บทที่ 3

โปรแกรมหลักและโปรแกรมย่อย

3.1 แนวคิดของการออกแบบโปรแกรม

โดยทั่วไปแล้ว โปรแกรมหนึ่ง ๆ จะมีคำสั่งเป็นจำนวนมากซึ่งทำหน้าที่แตกต่างกันไป ซึ่งเราอาจจะแบ่งคำสั่งเหล่านั้นออกเป็นกลุ่ม ๆ ตามวัตถุประสงค์ปลีกย่อยของงานที่ทำ นั่นคือ โปรแกรมที่เราใช้งานอยู่นั้น แท้จริงแล้วก็คือการทำงานเชื่อมโยงกันของกลุ่มคำสั่งเหล่านี้ เราเรียกกลุ่มคำสั่งที่แบ่งตามวัตถุประสงค์ปลีกย่อยของงานที่ทำว่า **โปรแกรมย่อย** (subprogram) ส่วนกลุ่มคำสั่งที่ทำหน้าที่เชื่อมโยงแต่ละโปรแกรมย่อยเข้าด้วยกัน และเป็นส่วนที่ผู้ใช้สามารถเข้าถึงได้นั้น เราเรียกว่า **โปรแกรมหลัก** (main program)

ความแตกต่างระหว่างโปรแกรมหลักและโปรแกรมย่อย

โปรแกรมหลัก	โปรแกรมย่อย
- ออกแบบตามวัตถุประสงค์หลักของงานที่ต้องการทำ (พิจารณาการทำงานในภาพรวม) - ออกแบบให้เป็นส่วนที่ผู้ใช้สามารถเข้าถึงและเรียกใช้งานได้ - ทำหน้าที่เชื่อมโยงโปรแกรมย่อยหลาย ๆ ตัวเข้าด้วยกัน เพื่อให้สามารถทำงานได้ - รับ input จากผู้ใช้โดยตรง - ส่ง output ไปยังผู้ใช้โดยตรง	- ออกแบบตามวัตถุประสงค์ปลีกย่อยของงานที่ต้องการทำ - ไม่อนุญาตให้ผู้ใช้เข้าถึงและเรียกใช้งานได้ - มักจะออกแบบให้อยู่เป็นเอกเทศ ไม่เชื่อมโยงกับโปรแกรมย่อยอื่น ๆ ถ้าไม่จำเป็น - รับ input จากโปรแกรมหลักหรือโปรแกรมย่อยอื่น ๆ - ส่ง output ไปยังโปรแกรมหลักหรือโปรแกรมย่อยอื่น

ข้อดีของโปรแกรมย่อย

1. สนับสนุนแนวคิดพื้นฐานในการทำงานที่ว่า มองการทำงานในภาพรวมก่อน แล้วจึงค่อยแบ่งงานออกเป็นส่วนย่อย ๆ (top-down principle)
2. ช่วยให้การค้นหาข้อผิดพลาดในโปรแกรมรวดเร็วยิ่งขึ้น
3. ผู้เขียนโปรแกรมสามารถปรับปรุงแก้ไขโปรแกรมย่อยได้ง่าย โดยไม่กระทบกับการทำงานในส่วนอื่น ๆ
4. สามารถนำโปรแกรมย่อยไปใช้ต่อ (recycle) ในโปรแกรมหลักตัวอื่นได้

3.2 การเขียนฟังก์ชันใน Scilab

เนื่องจากโปรแกรมย่อยมักจะมีลักษณะเป็นเอกเทศ เราจึงมักจะเขียนโปรแกรมย่อยในรูปของฟังก์ชัน (function) ถึงแม้ว่าในโปรแกรม Scilab เองจะมีฟังก์ชันเป็นจำนวนมากที่ทางผู้พัฒนาโปรแกรมได้เขียนเอาไว้อยู่ก่อนแล้ว แต่ก็ยังไม่สามารถตอบสนองต่อความต้องการที่แท้จริงของผู้เขียนโปรแกรมได้ เนื่องจากลักษณะของงานแต่ละประเภทที่แตกต่างกัน โปรแกรม Scilab จึงเปิดช่องให้ผู้เขียนโปรแกรมสามารถสร้างฟังก์ชันด้วยตนเองได้ โดยใช้คำสั่งต่อไปนี้

```
function [ตัวแปรที่เป็น output] = <ชื่อฟังก์ชัน>(ตัวแปรที่เป็น input)
        ⋮
       ชุดคำสั่ง
        ⋮
endfunction
```

หมายเหตุ

1. ถ้ามีตัวแปรที่เป็น input/output หลายตัว ให้คั่นระหว่างตัวแปรด้วย comma ถ้าไม่มีตัวแปรเลย ให้ว่างเอาไว้
2. ในกรณีที่ต้องการ output ออกจากฟังก์ชันทันที สามารถใช้คำสั่ง `return` <ตัวแปรที่เป็น output ทั้งหมด> บังคับได้

ตัวอย่าง 3.2.1 จงสร้างฟังก์ชันชื่อ `myfunc` ซึ่งรับตัวแปร a และ b เป็น input และส่งออกตัวแปร x, y และ z เป็น output โดยที่

$$x = a^2 + b, \qquad y = ab - 1, \qquad z = 3a + b^2$$

แล้วบันทึกลงในไฟล์ชื่อ `myfunc.sce`

วิธีทำ

```

```

อย่างไรก็ตาม ถ้าเราลอง run ไฟล์ดังกล่าวแล้ว จะพบว่าไม่มีการแสดงผลใด ๆ เลย เนื่องจากฟังก์ชันจะทำงานได้ก็ต่อเมื่อมี input แล้วเท่านั้น ถ้าเราพิมพ์คำสั่งต่อไปนี้ลงใน Scilab Console

```
-->[x,y,z] = myfunc(1,4)
```

จะได้ผลลัพธ์

```
z  =
    19.
y  =
    3.
 x  =
    5.
```

โดยหลักปฏิบัติแล้ว ผู้เขียนโปรแกรมควรเพิ่มคำอธิบายไว้ในบริเวณส่วนหัวของแต่ละฟังก์ชัน ซึ่งอาจกล่าวถึงวัตถุประสงค์ของฟังก์ชัน ลักษณะของ input และ output ความหมายของตัวแปรแต่ละตัว เป็นต้น ไว้อย่างพอสังเขป เพื่อช่วยให้ผู้ใช้สามารถเรียกใช้งานฟังก์ชันนั้นได้อย่างถูกต้อง และเป็นการอำนวยความสะดวกแก่ผู้พัฒนาโปรแกรมต่อไปอีกด้วย

ตัวอย่าง 3.2.2 ให้แทรกคำอธิบายต่อไปนี้ใต้บรรทัดที่นิยามฟังก์ชัน `myfunc` แล้วบันทึกลงในไฟล์ `myfunc.sce` เช่นเดิม

```
// This function computes x, y, and z from given values
// of a and b.
// Input:
//   a, b : real numbers
// Output:
//   x, y, z : real numbers
```

จากนั้นให้ run ไฟล์ดังกล่าว เมื่อเราพิมพ์คำสั่ง `head_comments` <ชื่อของฟังก์ชัน> (ในที่นี้คือ `myfunc`) ใน Scilab Console จะได้ผลลัพธ์ดังนี้

```
-->head_comments myfunc
function [x,y,z] = myfunc(a,b)
 This function computes x, y, and z from given values
 of a and b.
 Input:
   a, b : real numbers
 Output:
   x, y, z : real numbers
```

นั่นคือ ผู้ใช้สามารถเรียกดูคำอธิบายเพิ่มเติมของฟังก์ชัน `myfunc` ได้

ตัวอย่าง 3.2.3 จงสร้างฟังก์ชันชื่อ `myfunc2` ซึ่งรับตัวแปร a, b และ c เป็น input และส่งออกตัวแปร s, t, u, v เป็น output โดยให้

$$s = c - a^2, \qquad t = 2abc, \qquad u = a^2 + b^2, \qquad v = \frac{a+b}{c^2+1}$$

แล้วบันทึกต่อท้ายฟังก์ชัน `myfunc` ในไฟล์ `myfunc.sce` ทั้งนี้ ให้เพิ่มคำอธิบายการใช้งานฟังก์ชันนี้ด้วย

<u>วิธีทำ</u>

<u>หมายเหตุ</u>

1. ใน Scilab เราสามารถนิยามฟังก์ชันได้หลาย ๆ ฟังก์ชันภายในไฟล์เดียวกัน แต่ในบางโปรแกรม เช่น Matlab แต่ละไฟล์จะมีฟังก์ชันได้เพียงฟังก์ชันเดียวเท่านั้น และต้องตั้งชื่อไฟล์และชื่อฟังก์ชันให้ตรงกันด้วย เป็นต้น

2. ตัวแปรที่กำหนดไว้ภายในฟังก์ชันใด จะเป็นที่รู้จักเฉพาะภายในฟังก์ชันนั้นเท่านั้น (local variables) เช่น ตัวแปร a ในฟังก์ชัน `myfunc` เป็นคนละตัวกับตัวแปร a ในฟังก์ชัน `myfunc2` เป็นต้น

ต่อไปเราจะแสดงตัวอย่างการสร้างโปรแกรมหลัก ซึ่งทำหน้าที่เชื่อมโยงโปรแกรมย่อยหลาย ๆ ตัวเข้าด้วยกัน **<u>พึงสังเกตว่า ก่อนจะเรียกใช้ตัวแปรหรือฟังก์ชันใดก็ตาม จะต้องมีการนิยามตัวแปรหรือฟังก์ชันนั้น ๆ ก่อนเสมอ</u>**

ตัวอย่าง 3.2.4 จงเพิ่มชุดคำสั่งต่อไปนี้ต่อท้ายฟังก์ชัน `myfunc2` ที่อยู่ในไฟล์ `myfunc.sce`

```
// Main program
a = input("Enter a: ")
b = input("Enter b: ")
[x,y,z] = myfunc(a,b);
printf("x = %f\n", x)
printf("y = %f\n", y)
printf("z = %f\n", z)
[s,t,u,v] = myfunc2(x,y,z);
printf("s = %f\n", s);
printf("t = %f\n", t);
printf("u = %f\n", u);
printf("v = %f\n", v);
```

แล้วทดลอง run โดยให้ตัวแปร a และ b มีค่าแตกต่างกันไปหลาย ๆ ค่า จะได้ผลลัพธ์ดังนี้

```

```

จากตัวอย่างข้างต้น จะเห็นได้ว่าชุดคำสั่งที่เพิ่มเข้าไปใหม่นี้จะทำหน้าที่เชื่อมโยงฟังก์ชัน `myfunc` และ `myfunc2` เข้าด้วยกัน และเป็นส่วนที่ผู้ใช้สามารถเข้าถึงและเรียกใช้งานได้ จึงถือเป็นส่วนของโปรแกรมหลักนั่นเอง

<u>ข้อควรปฏิบัติ</u> โดยทั่วไปแล้ว เราควรแยกส่วนที่เป็นโปรแกรมหลักและโปรแกรมย่อยให้อยู่คนละไฟล์กัน เนื่องจากส่วนที่เป็นโปรแกรมย่อยนั้นอาจนำไปใช้กับโปรแกรมหลักตัวอื่น ๆ ได้อีก