

หลักการออกแบบโปรแกรมที่ดี

Principles of good programming

- สมบัติของโปรแกรมที่ดี
- หลักการออกแบบโปรแกรมแบบโครงสร้าง
- การออกแบบล้วนต่อประสานกับผู้ใช้



วิธีการเขียนโปรแกรมที่ดี

- ทำงานได้ตามที่ต้องการ
 - ถูกต้องแม่นยำ
 - อ่านเข้าใจได้
 - ปรับปรุงแก้ไขได้ง่าย
 - ง่ายต่อการใช้
 - มีวิธีการเขียนที่มีระบบแบบแผน
- วิเคราะห์และบทวนปัญหาอย่างถี่ถ้วน
 - ทดสอบการทำงาน
 - มีการย่อเยื้อง (indentation)
 - ตั้งชื่อตัวแปรอย่างมีความหมาย
 - ใช้ comment statement เพื่ออธิบายหรือระบุข้อสังเกตุ
 - แบ่งโปรแกรมออกเป็นส่วนๆ (Modularization)
 - ออกแบบส่วนต่อประสานกับผู้ใช้สอดคล้องวิธีธรรมชาติ
 - มีข้อความสื่อสารเพื่อโต้ตอบกับผู้ใช้เพื่อไกด์การทำงานของโปรแกรมหรือกำกับข้อมูลที่ปรากฏ
 - อาศัยหลักการการโปรแกรมแบบโครงสร้าง



- ทำงานได้ตามที่ต้องการ (Meet requirements)
- ถูกต้องแม่นยำ (Accurate)
- อ่านเข้าใจได้ (Readable)
- ปรับปรุงแก้ไขได้ง่าย (Maintainable)
- ง่ายต่อการใช้ (User friendly)
- มีวิธีการเขียนที่มีระบบแบบแผน



ตัวอย่างโปรแกรมที่ไม่เป็นแบบโครงสร้าง

Unstructured Program Example

:

```
100 FOR I = 1 TO N
110 FOR J = I + 1 TO N - 1
120 IF A(I) > A(J) THEN 140
130 GOTO 170
140 AA = A(I)
150 A(I) = A(J)
160 A(J) = AA
170 NEXT J
180 NEXT I
```



ตัวอย่างโปรแกรมแบบโครงสร้าง

Structured Program Example

```
/* ชุดคำสั่งต่อไปนี้ ทำการเรียงลำดับเลข N จำนวน  จากน้อยไปมาก*/
for (i = 1; i <= N; i++) /* ทำซ้ำต่อไปนี้ สำหรับทุกค่าของ i=1 ถึง N*/
    for (j = i+1; j < N; j++) /* ทำซ้ำต่อไปนี้ สำหรับทุกค่าของ j เริ่มตั้งแต่ i+1 ถึง N-1*/
        if (A[i] > A[j]) { /* สลับค่าระหว่างตัวที่ i กับ ค่าตัวที่ j*/
            AA = A[i];
            A[i] = A[j];
            A[j] = AA;
        }
```

204112 Structured Programming



5

หลักการนามธรรม Abstraction Concept

- **นามธรรม** หมายถึงการพิจารณาบางสิ่งบางอย่างที่แยกจากความเป็นจริงของสิ่งนั้น เป็นการทำให้ข้อเท็จจริงต่างๆ ดูง่ายขึ้น โดยอธิบายเพียงว่าสิ่งเหล่านั้น ทำอะไรบ้าง ยังไม่ต้องกล่าวถึงวิธีการว่าทำอย่างไร
- **WHAT to do, not HOW to do**
 - ทำการพิจารณารายละเอียดเพิ่มขึ้นทีละชั้น กลุ่มของฟังก์ชันหรือมодูลในระดับล่างสุดจะละเอียดที่สุด
 - มอดูลที่อยู่ในระดับเดียวกันหรือต่างระดับกัน จะสามารถสื่อสารถึงกันได้ ตามแบบที่กำหนด โดยมีกฎเกณฑ์สำคัญดังนี้
 - มอดูลในระดับต่ำจะไม่เกี่ยวข้องกับมอดูลในระดับสูงกว่า และไม่สามารถเรียกมอดูลในระดับสูงกว่าได้ แต่มอดูลในระดับสูงกว่าจะเรียกมอดูลในระดับต่ำกว่าได้
 - มอดูลในแต่ละระดับมีข้อมูลของตัวเอง ซึ่งมอดูลในระดับอื่นๆ ไม่สามารถเรียกใช้ได้

204112 Structured Programming



7

หลักการออกแบบการเขียนโปรแกรมแบบโครงสร้าง

Principles of structured programming design

ปรัชญาโครงสร้างประกอบด้วย

- ♥ หลักการนามธรรม
- ♥ หลักการความเป็นระเบียบ
- ♥ หลักการแบ่งแยก
- ♥ หลักการลำดับขั้น

ดร. ครรชิต มาลัยวงศ์ และ วิชิตาภูมิวัตร. เทคนิคการออกแบบโปรแกรม. ภาฯ ฯพ. : บริษัทที่อีดูเมาท์. จำกัด.
204112 Structured Programming



6

- **หลักการความเป็นระเบียบ**
การใช้วิธีการที่เข้มงวดอาจเริ่ง เอาจัง เพื่อให้มีพื้นฐานสามารถพิสูจน์ความถูกต้องของโปรแกรมได้
 - นำໄไปสู่ระเบียบวิธีการพัฒนาโปรแกรม (Program Development Process)
- **หลักการแบ่งแยก (Decomposition)**
 - แบ่งปัญหาที่ซับซ้อนออกเป็นส่วนๆ หรือปัญหาย่อยๆ ที่ยังคงความสัมพันธ์ระหว่างส่วนย่อยๆเหล่านั้น
- **หลักการลำดับขั้น (Hierarchy)**
 - สัมพันธ์กับหลักการแบ่งแยก คือแทนที่จะแบ่งเป็นส่วนย่อยๆ ก็แบ่งให้เป็นลำดับขั้น เช่น เป็นรูปต้นไม้
- **โปรแกรมโครงสร้าง จะมีการจัดเรียง หรือรูปแบบการทำงานอย่างชัดเจน**

204112 Structured Programming



8

การออกแบบเชิงโครงสร้าง

- การออกแบบจากบนลงล่าง
(Top – down design)



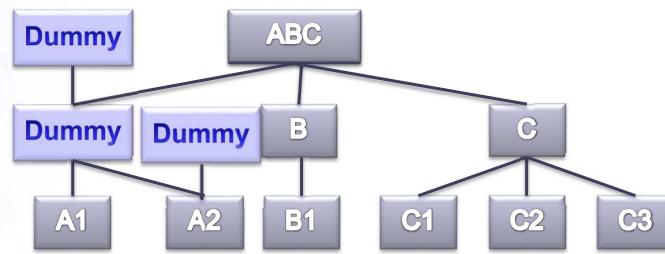
- การออกแบบจากล่างขึ้นบน
(Bottom – up design)



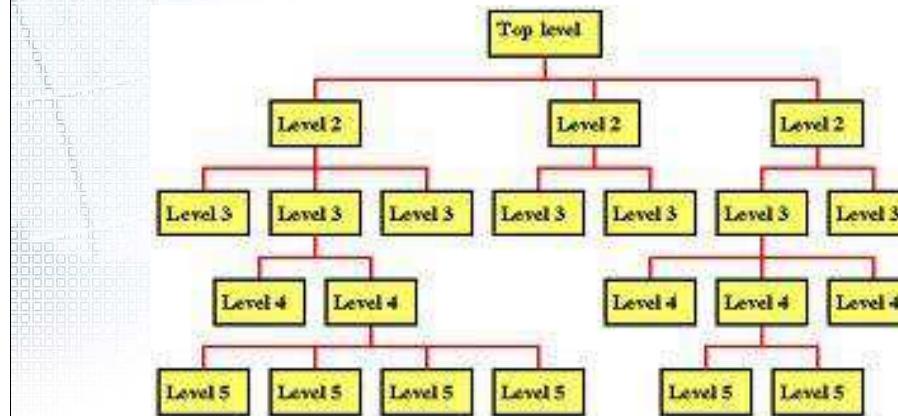
- การออกแบบแบบผสมผสาน
(Hybrid design)



Bottom-up Design

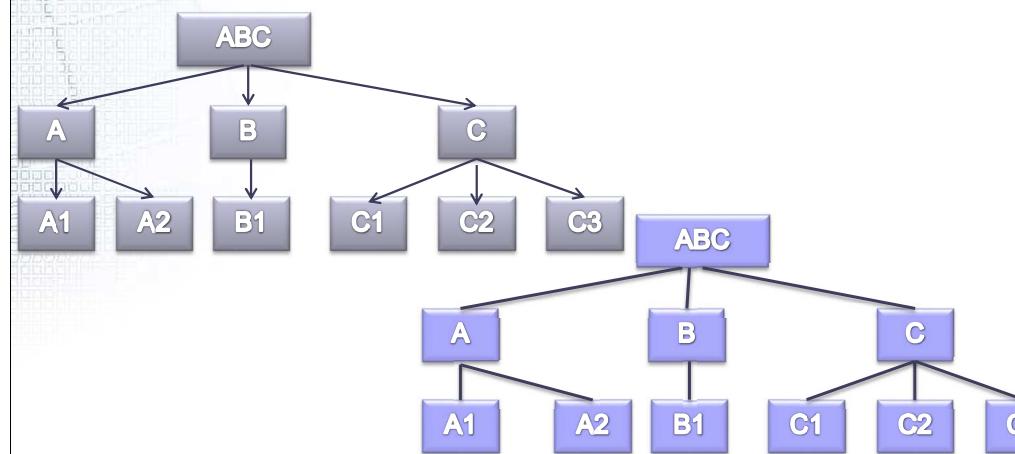


Top-down Design



การออกแบบเชิงโครงสร้าง

- การออกแบบจากบนลงล่าง
(Top – down design)



โครงสร้างของโปรแกรมทั่วไป



การออกแบบส่วนต่อประสานกับผู้ใช้ User interface design

- ข้อผิดพลาดของการออกแบบส่วนต่อประสาน
- กฎทองของการออกแบบ
- กิจกรรมในการออกแบบส่วนต่อประสาน
- วัฏจักรของการออกแบบ

ข้อผิดพลาดของการออกแบบส่วนต่อประสาน Typical Design Errors

- ขาดความสอดคล้อง
- ผู้ใช้ต้องจดจำมากเกินไป
- ไม่มีคำแนะนำหรือส่วนช่วยเหลือในขณะใช้งาน
- การต้องออกจากโปรแกรมมีน้อย เช่น ต้องการให้ผู้ใช้ป้อนข้อมูล แต่ไม่มีข้อความบอก หรือกรณีที่โปรแกรมอยู่ระหว่างการประมวลผล แต่ไม่มีข้อความใดๆแจ้ง กรณีนี้อาจทำให้ผู้ใช้เข้าใจผิดว่าโปรแกรมค้างหรือมีปัญหาได้
- ใช้งานยาก

- กำหนดบทบาทของผู้ใช้ให้ได้
- ลดภาระการจดจำของผู้ใช้
- ทำส่วนต่อประสานให้เป็นรูปแบบหรือแนวเดียวกัน(Make the interface consistent)

ลดภาระการจดจำของผู้ใช้

Reduce the user's memory load

- หลีกเลี่ยงที่จะต้องทำให้ผู้ใช้จำจำข้อมูลหรือรายละเอียดที่พึงผ่านตา
- กำหนดค่า default ที่สื่อความหมายได้ง่าย
- กำหนดแนวทางหรือมีคำแนะนำกำกับแต่ละขั้นตอน
- การจัดวางตำแหน่งของข้อมูล หรือลำดับการโต้ตอบควรเป็นหรือคล้ายกับแนวทางที่ผู้ใช้คุ้นเคย

Place the user in control

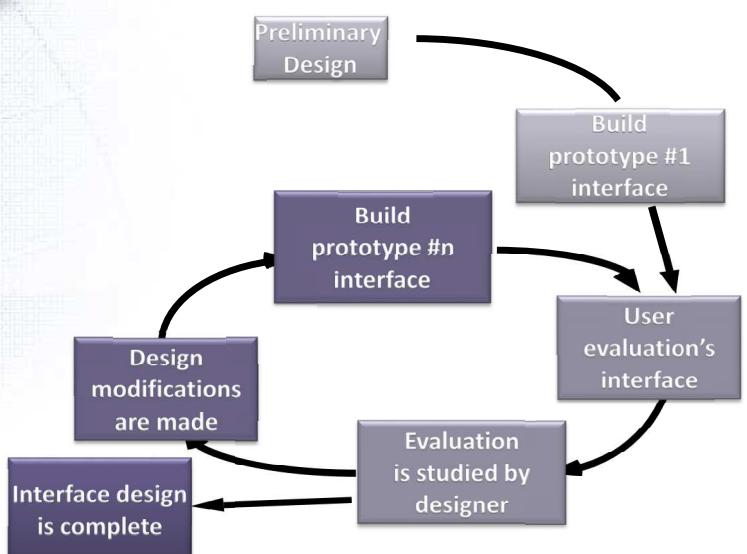
- กำหนดวิธีโต้ตอบกับผู้ใช้ที่ไม่เอ้อให้ผู้ใช้โต้ตอบด้วยสิ่งที่โปรแกรมไม่สามารถจัดการได้
- ให้ผู้ใช้สามารถโต้ตอบด้วยวิธีการที่มีความยืดหยุ่น เช่น หากต้องการให้ผู้ใช้กดปุ่ม Y เพื่อยืนยันการทำงาน โปรแกรมควรยอมให้ผู้ใช้ยืนยันด้วยการป้อน Y หรือ y ได้
- ยอมให้ผู้ใช้สามารถยกเลิกการทำงานกลางคันได้ แต่โปรแกรมต้องควบคุมได้หากเกิดกรณีนี้ขึ้น
- ควรซ่อนหรืออพาร์กรายละเอียดเชิงเทคนิคไว้ภายในโปรแกรม

กิจกรรมในการออกแบบส่วนต่อประสาน

Interface Design Activities

- กำหนดเป้าหมายหรือวัตถุประสงค์ของงาน
- ระบุหรือกำหนดขั้นตอนการดำเนินงานที่จะทำให้บรรลุแต่ละเป้าหมาย
- ระบุขั้นตอนของงานที่เกี่ยวข้องกับส่วนต่อประสาน
- กำหนดวิธีหรือแนวทางในการควบคุมให้ผู้ใช้ดำเนินการอย่างถูกต้อง เช่น จะกำหนดวิธีการอย่างไรให้ผู้ใช้ป้อนข้อมูลเฉพาะค่าที่โปรแกรมสามารถประมวลผลได้เท่านั้น ?

วัฏจักรของการออกแบบ Design Evaluation Cycle



204112 Structured Programming

21

บทส่งท้าย

- สิ่งที่ต้องส่งในคาบหน้า
- ภาพส่วนต่อประสานกับผู้ใช้ของโปรแกรมที่คุณประทับใจ พร้อมระบุเหตุผล
- การเตรียมตัวสำหรับภาคเรียนต่อไป
- ทบทวนขบวนการแก้ปัญหา

204112 Structured Programming



22