum และตัวแปร number และเก็บผลลัพธ์ทับลงในตัวแปร sum (เช่นเดียวกับการเก็บเลขตัวใหม่ลงในขวดโหลเดิม และนำตัวเลขเดิมออก)

(5) สัญลักษณ์ Decision “count > round” เป็นการตรวจสอบว่าจำนวนข้อมูลที่รับเข้ามาขณะนี้ มากกว่าหรือเท่ากับจำนวนตัวเลขทั้งหมดที่ผู้ใช้ต้องการจะกรอกหรือยัง ถ้ายังไม่เท่ากับหรือมากกว่า จะทำงานต่อโดยการวนรับค่าจากผู้ใช้และบวกค่าเพิ่มขึ้น 1 ต่อไป สามารถวางบล็อกคำสั่งได้ ดังนี้

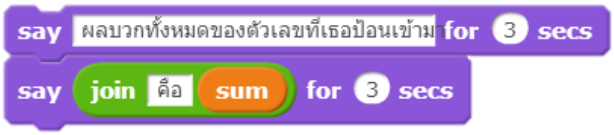


บล็อกคำสั่ง repeat until count > round บล็อกคำสั่งต่าง ๆ ที่วางภายในบล็อกคำสั่งนี้จะทำงานเมื่อ count > round เป็นเท็จ และจะหยุดทำงานเมื่อ count < round เป็นจริง หรือหยุดทำงานเมื่อจำนวนที่กรอกเข้ามาเพื่อหาผลรวมทั้งหมด ครบตามจำนวนที่ผู้ใช้กรอกเข้ามาในครั้งแรกที่ถูกเก็บในตัวแปร round แล้ว

(6) บล็อกคำสั่ง “set count = count + 1” เป็นการเพิ่มจำนวนข้อมูลที่ผู้ใช้จะกรอกเข้ามาเพื่อทำการหาผลรวม โดยจะทำงานเมื่อเงื่อนไขในผังงานเป็นจริงเท่านั้น สามารถวางบล็อกคำสั่งได้ ดังนี้

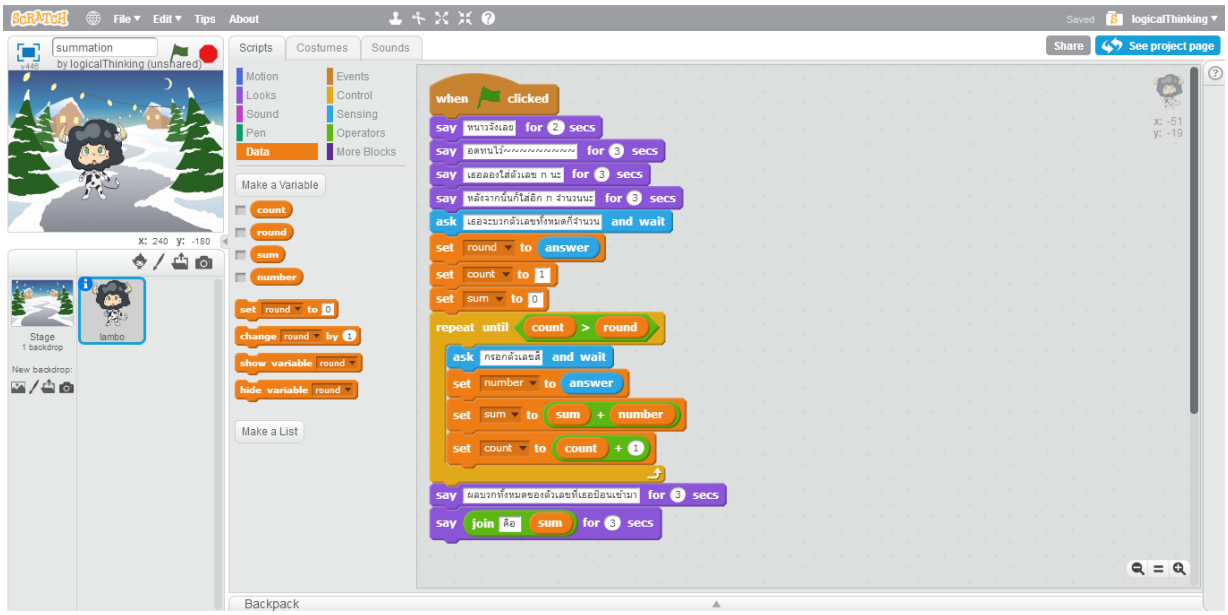
บล็อกคำสั่ง set count to count + 1 บล็อกคำสั่งนี้วางอยู่ภายในบล็อกคำสั่ง repeat until ..... และทำงานเมื่อเงื่อนไขด้านหลังของบล็อกคำสั่ง repeat until ..... เป็นเท็จเท่านั้น โดยมรการทำงานคือ นำค่าที่อยู่ในตัวแปร count ทำการบวกเพิ่มขึ้น 1

(7) บล็อกคำสั่ง “say join ผลบวกคือ sum” จากผังงานจะทำงานเมื่อเงื่อนไขจากสัญลักษณ์ Decision เป็นเท็จแล้ว โดยจะแสดงผลรวมของข้อมูลทั้งหมดที่ต้องการหาผลบวก โดยสามารถวางบล็อกคำสั่งได้ ดังนี้



บล็อกคำสั่ง say ..... for ... secs เป็นการเกริ่นนำก่อนที่จะแสดงผลลัพธ์ อาจมีหรือไม่ก็ได้ และบล็อกคำสั่ง say join “คือ” sum for ... secs เป็นการแสดงผลรวมของตัวเลขทั้งหมดที่ผู้ใช้ป้อนเข้ามาที่ถูกเก็บอยู่ในตัวแปร sum การ join ข้อความจะใช้หรือไม่ก็ได้เช่นกัน

สามารถชมตัวอย่างบล็อกคำสั่งได้ที่ https://scratch.mit.edu/projects/111933616/




ระวังสับสน
ผังงาน : ทำงานเมื่อเงื่อนไขเป็นจริง
บล็อกคำสั่ง repeat until ..... : ทำงานเมื่อเงื่อนไขเป็นเท็จ