2

พื้นฐานโปรแกรมภาษาฟอร์แทรน

(Basics of Fortran programming)

ภาษาฟอร์แทรนเป็นหนึ่งภาษาระดับสูงที่มีการใช้และยอมรับกันอย่างแพร่หลาย ชื่อภาษาฟอร์แทรนมาจากคำว่าสูตรและการแปร (**FOR**mula และ **TRAN**slation) ซึ่งไปถูกพัฒนาใช้สำหรับคอมพิวเตอร์ IBM 704 โดย John Backus และทีมนักโปรแกรม (Programmer) อีก 13 คน ในช่วงปี คศ. 1954 – 1957

หัวใจของระบบการคำนวณคือ หน่วยประมวลผลกลาง (Central processing unit) หรือ ที่เรียกว่า CPU โดย CPU จะเป็นตัวควบคุมการทำงานของ ระบบ การทำงาน การคำนวณและตรรกะ การเก็บค่า และกู้ข้อมูล

2.1 โครงสร้างเลขฐานสอง (Binary scheme)

ตัวเลขที่ใช้ในเลขฐานสอง (Binary digits, bits) มีแค่สองตัวคือ 0 และ 1 เท่านั้น คำว่า บิต (bit) จะหมายถึงตำแหน่งของตัวเลขที่ใช้ในเลขฐานสอง เช่น 8 bits จะแทนตัวเลข 0 และ 1 ได้ 8 ตำแหน่งดังนี้

10000000_2 จะหมายถึงค่า $1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$ ซึ่งมีค่าเท่ากับ 128 ในเลขฐานสิบ และ 8 บิตจะมีค่าเท่ากับ 1 ไบต (Byte) นอกจากนี้เรายังกำหนดว่า 2^{10} = 1024 ไบต ซึ่งถูกเรียกว่า 1K ของความจำ (Memory) ดังนั้นความจำ 512K จะมีค่าเท่ากับ $512\times2^{10} = 2^9\times 2^{10} = 2^{19}$ = 524,288 ไบต หรือเทียบเท่า $2^{19} \times2^3 = 2^{22}$ = 4,194,304 บิต

ความพิวเตอร์ส่วนมากเก็บเลขจำนวนเต็มในระบบตัวเลขฐานสอง เลขจำนวนเต็มหนึ่งจำนวนที่ใช้ที่เก็บในหน่วยความจำเท่ากับ 1 คอมพิวเตอร์เวิร์ด (Word) ซึ่งประกอบด้วยกี่บิตนั้นขึ้นอยู่กับระบบคอมพิวเตอร์ ทั้งนี้อาจมีตั้งแต่ 16 – 60 บิต ตัวอย่างแสดงดังในตารางที่ 2.1

จำนวนบิตใน 1 เวิร์ด	ค่าสูงสุดของเลขจำนวนเต็ม	จำนวนหลักของเลขฐานสิบ
16	2^{15}-1	5
32	2^{31}-1	10
60	2^{60}-1	16

ตารางที่ 2.1 จำนวนบิตใน 1 เวิร์ด

โดยขนาดของเวิร์ดที่ถูกกำหนดจะจำกัดช่วงของค่าจำนวนเต็ม (Integer) ซึ่งสามารถถูกเก็บไว้ภายในตัวอย่างเช่น ค่าจำนวนเต็มบวกที่มากที่สุดที่สามารถถูกเก็บในเวิร์ด 16 บิต คือ

$$0111111111111111_2 = 2^{15} - 1 = 32767$$

และตัวเลขลบที่น้อยที่สุดคือ

$$1000000000000000_2 = -2^{15} = -32768$$

ค่าของจำนวนเต็มที่ออกนอกช่วงที่กำหนดจะต้องการบิตมากกว่าที่สามารถถูกเก็บในเวิร์ดตัวหนึ่ง (A single word) ปรากฏการณ์นี้เรียกว่า overflow ข้อจำกัดดังกล่าวนี้สามารถแก้ไขได้โดยการใช้เวิร์ดมากกว่าหนึ่งเพื่อที่จะเก็บไว้ในเลขจำนวนเต็มตัวเดียว

2.2 เลขจำนวนจริง (Real number)

ตัวเลขที่ประกอบด้วยจุดทศนิยม (Decimal point) เรียกว่า เลขจำนวนจริง หรือ Floating point number ซึ่งแต่ละค่าจะเป็นค่าตัวเลขสัมประสิทธิ์ของเลขสิบยกกำลัง

ตัวอย่างเช่น 56.317 เขียนได้เป็น

$$(5\times10^{1})+(6\times10^{0})+(3\times10^{-1})+(1\times10^{-2})+(7\times10^{-3})$$

ซึ่งเทียบเท่ากับ

$$(5\times10)+(6\times1)+\left(3\times\frac{1}{10}\right)+\left(1\times\frac{1}{100}\right)+\left(7\times\frac{1}{1000}\right)$$

ส่วนเลขจำนวนจริงในระบบตัวเลขฐานสองก็คือตัวเลขค่าสัมประสิทธิ์ของเลขสองยกกำลัง ตัวอย่างเช่น 110.101_2 คือ

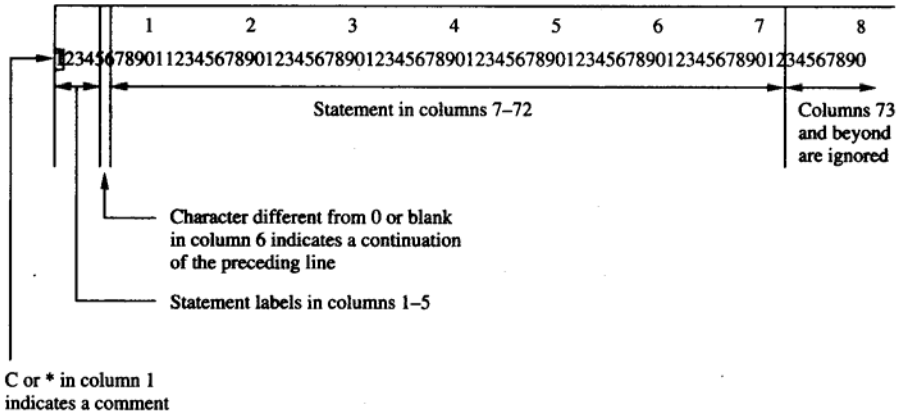
$$(1\times2^{2})+(1\times2^{1})+(0\times2^{0})+(1\times2^{-1})+(0\times2^{-2})+(1\times2^{-3})$$

ซึ่งในระบบเลขฐานสิบมีค่า

$$4+2+0+\frac{1}{2}+0+\frac{1}{8}=6.625$$

2.3 รูปแบบการเขียนภาษาฟอร์แทรน

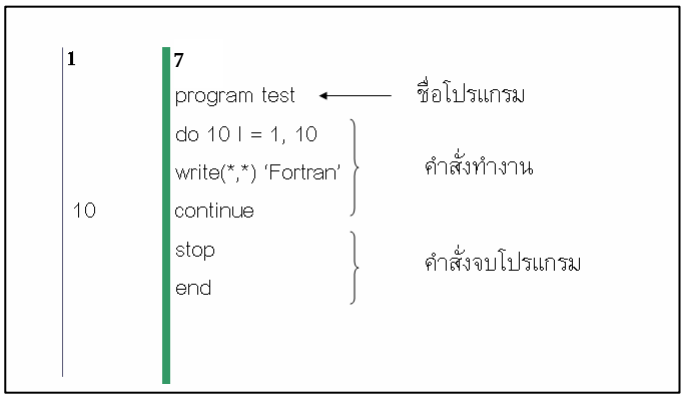
ตำแหน่งการเขียนโปรแกรมแสดงดังรูปที่ 2.1 ตัวคำสั่งต่างๆ ในสามารถฟอร์แทรนจะเริ่มต้นจากคอลัมน์ที่ 7 ความยาวในการเขียนตัวคำสั่งอยู่ระหว่างคอลัมน์ที่ 7 – 72 นอกช่วงนี้ (ตั้งแต่คอลัมน์ที่ 73) ถ้าหากเขียนคำสั่งทำงานโปรแกรมจะไม่ทำงานให้ และโปรแกรมจะแสดงความผิดพลาด (Syntax error) หลังจากทำการ compile แล้ว ในกรณีที่คำสั่งทำงานมีความยาวเกินคอลัมน์ที่ 72 ให้ขึ้นบรรทัดใหม่และพิมพ์อักษรหรือตัวเลขที่คอลัมน์ที่ 6 แล้ว ทำการเขียนคำสั่งส่วนที่เหลือต่อไป ส่วนในคอลัมน์ที่ 1 จะใส่ตัว C ในกรณีที่ต้องการเขียน comment และบรรทัดนี้คำสั่งจะไม่ถูกทำงาน นอกจากนี้



รูปที่ 2.1 ตำแหน่งการกำหนดคำสั่งในโปรแกรมฟอร์แทรน

2.4 โครงสร้างการเขียนโปรแกรม

ในการเขียนจะต้องกำหนดตัว main program ตัวอย่างเช่น เราต้องการใช้เขียนคำว่า Fortran เป็นจำนวน 10 ครั้งเรียงต่อกันลงมาในแนวดิ่ง สามารถเขียนคำสั่งได้เป็น



รูปที่ 2.2 ตัวอย่างการเขียนโปรแกรมฟอร์แทรน

ในรูปที่ 2 ประกอบด้วยส่วนหลัก 3 ส่วน คือ 1. ชื่อโปรแกรม 2. คำสั่งทำงาน และ 3. คำสั่งจบโปรแกรม

ชื่อโปรแกรม

ทุกครั้งของการเขียนโปรแกรมจะต้องกำหนดชื่อของตัวโปรแกรมและห้ามใช้คำสั่งเป็นชื่อโปรแกรม ในรูปที่ 2.2 ชื่อโปรแกรมกำหนดเป็น test

หมายเหตุ ชื่อโปรแกรมอาจจะเป็นชื่อเดียวกับชื่อตัวโปรแกรมหรือแตกต่างกันก็ได้ เช่น ชื่อตัวโปรแกรม คือ check.f แต่ชื่อโปรแกรมคือ program test เป็นต้น

คำสั่งทำงาน

ในรูปที่ 2.2 คำสั่งทำงานคือ การใช้ Do loop คือ พิมพ์คำว่า do แล้วตามด้วยหมายเลขตำแหน่งที่ต้องการวน loop ต่อจากนั้นตามด้วยจำนวนครั้งที่ต้องการให้วน loop บรรทัดต่อมาคือคำสั่งที่ต้องการให้

โปรแกรมทำงาน ซึ่งในที่นี้ เราใช้คำสั่ง write (*,*) เพื่อต้องการให้โปรแกรมแสดงผลบนหน้าจอคอมพิวเตอร์ โดยไม่บันทึกผลการทำงาน และ 'Fortran' หมายถึง การให้โปรแกรมพิมพ์คำว่า Fortran

หมายเหตุ (*,*) หมายถึง (รูปแบบของการแสดงค่า, การแสดงผล)

คำสั่งจบโปรแกรม

ในส่วนของ main program รูปทั่วไปคำสั่งจบโปรแกรมจะพิมพ์คำว่า stop แล้วขึ้นบรรทัดและพิมพ์คำว่า end ซึ่งหมายถึงสิ้นสุดการทำงานของโปรแกรม

หมายเหตุ ตัวอักษรที่พิมพ์อาจจะใช้เป็นตัวพิมพ์ใหญ่หรือพิมพ์เล็กก็ได้ ซึ่งจะให้ผลเหมือนกัน

คำสั่งทั่วไปที่มักใช้ในการเขียนโปรแกรมฟอร์แทรนมีดังนี้

End	หมายถึง	จบการทำงานของโปรแกรม
If(……)then	หมายถึง	ให้ทำงานถ้า (……..)
Else	หมายถึง	เป็นอย่างอื่น โดยใช้ควบคู่กับ If(……..)then
Endif	หมายถึง	สิ้นสุดเงื่อนไขโดยใช้ควบคู่กับ If(…….)then

รูปแบบคำสั่งตรรกะ (Logical type) ที่ใช้ควบคู่กับคำสั่ง If(….) then คือ

สัญลักษณ์	สัญลักษณ์ทางคณิตศาสตร์	ความหมาย
.LT.	$<$	น้อยกว่า
.GT.	$>$	มากกว่า
.EQ.	=	เท่ากับ
.LE.	$\leq$	น้อยกว่าหรือเท่ากับ
.GE.	$\geq$	มากกว่าหรือเท่ากับ
.NE.	$\neq$	ไม่เท่ากับ

ตารางที่ 2.2 รูปแบบคำสั่งตรรกะ

Format	หมายถึง	การกำหนดรูปแบบการแสดงผล
Open	หมายถึง	การเปิด file ที่ต้องการใช้งาน เช่น การเปิดไฟล์ข้อมูลเพื่อนำค่ามาใช้งาน หรือ การเปิดไฟล์ใหม่เพื่อบันทึกข้อมูลลงไป
Write	หมายถึง	การให้แสดงผล อาจจะแสดงผลบนหน้าจอคอมพิวเตอร์ หรือ บันทึกลงใน file ที่ต้องการเก็บ
Read	หมายถึง	ให้อ่านค่า
Stop	หมายถึง สิ้นสุดการทำงานของโปรแกรมโดยไม่มีการคำนวณใดๆ ต่อ	

2.5 การกำหนดตัวแปร

ค่าคงที่ (Constant)

เป็นปริมาณที่มีค่าไม่เปลี่ยนแปลงระหว่างการทำงานของโปรแกรม (Program execution) ค่าคงที่อาจจะเป็น ค่าจำนวนเต็ม (Integer number) ค่าจำนวนจริง (Real number) ค่าความละเอียดสองเท่า (Double precision) ค่าจำนวนเชิงซ้อน (Complex number), ตัวอักขระ (Character) หรือเป็นรูปตรรกะ (Logical type)

ค่าคงที่จำนวนเต็ม (Integer constant)

เป็นตัวเลขที่เรียงแถวกัน ซึ่งไม่ต้องใส่เครื่องหมายลูกน้ำ " , " (Comma) และต้องเป็นตัวเลขที่ไม่จุดทศนิยม อาจจะเป็นค่าบวกหรือลบก็ได้ ตัวอย่างเช่น

0

137

-2516

+17745

ส่วนรูปแบบที่เขียนไม่ถูกต้องสำหรับการกำหนดค่าคงที่จำนวนเต็มเช่น

5,280 (ห้ามใส่ลูกน้ำ)

16.0 (ห้ามใส่จุดทศนิยม)

- - 5 (ใส่เครื่องหมายพีชคณิตได้เพียงตัวเดียว)

7 - (เครื่องหมายพีชคณิตต้องอยู่หน้าตัวเลข)

ค่าคงที่จำนวนจริง (Real constant)

หรือที่รู้จักทั่วไปว่าค่าความละเอียดเดี่ยว (Single precision data) ซึ่งสามารถกำหนดอยู่ในรูปของค่าที่มีจุดทศนิยมหรือค่าที่อยู่ในรูปเอ๊กซ์โปเนนเชียล (Exponential notation) ได้ โดยใส่เครื่องหมายจุดทศนิยมที่ด้านหลังตัวเลขตัวสุดในกรณีที่ไม่มีทศนิยม และการกำหนดจะต้องไม่ใส่เครื่องหมายลูกน้ำในระหว่างตัวเลข เช่นเดียวกับการกำหนดค่าคงที่จำนวนเต็ม ตัวอย่างเช่น

1.234

- .01536

+ 56473.

ส่วนรูปแบบที่เขียนไม่ถูกต้องสำหรับการกำหนดค่าคงที่จำนวนจริงเช่น

12,345 (ห้ามใส่ลูกน้ำ)

63 (หลังตัวเลขตัวสุดท้ายต้องตามด้วยเครื่องหมายจุดทศนิยม)

การแสดงตัวเลขทางวิทยาศาสตร์ (Scientific representation)

ซึ่งเป็นค่าคงที่จำนวนจริงประกอบด้วยค่าจำนวนเต็มหรือค่าที่มีจุดทศนิยมแล้วตามด้วยอักษร E และเลขยกกำลังตามลำดับ ตัวเช่นค่า 337.456 สามารถเขียนได้เป็น 3.37456E2 ซึ่งมีความหมาย 3.37456×10^2 หรืออาจจะเขียนอยู่ในรูปแบบอื่นๆ ได้อีกเช่น

0.337456E3

337.456E0

33745.6E-2

33745.6E-3

หมายเหตุ ตัวเลขยกกำลังที่อยู่หลัง E อาจจะเป็นเลขจำนวนเต็มหรือเลขจำนวนจริงก็ได้

ค่าคงที่อักขระ (Character constant)

ซึ่งเรียกว่าค่า String เป็นลำดับของสัญลักษณ์ที่ถูกเลือกจากเซตอักขระของโปรแกรมฟอร์แทรน ค่าอักขระมาตรฐาน ANSI (American National Standards Institute standard character) กำหนดค่าสำหรับโปรแกรมฟอร์แทรนดังแสดงในตารางที่ 2.3

Character	Meaning
blank	blank or space
0, . . . , 9	digits
A, . . . , Z	uppercase letters
$	dollar sign
'	apostrophe (single quote)
(	left parenthesis
)	right parenthesis
*	asterisk
+	plus sign
−	minus sign
/	slash
,	comma
.	period
:	colon
=	equals sign

ตารางที่ 2.3 ค่าอักขระในโปรแกรมฟอร์แทรน

นอกจากนี้โปรแกรมฟอร์แทรนยังกำหนดรูปแบบอักขระในรูปของอักขระที่อยู่ในเครื่องหมายอัญประกาศ (Apostrophes) ตัวอย่างเช่น

'Run Ok'

ซึ่งประกอบด้วยอักขระทั้งหมด 6 ตัว

2.6 การประกาศค่าของตัวแปร (Variable)

การประกาศค่าของตัวแปรจะเขียนที่ส่วนต้นของตัวโปรแกรมเพื่อให้ทราบว่าตัวแปรที่ใช้คำนวณเป็นตัวแปรประเภทใด (จำนวนจริง จำนวนเต็ม หรือ อักขระ) เช่น

Real list1

ซึ่งหมายถึงตัวแปรที่ชื่อว่า list1 มีค่าเป็นเลขจำนวนจริง

และ

Integer list2

ซึ่งหมายถึงตัวแปรที่ชื่อว่า list2 มีค่าเป็นเลขจำนวนจริง แต่โดยทั่วไปแล้วถ้าหากกำหนดชื่อตัวแปรโดยขึ้นต้นด้วยอักษร a ถึง h หรือ o ถึง z จะหมายถึงการกำหนดตัวแปรตัวนั้นเป็นเลขจำนวนจริง ส่วนอักษร i ถึง n จะหมายถึงการกำหนดให้เป็นตัวแปรของเลขจำนวนเต็ม

Character*n, list3

จะหมายถึงการกำหนดตัวแปรให้เป็นอักขระและขนาดความยาวของตัวแปรมีค่าไม่เกิน n ซึ่งเป็นเลขจำนวนเต็มเท่านั้น ส่วนคำว่า list3 ที่แสดงด้านบน คือ ชื่อของตัวแปรไม่ใช่ค่าของตัวแปรที่เรากำหนด ดังนั้นถ้าหากเรากำหนดเป็น

Character*10 Fname, Lname*20, Int*1

จะหมายถึง ค่าของตัวแปรเป็นแบบอักขระ ชื่อตัวแปร Fname มีค่าจำนวนอักขระไม่เกิน 10 ตัว ชื่อตัวแปร Lname มีค่าจำนวนอักขระไม่เกิน 20 ตัว ส่วนชื่อตัวแปร Int มีค่าจำนวนอักขระไม่เกิน 1 ตัว เป็นต้น

2.7 การอ่าน (Read) การรับค่า (Input) และการแสดงค่า (Output)

การอ่าน (Read) เป็นคำสั่งที่ต้องการให้โปรแกรมอ่านค่าจากชุดข้อมูลจากไฟล์อื่นหรือจากข้อมูลที่เราป้อนให้แต่ละครั้ง เพื่อนำไปใช้ในการคำนวณหรือตัดสินใจ เช่น โปรแกรมตัดเกรด ผู้ใช้จะเป็นผู้ป้อนคะแนนที่ทำได้และเครื่องคอมพิวเตอร์จะเป็นตัวตัดสินว่าควรได้เกรดอะไร สมมติว่าเรากำหนดให้แต่ละเกรดอยู่ในช่วงของคะแนนดังต่อไปนี้

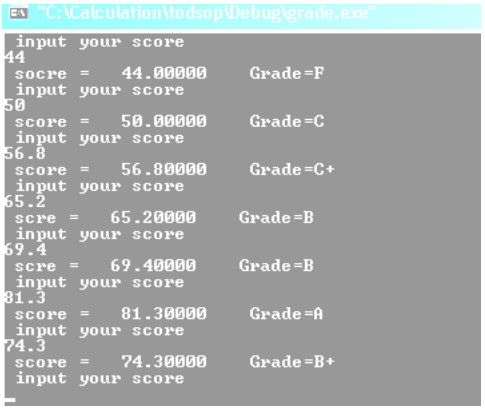
	A	≥ 80
$75 \leq$	**A -**	< 80
$70 \leq$	**B +**	< 75
$65 \leq$	**B**	< 70
$60 \leq$	**B -**	< 65
$55 \leq$	**C+**	< 60
$50 \leq$	**C**	< 55
$45 \leq$	**D**	< 50
$45 >$	**F**	

ตารางที่ 2.4 เกณฑ์ในการให้เกรด

สมมติว่าเราต้องการป้อนค่าคะแนนทีละคะแนนและดูผลของเกรดที่ได้ โปรแกรมสำหรับตัดเกรดสามารถถูกเขียนได้ ดังในรูปที่ 2.5 และผลที่เครื่องคอมพิวเตอร์พิจารณาแสดงในรูปที่ 2.6

```
      Program grade
      i = 0
10    i = i+1
      if(i.gt.10) goto 100
      write(*,*) 'input your score'
      read(*,*) score
      if(score.ge.80.0)then
            write(*,*) 'score =',score,'Grade=', 'A'
      else if((score.lt.80.0).and.(score.ge.75.0))then
            write(*,*) 'score =',score,'Grade=','A -'
      else if((score.lt.75.0).and.(score.ge.70.0))then
            write(*,*) 'score =',score, 'Grade=','B+'
      else if((score.lt.70.0).and.(score.ge.65))then
            write(*,*) 'scre =',score, 'Grade=', 'B'
      else if((score.lt.65.0).and.(score.ge.60))then
            write(*,*) 'score =',score, 'Grade=','B-'
      else if((score.lt.60.0).and.(score.ge.55))then
            write(*,*) 'score =',score, 'Grade=','C+'
      else if((score.lt.55.0).and.(score.ge.50))then
            write(*,*) 'score =',score, 'Grade=','C'
      else if((score.lt.50.0).and.(score.ge.45))then
            write(*,*) 'score =',score, 'Grade=','D'
      else
            write(*,*) 'socre =',score, 'Grade=','F'
      endif
      goto 10
100   stop
```

รูปที่ 2.3 โปรแกรมการตัดเกรด



รูปที่ 2.4 ผลการตัดเกรด

ในการเขียนโปรแกรมนอกจากการแสดงผลหรือแสดงค่าแล้วในบางครั้งอาจจะมีการนำชุดข้อมูลจากไฟล์อื่นเข้ามาใช้ในการคำนวณด้วย เช่น การคำนวณความร้อนที่อาคารได้รับในแต่ละเวลา เราต้องนำชุด

ข้อมูลอากาศ (Weather data) หรือชุดข้อมูลอุณหภูมิในแต่ละเวลา ซึ่งเป็นข้อมูลที่ตรวจวัดโดยกรมอุตุนิยมวิทยาทำขึ้นมาเป็นการรับค่า (Input) ของโปรแกรมที่เราใช้ในการคำนวณ

เพื่อจะนำค่าจากไฟล์อื่นมาใช้ในโปรแกรมมีลำดับขั้นดังนี้

1. การเปิดไฟล์ที่เป็นแหล่งของข้อมูลที่รับเข้า ใส่คำสั่ง open เช่น

   open(1,file='Weather data',status='old')

   ซึ่ง 1 หมายถึง การกำหนดให้ไฟล์ 'Weather data' เป็นไฟล์หมายเลข 1 ในตัวโปรแกรม ส่วน status ='old' หมายถึง การกำหนดให้ไฟล์ที่เราต้องการเปิดเพื่อรับค่าไม่เปลี่ยนแปลงข้อมูลใดๆ ได้ให้รับค่าเพียงอย่างเดียวแต่ถ้าหากต้องการให้เปลี่ยนแปลงข้อมูลให้ใช้กับค่า status='unknown' หมายเหตุ ถ้าหากไม่เขียนคำว่า status ='old' จะหมายถึงเหมือนกับ status='unknown'

1.00	20.00
2.00	20.50
3.00	21.00
4.00	21.50
5.00	22.00
6.00	22.50
7.00	23.00
8.00	23.50
9.00	24.00
10.00	24.50
11.00	25.00
12.00	25.50
13.00	26.00
14.00	26.50
15.00	27.00
16.00	27.50
17.00	27.00
18.00	26.50
19.00	26.00
20.00	25.50
21.00	25.00
22.00	24.50
23.00	24.00

รูปที่ 2.5 ตัวอย่างข้อมูลจากไฟล์ 'Weather data'

2. ใช้คำสั่ง Read เพื่อเก็บข้อมูลเข้ากับตัวแปรที่เรากำหนด เช่น

```
      Do 10 i = 1,23
      read(1,*) clock(i),temp(i)
10    continue
      close(1)
```

รูปที่ 2.6 โปรแกรมการรับและเก็บข้อมูล

ซึ่งให้อ่านค่าของไฟล์หมายเลข 1 โดยเครื่องหมายดอกจัน (*) ในวงเล็บหมายถึงรูปแบบของตัวเลขที่เราต้องการให้แสดงผล clock(i) หมายถึง เวลาที่ i มีค่าตั้ง 1 ถึง 23 และ temp(i) หมายถึง ค่าอุณหภูมิตั้งแต่ i = 1 ถึง 23

หมายเหตุ จะเห็นว่า ค่า clock(i) และ temp(i) เป็น Array ดังนั้นจะต้องกำหนดจำนวนของ Array ให้สอดคล้องกับจำนวนของข้อมูลที่รับเข้ามาด้วย ในตัวอย่างข้อมูลมีทั้งหมด 24 ตัว สำหรับ clock(i) และ temp(i)

3. ส่วนคำสั่ง close(i) จะใส่หรือไม่ใส่ก็ได้ เพราะเป็นตัวบอกว่าเราจะปิดไฟล์หมายเลข 1 ไม่มีการรับหรือส่งข้อมูลอีกต่อไประหว่างไฟล์

เพื่อจะนำค่าจากโปรแกรมคำนวณได้มาเก็บไว้ในไฟล์อื่นลำดับขั้นดังนี้

1. การเปิดไฟล์ที่เป็นแหล่งของข้อมูลที่ต้องการจัดเก็บให้คำสั่ง open เช่น
   open(1000,file='Heat Gen Result',status='unknown')
   ซึ่ง 1000 หมายถึง การกำหนดให้ไฟล์ 'Heat Gen Result' เป็นไฟล์หมายเลข 1000

2. ใช้คำสั่ง write หรือ print เพื่อเก็บข้อมูลเข้ากับตัวแปรที่เรากำหนด เช่น

```
      Do 10 i = 1,23
      write(1000,1001) clock(i), heatgen(i)
1000  format(e8.2,4x,e8.3)
  10  continue
      close(1000)
```

รูปที่ 2.7 โปรแกรมการบันทึกข้อมูลในไฟล์อื่น

ซึ่ง ตัวเลข 1001 ถูกใช้เพื่อกำหนดรูปแบบของค่าที่คำนวณได้ในที่นี้คือค่าของ clock(i) และ heatgen(i) ณ เวลาต่างๆ ตั้ง 1 ถึง 23 นั่นเอง โดยจะจัดเก็บค่าได้ทั้งหมด 8 ตำแหน่ง (รวมจุดทศนิยม) และมีเลขทศนิยม 2 ตำแหน่ง สำหรับ clock(i) และ มีเลขทศนิยม 3 ตำแหน่งสำหรับค่า heatgen(i) ตามลำดับ ส่วน 4x หมายถึง ให้เว้นช่องว่างจาก clock(i) ไป 4 ตำแหน่ง ตัวอย่างผลการคำนวณที่ถูกจัดเก็บแสดงดังรูปที่ 2.8

0.10E+01	.264E+02
0.20E+01	.695E+03
0.30E+01	.183E+05
0.40E+01	.482E+06
0.50E+01	.127E+08
0.60E+01	.335E+09
0.70E+01	.883E+10
0.80E+01	.233E+12
0.90E+01	.613E+13
0.10E+02	.162E+15
0.11E+02	.426E+16
0.12E+02	.112E+18
0.13E+02	.296E+19
0.14E+02	.780E+20
0.15E+02	.206E+22
0.16E+02	.542E+23
0.17E+02	.143E+25
0.18E+02	.376E+26
0.19E+02	.992E+27
0.20E+02	.261E+29
0.21E+02	.689E+30
0.22E+02	.182E+32
0.23E+02	.478E+33

รูปที่ 2.8 ตัวอย่างค่าที่ถูกจัดเก็บในไฟล์'Heat Gen Result'

หมายเหตุในกรณีที่ต้องการให้แสดงค่าบนหน้าจอคอมพิวเตอร์ให้ใช้คำสั่ง

write(1000,*) clock(i), heatgen(i)

3. ในกรณีที่ไม่ต้องการบันทึกค่าใดๆในไฟล์ 'Heat Gen Result' แล้วให้ใช้คำสั่ง close(1000) ซึ่งหมายถึงปิดไฟล์หมายเลข 1000

   หมายเหตุ ต้องระวังเรื่องการกำหนด Array ของตัวแปร heatgen(i) ด้วยซึ่งในที่นี้บันทึกค่าได้ทั้งหมดเพียง 24 ค่าเท่านั้น ถ้าเราบันทึกมากกว่า 24 แล้ว เวลา Compile โปรแกรมจะแสดง error ออกมา

2.8 การกำหนด Array ของตัวแปร

ดังที่ได้กล่าวมาแล้วในกรณีที่ตัวแปรที่เราต้องการรับข้อมูลหรือจัดเก็บมีหลายค่าเราจำเป็นต้องกำหนดขนาดของ Array ของตัวแปรเหล่านั้นที่ตอนต้นของตัวโปรแกรม เช่น

```
program heatload
real clock(24),temp(24),heatgen(24)
open(1,file='Weather data',status='old')
open(1000,file='Heat Gen Result',status='unknown')
```

รูปที่ 2.9 ตัวอย่างการกำหนด Array ให้กับตัวแปร

โดยทั่วไปในการคำนวณปัญหาทางพลศาสตร์ของไหลใน 2 มิติ (2-dimensional problem) เราอาจจะต้องการบันทึกค่าของความเร็วซึ่งประกอบด้วยความเร็วในแนวแกนการไหล (Axial flow) หรือในแนวแกน x และความเร็วในแนวตั้งฉากการไหล (Normal flow) หรือในแนวแกน y เราจำเป็นต้องกำหนดตัวแปรในทั้งสองแนวเป็น $u(i, j)$ และ $v(i, j)$ ตามลำดับ (ซึ่ง i และ j) เป็นตำแหน่งพิกัดของ Grid ที่เราใช้คำนวณ ในแนวแกน x และ y ตามลำดับ) ดังนั้นการกำหนด Array จะต้องระบุเท่ากับจำนวนของ Grid ที่เราใช้คำนวณ เช่น จำนวน Grid ในแนวแกน x = 24 และ y = 24 ดังนั้น การกำหนด Array เขียนได้เป็น

```
real u(24,24),v(24,24)
```

<u>ตัวอย่าง</u>

จงหาโปรไฟล์ความเร็ว (Velocity profile) ในแนวการไหลของของไหลที่ไหลในท่อกลมแบบราบเรียบ (Laminar pipe flow) โดยสมการของการไหล คือ

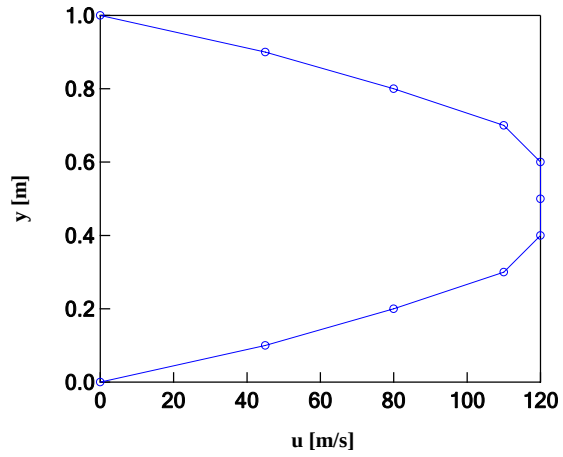
$$u(y) = \frac{a^2}{2\mu}\left(\frac{\partial p}{\partial x}\right)\left[\left(\frac{y}{a}\right)^2 - \left(\frac{y}{a}\right)\right]$$

กำหนดให้ $\partial p / \partial x$ เป็นค่าคงที่เท่ากับ -1 N/m^3 a เป็นรัศมีของท่อมีค่าเท่ากับ 1 เมตร และ μ เป็นค่าความหนืดของของไหล มีค่าเท่ากับ 10^{-3} kg/(m.s) ตามลำดับ

โปรแกรมสำหรับคำนวณปัญหาตัวอย่างแสดงดังในรูปที่ 2.10 โดยการคำนวณได้แบ่งตำแหน่งพิกัด y สำหรับการคำนวณออกเป็น 10 ส่วน คือ ตั้งแต่ตำแหน่งที่ 0 ถึง 10 ดังนั้น Array สำหรับ ค่า y และ u กำหนดให้เป็น $y(0:10)$ และ $u(0:10)$ และผลการคำนวณแสดงดังในรูปที่ 2.11

```
      program velocity
      real u(0:10),y(0:10)
      open(1000,file='U-velocity',status='unknown')
      ny = 10
      a = 1.0
      dy = a/real(ny)
      vis = 10.0**(-3)
      dpdx = -1.0
      do 10 i = 0,10
      y(i) = real(i)*dy
      u(i) = a*a/(2.0*vis)*dpdx*((y(i)/a)**2-(y(i)/a))
      write(1000,1001) y(i),u(i)
1001  format(e8.2,6x,e8.2)
10    continue
      close(1000)
      stop
      end
```

รูปที่ 2.10 ตัวอย่างโปรแกรม



รูปที่ 2.11 โปรไฟล์ความเร็วของของไหลในแนวแกนการไหล

2.9 การเขียนโปรแกรมย่อย (Subroutine)

โดยทั่วไปถ้าหากโปรแกรมมีความยาวมากๆ จะทำการแยกส่วนที่เป็นคำสั่งการคำนวณไว้ต่างหากที่โปรแกรมย่อย (Subroutine) เพื่อสะดวกในการตรวจสอบแก้ไขโปรแกรม ในโปรแกรมหนึ่งๆ อาจจะประกอบด้วยโปรแกรมย่อยหลายๆ โปรแกรมก็ได้

การเขียนโปรแกรมย่อย ประกอบด้วยส่วนที่คล้ายๆ กับการเขียนโปรแกรมทั่วๆ ไป แตกต่างตรงที่ชื่อโปรแกรมย่อย และการคำสั่งจบโปรแกรม เช่น

การกำหนดชื่อโปรแกรมย่อย เขียนในรูป

Subroutine *name (formal-argument-list)*

โดย *name* หมายถึง ชื่อของโปรแกรมย่อย ซึ่งจะต้องเป็นชื่อที่ไม่ซ้ำกับโปรแกรมหลัก (Main program) หรือ โปรแกรมย่อยอื่นๆ และต้องไม่เป็นชื่อของคำสั่งทำงาน ส่วน *formal-argument-list* เป็นตัวแปรที่เราต้องการนำไปคำนวณ ซึ่งตัวแปรที่ส่งไปคำนวณระหว่างโปรแกรมหลักและโปรแกรมย่อยจะต้องมี Array ที่เท่ากัน การจบโปรแกรมจะใช้คำสั่ง

End

แต่ในกรณีที่มีการส่งค่าที่คำนวณได้คืนกับไปที่โปรแกรมหลักจะใช้คำสั่ง

Return

End

การเรียกใช้โปรแกรมย่อย จะใช้คำสั่ง

Call *name (actual-argument-list)*

ตัวอย่างการใช้โปรแกรมย่อย

จงเขียนโปรแกรมเพื่อคำนวณหาค่า x จากสมการ

$$x = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

โปรแกรม

```
      Program root
      real a(10),b(10),c(10),x(10)
      open(1000,file='Root data',status='unknown')
      a(1) = 1.0
      b(1) = 9.0
      c(1) = 2.0
      call Cal(a(1),b(1),c(1))
      do 10 i = 2,5
      a(i) = a(i-1)+1.0
      b(i) = b(i-1)+1.0
      c(i) = c(i-1)+1.0
      call Cal(a(i),b(i),c(i))
10    continue
      stop
      end

      subroutine Cal(a1,b1,c1)
      real a1,b1,c1,x1,x2
      x1 = (b1+sqrt((b1*b1)-4.0*a1*c1))/(2.0*a1)
      x2 = (b1-sqrt((b1*b1)-4.0*a1*c1))/(2.0*a1)
      write(1000,*) a1,b1,c1,x1,x2
      end
```

รูปที่ 2.12 โปรแกรมการหาค่ารากที่สอง

a	b	c	x1	x2
1.000000	9.000000	2.000000	8.772002	0.2279981
2.000000	10.00000	3.000000	4.679450	0.3205505
3.000000	11.00000	4.000000	3.257334	0.4093327
4.000000	12.00000	5.000000	2.500000	0.5000000
5.000000	13.00000	6.000000	2.000000	0.6000000

รูปที่ 2.13 ผลแสดงของโปรแกรมการหารากที่สอง

จากรูปที่ 2.12 ถึงแม้ตัวแปรที่ถูกส่งไปคำนวณในโปรแกรมย่อยจะมี Array เท่ากับ 10 แต่ในความเป็นจริงแล้วค่าที่ถูกส่งไปคำนวณมีทีละค่าเท่านั้น หรือ Array เท่ากับ 1 นั่นเอง ตัวอย่างคำสั่งการคำนวณทางคณิตศาสตร์ที่นิยมใช้ในการเขียนโปรแกรมฟอร์แทรนแสดงดังตารางที่ 10

Function	Description	Type of Argument(s)*	Type of Value
ABS(x)	Absolute value of x	Integer or real	Same as arguments
COS(x)	Cosine of x radians	Real	Real
EXP(x)	Exponential function	Real	Real
INT(x)	Integer part of x	Real	Integer
LOG(x)	Natural logarithm of x	Real	Real
MAX($x_1, \ldots, x_n$)	Maximum of $x_1, \ldots, x_n$	Integer or real	Same as arguments
MIN($x_1, \ldots, x_n$)	Minimum of $x_1, \ldots, x_n$	Integer or real	Same as arguments
MOD(x, y)	x (mod y); $x - \text{INT}(x/y) * y$	Integer or real	Same as arguments
NINT(x)	x rounded to nearest integer	Real	Integer
REAL(x)	Conversion of x to real type	Integer	Real
SIN(x)	Sine of x radians	Real	Real
SQRT(x)	Square root of x	Real	Real

ตารางที่ 2.5 คำสั่งการคำนวณทางคณิตศาสตร์

จากตารางที่ 2.5 แต่ละคำสั่งใช้งานดังต่อไปนี้

ABS(x) หมายถึง การทำให้ค่า x มีค่าเป็นบวก

COS(x) หมายถึง การหาค่าโคซายน์ของค่า x ซึ่งค่า x มีหน่วยเป็นเรเดียลเสมอ นอกจากนี้การหากเราต้องการใช้ค่า π แต่ไม่มีสัญลักษณ์ π ในการเขียนโปรแกรม เราสามารถหาค่าดังกล่าวได้โดยกำหนดให้ PAI=2.0*ASIN(1.0)

EXP(x) หมายถึง e^x

INT(x) หมายถึง การทำให้ค่า x เป็นเลขจำนวนเต็มโดยจะตัดเศษออก

MAX(x1,…,xn) หมายถึงการหาค่าสูงสุดของค่า x

MOD(x,y) หมายถึง การเลือกใช้ค่าเศษ เช่น 11 หารด้วย 10 ค่าเศษมีค่าเท่ากับ 1 และ MOD(223,20) = 3

Real(x) หมายถึง การทำให้ค่า x เป็นเลขจำนวนจริง

2.10 ฟังก์ชั่น (Function)

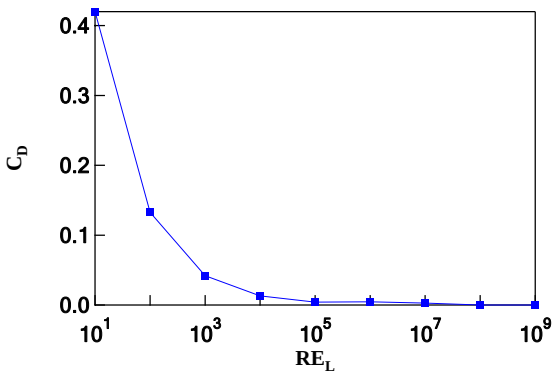
การใช้คำสั่งฟังก์ชั่นมีจุดประสงค์คล้ายกับการใช้โปรแกรมย่อย แต่ไม่นิยมใช้ในการเขียนโปรแกรม เพราะอาจจะทำให้เกิดความสับสนระหว่างตัวแปรที่เป็น Array และคำสั่งฟังก์ชั่น ตัวอย่างเช่น เราต้องการคำนวณสัมประสิทธิ์แรงต้านของอากาศ (Drag coefficient) ที่ไหลผ่านแผ่นราบ (Flat plate) ด้วยค่าตัวเลขเรย์โนลด์ที่แตกต่างกัน ถ้าสมมติให้

$$C_D = \begin{cases} \dfrac{1.328}{\sqrt{\mathrm{Re}_L}} & when \quad \mathrm{Re}_L < 5\times10^5 \\ \dfrac{0.074}{\mathrm{Re}_L^{1/5}} & when \quad 5\times10^5 \le \mathrm{Re}_L < 10^7 \\ \dfrac{0.455}{\left(\log \mathrm{Re}_L\right)^{2.58}} & when \quad 10^7 \le \mathrm{Re}_L \le 10^9 \end{cases}$$

```
      program drag
      REAL CD(10),RE(10)
      OPEN(1000,FILE='DRAG')
      RE(1) = 10.0
      CD(1) = F(RE(1))
      WRITE(1000,*) RE(1),CD(1)
      DO 10 I = 2,9
      RE(I) = RE(I-1)*10.0
      CD(I) = F(RE(I))
      WRITE(1000,*) RE(I),CD(I)
10    CONTINUE
      STOP
      END

      FUNCTION F(W)
      IF(W.GE.10.E+7.AND.W.LE.10.E+9) F = 0.455/(LOG(W)**2.58)
      IF(W.GE.10.E+5.AND.W.LT.10.E+7) F = 0.074/(W**0.2)
      IF(W.LT.10.E+5) F = 1.328/SQRT(W)
      RETURN
      END
```

รูปที่ 2.14 การเขียนโปรแกรมด้วยคำสั่งฟังก์ชั่น



รูปที่ 2.15 ผลการคำนวณค่าสัมประสิทธิ์การต้านที่แปรผันตามตัวเลขเรย์โนลด์

2.11 การกำหนดตัวแปรร่วม (Common)

การกำหนดตัวแปรร่วมถูกใช้อยู่บ่อยๆ เพื่อกำหนดค่าตัวแปรระหว่างโปรแกรมหลัก (Main program) กับ โปรแกรมย่อย (Subroutine) และ ฟังก์ชั่น (Function) เป็นค่าเดียวกัน ข้อควรระวังในการใช้คำสั่งคอมมอน คือ ค่าตัวแปรที่เราส่งไปที่โปรแกรมย่อยหรือฟังก์ชั่นอาจจะถูกเปลี่ยนแปลงได้ถ้าเรากำหนดค่าตัวแปรนั้นๆ ใหม่ในโปรแกรมย่อยหรือฟังก์ชั่น ตัวอย่างเช่น ในรูปที่ 2.16 ค่า dens, vis, และ dia ถูกประกาศ (Declare) ในโปรแกรมหลัก โดยใช้คำสั่งคอมมอนทั้งในโปรแกรมหลักและฟังก์ชั่น ค่าเหล่านี้จะถูกส่งไปยังฟังกชั่นด้วย

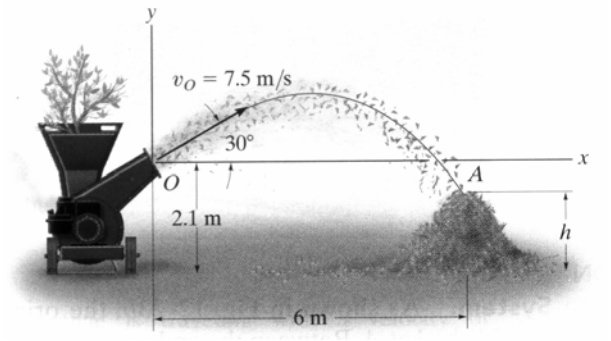
```
      program common1
      common dens,vis,dia
      real RE(10)
      dens = 10.e+3
      vis = 10.0e-3
      dia = 1.0
      do 10 i = 1,5
      U  = 2.0*10.0**(i)
      RE(i) = F1(U)
      write(*,*) U,RE(i)
10    continue
      stop
      end

      function F1(U)
      common dens,vis,dia
      F1 = U*dens*dia/vis
      return
      end
```

รูปที่ 2.16 ตัวอย่างการใช้คำสั่งคอมมอน

คำถามท้ายบท

1. จากรูปปัญหาที่ 1 จงเขียนโปรแกรมเพื่อคำนวณลักษณะการเคลื่อนที่ (u, v, x, y) ของหญ้าที่ถูกตัดจากเครื่องตัดหญ้า โดยกำหนดให้ค่าการเปลี่ยนแปลงของเวลามีค่า Δt = 0.05, 0.1 และ 0.15 วินาที และเปรียบเทียบผลกับคำตอบแม่นตรง



รูปปัญหาที่ 1

เอกสารอ้างอิง

1. Larry Nyhoff, and Sanford Leestma (1992). FORTRAN 77 For Engineers and Scientists, 3rd ed., Maxwell Macmillan International.
2. Robert W. Fox, and Alan T. McDonald (1994). Introduction to Fluid Mechanics, 4th ed., Wiley.