ภาพรวมของการพัฒนาโปรแกรม

Program Development — overview

วัฏจักรการพัฒนาโปรแกรม

1. การวิเคราะห์ปัญหา (Analysis the Problem)
2. การออกแบบโปรแกรม (Design the Program)
3. การเขียนโปรแกรม (Coding)
4. การตรวจสอบข้อผิดพลาดของโปรแกรม (Debugging)
5. การทดสอบความถูกต้องของโปรแกรม (Validating)
6. การทำเอกสารประกอบโปรแกรม (Documentation)
7. การบำรุงรักษาโปรแกรม (Program Maintenance)

- **แต่ละข้อ ไม่ใช่ทำเสร็จแล้ว เสร็จเลย อาจต้องย้อนกลับมาทำใหม่/ทำหลาย ๆรอบ**
- **ไม่จำเป็นต้องทำทีละข้อ อาจต้องทำบางข้อพร้อม ๆกัน**

การวิเคราะห์ปัญหา (Problem Analysis)

- เป็นการนำโจทย์ปัญหามา
  - แยกส่วนประกอบ
  - วิเคราะห์ขั้นตอนการทำงานในการแก้ปัญหานั้น
- สิ่งที่ต้องกระทำ
  - การระบุข้อมูลออก (Output Specification) — ผลลัพธ์
  - การระบุข้อมูลเข้า (Input Specification) — ต้องใช้ข้อมูลอะไรบ้าง
  - การระบุวิธีการประมวลผล (Process Specification)
    - ทำอย่างไรเพื่อให้ได้ผลลัพธ์ที่ต้องการจากข้อมูลนำเข้า

ตัวอย่างการวิเคราะห์ปัญหา #01

“จงหาพื้นที่ของรูปสี่เหลี่ยมผืนผ้าที่กำหนดให้”

การวิเคราะห์ปัญหาได้ผลดังนี้

- ข้อมูลออก พื้นที่ของรูปสี่เหลี่ยมผืนผ้า
- ข้อมูลเข้า ความกว้างและความยาวของรูปสี่เหลี่ยมผืนผ้า
- การประมวลผล นำความกว้างมาคูณกับความยาว เพื่อให้ได้พื้นที่

ตัวอย่างการวิเคราะห์ปัญหา #02

"จงหาสมการเชิงเส้นจากชุดข้อมูลที่กำหนดให้"

ผลการวิเคราะห์ปัญหา

- ข้อมูลออก — สมการเชิงเส้น
  - a, b จากรูปสมการ $y = ax + b$
- ข้อมูลเข้า — ชุดข้อมูล
  - 2 ชุดสำหรับค่า x และ y โดยมีการระบุว่าข้อมูลตัวไหนในชุด x มีความสัมพันธ์กับข้อมูลตัวไหนในชุด y
- การประมวลผล — หาสมการเชิงเส้นจากชุดข้อมูล เช่นการใช้ linear regression

ตัวอย่างการวิเคราะห์ปัญหา #03

"หาเนื้อคู่ให้หน่อย"

ผลการวิเคราะห์ปัญหา

- ข้อมูลออก ข้อมูลเกี่ยวกับเนื้อคู่ เช่น ???
- ข้อมูลเข้า ข้อมูลเกี่ยวกับผู้รับคำทำนาย เช่น ???
- การประมวลผล ???

การออกแบบโปรแกรม (Program Design)

- ออกแบบโครงสร้างโดยรวมของโปรแกรม
- เปรียบเทียบได้กับการสร้างแบบแปลนอาคาร โดยที่ยังไม่ได้สร้างอาคาร
- ใช้เครื่องมือช่วยในการออกแบบโปรแกรม
  - ขั้นตอนวิธี (Algorithm)
  - รหัสเทียม (Pseudocode)
  - ผังงาน (Flowchart)

ขั้นตอนวิธี (Algorithm)

- เป็นภาษาพูดในการอธิบายการทำงานของโปรแกรม
- แบ่งเป็นข้อๆ ที่มีการทำงานที่ชัดเจน ไม่กำกวม และต้องทำตามลำดับ
- ตัวอย่าง: ขั้นตอนวิธีในการหาพื้นที่สี่เหลี่ยมผืนผ้า

1. รับค่าความกว้างของรูปสี่เหลี่ยมผืนผ้า
2. รับค่าความยาวของรูปสี่เหลี่ยมผืนผ้า
3. คำนวณค่าพื้นที่รูปสี่เหลี่ยมผืนผ้า
   โดยใช้สูตร พื้นที่ = ความกว้าง × ความยาว
4. แสดงผลค่าพื้นที่รูปสี่เหลี่ยมผืนผ้า

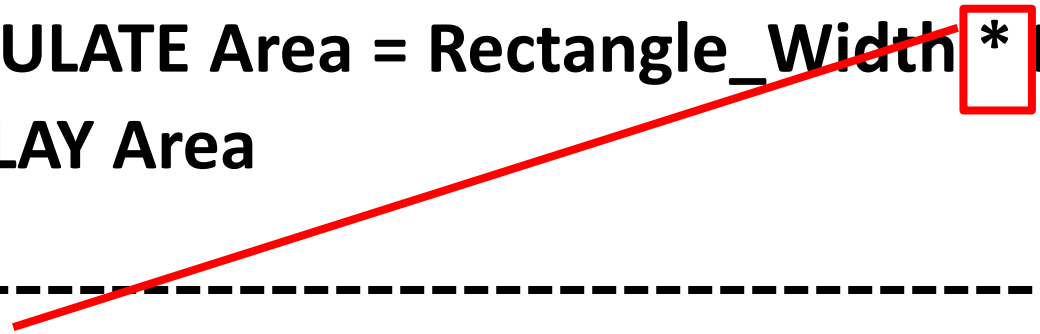
รหัสเทียม (Pseudocode)

- มีลักษณะเป็นคำสั่งที่คล้ายกับคำสั่งในภาษาคอมพิวเตอร์ทั่วไป
- มีความคล้ายกับภาษาคอมพิวเตอร์มาก
- มีการกำหนดตัวแปรสำหรับเก็บข้อมูล ข้อมูลนำเข้า ผลลัพธ์ และ/หรือข้อมูลที่ใช้ระหว่างการประมวลผล

รหัสเทียม (Pseudocode) (ต่อ)

- ตัวอย่าง: รหัสเทียมของการหาพื้นที่สี่เหลี่ยมผืนผ้า

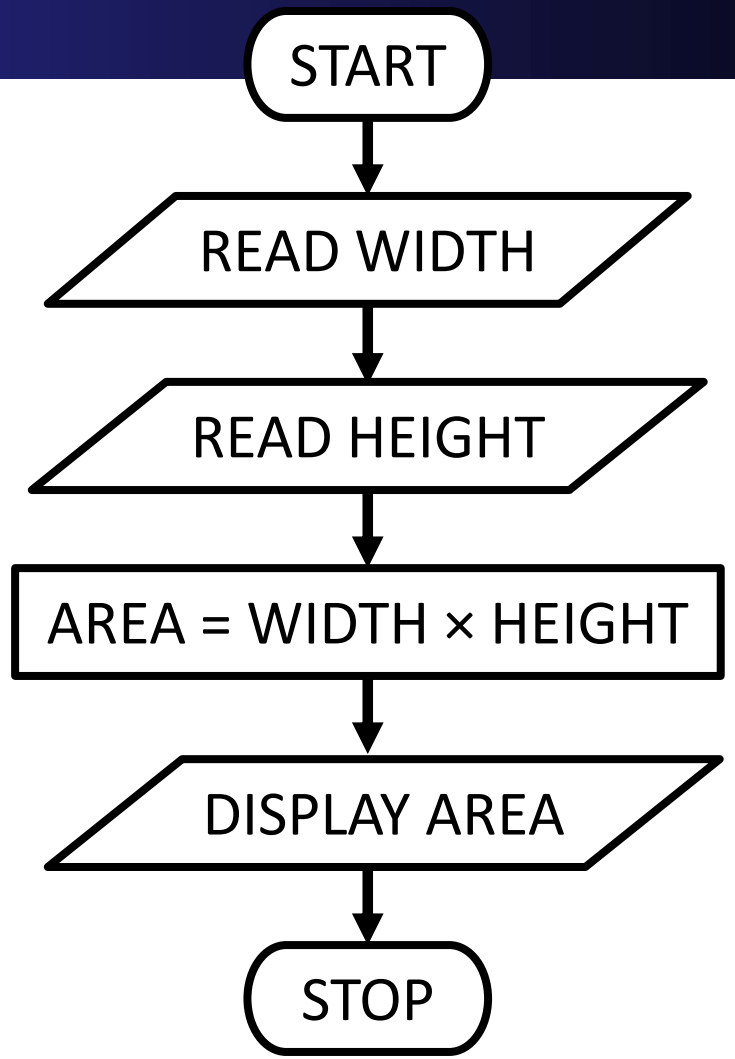
```
START
        READ Rectangle_Width
        READ Rectangle_Height
        CALCULATE Area = Rectangle_Width * Rectangle_Height
        DISPLAY Area
STOP
```



ใช้แทนตัวคูณบนคอมพิวเตอร์

ผังงาน (Flowchart)

- อธิบายด้วยรูปภาพโดยใช้สัญลักษณ์ต่าง ๆ
- มีเส้นแสดงทิศทางการทำงาน
- จะเห็นภาพรวมของโปรแกรมได้ง่าย
- ตัวอย่าง: ผังงานของตัวอย่างปัญหาการหาพื้นที่สี่เหลี่ยมผืนผ้า

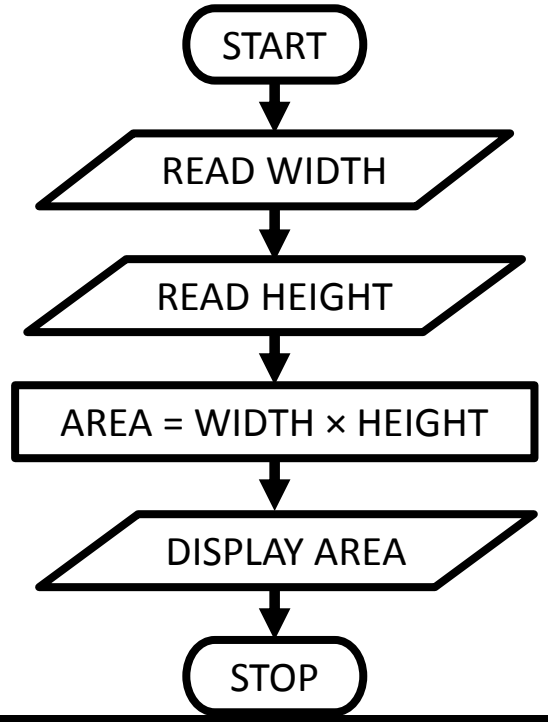


การเขียนโปรแกรม (Coding)

- เขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ เช่น Python
- เป็นการแปลง ขั้นตอนวิธี รหัสเทียม หรือ ผังงาน เป็นชุดคำสั่ง (Code) ภาษาคอมพิวเตอร์แล้วสามารถนำไปแปลเป็นโปรแกรมคอมพิวเตอร์ได้

การเขียนโปรแกรม (Coding) (ต่อ)

- ตัวอย่าง:การหาพื้นที่สี่เหลี่ยมผืนผ้า จากผังงาน ไปเป็นชุดคำสั่งภาษา Python




```
Width = int(input())
Length = int(input())
Area = Width * Length
print(Area)
```

การตรวจสอบข้อผิดพลาดของโปรแกรม (Debugging)

- เป็นการหาข้อผิดพลาด (Error) ของโปรแกรม และแก้ไข เพื่อให้โปรแกรมทำงานได้อย่างถูกต้อง
- ประเภทของข้อผิดพลาดของโปรแกรม
  - ข้อผิดพลาดทางไวยากรณ์ของภาษา (Syntax Error)
  - ข้อผิดพลาดในระหว่างการทำงานของโปรแกรม (Run-Time Error)
  - ข้อผิดพลาดทางในการตีความหมายของโปรแกรมผิด (Logical Error)

ข้อผิดพลาดทางไวยากรณ์ของภาษา (Syntax Error)

- เป็นข้อผิดพลาดที่ตรวจพบและแก้ไขได้ง่ายที่สุด
- พบในระหว่างการแปลโปรแกรมจากภาษาคอมพิวเตอร์เป็นภาษาเครื่อง ซึ่งเป็นหน้าที่ของ Compiler หรือ Interpreter
- เกิดจากการพิมพ์คำสั่งต่าง ๆ หรือโครงสร้างในโปรแกรมผิด เช่น
  - ในภาษา Python ถ้าต้องการพิมพ์ข้อความต้องใช้คำสั่ง

    print เช่น print(‘Hello World’)

    ถ้าเขียนผิดเป็น primt(‘Hello World’)

    เมื่อคอมไพเลอร์แปลโปรแกรม คอมไพเลอร์จะแจ้งให้ทราบว่าคำสั่งผิด

ข้อผิดพลาดในระหว่างการทำงานของโปรแกรม (Run-Time Error)

- คอมไพเลอร์ไม่สามารถตรวจสอบได้ในระหว่างการแปลโปรแกรม
- ข้อผิดพลาดเกิดขึ้นในระหว่างการทำงานของโปรแกรม ที่โปรแกรม สามารถตรวจสอบได้เอง
- เช่นเขียนโปรแกรมในการหาค่า 1/x โดยให้ผู้ใช้ใส่ค่า x ข้อผิดพลาดจะเกิดขึ้นในกรณีที่ผู้ใช้ใส่ค่า x เป็น 0 เนื่องจากในทางคณิตศาสตร์ห้ามให้ 0 เป็นตัวหาร

ข้อผิดพลาดทางในการตีความหมายของโปรแกรมผิด (Logical Error)

- เกิดจากการ
  - เข้าใจโจทย์ผิด
  - เข้าใจ Requirement ผิด
  - เข้าใจการทำงานของ Function ในโปรแกรมผิด
  - พิมพ์ผิดแบบที่ไม่ทำให้เกิด syntax error หรือ runtime error

- เป็นข้อผิดพลาดที่ค้นหาและแก้ไขได้ยากที่สุด
- คอมไพเลอร์ไม่สามารถตรวจสอบได้ในระหว่างการแปลโปรแกรม
- จะไม่มีการเตือนเกี่ยวกับข้อผิดพลาดในระหว่างการทำงานของโปรแกรม
- ข้อผิดพลาดจะอยู่ในรูปของผลลัพธ์การทำงานที่ไม่ถูกต้อง
- จะต้องทำการเปรียบเทียบระหว่างผลลัพธ์จากการทำงาน และผลลัพธ์ที่ควรจะได้ด้วยตัวเอง

ข้อผิดพลาดทางในการตีความหมายของโปรแกรมผิด: ตัวอย่าง

- สิ่งที่ต้องการ

"จงหาค่า *a* จากสูตร $a = x + y - z$"

- รหัสเทียมแสดงการประมวลผล

a = x + y + z

- ข้อผิดพลาด

การคำนวณไม่ตรงกับสูตรที่กำหนดให้ เนื่องจากค่า z ถูกบวกเข้าไปแทนที่จะถูกลบออก

- การแก้ไข

a = x + y - z

การทดสอบความถูกต้องของโปรแกรม (Validating)

- การตรวจสอบขอบเขตของข้อมูล (Range Check)
  - เช่น การป้อนข้อมูลวันที่ในส่วนเดือน ต้องมีค่าไม่น้อยไปกว่า 1 ลำไม่เกิน 12 ถ้าเกิน ต้องให้ป้อนข้อมูลใหม่
- การตรวจสอบความสมบูรณ์ของข้อมูล (Completeness Check)
  - เช่น การป้อนข้อมูลวันที่ในรูปแบบ ddmmyy ต้องป้อน 6 หลักเช่น 140543 ถ้าป้อนไม่ครบ เช่น 1405 ต้องให้ป้อนข้อมูลใหม่
- การตรวจสอบชนิดของข้อมูล (Data Type Check)
  - เช่น ข้อมูลที่เป็นอายุ ผู้ใช้ต้องป้อนเป็นตัวเลขเท่านั้น ถ้าป้อนเป็นตัวอักษร ต้องป้อนข้อมูลใหม่

การทำเอกสารประกอบโปรแกรม (Documentation)

- เอกสารประกอบโปรแกรมสำหรับผู้ใช้ (User Documentation)
  - อธิบายการใช้งานโปรแกรม
  - คู่มือการใช้งาน
- เอกสารประกอบสำหรับผู้พัฒนาโปรแกรม (Technical Documentation)
  - อธิบายการทำงานของโปรแกรม
  - อธิบายชุดคำสั่ง (Source Code=โปรแกรมที่ยังไม่ได้ Compile) ของโปรแกรม
  - อาจมีการอธิบายในส่วนของ ขั้นตอนวิธี รหัสเทียม หรือ ผังงาน
  - มีประโยชน์ในการนำโปรแกรมไปพัฒนาต่อไป หรือ ตรวจสอบข้อผิดพลาด

!!!ไม่ควรทำแค่ตอนที่เขียนโปรแกรมเสร็จ ควรจะมีการทำเอกสารควบคู่ไปกับทุกขั้นตอน!!!

ตัวอย่างเอกสารประกอบโปรแกรม

sum(a, b)

คืนค่า a + b

a และ b ต้องเป็นตัวเลข

- แสดงชื่อฟังก์ชัน (โปรแกรมย่อย) และตัวแปรข้อมูลขาเข้า
- แสดงการประมวลผล และข้อมูลขาออก
- แสดงข้อจำกัดของข้อมูลขาเข้า

การบำรุงรักษาโปรแกรม (Program Maintenance)

- เมื่อนำโปรแกรมไปใช้งานอาจมีข้อผิดพลาดเล็กๆน้อยๆเกิดขึ้น ต้องทำการแก้ไขข้อผิดพลาด (Corrective maintenance) นั้น
- การปรับปรุง พัฒนา โปรแกรมให้ดีขึ้น เช่น ปรับเปลี่ยนหน้าจอให้ง่ายต่อการใช้งาน หรือ เพิ่มการทำงานใหม่ ๆ (Perfective maintenance) ที่ไม่ใช่ Function ใหญ่ เป็นการเพิ่ม Function เล็กๆ เพื่อเสริมการทำงานของโปรแกรมให้สมบูรณ์ขึ้น
- การปรับปรุง พัฒนาโปรแกรมให้ทำงานกับ Hardware หรือ OS version ใหม่ๆ (Adaptive maintenance) เช่น การต่อเพิ่มอุปกรณ์ IO ใหม่ เช่นอุปกรณ์นำข้อมูลเข้าที่เป็นที่อ่านลายนิ้วมือ