Computer Programming I:
การเขียนโปรแกรมคอมพิวเตอร์ I

Introduction to Compro I

อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

Outline

- **รู้จักกับคอมพิวเตอร์และการเขียนโปรแกรม**
- คอมพิวเตอร์กับโปรแกรม
  - การวางแผนการเขียนโปรแกรม
  - ปัญหาที่แก้ไขด้วยคอมพิวเตอร์ได้
  - การแก้ปัญหาการคำนวณด้วยมนุษย์
  - การแก้ปัญหาการคำนวณด้วยคอมพิวเตอร์
- ขั้นตอนและตัวอย่างการเขียนโปรแกรม
  - การวิเคราะห์ปัญหา
  - ซูโดโค้ดและโฟลวชาร์ต

แนะนำวิชา

- รหัสและชื่อวิชา 517111 การเขียนโปรแกรมคอมพิวเตอร์ I
- ชื่อภาษาอังกฤษ Computer Programming I
- จำนวนหน่วยกิต: 3 (2-2-5)
- เป็นวิชาบังคับสำหรับสาขาวิทยาการคอมพิวเตอร์ และเทคโนโลยีสารสนเทศ ชั้นปี 1
- มีการเรียนการสอนภาคทฤษฎีและภาคปฏิบัติการ (Lab)

แนะนำผู้สอน

- ผู้สอน อาจารย์ ดร.ปัญญนัท อ้นพงษ์
- ห้องทำงาน 1642/1 ชั้น 6 อาคารวิทยาศาสตร์ 1
- ช่องทางการติดต่อ aonpong_p@su.ac.th

การประเมินผลการศึกษา

- ตัด F ที่ 40 คะแนน (ตัดอิงเกณฑ์ ดังนั้นคะแนนที่ได้จะสามารถประเมินเกรดได้ทันที)
- คะแนนที่ใช้วัดผล
  - ภาคทฤษฎี กลางภาค+ปลายภาค 15+15=30 คะแนน
  - ภาคปฏิบัติ กลางภาค+ปลายภาค 25+25=50 คะแนน
  - สอบปฏิบัติการย่อย (Quiz) 25 คะแนน
  - รวม 105 คะแนน
- ตัวช่วยการเก็บคะแนน
  - การเข้าเรียนและส่งการบ้าน 4 คะแนน + คะแนนแถมอื่นๆ ประมาณ 20 คะแนน
- สอบร่วมกัน ใช้ข้อสอบร่วมกันทุกกลุ่ม

หนังสือประกอบการเรียนการสอน

- “คู่มือเรียนภาษาซี ฉบับปรับปรุงใหม่ ” โดย อรพิน ประวัติบริสุทธิ์

  สำนักพิมพ์โปรวิชัน ราคา 199 บาท อ่านง่าย มีแบบฝึกหัดท้ายบทพร้อมเฉลย

- เน้นทำแบบฝึกหัดเพิ่มเติมและข้อสอบเก่าด้วยตนเองจะให้ผลมากกว่า เพราะมักจะชี้ปัญหาและสิ่งที่อาจทำให้เกิดความผิดพลาดได้ตรงจุด

คำแนะนำการเรียน

- ควรทำโจทย์ที่มีเตรียมไว้ให้ล่วงหน้าและลองทำก่อนเข้าเรียน (เวลาในแล็บมีน้อยและมักไม่เพียงพอต่อการตกผลึกความคิดระหว่างเขียนโปรแกรม)
- ถ้าลองทำมาก่อน สามารถนำปัญหาที่พบมาถามในห้องเรียนได้
- พยายามทำด้วยตนเองก่อนจะเปิดเฉลย หรือถามอาจารย์/เพื่อน (ถ้าที่สุดแล้วยังไม่ได้ก็สามารถถามได้)
- วิชานี้ไม่ได้เน้นเรื่องความจำ แต่เป็นความเข้าใจและการคิด วิเคราะห์โจทย์มากกว่า ดังนั้น การอ่านและไม่ลองทำโจทย์อาจจะไม่ตอบโจทย์วิชานี้เท่าไหร่นัก
- ชีทใหม่และชีทเดิมมีเนื้อหาที่ครอบคลุมและอ้างอิงถึงกัน แต่นักศึกษาอาจลองทำความเข้าใจทั้งสองชีทเลยก็ได้ (ถ้าทำโจทย์เท่ากัน อ่านเยอะกว่า ยังไงก็ดีกว่า)

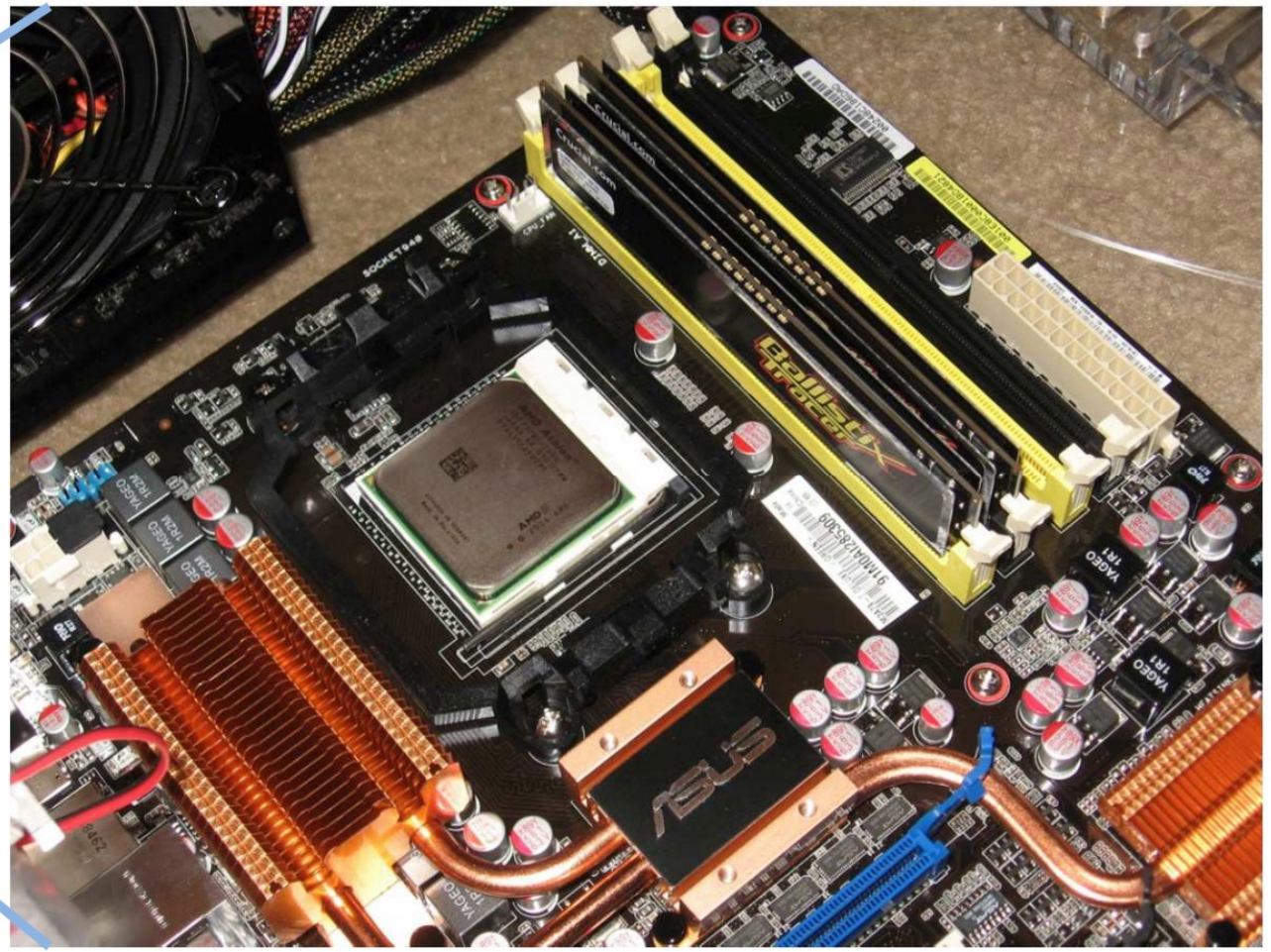
รู้จักกับคอมพิวเตอร์

- คอมพิวเตอร์คือเครื่องคำนวณชนิดหนึ่ง
- แต่เป็นเครื่องคำนวณที่มีสมรรถนะสูงมาก
  - เรามักจะสามารถดำเนินการจำนวนมากได้สำเร็จอย่างรวดเร็ว
  - ในการเรียนระดับของเรา เราสามารถทำการคำนวณได้เสร็จสิ้นภายในไม่กี่วินาทีด้วยคอมพิวเตอร์ส่วนบุคคล (โน๊ตบุ๊คหรือเดสท็อปทั่วไป)
  - ในอนาคต นักศึกษาอาจได้ไปทำงานในระบบคอมพิวเตอร์ที่มีขนาดใหญ่กว่าการทำงานด้วยคอมพิวเตอร์ส่วนบุคคล เช่น ซูเปอร์คอมพิวเตอร์ขนาดใหญ่ ที่สามารถทำงานที่มีความซับซ้อนมากกว่าที่คอมพิวเตอร์ส่วนบุคคลสามารถทำได้อย่างมาก

องค์ประกอบของคอมพิวเตอร์

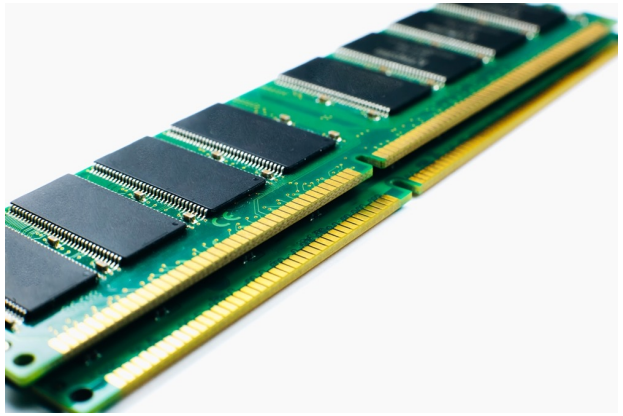
- อุปกรณ์พื้นฐาน (Hardware)
  - เม้าส์, คีย์บอร์ด, จอภาพ, หน่วยความจำ, หน่วยประมวลผลกลาง (CPU) และอุปกรณ์อื่นๆ
- ในการทำงาน อุปกรณ์ทั้งหมดจำเป็นต้องทำงานร่วมกัน โดยเฉพาะอย่างอย่างหน่วยความจำและหน่วยประมวลผลกลาง ที่จำเป็นต้องแลกเปลี่ยนข้อมูลกันบ่อยครั้ง
  - เราไม่สามารถคำนวณอะไรได้เลย ถ้าเราลืมวิธีการคำนวณ คอมพิวเตอร์ก็เช่นกัน
  - เราไม่สามารถคำนวณอะไรได้เลย ถ้าเราลืมว่าเลขที่เราจะนำมาคำนวณคืออะไร แม้จะเป็นคำสั่งที่ง่ายที่สุด คอมพิวเตอร์ก็เช่นกัน
  - เช่น โจทย์บอกว่า 5+3 ถ้าเราลืมว่าการบวกคิดอย่างไร หรือลืมว่านำเลขอะไรมาบวกกัน เราก็ไม่สามารถคำนวณคำตอบออกมาได้ คอมพิวเตอร์ก็เช่นกัน

หน่วยความจำกับการคำนวณ



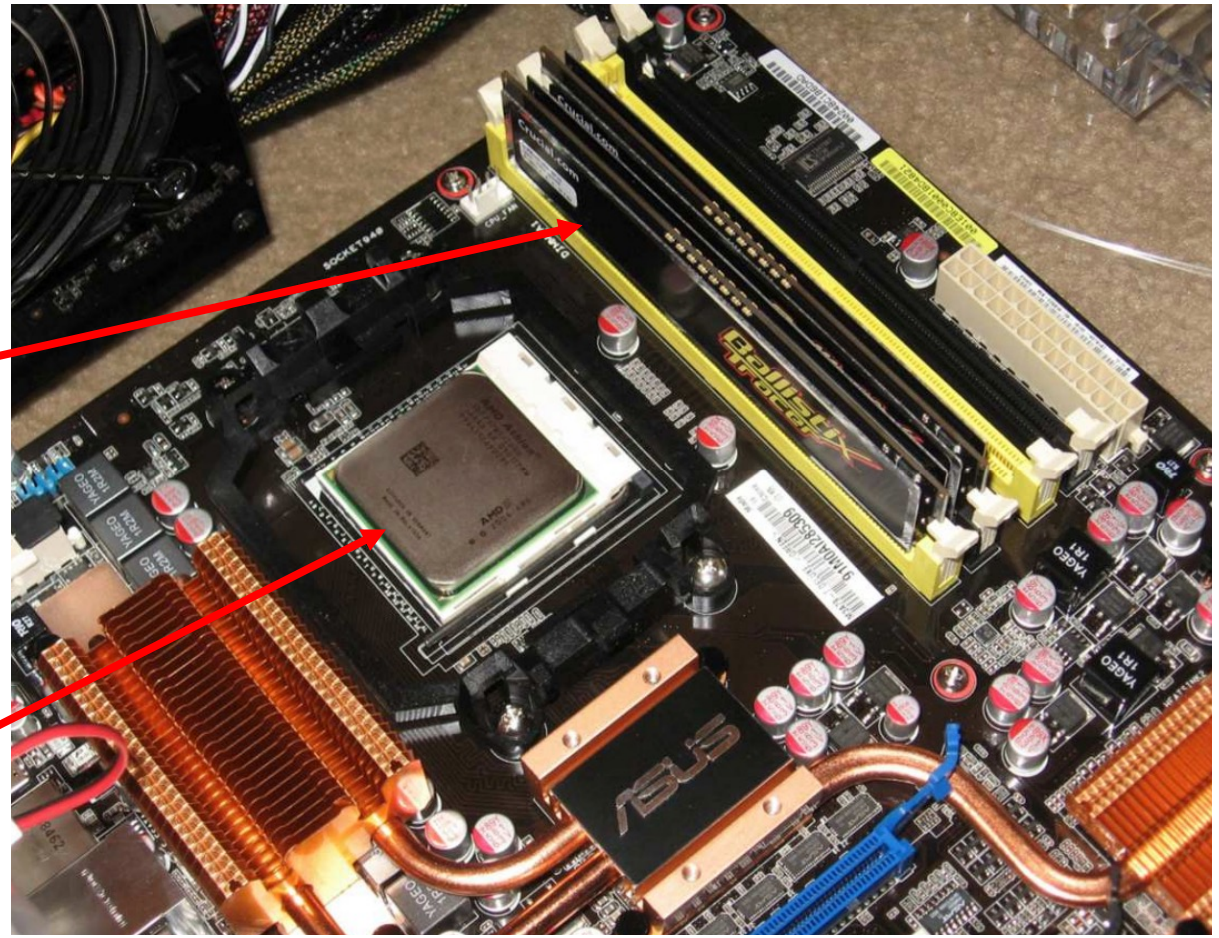
ภาพ overclockzone.com และ jedihawk.com

หน่วยความจำกับการคำนวณ



หน่วยความจำ (RAM)

หน่วยประมวลผลกลาง



ภาพ overclockzone.com, kapok.com และ hardware.info

หน่วยความจำกับการคำนวณ

- สิ่งที่นักศึกษาควรให้ความสำคัญในวิชานี้ไม่ใช่เพียงแต่ขั้นตอนวิธีการ แต่ต้องคำนึงถึงหน่วยความจำด้วย
- แต่ในระดับเริ่มต้น เราจะยังไม่พูดถึงเรื่องนี้มากนัก แต่จะให้นักศึกษาคุ้นเคยกับการเขียนโปรแกรมเป็นลำดับขั้นก่อน

- ส่วนของอุปกรณ์อื่น ๆ เช่นคีย์บอร์ดและเมาส์ ก็เป็นอุปกรณ์เพื่อรับข้อมูลนำเข้าอย่างหนึ่ง ซึ่งเราจะใช้บ่อยเป็นอย่างมากในการใช้งานคอมพิวเตอร์

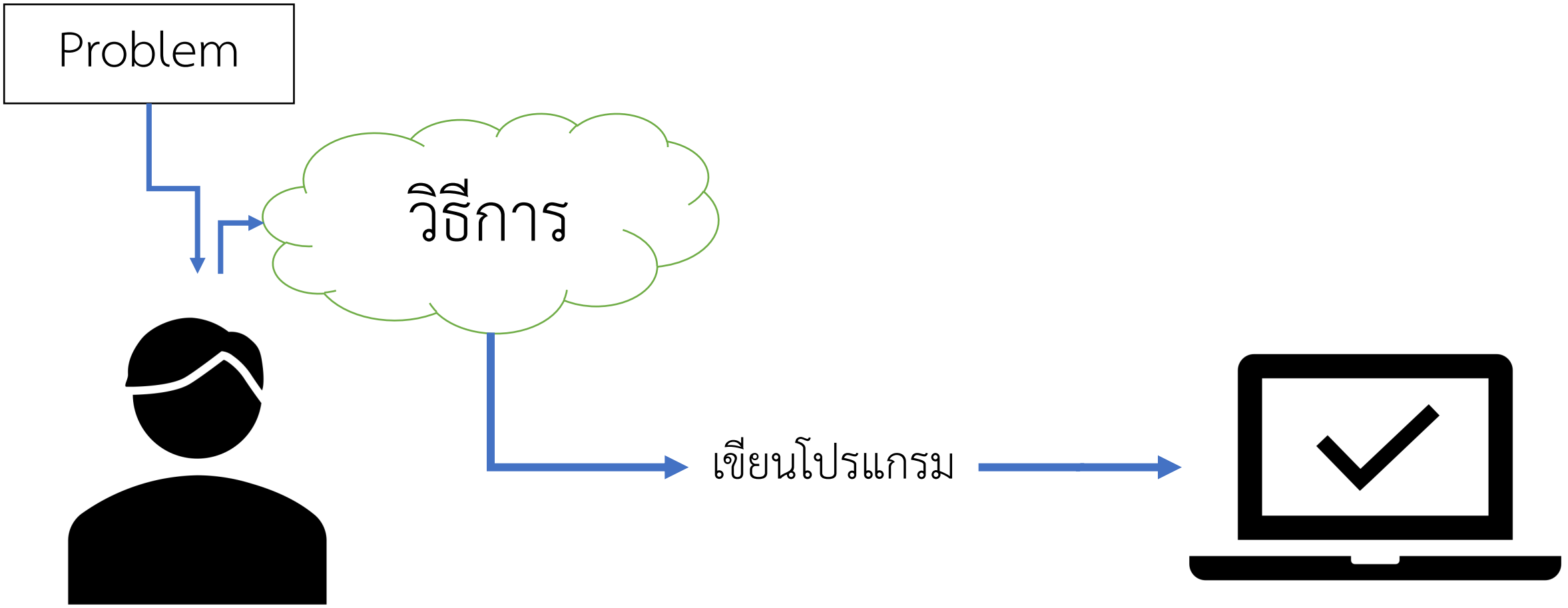
Outline

- รู้จักกับคอมพิวเตอร์และการเขียนโปรแกรม
- **คอมพิวเตอร์กับโปรแกรม**
  - **การวางแผนการเขียนโปรแกรม**
  - **ปัญหาที่แก้ไขด้วยคอมพิวเตอร์ได้**
  - **การแก้ปัญหาการคำนวณด้วยมนุษย์**
  - **การแก้ปัญหาการคำนวณด้วยคอมพิวเตอร์**
- ขั้นตอนและตัวอย่างการเขียนโปรแกรม
  - การวิเคราะห์ปัญหา
  - ซูโดโค้ดและโฟลวชาร์ต

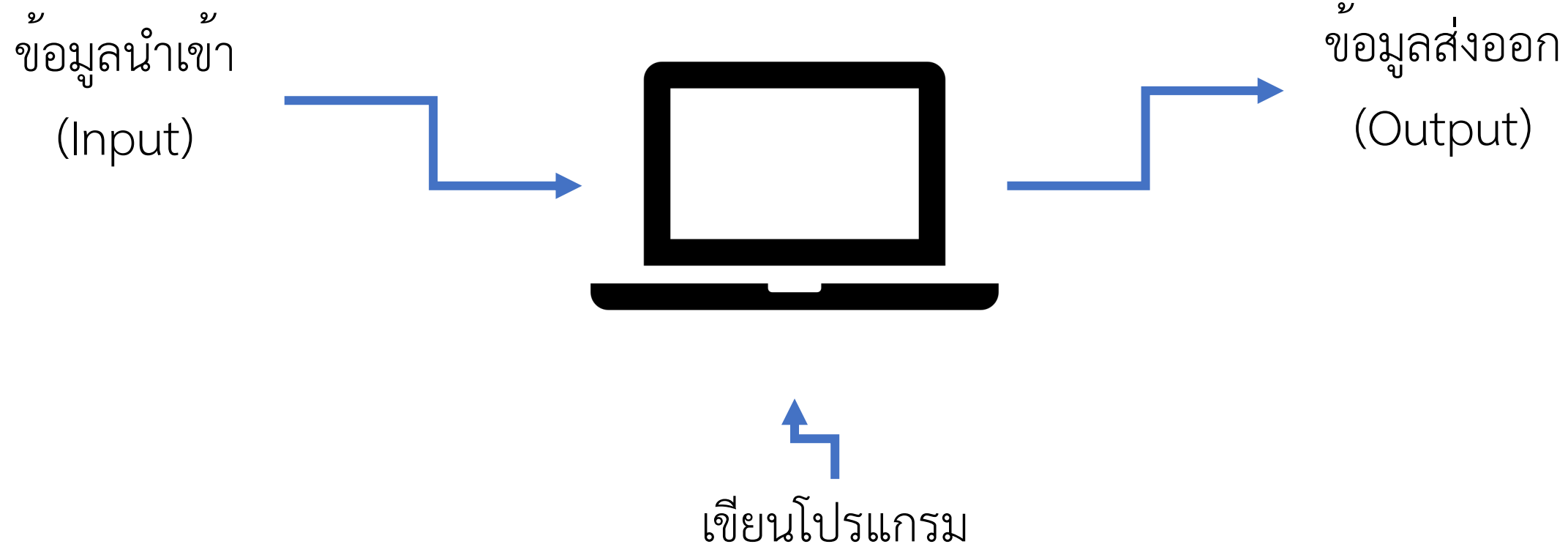
คอมพิวเตอร์กับโปรแกรม

- คอมพิวเตอร์คือเครื่องคิดเลข
- คอมพิวเตอร์แก้ไขปัญหาให้เราได้ แต่เราต้องบอกคอมพิวเตอร์ให้ได้ว่ามันจะต้องทำอะไรบ้าง
- ให้คิดว่าคอมพิวเตอร์เป็นเด็กที่ไม่ฉลาดนัก มันรู้วิธีการคำนวณแค่เบื้องต้นเท่านั้น แต่โจทย์ปัญหาที่ต้องให้คอมพิวเตอร์แก้มักจะไม่เบื้องต้นด้วย
  - การคำนวณมักจะมาเป็นชุด ๆ ไม่ใช่บวกลบคูณหารแล้วจบในครั้งเดียว
  - เราจะต้องบอกคอมพิวเตอร์ว่ามันต้องทำอะไรบ้าง ทีละขั้นตอน (แต่บอกครั้งเดียว เมื่อคอมพิวเตอร์ได้ลำดับขั้นตอนที่ถูกต้องแล้ว ครั้งต่อไปมันจะคิดให้เราได้)

คอมพิวเตอร์กับโปรแกรม



คอมพิวเตอร์กับโปรแกรม



คอมพิวเตอร์กับโปรแกรม

- เราจะสอนลำดับขั้นตอนให้กับคอมพิวเตอร์ได้อย่างไร คอมพิวเตอร์จะคุยกับเรารู้เรื่องหรือ?
  - รู้เรื่อง แต่เราต้องพูดภาษาเดียวกับมัน ซึ่งคอมพิวเตอร์ก็สามารถเข้าใจได้หลายภาษา ทั้งภาษาระดับสูง ที่ใกล้กับการพูดคุยของมนุษย์ และภาษาระดับต่ำ ที่ใช้คำสั่งเกือบจะเป็นเลขฐาน 2 ทั้งหมด (Binary)
  - ในวิชานี้เราจะใช้ภาษาระดับสูงตัวหนึ่งซึ่งได้รับความนิยมมาเป็นเวลานาน คือ ภาษาซี
  - การใช้ภาษาซีก็จะต้องแปลให้เป็นภาษาเครื่องอีกครั้งโดยใช้ Compiler (เป็นเครื่องมือที่มีให้ เราแค่เขียนภาษาซีให้ถูกต้องก็พอ)

การวางแผนการเขียนโปรแกรม

- ก่อนจะเริ่มเขียนโปรแกรม ต้องแน่ใจว่าคอมพิวเตอร์สามารถแก้ไขปัญหาที่ต้องการได้ (แก้ไขได้ด้วยการคำนวณ)
- โดยส่วนมาก ปัญหาส่วนใหญ่มักจะสามารถแก้ไขด้วยระบบคอมพิวเตอร์ได้ โดยเฉพาะโจทย์ที่นำมาใช้ในวิชานี้ สามารถแก้ไขด้วยการเขียนโปรแกรมได้ทั้งหมด
- ปัญหาที่ไม่สามารถแก้ไขได้ส่วนใหญ่คือปัญหาที่แม้แต่ผู้เขียนโปรแกรมก็ไม่รู้ขั้นตอนการคิด ทำให้ไม่สามารถบอกคอมพิวเตอร์ได้ว่าจะต้องทำการคำนวณอย่างไร

ปัญหาที่แก้ไขด้วยการเขียนโปรแกรมคอมพิวเตอร์ได้

- ถ้าสามารถคิดวิธีคำนวณออกมาได้ หรือระบุเป็นขั้นตอนที่แน่ชัดได้ ก็สามารถแก้ไขด้วยการเขียนโปรแกรมคอมพิวเตอร์ได้
  - เพราะคอมพิวเตอร์จะทำได้ก็ต่อเมื่อเราสั่ง ถ้าเราสั่งคอมพิวเตอร์ให้เป็นขั้นตอนไม่ได้ คอมพิวเตอร์ก็ทำงานให้เราไม่ได้
  - และถ้าเป็นปัญหาที่วิธีคิดไม่แน่นอน โปรแกรมก็อาจจะเขียนได้แต่มีความซับซ้อน หรืออาจเขียนไม่ได้เลย และผลลัพธ์มีความเสี่ยงที่จะผิดและไม่น่าเชื่อถือได้ (ต้องมีการตรวจสอบอย่างหนัก) วิชานี้ไม่มีการเขียนโปรแกรมในลักษณะนี้
- ปัญหาที่เราจะแก้ต้องไม่เกินกำลังของคอมพิวเตอร์ของเรา (ทรัพยากรต้องเพียงพอ)

การแก้ปัญหาการคำนวณด้วยมนุษย์

- ปัญหาด้านการคำนวณของมนุษย์ได้รับการแก้ปัญหามาช้านานแล้ว และเรารู้จักมันในนามของวิชาคณิตศาสตร์
  - เช่น การแก้สมการต่างๆ การแก้สมการสองตัวแปร สามตัวแปร การตรวจสอบจำนวณเฉพาะ การหาตัวหารร่วมมาก การหาตัวคูณร่วมน้อย เป็นต้น
  - จะเห็นว่าปัญหาเหล่านี้มีวิธีแก้ที่ชัดเจน และเราได้เล่าเรียนมาแล้วอย่างยากลำบาก
  - การใช้คอมพิวเตอร์ไม่ได้ทำให้หาวิธีการคำนวณแบบใหม่ได้ เพียงแต่ทำให้เร็วขึ้น
- ปัญหาหลักคือ โปรแกรมเมอร์มือใหม่มักไม่รู้ว่าจะแก้ปัญหานั้นๆได้อย่างไร และอาจต้องใช้การลองผิดลองถูกเข้าช่วย ในขณะที่โปรแกรมเมอร์ที่มีประสบการณ์อาจพบเจอการแก้ปัญหาที่มีลักษณะคล้ายกันนั้นมาแล้ว จึงสามารถมองออกได้โดยง่าย

การแก้ปัญหาการคำนวณด้วยมนุษย์

- ปัญหาเหล่านี้มีวิธีการแก้ไขปัญหาที่แน่นอนหรือไม่
- ตัวอย่างการแก้ไขปัญหาการแก้สมการ 1 ตัวแปร
  - 3x + 2 = 8
  - เราต้องการหาค่าของตัวแปร x (ให้ค่าของตัวแปร x เป็นผลลัพธ์)
- ตัวอย่างการแก้ไขปัญหาการแก้สมการ 2 ตัวแปร
  - 2x + 3y = 25
  - 3x + 2y = 20
  - เราต้องการหาค่าของตัวแปร x และตัวแปร y (ให้ค่าของตัวแปร x และ y เป็นผลลัพธ์)
- เราแก้ปัญหาเหล่านี้ด้วยการไม่ลองผิดลองถูกหรือไม่/อธิบายให้คนอื่นฟังได้หรือไม่

การแก้ปัญหาการคำนวณด้วยคอมพิวเตอร์

- มนุษย์ใช้ขั้นตอนแบบใด คอมพิวเตอร์ก็ใช้ขั้นตอนแบบเดียวกัน
- การเขียนภาษาคอมพิวเตอร์คือการสอนคอมพิวเตอร์ให้เข้าใจการทำงานนั้น
- คอมพิวเตอร์จะทำการคำนวณตามที่เรากำหนดเอาไว้ในภาษา

ภาษาคอมพิวเตอร์

- เป็นภาษาสื่อกลางที่ใช้ในการออกคำสั่งให้คอมพิวเตอร์
- มีหลายภาษา มีจุดเด่นจุดด้อยต่างกันออกไป
- ส่วนใหญ่ ภาษาคอมพิวเตอร์ มักมีลักษณะพื้นฐานดังต่อไปนี้
  - มีความเข้มงวดทางภาษามาก
  - มีการนิยามและกำหนดค่าต่าง ๆ
  - วิธีคิดต้องชัดเจน (โปรแกรมไม่สามารถคิดการทำงานขึ้นใหม่เองได้)
  - ไม่ยอมรับความกำกวม คือโปรแกรมจะต้องตีความหมายได้อย่างเดียว
- ผู้เขียนโปรแกรมโดยใช้ภาษาคอมพิวเตอร์จะต้องมีความเข้าใจกฎเกณฑ์ทางภาษาเป็นอย่างดีหรือสามารถสืบค้นและใช้คู่มือของคำสั่งแต่ละคำสั่งได้ดีด้วย

Outline

- รู้จักกับคอมพิวเตอร์และการเขียนโปรแกรม
- คอมพิวเตอร์กับโปรแกรม
  - การวางแผนการเขียนโปรแกรม
  - ปัญหาที่แก้ไขด้วยคอมพิวเตอร์ได้
  - การแก้ปัญหาการคำนวณด้วยมนุษย์
  - การแก้ปัญหาการคำนวณด้วยคอมพิวเตอร์
- **ขั้นตอนและตัวอย่างการเขียนโปรแกรม**
  - **การวิเคราะห์ปัญหา**
  - **ซูโดโค้ดและโฟลวชาร์ต**

ขั้นตอนการพัฒนาโปรแกรม

- ประกอบด้วย 5 ขั้นตอน
  - **วิเคราะห์ปัญหา (Analysis)**
  - **วางแผนและออกแบบ (Planning and Design)**
  - **การเขียนโปรแกรม (Coding)**
  - การทดสอบโปรแกรม (Testing)
  - จัดทำเอกสารและคู่มือการใช้งานหรือพัฒนา (Documentation)
- ในรายวิชานี้จะเป็น 3 ขั้นตอนแรกของกระบวนการทั้งหมด

การวิเคราะห์ปัญหา

- ตัวอย่างปัญหา จงเขียนโปรแกรมรับค่าจำนวนเต็ม 2 จำนวน และแสดงผลบวกของจำนวนเต็มทั้ง 2 จำนวนนั้น

1. **วิเคราะห์ปัญหา**
    1. เป็นขั้นตอนที่สำคัญที่สุด เพราะจะเป็นการบอกเราว่าเราเข้าใจปัญหานั้นอย่างไรบ้าง และเรามีความสามารถพอที่จะแก้ไขปัญหานี้ได้หรือไม่ ถ้าเราไม่สามารถแก้ไขปัญหานี้ได้ เราก็ไม่สามารถเขียนโปรแกรมเพื่อแก้ไขปัญหานี้ได้
    2. เริ่มจากการคัดแยกข้อมูลนำเข้าและข้อมูลขาออก
    3. จากนั้นสังเกตความสัมพันธ์ของข้อมูลระหว่างข้อมูลขาเข้าและข้อมูลขาออก

การวิเคราะห์ปัญหา

- ตัวอย่างปัญหา จงเขียนโปรแกรมรับค่าจำนวนเต็ม 2 จำนวน และแสดงผลบวกของจำนวนเต็มทั้ง 2 จำนวนนั้น

1. วิเคราะห์ปัญหา

1. ข้อมูลนำเข้า : จำนวนเต็ม 2 ตัว

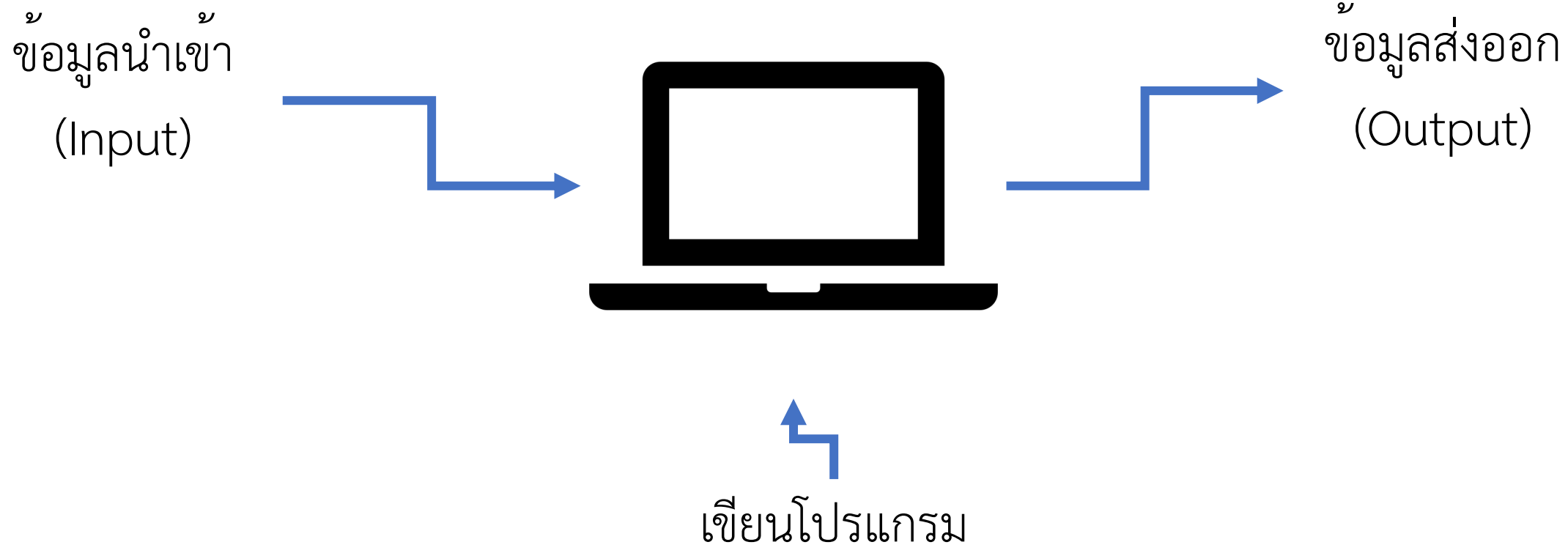
   กำหนดให้ชื่อ ตัวแปร x และ ตัวแปร y (จริง ๆ แล้ว จะตั้งชื่อตัวแปรอย่างไรก็ได้ แต่ควรมีความเกี่ยวข้องกับหน้าที่ของมัน)

2. ข้อมูลส่งออก : ผลบวกของจำนวนเต็มทั้ง 2 ตัว

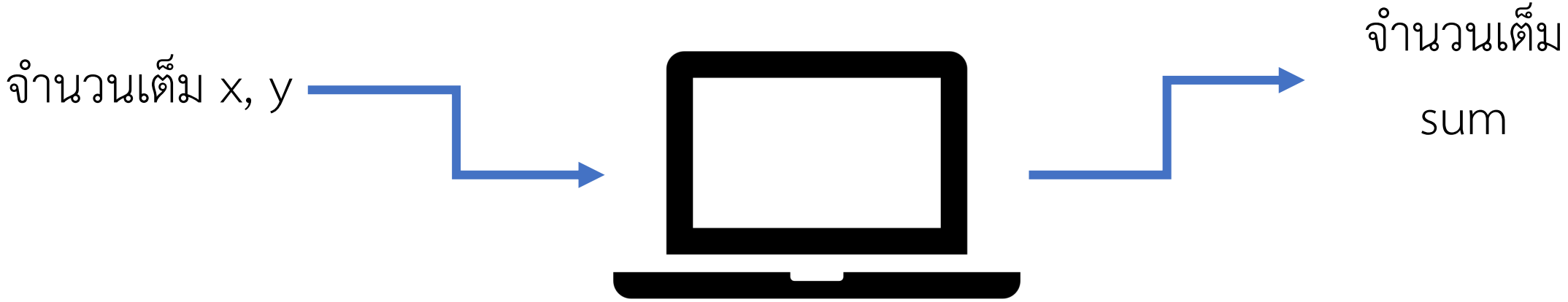
   กำหนดให้ชื่อ ตัวแปร sum (สามารถตั้งชื่อตัวแปรอย่างไรก็ได้เช่นเดียวกัน)

   จะได้ลักษณะความสัมพันธ์ดังนี้

การวิเคราะห์ปัญหา



การวิเคราะห์ปัญหา



การวิเคราะห์ปัญหา

วิเคราะห์ปัญหา

เมื่อวิเคราะห์แล้วได้ข้อมูลทั้งข้อมูลขาเข้าและข้อมูลขาออกแล้ว เราจะต้องมองหาความสัมพันธ์ของทั้งข้อมูลขาเข้าและขาออก

พูดให้ง่าย คือ ทำอย่างไรให้เมื่อคอมพิวเตอร์ได้รับข้อมูลขาเข้าเข้ามาแล้ว สามารถนำข้อมูลขาออกที่ต้องการออกมาได้

จากโจทย์ โจทย์นี้ไม่ได้มีความซับซ้อนมากนัก อย่างไรก็ดี ข้อมูลขาออกสามารถคำนวณได้จากการนำข้อมูลขาเข้าทั้งสองตัวมาบวกกัน จึงได้ว่า

$$sum = x + y$$

การวางแผนและออกแบบ (Planning and Design)

ถัดจากการวิเคราะห์ปัญหา เราจะนำสิ่งที่วิเคราะห์มาเป็นอย่างดีแล้วมาวางแผนอีกครั้งอย่างเป็นขั้นตอน เพื่อให้ขั้นตอนของการดำเนินการเป็นไปตามลักษณะของการเขียนโปรแกรมโดยชัดเจนยิ่งขึ้น

- แผนการแก้ไขปัญหานี้ มักถูกเรียกว่า **อัลกอริทึม (Algorithm)**
- วิธีการอธิบายอัลกอริทึมที่นิยม แบ่งออกเป็น 2 รูปแบบ ได้แก่
  - **ซูโดโค้ด (Pseudocode)** เป็นการอธิบายด้วยข้อความสั้น ๆ ที่สื่อความหมายง่าย ๆ
  - **โฟลวชาร์ต (Flowchart)** เป็นการใช้แผนภาพอธิบายทิศทางของการคิดและสื่อความหมาย

ซูโดโค้ด (Pseudocode)

ไม่จำเป็นต้องนำไปคอมไพล์ได้ แต่ต้องอ่านแล้วเข้าใจว่าขั้นตอนของการทำงานเป็นอย่างไร สิ่งที่ในซูโดโค้ดควรพิจารณาให้มี ได้แก่

- จุดเริ่มต้น (Start)
- ข้อมูลขาเข้า (Input)
- วิธีการคำนวณ
- การแสดงผลลัพธ์
- จุดสิ้นสุด (End, Stop)

ถ้าไม่มีสิ่งเหล่านี้อยู่ในซูโดโค้ด ให้กลับไปพิจารณาอีกครั้งว่าเป็นโปรแกรมที่เราต้องการจริงหรือเปล่า ซึ่งถ้าเป็นในวิชานี้ ถ้าขาดสิ่งใดสิ่งหนึ่งไปก็มักจะผิด

ซูโดโค้ด (Pseudocode)

จากตัวอย่างโจทย์ข้อที่แล้ว สามารถเขียนซูโดโค้ดได้ดังนี้

```
START
        READ X
        READ Y
        COMPUTE SUM=X+Y
        PRINT SUM
END
```

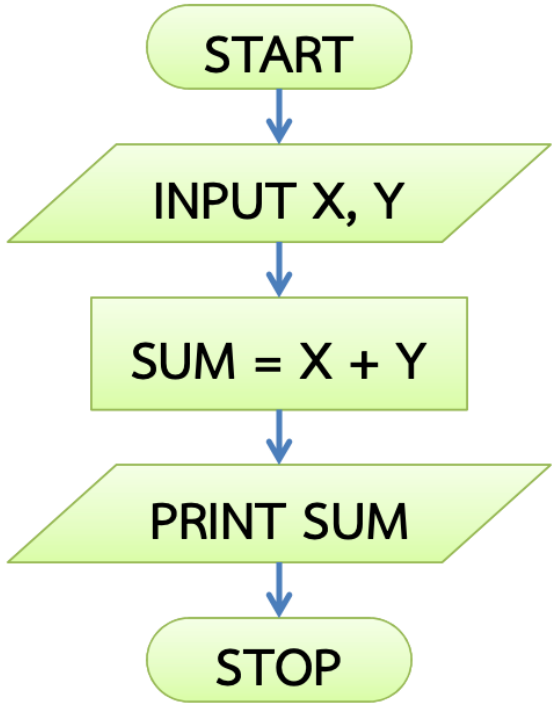
```
START
        READ X, Y
        SUM=X+Y
        PRINT SUM
END
```

ทั้งสองแบบล้วนถูกทั้งคู่ จะเห็นได้ว่าการเขียนซูโดโค้ดไม่ได้มีข้อจำกัดมากนัก
(ใช้เพื่อแสดงเฉพาะแนวคิดเท่านั้น ไม่ได้นำไปใช้ในโค้ดโดยตรง)

โฟลวชาร์ต (Flowchart)

เมื่อเขียนแล้วต้องอ่านแล้วเข้าใจว่าขั้นตอนของการทำงานเป็นอย่างไร สิ่งที่ในโฟลวชาร์ตควรพิจารณาให้มี ได้แก่

- จุดเริ่มต้น (Start)
- ข้อมูลขาเข้า (Input)
- วิธีการคำนวณ
- การแสดงผลลัพธ์
- จุดสิ้นสุด (End, Stop)



เช่นเดียวกับซูโดโค้ด ถ้าไม่มีสิ่งเหล่านี้อยู่ในโฟลวชาร์ต ให้กลับไปพิจารณาอีกครั้งว่าเป็นโปรแกรมที่เราต้องการจริงหรือเปล่า ซึ่งถ้าเป็นในวิชานี้ การขาดสิ่งใดสิ่งหนึ่งไปก็มักจะผิด

โฟลวชาร์ต (Flowchart)

องค์ประกอบพื้นฐานของโฟลวชาร์ต

จุดเริ่มต้น (Start) จุดสิ้นสุด (Stop)

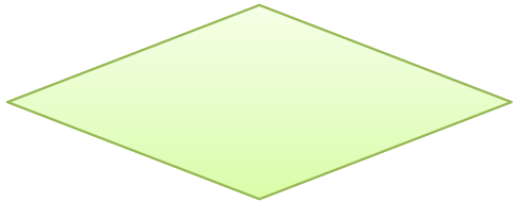
การรับค่า (Input; scanf) การส่งออก (Output; printf)

การคำนวณและประมวลผล

ระบุลำดับการทำงาน

โฟลวชาร์ต (Flowchart)

องค์ประกอบพื้นฐานของโฟลวชาร์ต

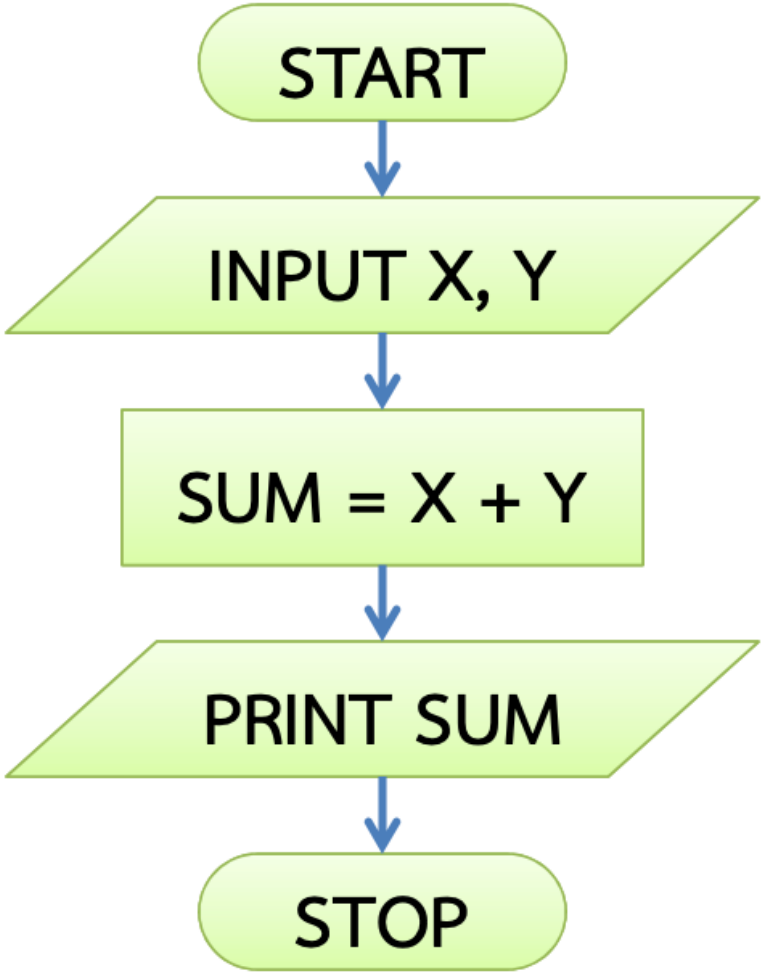


ใช้กำหนดทางเลือกเส้นทาง (ทางแยก) แบ่งออกเป็น 2 ทาง

ใช้สร้าง node หรือจุดเชื่อมต่อ

โฟลวชาร์ต (Flowchart)

เมื่อนำทุกอย่างมาประกอบกันให้เป็นลำดับและแนวทางการคิดที่ถูกต้อง จะได้



ข้อสังเกตเกี่ยวกับซูโดโค้ดและโฟลวชาร์ต

- การเขียนซูโดโค้ดและโฟลวชาร์ตเป็นการแสดงแนวทางและวิธีที่นักศึกษาใช้คิด ดังนั้นการเขียนทั้งซูโดโค้ดและโฟลวชาร์ตมีความเป็นไปได้ที่จะเขียนได้หลายแบบ
- รูปแบบการเขียนไม่ตายตัว เช่น อาจใช้ input X แยกกับ input Y หรือจะเขียนเป็น input X, Y ก็ได้ ถ้ายังสามารถสื่อความหมายแบบเดิมได้อยู่
- ไม่ได้เน้นภาษาอังกฤษ เช่น ถ้าเขียนคำสั่งผิดเล็กน้อย หรือใช้คำที่ยังคงความหมายเดิม เช่น STOP หรือ END ก็ยังคงถูกต้อง เพราะยังสื่อความหมายของการสิ้นสุดโปรแกรมได้เหมือนเดิม
- หลีกเลี่ยงการเขียนด้วยภาษาไทย (เขียนได้ แต่อาจสร้างความสับสน)

ขั้นตอนการพัฒนาโปรแกรม

- ประกอบด้วย 5 ขั้นตอน
  - วิเคราะห์ปัญหา (Analysis)
  - วางแผนและออกแบบ (Planning and Design)
  - **การเขียนโปรแกรม (Coding)**
  - **การทดสอบโปรแกรม (Testing)**
  - จัดทำเอกสารและคู่มือการใช้งานหรือพัฒนา (Documentation)

การเขียนโปรแกรม (Coding)

- คือการนำโปรแกรมที่ออกแบบไว้จากขั้นตอนที่ 1 และ 2 มาเขียนใหม่ให้ถูกต้องตามหลักไวยากรณ์คอมพิวเตอร์
- เช่นเดียวกับภาษาไทยและอังกฤษ ภาษาของคอมพิวเตอร์ก็มีหลักไวยากรณ์ แต่จะมีความเคร่งครัดมากกว่า
- หลักไวยากรณ์ ไม่ใช่ขั้นตอนการทำงาน เป็นแค่การออกคำสั่งคำสั่งเดียวเท่านั้น

ตัวอย่างการเขียนโปรแกรม (Coding)


```
#include<stdio.h>                    //เตรียมการทำงาน
void main(){                         //กลุ่มของคำสั่งหลัก
        int x, y, sum;               //ประกาศว่าจะใช้ตัวแปร 3 ตัว
        scanf("%d", &x);             //รับค่าให้ตัวแปร x
        scanf("%d", &y);             //รับค่าให้ตัวแปร y
        sum = x+y;          //หาผลรวมของ x และ y, เก็บค่าไว้ใน sum
        printf("%d", sum);           //แสดงผลทางจอภาพ
}                                    //จบการทำงาน
```

การทดสอบโปรแกรม (testing)

- คือการตรวจสอบว่าโปรแกรมที่เราเขียนทำงานถูกต้องหรือไม่
- ส่วนใหญ่จะลอง input ข้อมูล แล้วเทียบ output ที่ได้ และ output ที่คาดหวัง
- จากตัวอย่างที่แล้ว เราจะลองใส่ข้อมูล x และ y เข้าไปและเปรียบเทียบผลลัพธ์ที่ได้จากโปรแกรมกับค่าผลบวกที่ถูกต้อง
- ถ้าหาผลที่คาดหวัง (เฉลย) เองไม่ได้ ก็ไม่อาจรู้ได้ว่าโปรแกรมทำงานถูกต้องหรือไม่
- การทดสอบที่น่าเชื่อถือจะต้องทำการทดสอบหลายครั้งกับหลายๆรูปแบบ

ตัวอย่างการทดสอบโปรแกรม (testing)

- ลองรันโปรแกรมที่เขียนขึ้น และใส่ข้อมูลนำเข้าเป็น 4 และ 5
  - โปรแกรมควรได้ผลลัพธ์เป็น 9
- ลองรันโปรแกรมที่เขียนขึ้นอีกครั้ง และใส่ข้อมูลนำเข้าเป็น 20 และ -5
  - โปรแกรมควรได้ผลลัพธ์เป็นเท่าไหร่
- สังเกตว่า การทดลองจะลองกับข้อมูลหลายๆแบบ โดยเฉพาะในการสอบ มักจะทดสอบโปรแกรมของคุณต้องชุดข้อมูลสุดโต่ง คือสุดขีดจำกัดของโจทย์ที่กำหนดให้ ในการทดสอบโปรแกรมเองจึงควรทดสอบให้รอบคอบ

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

- **ปัญหา** จงวิเคราะห์ปัญหาและเขียนโฟลวชาร์ตสำหรับการหาพื้นที่สี่เหลี่ยมคางหมู เมื่อทราบความยาวคู่ขนานและระยะห่างระหว่างเส้นทั้งสอง
- สมการหาพื้นที่สี่เหลี่ยมคางหมู :
- ข้อมูลขาเข้า :
- ข้อมูลขาออก :

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

- **ปัญหา** จงวิเคราะห์ปัญหาและเขียนโฟลวชาร์ตสำหรับการหาพื้นที่สี่เหลี่ยมคางหมู เมื่อทราบความยาวคู่ขนานและระยะห่างระหว่างเส้นทั้งสอง
- โฟลวชาร์ต:

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

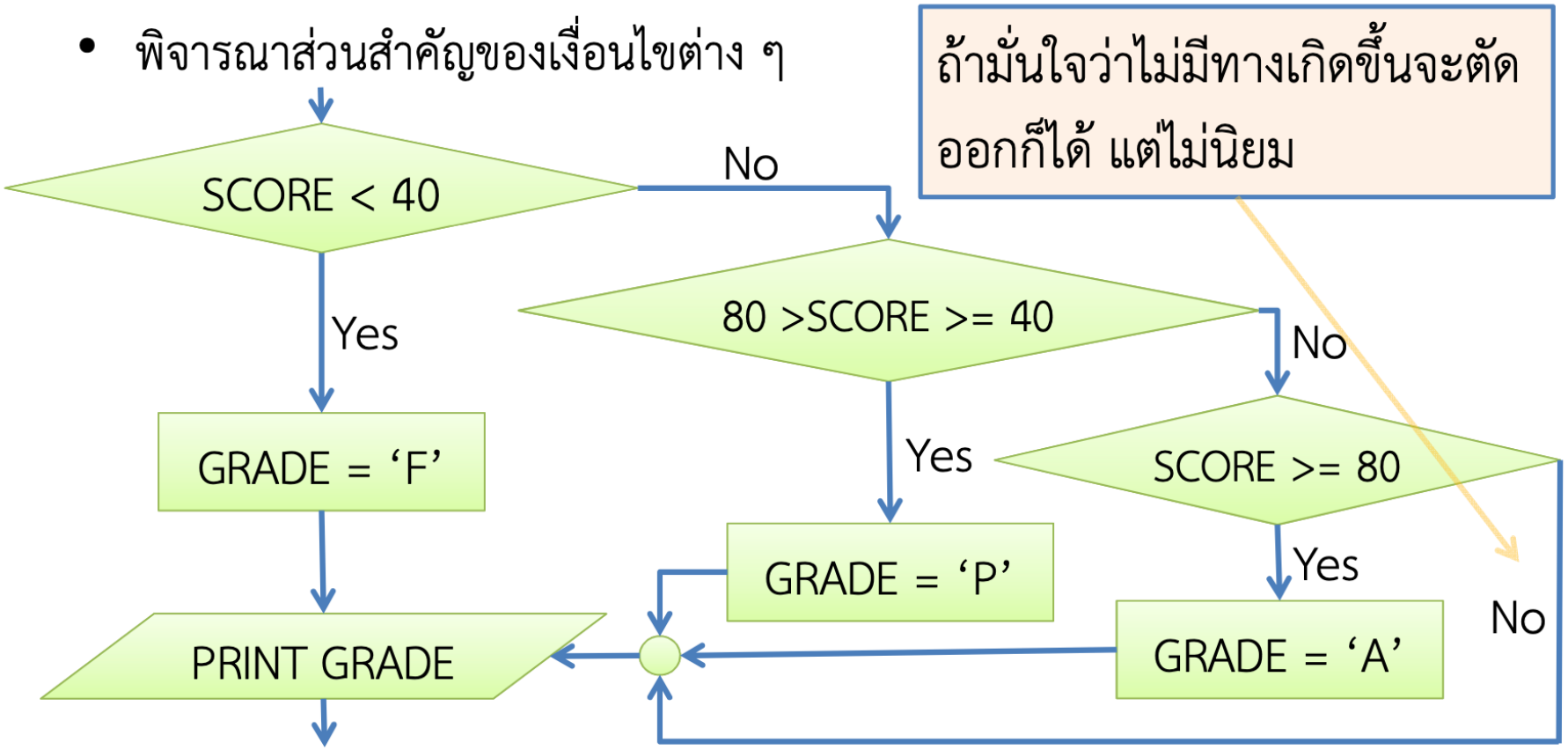
- **ปัญหา** ในการสอบบางวิชาจะกำหนดการวัดผลเป็น ผ่าน (s), ไม่ผ่าน (u) และยอดเยี่ยม ดังนี้ “ถ้าคะแนนมากกว่าหรือเท่ากับ 40 ให้วัดผลเป็น ผ่าน (s)” “ถ้าคะแนนน้อยกว่า 40 ให้วัดผลเป็น ไม่ผ่าน (u)” แต่ถ้า “คะแนนสูงกว่า 80 คะแนน ให้วัดผลเป็น ยอดเยี่ยม (A)” จงวิเคราะห์โจทย์และเขียนโฟลวชาร์ตเพื่อแก้ปัญหานี้
- ข้อมูลขาเข้า :
- ข้อมูลขาออก :

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

- **ปัญหา** ในการสอบบางวิชาจะกำหนดการวัดผลเป็น ผ่าน (s), ไม่ผ่าน (u) และ ยอดเยี่ยม ดังนี้ “ถ้าคะแนนมากกว่าหรือเท่ากับ 40 ให้วัดผลเป็น ผ่าน (s)” “ถ้าคะแนนน้อยกว่า 40 ให้วัดผลเป็น ไม่ผ่าน (u)” แต่ถ้า “คะแนนสูงกว่า 80 คะแนน ให้วัดผลเป็น ยอดเยี่ยม (A)” จงวิเคราะห์โจทย์และเขียนโฟลวชาร์ตเพื่อแก้ปัญหานี้
- โฟลวชาร์ต:

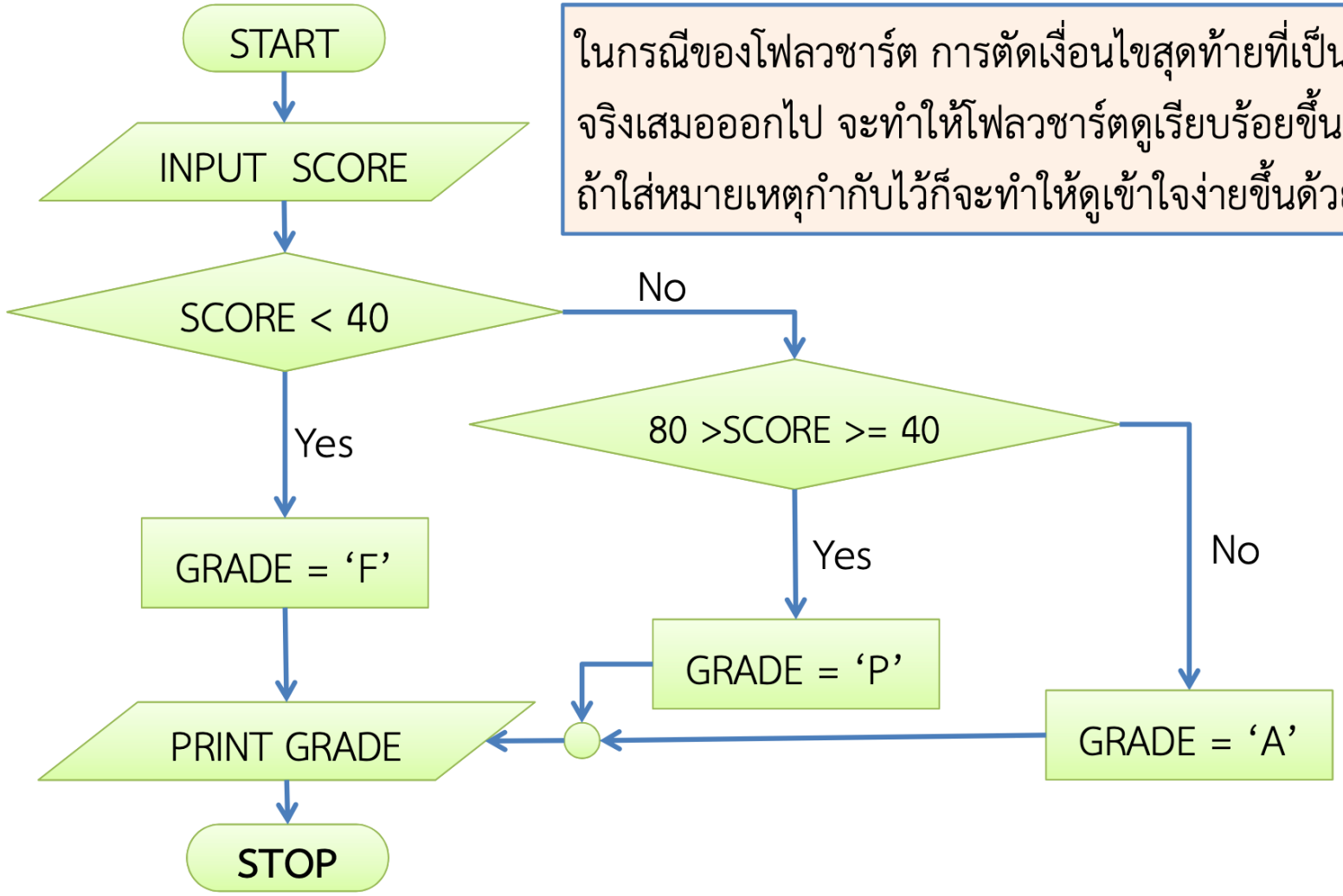
ขั้นตอนวิธีกับเส้นทางที่ไม่มีทางเกิดขึ้น

- โฟลวชาร์ต:



ขั้นตอนวิธีกับเส้นทางที่ไม่มีทางเกิดขึ้น

- โฟลวชาร์ต:



ในกรณีของโฟลวชาร์ต การตัดเงื่อนไขสุดท้ายที่เป็นจริงเสมอออกไป จะทำให้โฟลวชาร์ตดูเรียบร้อยขึ้น ถ้าใส่หมายเหตุกำกับไว้ก็จะทำให้ดูเข้าใจง่ายขึ้นด้วย

ลองเขียนซูโดโค้ด


```
START
READ SCORE
IF SCORE > 40 THEN
        GRADE ‘F’
ELSE IF SCORE >= 40 AND SCORE < 80 THEN
        GRADE ‘P’
ELSE
        GRADE ‘A’
END IF
PRINT GRADE
STOP
```

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

- **ปัญหา** จงพิมพ์คำว่า“Hello”ถ้าเลขที่อ่านเข้ามามีค่าเท่ากับ 1

- ข้อมูลขาเข้า :
- ข้อมูลขาออก :

ตัวอย่างการวิเคราะห์โจทย์ปัญหา (ทำไปพร้อมกัน)

- **ปัญหา** จงพิมพ์คำว่า“Hello”ถ้าเลขที่อ่านเข้ามามีค่าเท่ากับ 1
- ซูโดโค้ด:

การวนซ้ำ (Loop)

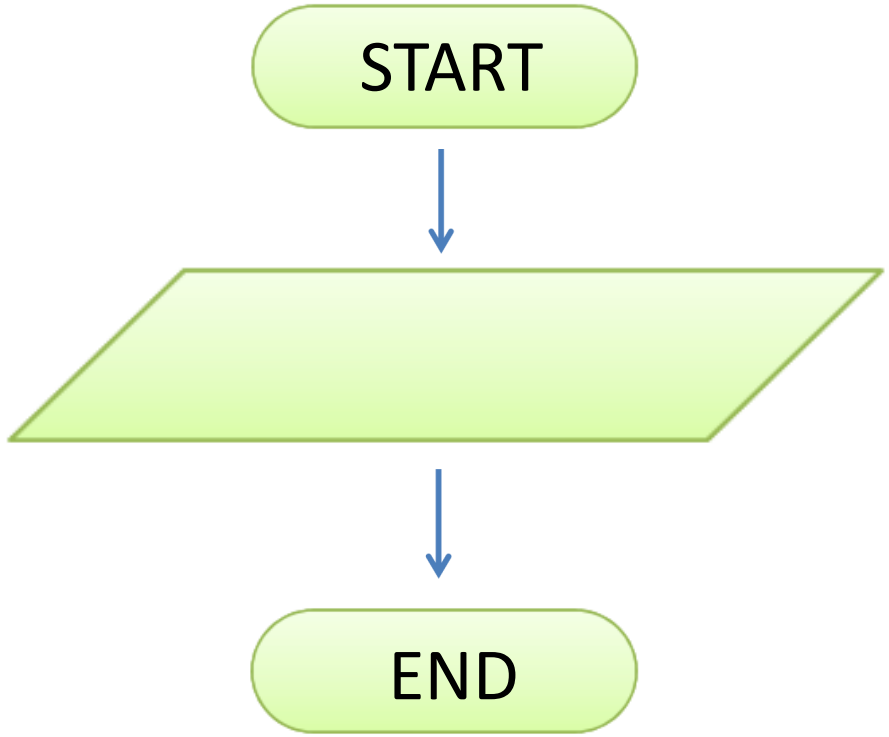
- เราสามารถสร้างเส้นทางของการทำงานที่วนซ้ำได้
- รายละเอียดของการวนซ้ำอาจเปลี่ยนไปได้เรื่อย ๆ คือในแต่ละรอบ อาจเหมือนเดิมทุกประการหรือมีการอัพเดทอะไรบางอย่างได้
- การวนซ้ำมักจะต้องมีเส้นทางสำหรับการออกจากวังวน

การวนซ้ำ (Loop)

- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 5 ครั้ง ออกทางจอภาพ
- **วิเคราะห์โจทย์**
- ข้อมูลนำเข้า: -
- ข้อมูลส่งออก: คำว่า Hello ซ้ำกันจำนวน 5 ครั้ง
- a.k.a HelloHelloHelloHelloHello

การวนซ้ำ (Loop)

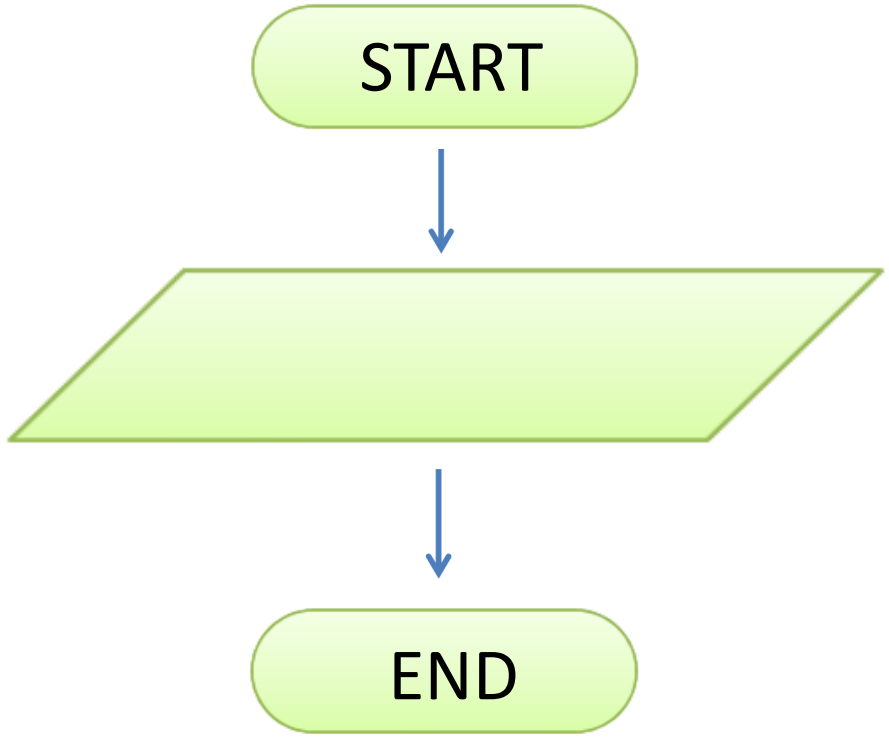
- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 10 ครั้ง ออกทางจอภาพ
- Flowchart: แบบง่าย ๆ



การวนซ้ำ (Loop)

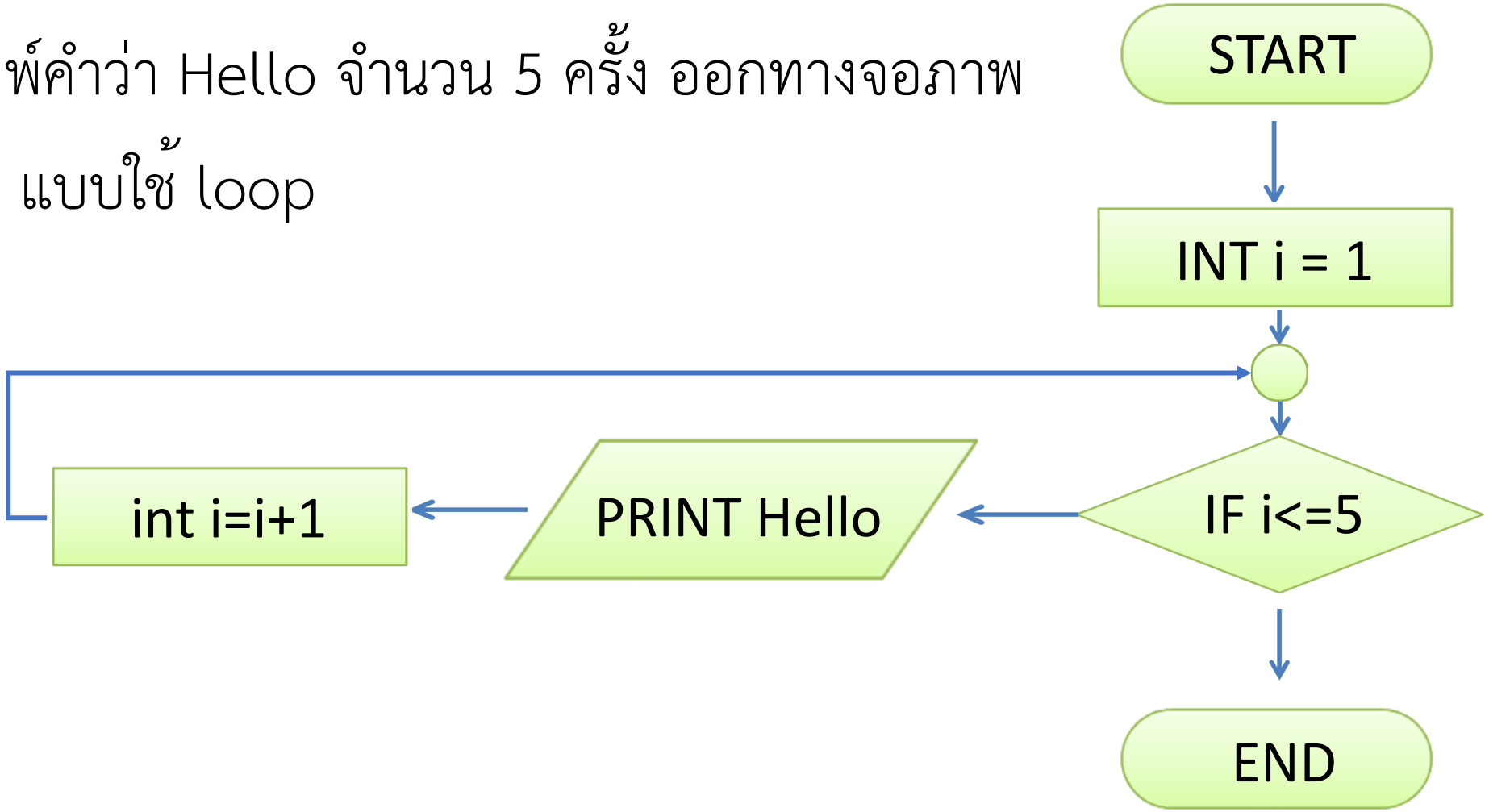
- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 1000 ครั้ง ออกทางจอภาพ
- Flowchart: แบบง่าย ๆ

  (มีใครทำได้บ้าง)



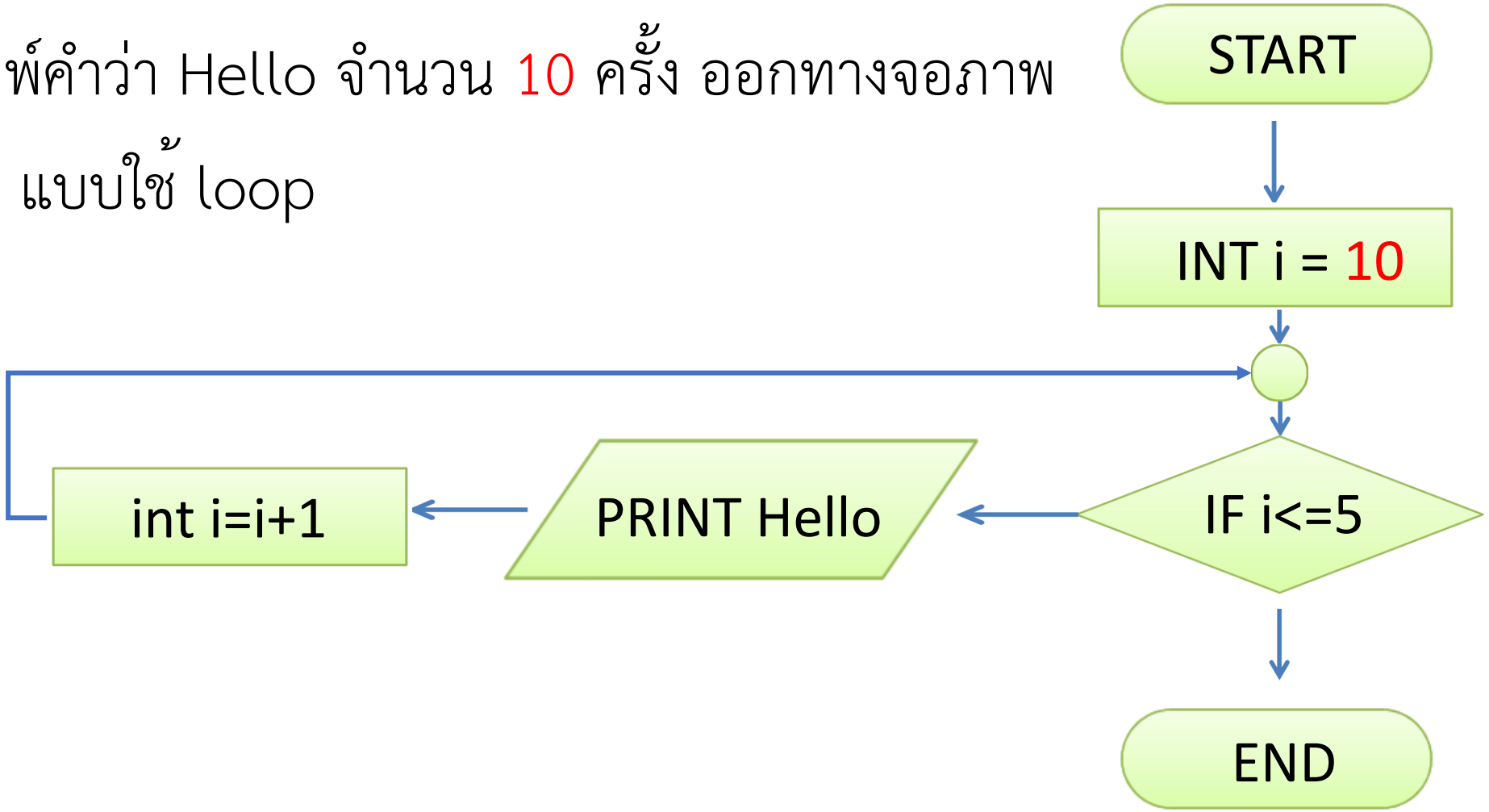
การวนซ้ำ (Loop)

- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 5 ครั้ง ออกทางจอภาพ
- Flowchart: แบบใช้ loop



การวนซ้ำ (Loop)

- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 10 ครั้ง ออกทางจอภาพ
- Flowchart: แบบใช้ loop

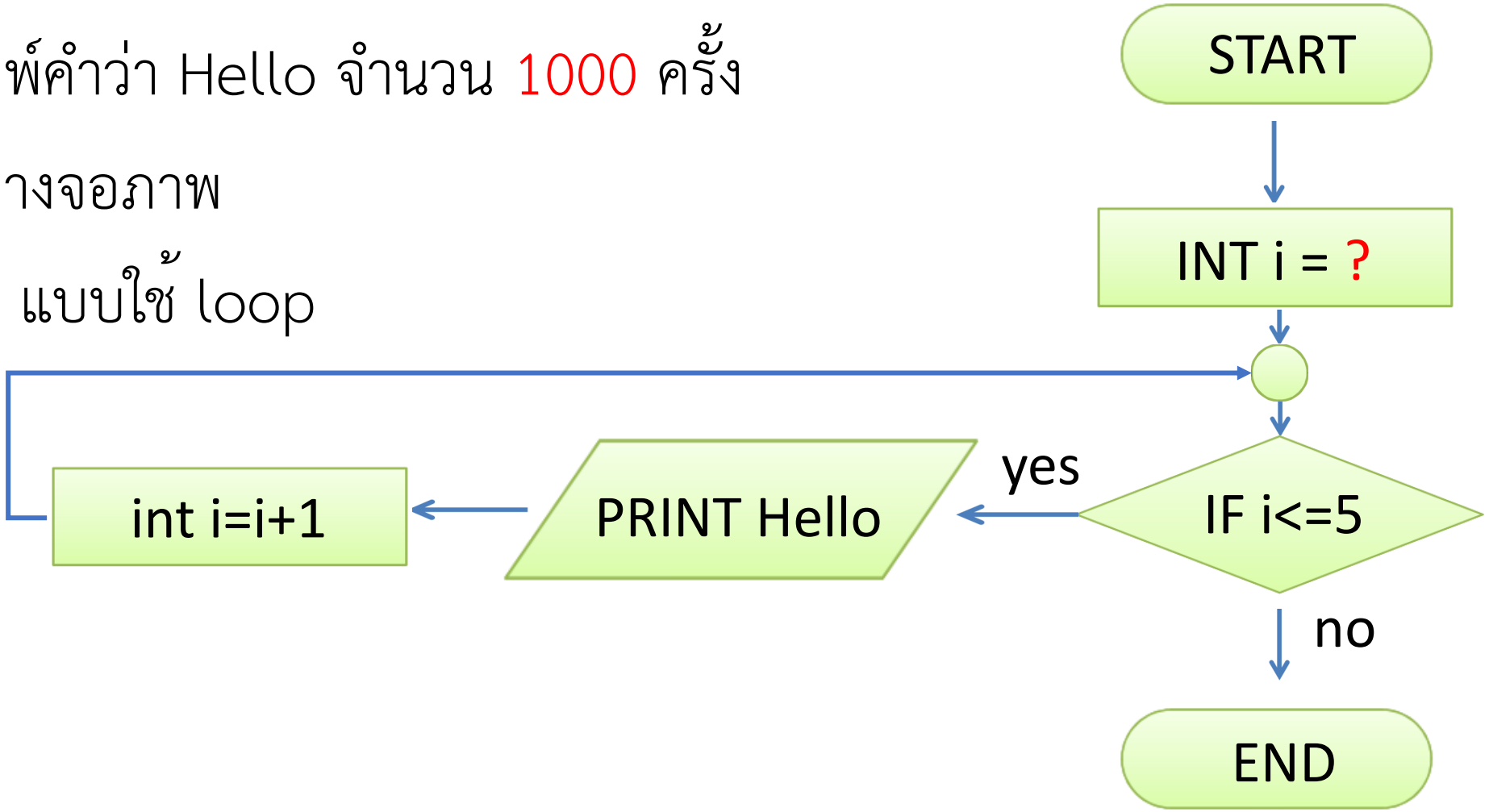


การวนซ้ำ (Loop)

- **โจทย์** จงพิมพ์คำว่า Hello จำนวน 1000 ครั้ง

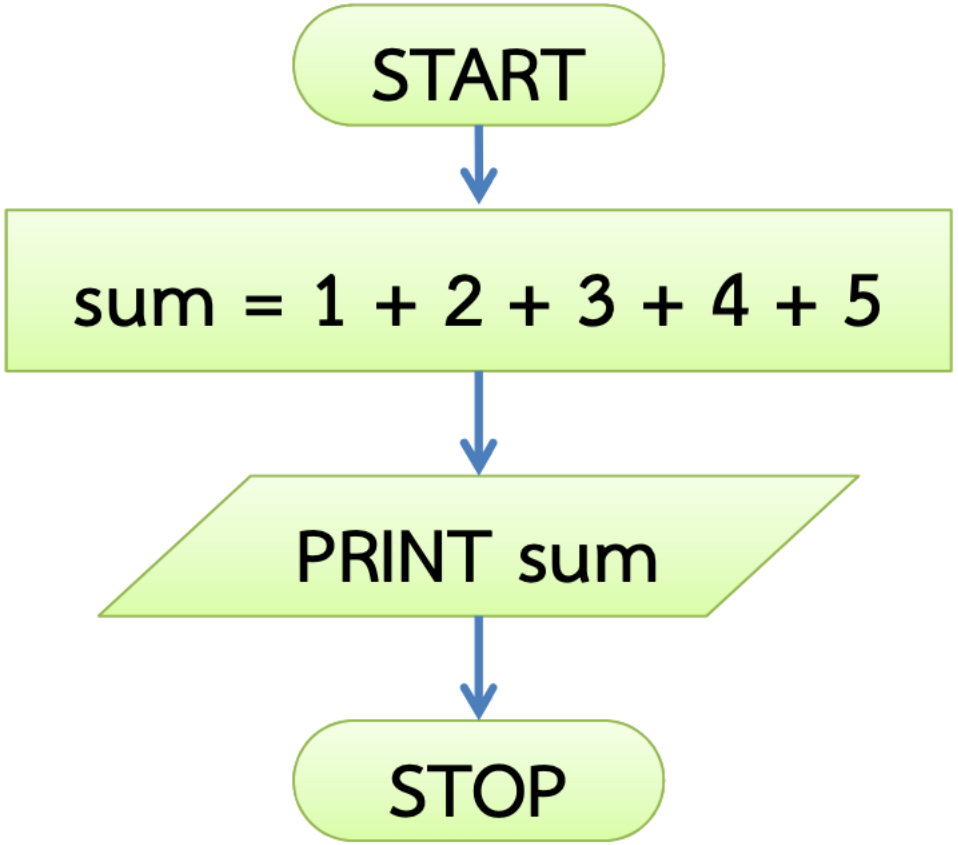
  ออกทางจอภาพ

- Flowchart: แบบใช้ loop



การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของ 1+2+3+4+5
- Flowchart:



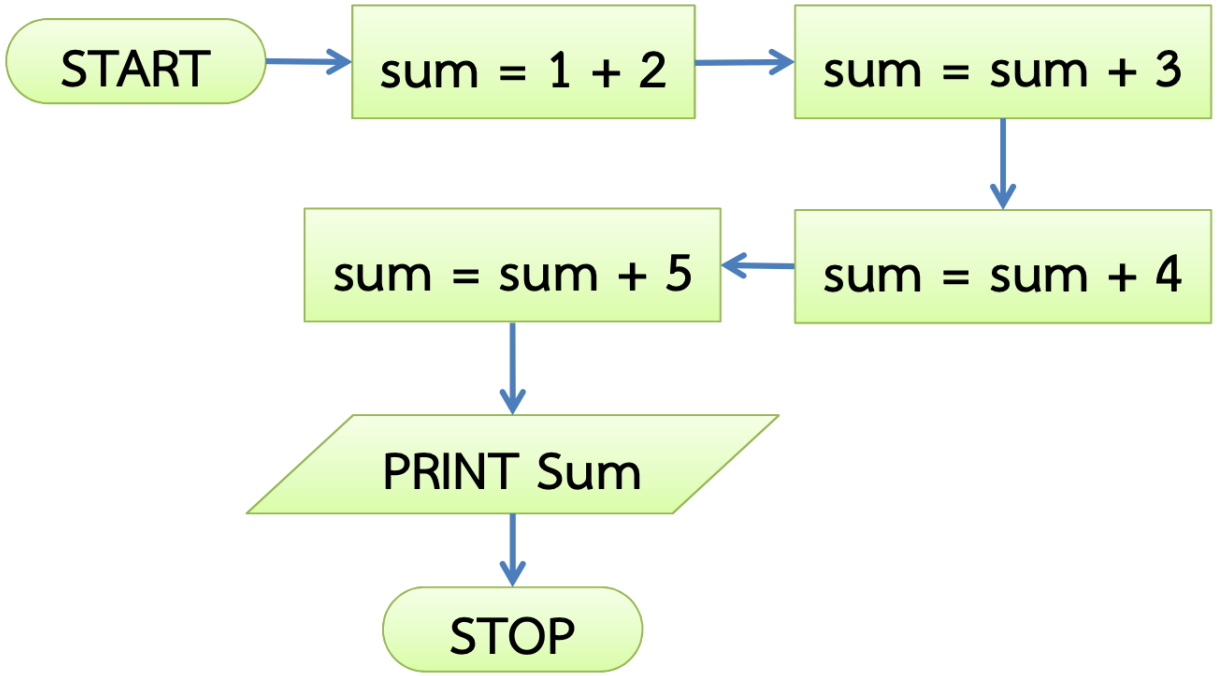
การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของ 1+2+3+4+5+...+1000
- Flowchart:

การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของ 1+2+3+4+5

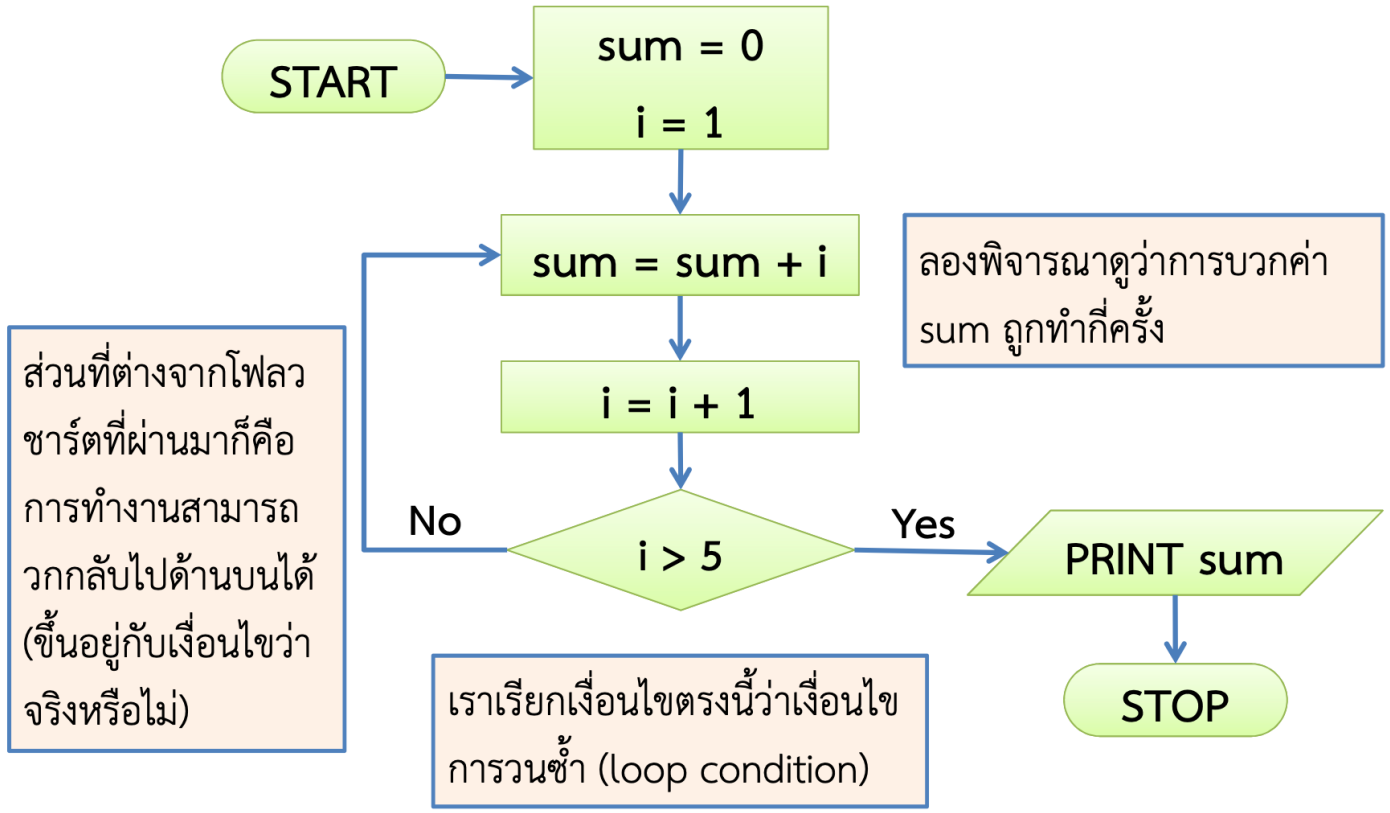
Flowchart (ลองคิดหาช่องทางให้ loop; เก็บค่าไว้บวกต่อ):



การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของ 1+2+3+4+5

Flowchart (ใช้ loop):



การวนซ้ำ (Loop): ซูโดโค้ดของการวนซ้ำ

- **โจทย์** จงหาผลบวกของ 1+2+3+4+5

ซูโดโค้ด (ใช้ loop):

```
START
sum = 0
i = 1
WHILE i <= 5 DO
        sum = sum + i
        i =i + 1
END WHILE
PRINT sum
END
```

การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของ 1+2+3+...+9+10

Flowchart หรือ ซูโดโค้ด (ใช้ loop):

การวนซ้ำ (Loop)

- **โจทย์** จงหาผลบวกของเลข 10 ค่าที่ป้อนเข้าไป (ครั้งนี้อาจไม่เรียงตามลำดับ และแต่ละค่าอาจมีค่าเป็นเท่าไรก็ได้)

ความเป็นไปได้ที่หลากหลายของการเขียนโปรแกรม

- **โจทย์** จงหาผลบวกของเลขคู่ทั้งหมด ที่มีค่าระหว่าง 0 ถึง 100
- **วิเคราะห์โจทย์**
  - **ข้อมูลนำเข้า :** ไม่มี
  - **ข้อมูลขาออก :** ผลบวกของเลขคู่ทั้งหมด ที่มีค่าระหว่าง 0 ถึง 100

ความเป็นไปได้ที่หลากหลายของการเขียนโปรแกรม

- **โจทย์** จงหาผลบวกของเลขคู่ทั้งหมด ที่มีค่าระหว่าง 0 ถึง 100
- **แนวทาง**
  1. ใช้สมการอนุกรมเลขคณิต นำค่า 0 และ 100 ใส่ลงในสมการ โปรแกรมทำการคำนวณและ Output ออกมาได้เลย
  2. ใช้ if-else ร่วมกับ loop ในการวนซ้ำและเพิ่มค่าตัวเลขทีละ 1 โดยเริ่มจาก 0 ตรวจสอบตัวเลขว่าเป็นเลขคู่หรือไม่ และนำเลขคู่ที่พบไปบวกกับค่าผลรวม
  3. ใช้ loop และเพิ่มค่า i ทีละ 2 โดยให้ i เริ่มต้นที่ 0

ความเป็นไปได้ที่หลากหลายของการเขียนโปรแกรม

- **โจทย์** จงหาผลบวกของเลขคู่ทั้งหมด ที่มีค่าระหว่าง 0 ถึง 100

ความเป็นไปได้ที่หลากหลายของการเขียนโปรแกรม

- **โจทย์** จงหาผลบวกของเลขคู่ทั้งหมด ที่มีค่าระหว่าง 0 ถึง 100

สรุป Introduction to Compro I

- แผนการเรียนรู้
- การวิเคราะห์โจทย์
  - ข้อมูลนำเข้า
  - ข้อมูลขาออก
  - โฟลวชาร์ต
  - ซูโดโค้ด

เมื่อนักศึกษาปี 1 พบกับรุ่นพี่ในรายวิชาคอมโปร1