บทที่ 4
โครงสร้างการเลือก : ข้อความสั่ง if และ case
(Selection Structures : if and case Statements)

4.1 โครงสร้างควบคุม
4.2 นิพจน์แบบบูล
4.3 ข้อความสั่ง if
4.4 แผนภาพวากยสัมพันธ์
4.5 ข้อความสั่ง if ที่มีข้อความสั่งประกอบ
4.6 ขั้นตอนตัดสินใจในอัลกอริทึม
กรณีศึกษา : ปัญหาบัญชีเงินเดือน
4.7 Hand - Tracing an Algorithm
4.8 ข้อความสั่ง Nested if และการตัดสินใจหลายแบบทางเลือก
4.9 ข้อความสั่ง case
4.10 ข้อผิดพลาดร่วมของการเขียนโปรแกรม

ในบทนี้เริ่มต้นเราศึกษาข้อความสั่งซึ่งควบคุมสายงานของการกระทำการโปรแกรม (Flow of program execution) เราจะเรียนรู้การใช้ข้อความสั่ง if และ ข้อความสั่ง case เพื่อให้เลือกหนึ่งข้อความสั่งมากระทำการจากหลายๆ ทางเลือกขั้นแรก อภิปรายนิพนธ์แบบบูล เพราะว่าข้อความสั่ง if ขึ้นอยู่กับค่าของมัน

กรณีศึกษาในบทนี้ขยายความสามารถในการแก้ปัญหาของเราโดยแนะนำเทคนิคของผลเฉลย โดยการอุปมาและกระทำการ hand - tracing ของอัลกอริทึม ในบทนี้ยังแนะนำเทคนิคเชิงภาพเพื่อแสดงสายงานของข้อมูล (data flow) ในผังโครงสร้างและการแทนตัวสร้าง Pascal (สมาชิกของภาษา) ในแผนภาพวากยสัมพันธ์

4.1 โครงสร้างควบคุม (Control Structures)

การเขียนโปรแกรมเชิงโครงสร้างใช้โครงสร้างควบคุม เพื่อควบคุมสาย (flow) ของการกระทำการในโปรแกรมหรือกระบวนงาน โครงสร้างควบคุมของ Pascal ทำให้เราสามารถรวม (combine) แต่ละคำสั่งให้เป็นหน่วยตรรกะหนึ่งหน่วยที่มีหนึ่งทางเข้า และหนึ่งทางออก จากนั้นเราสามารถเขียนโปรแกรมเป็นลำดับของโครงสร้างควบคุมซึ่งไม่ใช่ลำดับของแต่ละคำสั่ง (รูป 4.1)

โครงสร้างควบคุม หมายถึง การรวมของแต่ละคำสั่งให้เป็นหนึ่งหน่วยตรรกะที่มีหนึ่งทางเข้าและหนึ่งทางออก (The control structures are combinations of individual instructions into a single logical unit with one entry point and one exit point.)

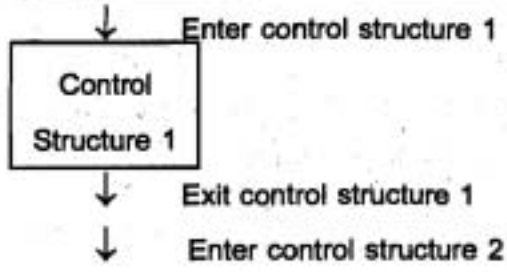
คำสั่ง ถูกจัดระเบียบเป็นสามชนิดของโครงสร้างควบคุม เพื่อควบคุมสายการกระทำการ ได้แก่ แบบลำดับ การเลือก และการทำซ้ำ จนกระทั่งขณะนี้เรามีเฉพาะสายแบบลำดับเท่านั้น

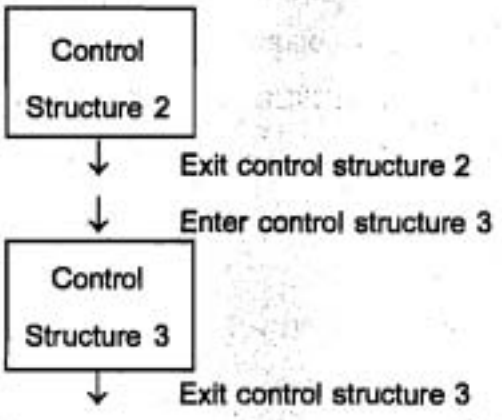
ข้อความสั่งประกอบ เขียนเป็นกลุ่มของข้อความสั่งอยู่ใน begin และ end ซึ่งถูกระบุเป็นสายแบบลำดับ :

```
begin
   statement1 ;
   statement2 ;
      :
   statementn
end
```

สายงานควบคุมจาก $statement_1$ ไป $statement_2$ เช่นนี้เรื่อยไป

เราใช้ข้อความสั่งประกอบไปตลอดทั้งหมด ตัวโปรแกรมหรือกระบวนงานประกอบด้วย ข้อความสั่งประกอบหนึ่งชุด





รูป 4.1 โปรแกรมคือลำดับของโครงสร้างควบคุมสามชนิด

ข้อความสั่งประกอบ หมายถึง กลุ่มของข้อความสั่งอยู่ภายใน begin และ end ซึ่งถูกกระทำการแบบลำดับ (A compound statement is a gronp of statements brackited by begin and end that are executed sequentially.)

ในบทนี้อธิบายโครงสร้างควบคุมของ Pascal สำหรับการเลือกผลเฉลยของปัญหาบางอย่างต้องเลือกซึ่งกระทำระหว่างขั้นตอนทางเลือกซึ่งข้อความสั่ง Pasual อาจจะถูกกระทำการ หรือไม่ถูกกระทำการขึ้นอยู่กับข้อมูลอินพุต โครงสร้างควบคุมการเลือกให้เลือกว่าทางเลือกใดจะให้กระทำการ

โครงสร้างควบคุมการเลือก หมายถึง โครงสร้างควบคุมซึ่งให้เลือกระหว่างข้อความสั่งโปรแกรมทางเลือกต่างๆ (A selection control structure is a control structure that chooses among alternative prognam statement.)

4.2 นิพจน์แบบบูล (Boolean Expressions)

โปรแกรมเลือกระหว่างขั้นตอนทางเลือกต่างๆ โดยการทดสอบค่าของตัวแปรหลัก (key variables) ตัวอย่างเช่น เนื่องจากอัตราภาษีที่แตกต่างกันใช้กับระดับเงินเดือนหลากหลายโปรแกรมภาษีเงินได้ต้องเลือกอัตราที่ถูกต้องสำหรับเงินเดือนของพนักงานแต่ละคน โดยการเปรียบเทียบค่าเงินเดือนกับเงินเดือนสูงสุดสำหรับกลุ่มภาษีเงินได้เฉพาะใน Pascal นิพจน์แบบบูล หรือเงื่อนไข (conditions) กระทำการเปรียบเทียบเช่นนั้น นิพจน์แบบบูลแต่ละตัวมีค่าที่เป็นไปได้สองค่า คือ True หรือ False เมื่อ True แสดงว่า การทดสอบประสบผลสำเร็จและ False แสดงว่าการทดสอบไม่ประสบผลสำเร็จ

ตัวแปรแบบบูลและค่าคงตัว (Boolean Variables and Constants)

นิพจน์แบบบูลที่ง่ายที่สุด คือ ตัวแปรแบบบูล หรือค่าคงตัว

ข้อความสั่ง

```
const
    LeapYear = True ;
```

ระบุ ค่าคงตัวแบบบูล LeapYear มีค่าเป็น True

ข้อความสั่ง

```
var
    switch, Flag : Boolean ;
```

ประกาศ Switch และ Flag เป็นตัวแปรแบบบูล - ตัวแปรซึ่งอาจกำหนดได้เฉพาะค่า True หรือ False เท่านั้น

กำหนดการประกาศเหล่านี้ให้ ข้อความสั่งกำหนดค่าทั้งหมดนี้ถูกต้อง (valid)

```
Switch : = True ;     {Switch gets True}
Flaga :  = False ;    {Flag gets False}
Switch : = Flag ;     {Switch gets value of Flag}
```

หลังจากข้อความสั่งเหล่านี้กระทำการทั้ง Flag และ Switch มีค่าเป็น False ทั้งคู่

นิพจน์แบบบูลกับตัวดำเนินการสัมพันธ์ (Boolean Expressions with Relational operators) นิพจน์แบบบูล ซึ่งกระทำการเปรียบเทียบมีรูปแบบดังนี้

variable relational-operator variable
variable relational-operator constant

ตัวดำเนินการสัมพันธ์ คือ สัญลักษณ์ต่อไปนี้

<	less than
< =	less than or equal
=	equal
>	greater than
> =	greater than or equal
< >	not equal

Items ซึ่งนำมาเปรียบเทียบ หรือตัวถูกดำเนินการ (operands) บ่อยครั้งเป็นตัวแปรสองตัว หรือตัวแปรกับค่าคงตัว ถ้า I เป็นชนิด Integer, เงื่อนไข I < 3 เป็นจริง เมื่อ I เป็นค่าลบ หรือ 0, 1 หรือ 2

ตัวถูกดำเนินการสองตัวของตัวดำเนินการสัมพันธ์ต้องมีแบบชนิดข้อมูลเหมือนกัน (เป็นชนิด Integer, Real, Char, หรือ Boolean ทั้งคู่) หรือตัวถูกดำเนินการตัวหนึ่งอาจเป็นชนิด Real และตัวถูกดำเนินการอีกตัวหนึ่งเป็นชนิด Integer ถ้า I เป็นชนิด Integer, เงื่อนไข I < '3' เกิดข้อผิดพลาดวากยสัมพันธชนิด mismatch เพราะว่า '3' เป็นสัญพจน์ (literal) ชนิด Char

ตัวอย่าง 4.1

อัตราภาษีของแต่ละบุคคลขึ้นอยู่กับเงินเดือนของเขาหรือเธอ คนโสดมีเงินเดือนน้อยกว่า $18,550 ภาษีอยู่ที่อัตรา 15% คนที่มีเงินเดือน ระหว่าง $18,550 ถึง $44,900 จ่าย 15% ของเงิน $18,550 แรก และ 28% ของเงินส่วนที่เหลือ ถ้าเงินได้ซึ่งต้องเสียภาษีเก็บในตัวแปรชนิด Real ชื่อ Income นิพจน์แบบบูล ซึ่งสมนัยกับคำถาม "Is annual income less than $18,550"? คือ

Income < 18550.00

นิพจน์ ประเมินผลเป็น True เมื่อคำตอบคือ ใช่ (yes) และเป็น False เมื่อคำตอบคือ ไม่ใช่ (no)

ตัวอย่าง 4.2

ตาราง 4.1 แสดงรายการตัวดำเนินการสัมพันธ์ และเงื่อนไขตัวอย่างเงื่อนไขแต่ละชุดถูกประเมินผล โดยสมมติว่า ตัวแปรต่างๆ มีค่าดังนี้

X	Power	MaxPow	Y	Item	MinItem
-5	1024	1024	7	1.5	-999.0

MomOrDad	Num	Sentinel
M	999	999

ตาราง 4.1 ตัวดำเนินการสัมพันธ์ของ Pascal และเงื่อนไขตัวอย่าง

Operator	Condition	Meaning	Boolean Value
< =	X <= 0	x น้อยกว่าหรือเท่ากับ 0	True
<	Power < MaxPow	Power น้อยกว่า MaxPow	False
> =	X > = Y	x มากกว่าหรือเท่ากับ Y	False
>	Item > MinItem	Item มากกว่า MinItem	True
=	MonOrDad = 'M'	MomOrDad เท่ากับ 'M'	True
< >	Num < > Sentinel	Num ไม่เท่ากับ Sentinel	False

ตัวดำเนินการแบบบูล (Boolean Operators)

เราสามารถประกอบนิพจน์แบบบูลที่ซับซ้อนมากขึ้น โดยใช้ตัวดำเนินการแบบบูลสามชนิด ได้แก่ and, or, not ซึ่งต้องใช้ตัวถูกดำเนินการ ชนิด Boolean

ตัวอย่าง นิพจน์แบบบูล

(Salary < MinimumSalary) or (Dependents > 5)

(Temperature > 90.0) and (Humidity > 0.90)

ตัวแปรแบบบูล คือ นิพจน์แบบบูล ดังนั้น จึงนำมาใช้เป็นตัวถูกดำเนินการของตัวดำเนินการแบบบูลได้

นิพจน์แบบบูล WinningRecord and (not Probation)

จัดดำเนินการตัวแปรแบบบูลสองตัว (WinningRecord และ Probation)

โปรดสังเกตนิพจน์ (WinningRecord = True) and (Probation = False)

ซึ่งสมมูลเชิงตรรกะกับนิพจน์ชุดก่อนหน้า แต่ชุดแรกดีกว่า เพราะว่าเขียนกระชับกว่า และอ่านง่ายกว่า

ตาราง 4.2 แสดงให้เห็นว่าตัวดำเนินการ and ซึ่งให้ผลลัพธ์เป็นจริงเฉพาะเมื่อตัวถูกดำเนินการของมันทั้งคู่เป็นจริง

ตาราง 4.3 แสดงให้เห็นว่าตัวดำเนินการ or ให้ผลลัพธ์เป็นเท็จเฉพาะเมื่อตัวถูกดำเนินการของมันทั้งคู่เป็นเท็จ ตัวดำเนินการ not มีตัวถูกดำเนินการหนึ่งตัว ตาราง 4.4 แสดงให้เห็นว่าตัวดำเนินการ not ให้ผลลัพธ์เป็นส่วนเติมเต็มเชิงตรรกะ หรือ นิเสธ ของตัว

ถูกดำเนินการของมัน (นั่นคือ ถ้า Switch เป็น True, not Switch จะเป็น False และในทางกลับกันเป็นจริง (and vice versa)

ส่วนเติมเต็มเชิงตรรกะ (นิเสธ) หมายถึง ส่วนเติมเต็มเชิงตรรกะของ True และ False และในทางกลับกันเป็นจริง (Logical complement (negation) is the logical complement of True is False and vice versa.)

ตาราง 4.2 ตัวดำเนินการ and

operand 1	operand 2	operand 1 and operand 2
true	true	true
true	false	false
false	true	false
false	false	false

ตาราง 4.3 ตัวดำเนินการ or

operand 1	operand 2	operand 1 or operand 2
true	true	true
true	false	true
false	true	true
false	false	false

ตาราง 4.4 ตัวดำเนินการ not

operand 1	not operand 1
true	false
false	true

การทำก่อนของตัวดำเนินการ กำหนดอันดับการประเมินผลของมัน ตาราง 4.5 แสดงให้เห็นการทำก่อนของตัวดำเนินการทั้งหมดของ Pascal รวมทั้งตัวดำเนินการสัมพันธ์ ตัวดำเนินการ not มีการทำก่อนสูงสุด ตามด้วย multiplicative operators (รวม and), additive operators (รวม or) สุดท้ายคือตัวดำเนินการสัมพันธ์

ตาราง 4.5 การทำก่อนของตัวดำเนินการ

operator	Precedence
not	Highest (evaluate Lirst)
*, /, div, mod, and	↓
+, -, or	↓
<, <=, =, < >, >=, >	Lowest (evaluted last)

เนื่องจากตัวดำเนินการสัมพันธ์มีการทำก่อนต่ำสุด เราอาจจำเป็นต้องใส่เครื่องหมายวงเล็บในนิพจน์ การใช้วงเล็บเพื่อป้องกันข้อผิดพลาดวากยสัมพันธ์

ตัวอย่างเช่น นิพจน์

X < Mix + Max

ทำให้ถูกต้องเป็น

X < (Mix + Max)

เพราะว่า + มีการทำก่อนสูงกว่า < อย่างไรก็ตามนิพจน์ Min < = X and X < = Max (นิพจน์บูลีนไม่ถูกต้อง) เกิดข้อผิดพลาดวากยสัมพันธ์ชนิด mismatch เพราะว่าคอมไพเลอร์ Pascal ให้ความหมายเป็น

Min < = (X and X) < = Max (นิพจน์บูลีนไม่ถูกต้อง)

เพราะว่า and มีการทำก่อนสูงกว่า < = นี่คือข้อผิดพลาด

เพราะว่า ตัวแปร X ชนิด Real ไม่สามารถเป็นตัวถูกดำเนินการของตัวดำเนินการแบบบูล and การใส่วงเล็บทำกับหลีกเลี่ยงข้อผิดพลาดวากยสัมพันธ์

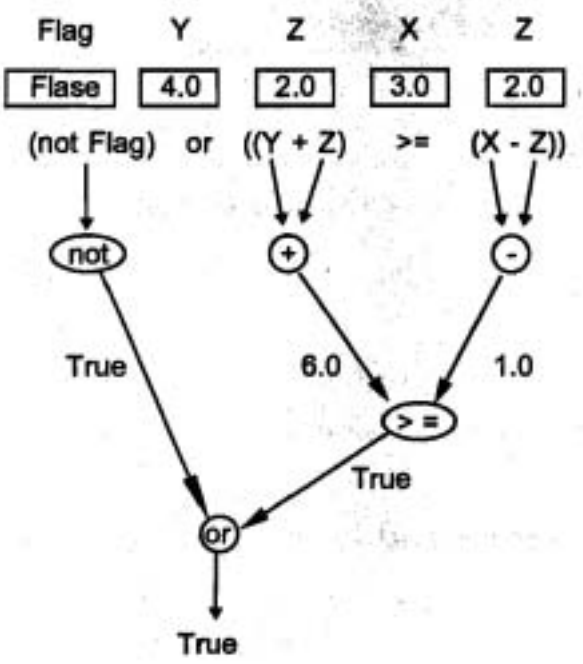
(Min < = X) and (X < = Max)

ตัวอย่าง 4.3

นิพจน์ 1 ถึงนิพจน์ 4 ประกอบด้วยตัวถูกดำเนินการและตัวดำเนินการแตกต่างกัน ค่าของนิพจน์แต่ละชุดในคอมเมนต์ สมมติว่า X, Y และ Z เป็นชนิด Real, Flag เป็นชนิด Boolean และตัวแปรต่างๆ มีค่าดังนี้

X	Y	Z	Flag
3.0	4.0	2.0	False

1. not Flag {not False is True}
2. (X + Y / Z) < = 3.5 {5.0 < = 3.5 is False}
3. (not Flag) or ((Y + Z) > = (X - Z) {True or True is True}
4. not (Flag or ((Y + Z) > = (X - Z))) {not (Flaise or True) is False}



รูป 4.2 ต้นไม้การประเมินผลสำหรับ (not Flag) or ((Y + Z) > = (X - Z))

การเขียนเงื่อนไขภาษาอังกฤษใน Pascal (Writing English Conditions in Pascal)

การแก้ปัญหาการเขียนโปรแกรมเราต้องเปลี่ยนเงื่อนไขที่แสดงด้วยภาษาอังกฤษให้เป็น Pascal ขั้นตอนอัลกอริทึม จำนวนมากทดสอบว่าค่าของตัวแปรอยู่ภายในพิสัยที่กำหนดของค่าต่างๆ หรือไม่ ตัวอย่างเช่น ถ้า Min แทนขอบเขตล่างของพิสัยของค่าต่างๆ และ Max แทนขอบเขตบน (Min น้อยกว่า Max)

นิพจน์

(Min < = X) and (X < = Max)

ทดสอบว่า X อยู่ภายในพิสัย Min จนถึง Max หรือไม่นับรวมขอบเขตล่างและขอบเขตบนด้วย ในรูป 4.3 พิสัยนี้คือส่วนที่แรเงา

นิพจน์เป็นจริง ถ้า X อยู่ภายในพิสัยนี้ และเป็นเท็จ ถ้า X อยู่นอกพิสัย

รูป 4.3 พิสัยของค่า True สำหรับ (Min < = X) and (X < = Max)

ตัวอย่าง 4.4

นิพจน์ Pascal 1 ถึงนิพจน์ 5 implement เงื่อนไขภาษาอังกฤษ ซึ่งแสดงในคอมเมนต์ทางขวามือ คอมเมนต์นี้แสดงการประเมินผลของนิพจน์แต่ละชุด สมมติว่า X คือ 3.0, Y คือ 4.0 และ Z คือ 2.0

1. (Z > X) or (Z > Y)	{Z มีค่ามากกว่า X หรือ y - False หรือ False คือ False}
2. (X = 1.0) or (x = 3.0)	{X เท่ากับ + 1.0 หรือ 3.0 - False หรือ True คือ True}
3. (X > Z) and (Y > Z)	{X และ Y มีค่ามากกว่า Z - True และ True คือ True}
4. (Z < = X) and (X < = Y)	{X อยู่ในพิสัย จาก Z ถึง Y - True และ True คือ True}
5. (X < Z) or (X >Y)	{X อยู่นอกพิสัย Z ถึง Y - False หรือ False คือ False}

นิพจน์ 1 คือ รหัส Pascal สำหรับเงื่อนไขภาษาอังกฤษ

" Z greater than X or Y "

เราอาจเขียนเงื่อนไขนี้เป็น

Z > X or Y

แต่นิพจน์นี้ไม่ถูกต้อง (invalid) เพราะว่า ตัวแปร X และ Y เป็นชนิด Real ไม่สามารถเป็นตัวถูกกระทำของตัวดำเนินการแบบบูล or

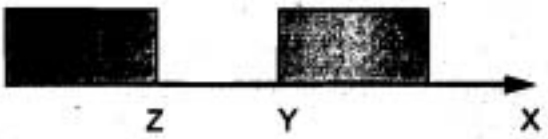
นิพจน์ 4 คือรหัส Pascal สำหรับความสัมพันธ์

Z < = X < = Y ค่าขอบเขต 2.0 และ 4.0

อยู่ในพิสัยของค่า X ซึ่งให้ผลลัพธ์ เป็น True

นิพจน์ 5 เป็นจริง ถ้า X อยู่นอกพิสัยขอบเขตโดย Z และ Y ในรูป 4.4 พื้นที่แรเงาแทนค่าของ X ซึ่งให้ผลลัพธ์เป็น True

Y และ Z ทั้งคู่ ถูกตัดออกเซตของค่าซึ่งให้ผลลัพธ์เป็น True



รูป 4.4 พิสัยของค่า True สำหรับ (x < z) or (X > Y)

การกำหนดค่าแบบบูล (Boolean Assignment)

เราสามารถเขียนข้อความสั่งเพื่อกำหนดค่าแบบบูล ให้กับตัวแปรแบบบูล ถ้าตัวแปร Same มีชนิดเป็น Boolean

ข้อความสั่ง

```
Same := True
```

กำหนดค่า True ให้กับ Same เนื่องจากข้อความสั่งกำหนดค่ามีรูปแบบทั่วไปดังนี้

```
variable := expression
```

เราสามารถใช้ข้อความสั่ง

Same := (X = Y)

เพื่อกำหนดค่าของนิพจน์แบบบูล (X = Y) ให้ Same ค่าของ Same เป็น True เมื่อ X และ Y เท่ากัน กรณีอื่นๆ Same เป็น False

ตัวอย่าง 4.5

ข้อความสั่งกำหนดค่าข้างล่างนี้ กำหนดค่าต่างๆ ให้กับตัวแปรแบบบูลสองตัว คือ InRange และ IsLetter

InRange เป็น True ถ้าค่าของ N อยู่ในพิสัย -10 ถึง 10

IsLetter เป็น True ถ้า Ch เป็นอักษรตัวใหญ่ หรืออักษรตัวเล็ก

InRange := (N > - 10) and (N < 10);

IsLetter := (('A' <= Ch) and (Ch <= 'Z')) or (('a' <= Ch) and (Ch <= 'z'))

นิพจน์ในข้อความสั่งกำหนดค่าชุดแรกเป็นจริง ถ้า N เป็นไปตามรายการเงื่อนไขทั้งคู่ (N มากกว่า -10 และ N น้อยกว่า 10)

กรณีอื่นๆ นิพจน์เป็นเท็จ

นิพจน์ในข้อความสั่งกำหนดค่าชุดที่สองใช้ตัวดำเนินการแบบบูล and, or นิพจน์ย่อยบนบรรทัดแรกเป็น True ถ้า Ch เป็นอักษรตัวใหญ่ นิพจน์ย่อยบนบรรทัดที่สองเป็น True ถ้า Ch เป็นอักษรตัวเล็ก ดังนั้น IsLetter เป็น True ถ้า Ch เป็นตัวอักษร กรณีอื่นๆ IsLetter เป็น False

ตัวอย่าง 4.6

ข้อความสั่งถัดไป กำหนดค่า True ให้กับ Is Even (ชนิด Boolean) ถ้า 2 เป็นตัวหารของ N ชนิด Integer

IsEven := (N mod 2 = 0)

เพราะว่า เลขจำนวนคู่ทั้งหมด จะหารด้วย 2 ลงตัว

ค่าซึ่งกำหนดให้ IsEven แสดงว่า N เป็นจำนวนคู่

(IsEven เป็น True) หรือ จำนวนคี่ (IsEven เป็น False)

การเขียนค่าแบบบูล (Writing Boolean Values)

นิพจน์แบบบูลส่วนใหญ่ปรากฏในโครงสร้างควบคุม ซึ่งกำหนดลำดับ ข้อความสั่ง Pascal ให้กระทำการ เนื่องจากโปรแกรมไม่ประมวลผลข้อมูลแบบบูลในวิธีเดียวกับที่

ประมวลผลข้อมูลเชิงตัวเลข (numerical data) โปรแกรมของเราแทบจะไม่ค่อยมีการอ่านค่าแบบบูล เป็นข้อมูลอินพุต หรือแสดงผลค่าแบบบูลเป็นผลลัพธ์ โปรแกรมถ้าจำเป็น เราสามารถแสดงผลค่าของตัวแปรแบบบูล ด้วยกระบวนงาน Write หรือ WriteLn แต่เราไม่สามารถใช้กระบวนงาน ReadLn อ่านตัวแปรแบบบูล ถ้า Switch เป็น False ข้อความสั่ง WriteLn (' Value of Swritch is ' , Switch)

แสดงผลดังนี้

Value of Switch is FALSE

แบบฝึกหัด 4.2 (Self - Check)

1. นิพจน์แบบบูลต่อไปนี้ชุดใดไม่ถูกต้อง และทำไม สมมติว่า X และ Y เป็นชนิด Real แล P, Q, R เป็นชนิด Boolean

   a) X < 5.1 and y > 22.3

   b) P and Q or Q and R

2. จงวาดรูปต้นไม้การประเมินผลของนิพจน์ต่อไปนี้

   a) A = (B + A - B)

   b) (C = (A +B)) or not Flag

   c) (A < > 7) and (C > = 6) or Flag

   d) not (B < = 12) and (A mod 2 = 0)

   e) not ((A > 5) or (C < (A + B)))

3. จงประเมินผลนิพจน์แต่ละชุดในแบบฝึกหัดข้อ 2 ถ้า A เท่ากับ 5, B เท่ากับ 10, C เท่ากับ 15 และ Flag เท่ากับ True

การเขียนโปรแกรม (Programming)

1. จงเขียนนิพจน์แบบบูล ของความสัมพันธ์แต่ละชุดข้างล่างนี้

   a) Age จาก 18 ถึง 21 inclusive

   b) Water มีค่าน้อยกว่า 1.5 และมีค่ามากกว่า 0.1

   c) Year หารด้วย 4 ลงตัว (ข้อแนะนำ ใช้ mod)

   d) Speed ไม่มากกว่า 55

2. จงเขียนข้อความสั่งกำหนดค่าแบบบูลสำหรับสิ่งต่อไปนี้

a) กำหนดค่าของ True ให้ Between ถ้า N อยู่ในพิสัย -k ถึง +k, inclusive; กรณีอื่นๆ กำหนดค่าของ False

b) กำหนดค่าของ True ให้ UpperCase ถ้า Ch เป็นอักษรตัวใหญ่ กรณีอื่นๆ กำหนดค่าของ False

c) กำหนดค่าของ True ให้ Divisor ถ้า M เป็นตัวหารของ N กรณีอื่นๆ กำหนดค่าของ False

4.3 ข้อความสั่ง if (The if Statement)

ใน Pascal โครงสร้างควบคุมการเลือกแบบแรก ประกอบด้วยข้อความสั่ง if ซึ่งต้องมีนิพจน์แบบบูลเสมอข้อความสั่ง if กำหนดว่า ส่วนรหัสทางเลือกชุดใดจะกระทำการในสถานการณ์หนึ่ง ข้อความสั่ง if ในหัวข้อนี้ เลือกระหว่างหนึ่งทางเลือกหรือสองทางเลือก

ข้อความสั่ง if ที่มีสองทางเลือก (if Statement with Two Alternatives)

ข้อความสั่ง if

```
if Gross > 100.00 then
   Net := Gross - Tax
else
   Net := Gross
```

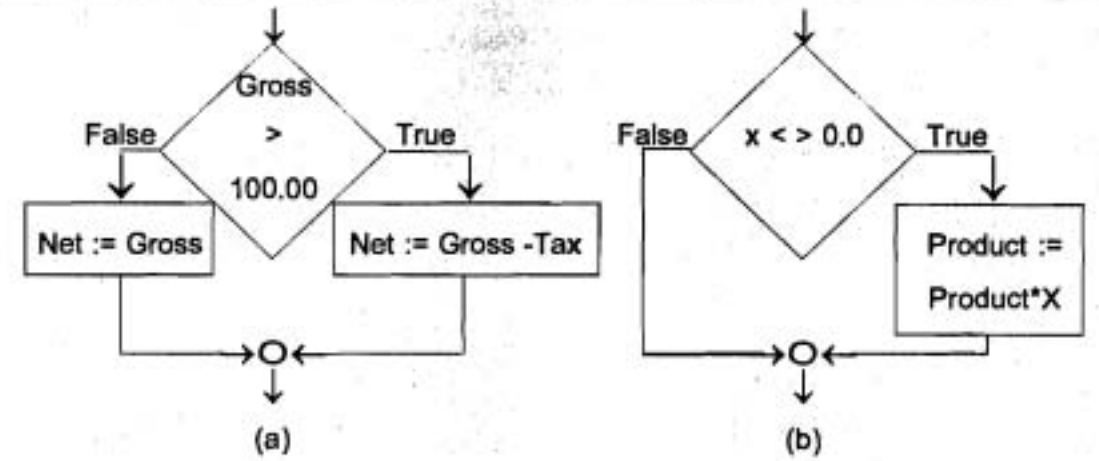
เลือกข้อความสั่งกำหนดค่าหนึ่งคำสั่งในสองคำสั่ง

ถ้านิพจน์แบบบูลเป็นจริง เลือกข้อความสั่งหลัง then (นั่นคือ Gross มากกว่า 100.0) หรือเลือกข้อความสั่งหลัง else ถ้านิพจน์แบบบูลเป็นเท็จ (นั่นคือ Gross ไม่มากกว่า 100.00)

รูป 4.5 (a) เป็นผังงานของข้อความสั่ง if ข้างต้น

ในผังงาน แผนภาพกล่องและลูกศร หมายถึง การกระทำการทีละขั้นตอนของโครงสร้างควบคุม หรือส่วนของโปรแกรมกล่องรูปข้าวหลามตัดในผังงานแทนการตัดสินใจ ซึ่งปกติมีหนึ่งทางเข้าและสองทางออก (ระบุ True หรือ False) กล่องรูปสี่เหลี่ยมผืนผ้าแทนข้อความสั่งกำหนดค่า หรือการประมวลผล

ผังงาน หมายถึงแผนภาพซึ่งแสดงการกระทำการทีละขั้นตอนของโครงสร้างควบคุม หรือส่วนของโปรแกรม (A flowchart is a diagram that shows the step-by-step execution of a control Structure or a program fragment.)



รูป 4.5 ผังงานของข้อความสั่ง if ที่มี a) สองทางเลือก และ b) หนึ่งทางเลือก

รูป 4.5 (a) แสดงให้เห็นว่าเงื่อนไข (Gross > 100.00) ถูกประเมินผลเป็นอันดับแรก ถ้าเงื่อนไขเป็นจริง การควบคุมโปรแกรมจะไปตามลูกศรที่มี True กำกับ และข้อความสั่งกำหนดค่าในรูปสี่เหลี่ยมผืนผ้าขวามือถูกกระทำการ ถ้าเงื่อนไขเป็นเท็จ การควบคุมโปรแกรมจะไปตามลูกศรที่มี False กำกับ และข้อความสั่งกำหนดค่าในรูปสี่เหลี่ยมผืนผ้าซ้ายมือถูกกระทำการ

ข้อความสั่ง if ที่มีหนึ่งทางเลือก (if Statement with One Alternative)

ข้อความสั่ง if ในหัวข้อที่แล้ว มีสองทางเลือกแต่กระทำการเพียงหนึ่งทางเลือกสำหรับค่าที่กำหนดให้ของ Gross เราสามารถเขียนข้อความสั่ง if ที่มีหนึ่งทางเลือกเพื่อให้กระทำการเฉพาะเมื่อเงื่อนไขเป็นจริงเท่านั้น

ข้อความสั่ง if ในรูป 4.5 (6)

```
{Multiply Product by a non zero X only}
if X < > 0.0 then
   Product := Product * X
```

มีหนึ่งทางเลือก ซึ่งจะกระทำการเฉพาะเมื่อ X ไม่เท่ากับ 0 เท่านั้น ทำให้ Product คูณกับ X และค่าใหม่เก็บใน Product แทนที่ค่าเก่า ถ้า X เท่ากับ 0 ไม่ต้องทำการคูณ

การเปรียบเทียบข้อความสั่ง if หนึ่งทางเลือก และสองทางเลือก (A Comparison of One - and Two - Alternative if Statements)

เพื่อให้มีความแตกต่างกันระหว่างข้อความสั่ง if หนึ่งทางเลือกและข้อความสั่ง if สองทางเลือกบ่อยครั้งที่โปรแกรมเมอร์เรียกข้อความสั่ง if ที่มีสองทางเลือกว่า if - then - else และเรียกข้อความสั่ง if ที่มีหนึ่งทางเลือกว่า if - then

ตัวอย่าง 4.7

ข้อความสั่ง if

```
if MonOrDad = 'M' then
   WriteLn ('Hi Mom')
else
   WriteLn ('Hi Dad')
```

มีสองทางเลือก ซึ่งแสดงผล 'Hi Mom' หรือ 'Hi Dad' ขึ้นอยู่กับตัวอักขระ ซึ่งเก็บในตัวแปร MomOrDad ชนิด Char

ตัวอย่าง 4.8

ข้อความสั่ง if ต่อไปนี้มีหนึ่งทางเลือกมันแสดงผลข้อความ 'Hi Mom' เฉพาะเมื่อ MomOrDad มีค่าเป็น 'M' ไม่ว่า 'Hi Mom' จะแสดงผลหรือไม่ก็ตาม ข้อความ 'Hi Dad' จะแสดงผลเสมอ ตัวอย่างนี้มีสองข้อความสั่ง เครื่องหมาย semicolon ตอนจบคำสั่ง if ใช้คั่นข้อความสั่ง if จากข้อความสั่งชุดที่สอง คือ WriteLn

```
if MomOrDad = 'M' then
   WriteLn ('Hi Mom') ;
WriteLn ('Hi Dad')
```

ข้อความสั่ง if ถัดไปไม่ถูกต้องเพราะว่ามี semicolon หน้าคำ else คอมไพเลอร์จะตรวจพบ syntax error เมื่อมันพบ else เพราะว่า semicolon ใช้จบข้อความสั่ง if และข้อความสั่งชุดใหม่ขึ้นต้นด้วย else ไม่ได้

```
if MonOrDad = 'M' then
   WriteLn ('Hi Mom');
else          {error - new statement begin with else}
   WriteIn ('Hi Dad')
```

Syntax Display

If Statement (One Alternative)

Form :

```
if condition then
    Statement_T
```

ตัวอย่าง

```
if X > 0.0 then
    PosProd := PosProd * X
```

มีความหมายดังนี้ ถ้าเงื่อนไขประเมินผลเป็น True ข้อความสั่งหลัง then คือ $Statement_T$ จะถูกกระทำการ กรณีอื่นๆ $Statement_T$ จะถูกข้าม

Syntax Display

If Statement (Twor Alternatives)

Form :

```
if condition then
    Statement_T
else
    Statement_F
```

ตัวอย่าง

```
if x > = 0.0 then
    Write ('Positive')
else
    Write ('Negative')
```

มีความหมายดังนี้ ถ้า condition ประเมินผลเป็น True แล้ว $Statement_T$ (งานที่เป็นจริง) จะถูกกระทำการและ Statement F จะถูกข้าม กรณีอื่นๆ (otherwise) $statement_T$ ถูกข้ามและ $Statement_F$ (งานที่เป็นเท็จ) จะถูกกระทำการ

สไตล์ของโปรแกรม (Program Style)

รูปแบบของข้อความสั่ง if (Format of the if Statement)

ตัวอย่าง ข้อความสั่ง if ทั้งหมดในตำราเล่มนี้ $statement_T$ และ $statement_F$ ย่อหน้าตรงกัน คำว่า else พิมพ์บนหนึ่งบรรทัดแยกต่างหากปรับให้ตรงกับคำว่า if การจัดรูปแบบของข้อความสั่งทำให้โปรแกรมอ่านง่ายขึ้น การจัดรูปแบบไม่ทำให้เกิดความแตกต่างกับคอมไพเลอร์

แบบฝึกหัด 4.3 (Self - Check)

1. จงบอกผลลัพธ์ที่ข้อความสั่งเหล่านี้แสดงผล

a)
```
if 12 < 12 then
    WriteLn ('Less')
else
    WriteLn ('Not Less')
```

b)
```
Var1 := 25.12 ;
Var2 := 15.00 ;
If var1 < = var2 then
    WriteLn ('Less or equal')
else
    WriteLn ('Greater than')
```

2. จงหาค่า X เมื่อ Y คือ 15.0

a)
```
X := 25.0 ;
if Y < > (X - 10.0) then
    X := X - 10.0
else
    X := X / 2.0
```

b)
```
if (Y < 15.0) and (Y > - 0.0) then
    X := 5 * Y
else
    X := 2 * Y
```

เขียนโปรแกรม (Programming)

1. จงเขียนข้อความสั่ง Pascal ให้กระทำขั้นตอนต่อไปนี้

a) ถ้า Item ไม่เท่ากับศูนย์ให้คูณ Product ด้วย Item และเก็บผลลัพธ์ใน Product กรณีอื่นๆ ข้ามการคูณไม่ว่าจะเป็นกรณีใดก็ตาม ให้พิมพ์ค่าของ Product

b) เก็บผลต่างสัมบูรณ์ (absolute difference) ของ X และ Y ใน Y เมื่อผลต่างสัมบูรณ์ คือ (X - Y) หรือ (Y - X) ซึ่งเป็นค่าบวก ไม่ให้ใช้ฟังก์ชัน Abs ในผลเฉลย

c) ถ้า X เป็น 0 ให้บวก 1 กับ ZeroCount

ถ้า X เป็นค่าลบ ให้บวก X กับ MinusSum

ถ้า X มีค่ามากกว่า 0 ให้บวก X กับ PlusSum

4.4 แผนภาพวากยสัมพันธ์ (Syntax Diagrams)

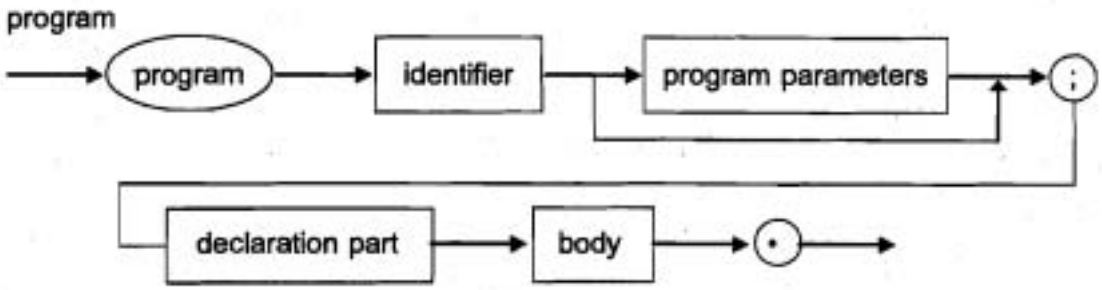
นอกเหนือจาก syntax displays แล้วเราใช้ แผนภาพวากยสัมพันธ์อธิบายตัวสร้าง Pascal บางครั้งเราเรียกแผนภาพวากยสัมพันธ์ว่า railroad diagrams เพราะว่ามันคล้ายแผนภาพการวางผังสำหรับแบบจำลองรางรถไฟ

แผนภาพวากยสัมพันธ์ หมายถึง การแทนที่เชิงภาพของวากยสัมพันธ์ของตัวสร้าง Pascal (Syntax diagram is a graphical representation of the syntax of a pascal construct.)

การเรียนว่าจะอ่านแผนภาพวากยสัมพันธ์อย่างไร ให้ศึกษาแผนภาพวากยสัมพันธ์ program (รูป 4.6)

แผนภาพประกอบด้วยกลุ่มของสมาชิกเชิงวากยสัมพันธ์ต่อกันด้วยลูกศร การจำแนกชนิดของสมาชิกเชิงวากยสัมพันธ์แต่ละตัวแสดงด้วยรูปร่างของมัน

- คำสงวน อยู่ในรูปไข่
- Special symbols (เครื่องหมายกำกับวรรคตอนและตัวดำเนินการ) อยู่ในวงกลม
- สมาชิกเชิงวากยสัมพันธ์ที่มีแผนภาพวากยสัมพันธ์ของตัวมันเองอยู่ในรูปสี่เหลี่ยมผืนผ้า



รูป 4.6 แผนภาพวากยสัมพันธ์ สำหรับ program

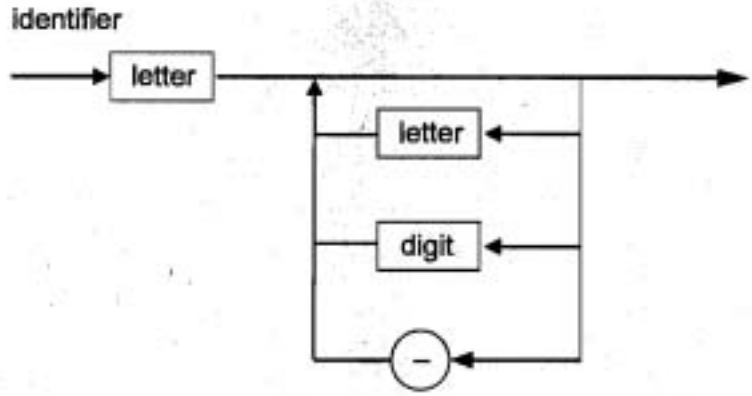
ตามรอยตลอดแผนภาพวากยสัมพันธ์ ในรูป 4.6 ตามลูกศร เริ่มต้นที่ลูกศรซ้ายมือสุด (เข้าสู่ program) และทางออกที่ลูกศรขวามือสุด (ออกจากสัญลักษณ์ .) จากแถวบนสุด โปรดสังเกตว่าโปรแกรม Pascal เริ่มต้นด้วยคำสงวน program ตามด้วย identifier ลูกศรจาก identifier แบ่งเป็นสองทาง : ทางหนึ่งไปยัง program parameters ไปยังสัญลักษณ์ ; และอีกทางหนึ่งข้าม program parameters หมายความว่า สมาชิกเชิงวากยสัมพันธ์ program parameters ละเว้นได้ ส่วนหัวโปรแกรม

program First ;

เขียนถูกต้องตามแผนภาพวากยสัมพันธ์ เพราะว่า First คือ identifier ส่วนที่เหลือของแผนภาพวากยสัมพันธ์ แสดงว่า สัญลักษณ์ ; ต้องตามด้วยสมาชิกเชิงวากยสัมพันธ์ dedaration part, body และสัญลักษณ์ .

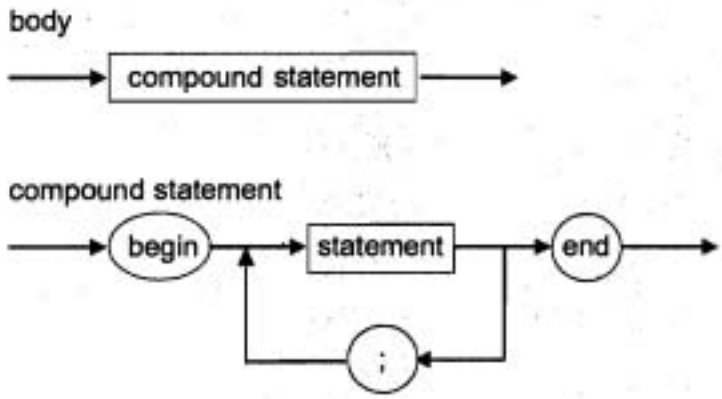
แผนภาพวากยสัมพันธ์ สำหรับ identifier (รูป 4.7) แสดงว่า identifier อาจเป็นตัวอักษรหนึ่งตัว (A-Z, a-z) เราตรวจดูสิ่งนี้ โดยตามรอยลูกศรแนวนอนที่ตอนบนในแผนภาพจากซ้ายไปขวา จากการตามรอยตลอดส่วนวนซ้ำในแผนภาพจะเห็นว่า ตัวอักษรตัวแรกอาจตามด้วยตัวอักษรหนึ่งตัวหรือมากกว่าหนึ่งตัว เลขโดด (0-9) หรือสัญลักษณ์ขีดเส้นใต้ (ตัวอย่างเช่น R2D2, First) สิ่งนี้สมนัยกับบทนิยามก่อนหน้าของ identifier (ดูหัวข้อ 2.3)

แผนภาพวากยสัมพันธ์ สำหรับสมาชิกเชิงวากยสัมพันธ์ declaration part และ body ทำให้การอธิบายสมาชิกวากยสัมพันธ์ program ครบถ้วน แผนภาพสำหรับ declaration part อยู่ในภาคผนวก C รวมทั้งสมาชิกวากยสัมพันธ์ตัวอื่นๆ ของ Pascal



รูป 4.7 แผนภาพวากยสัมพันธ์ สำหรับ identifier

แผนภาพวากยสัมพันธ์ สำหรับ body (รูป 4.8) แสดงให้เห็นว่าโปรแกรม body คือ compound statement หรือลำดับของสมาชิก statement 1 คำสั่งหรือมากกว่า 1 คำสั่ง คั่นด้วย semicolon และปิดล้อมด้วย begin และ end



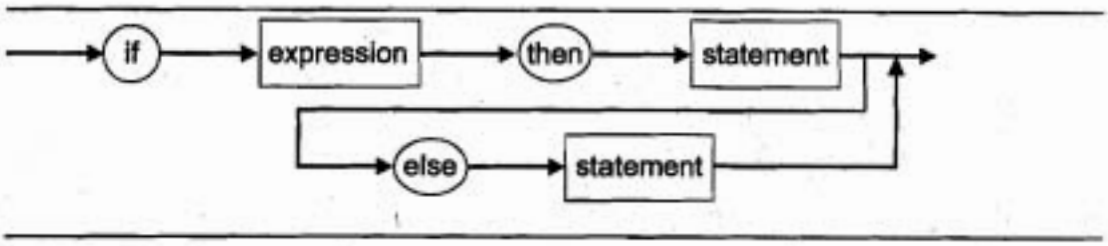
รูป 4.8 แผนภาพวากยสัมพันธ์ สำหรับ body และ compound statement

รูป 4.9 แสดงให้เห็นแผนภาพวากยสัมพันธ์สำหรับ if statement

ลูกศรออกจากจากสมาชิก statement ที่แถวบนสุดแบ่งออกเป็นสองทาง : ลูกศรชี้ไปทางขวามือให้นิยามของข้อความสั่ง if ที่มีหนึ่งทางเลือก (if - then) ส่วนลูกศรที่ชี้ลง และ

ไปทางซ้ายมือ จนถึงคำสงวน else ให้นิยาม ข้อความสั่ง if ที่มีสองทางเลือก (if - then - else) ข้อความสั่งซึ่งตามหลังคำ then หรือ else อาจเป็น executable statement หนึ่งคำสั่ง หรือ compound statement ตัวอย่าง ข้อความสั่งเช่นนี้ ได้แก่ assignment statements, procedure call statements และ if statement อื่นๆ

โดยการเปรียบเทียบ ข้อความสั่งโปรแกรมกับแผนภาพวากยสัมพันธ์ที่สมนัยกันของมัน เราสามารถทวนสอบได้ว่าข้อความสั่งนั้นถูกต้อง ถ้าข้อผิดพลาดวากยสัมพันธ์เกิดขึ้นระหว่างการแก้ไขจุดบกพร่อง (debngging) เราสามารถอ้างถึงแผนภาพวากยสัมพันธ์ที่ถูกต้องเพื่อตรวจรูปแบบที่ถูกต้องของสมาชิกที่เป็นเหตุให้เกิดปัญหา ภาคผนวก C ประกอบด้วยแผนภาพวากยสัมพันธ์ทั้งหมดของ Pascal

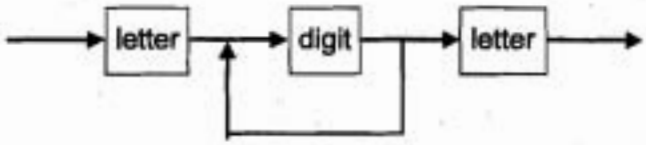


รูป 4.9 แผนภาพวากยสัมพันธ์ สำหรับข้อความสั่ง if

แบบฝึกหัด 4.4 (Self - Check)

1. ไอเดนติไฟเออร์ต่อไปนี้ ตัวใดที่ถูกต้องตามแผนภาพวากยสัมพันธ์

   Ace, R2D2, R245, A23B, A1c, B34d5c, A23cd



2. จงวาดรูปแผนภาพวากยสัมพันธ์ เพื่ออธิบายสัญพจน์เชิงตัวเลขคล้าย Real (Real - like numeric (literals) ซึ่งขึ้นต้นด้วยเลขโดด และจบด้วยเลขโดด และประกอบด้วยจุดทศนิยมหนึ่งตัวที่ใดที่หนึ่งระหว่างเลขโดด

3. จงดัดแปรแผนภาพวากยสัมพันธ์สำหรับ identifier เพื่อแสดงให้เห็นว่า identifier อาจขึ้นต้นด้วยด้วยสัญลักษณ์ underscore หรือตัวอักษรก็ได้

4.5 ข้อความสั่ง if ที่มีข้อความสั่งประกอบ (if Statement with Compound Statements)

ข้อความสั่ง if บางชุดมีข้อความสั่งประกอบตามหลัง then หรือตามหลัง else เมื่อคำสงวน begin ตามหลัง then หรือ else คอมไพเลอร์ Pascal จะแปลข้อความสั่งต่างๆ ระหว่าง begin และ end เป็นข้อความสั่งประกอบ

ตัวอย่าง 4.9

ปัญหาการเขียนโปรแกรมจำนวนมาก เราต้องเรียงอันดับคู่ของค่าข้อมูลในหน่วยความจำ เพื่อให้ค่าเล็กกว่าเก็บในตัวแปรหนึ่งตัว (ชื่อ X) และค่าใหญ่กว่าเก็บที่ตัวแปรอีกหนึ่งตัว (ชื่อ Y) ในรูป 4.10 ข้อความสั่ง if จัดเรียงใหม่ค่าสองค่าใดๆ ก็ตาม ซึ่งเก็บใน X และ Y เพื่อให้เลขตัวเล็กกว่าๆ อยู่ใน X และเลขตัวใหญ่กว่าอยู่ใน Y ถ้าเลขสองตัวนั้นเรียงอันดับถูกต้องมาแล้ว ข้อความสั่งประกอบจะไม่ถูกกระทำการ ตัวแปร X, Y และ Temp ทั้งหมดนี้ควรจะมีแบบชนิดข้อมูลเหมือนกันถึงแม้ว่าค่าของ X และ Y กำลังจะสลับที่กัน ตัวแปร Temp จำเป็นต้องเก็บสำเนาของหนึ่งค่าในสองค่านี้อยู่

```
if X > Y then
   begin {switch X and Y}
      Temp  :=  X ;      {stores old X in Temp}
         X  :=  Y ;      {stores old Y in X}
         Y  : =  Temp    {stores old X in Y}
   End {if}
```

รูป 4.10 ข้อความสั่ง if เพื่อเรียงอันดับ x และ y

ตาราง 4.6 จำลองแบบการกระทำการของข้อความสั่ง if ชุดนี้ เมื่อ X คือ 12.5 และ Y คือ 5.0 ในตารางแสดงให้เห็นว่า Temp เริ่มต้นไม่ถูกนิยาม (แสดงด้วยเครื่องหมาย ?) แต่ละบรรทัดแสดงส่วนของข้อความสั่ง if ซึ่งกำลังกระทำการตามด้วยผลของมัน ถ้าตัวแปรใดก็ตามรับค่าใหม่ ค่าใหม่ของมันจะแสดงให้เห็นบนบรรทัดนั้น ถ้าไม่มีค่าใหม่ปรากฏให้เห็นตัวแปรนั้น ยังคงเก็บค่าก่อนหน้าของมัน ค่าสุดท้ายซึ่งเก็บใน X คือ 5.0 และ ค่าสุดท้ายซึ่งเก็บใน Y คือ 12.5

ตาราง 4.6 การจำลองแบบทีละขั้นตอนของข้อความสั่ง if

statement Part	X	Y	Temp	Effect
	12.5	5.0 ?		
if X > Y then				12.5 > 5.0 is true
Temp := X ;			12.5	Stores old X in Temp
X := Y;	5.0			Stores old Y in X
Y := Temp		12.5		Stores old X in Y

ตัวอย่าง 4.10

ในฐานะที่เราเป็นผู้จัดการแผนกเสื้อผ้า เราต้องเก็บระเบียนของการตรวจสอบรายการเปลี่ยนแปลง เมื่อ TransType is 'C' ใน if statement ถัดไปข้อความสั่งประกอบหลัง then ประมวลผลรายการเปลี่ยนแปลง (TransAmount) ซึ่งแทนการตรวจสอบที่เราเขียนเพื่อจ่ายเงินค่าสินค้าที่รับไว้ กรณีอื่นๆ ข้อความสั่งประกอบหลัง else ประมวลผลการฝากเงิน ซึ่งกระทำในบัญชีการตรวจสอบ ข้อความสั่งประกอบทั้งคู่ แสดงผลข้อความที่เหมาะสมและปรับ (update) งบดุลบัญชี (account balance) ชื่อ BALANCE

```
If  Trans Type = 'C' then
    begin {check}
       Write (' Check for $ ', TransAmount : 4 : 2);
       Balance := Balance - TransAmount {Deduct check}
    end {check}
else
    begin {deposite}
       Write ('Deposite of $ ' , Trans Amoun : 4 : 2);
       Balance := Balance + TransAmount {Add deposit}
    end      {deposit and if}
```

เครื่องหมาย semicolon ในข้อความสั่ง if คั่นแต่ละข้อความสั่งในแต่ละทางเลือก ข้อผิดพลาดร่วมซึ่งอาจจะเกิดขึ้นถ้าใส่ semicolon หลัง end ตัวแรก (end; {check}) ซึ่งจะทำให้ข้อความสั่ง if ถูกแยกเป็นข้อความสั่งสองคำสั่ง เนื่องจากข้อความสั่งชุดที่สอง ขึ้นต้นด้วย else ไม่ได้ คอมไพเลอร์ จะแสดงผลข้อความผิดพลาดว่า unexpected symbol

สไตล์การเขียนโปรแกรม

การเขียนข้อความสั่ง if ที่มีข้อความสั่งประกอบ True หรือ False (Writing if Statement with Compound True or False Statements)

ข้อความสั่ง if แต่ละคำสั่งในหัวข้อนี้ ประกอบด้วยข้อความสั่งประกอบอย่างน้อยหนึ่งคำสั่ง ปิดล้อมด้วย begin และ end เพื่อให้การอ่านหรือการทำความเข้าใจข้อความสั่ง if ปรับปรุงให้ดีขึ้น

ข้อความสั่งประกอบแต่ละชุดให้ย่อหน้า ทั้งนี้คอมไพเลอร์ Pascal ไม่สนใจ (ignore) การย่อหน้า

คอมเมนต์หลัง end แต่ละชุด ช่วยให้มีการเกี่ยวข้อง end กับ begin ซึ่งสัมนัยกัน คอมเมนต์อาจไม่เขียนก็ได้แต่การเขียนคอมเมนต์ไว้ทำให้อ่านโปรแกรมดีขึ้น

เครื่องหมาย semicolon ต้องใส่ระหว่างข้อความสั่งแต่ละชุดภายในข้อความสั่งประกอบ แต่ไม่ใส่ semicolon ก่อนหรือหลังคำสงวน then, else หรือ begin

semicolon อาจอยู่หลัง end ซึ่งเป็นตัวจบข้อความสั่ง if ทั้งหมด

แบบฝึกหัด 4.5 (Self - Check)

1. จงใส่ semicolons ในตำแหน่งที่จำเป็นเพื่อหลีกเลี่ยงข้อผิดพลาดวากยสัมพันธ์ และเขียนใหม่โดยย่อหน้าเพื่อทำให้อ่านโปรแกรมดีขึ้น

```
{incorrect if statement}
if X > Y then
begin
X := X + 10.0
WriteLn ('X Bigger')
end
else
WriteLn ('X Smaller')
WriteLn ('Y is ' , Y)
```

2. จงอธิบายว่าทำไม โปรแกรมจึงคอมไพล์ (compile) ไม่ผ่านถ้าเราลบ begin และ end ออกไป

3. จงอธิบายผลของการใส่ begin และ end ปิดล้อมสองบรรทัดท้ายสุดในแบบฝึกหัดข้อ (1)

4. จงหา syntax error และ logic error ในข้อความสั่ง if :

```
if  num 1 < 0 then
    begin
       Product := Num 1 * Num 2 * Num 3 ;
       WriteLn ('Product is ', Product : 1)
    end ;
else
    Sum := Num 1 + Num 2 + Num 3 ;
    WriteLn ('Sum is ' , Sum : 1)
```

5. แผนภาพวากยสัมพันธ์ อะไรบ้างซึ่งเราจะต้องใช้เพื่อตรวจสอบ ความถูกต้องของข้อความสั่ง if ข้างล่างนี้

```
if X > 0 then
   begin
      x := 25.0 ;
WriteLn ('Positive')
  End
```

เขียนโปรแกรม

1. จงเขียนข้อความ if ซึ่งกำหนดค่าจำนวนจริง 2 ค่าคือ X และ Y ให้เป็นนิเสธสองค่า ถ้าเป็นค่าลบทั้งคู่ หรือเป็นค่าบวกทั้งคู่

2. จงเขียนโปรแกรมเชิงโต้ตอบ (interactive program) ซึ่งคำนวณพื้นที่ของรูปสี่เหลี่ยมผืนผ้า (area = base x height) หรือพื้นที่ของรูปสามเหลี่ยม (area = 1/2 x base x height) หลังจากแจ้งผู้ใช้ให้พิมพ์อักขระตัวแรกของชื่อรูป (R หรือ T)

4.6 ขั้นตอนการตัดสินใจในอัลกอริทึม (Decision Steps in Algorithms)

ขั้นตอนอัลกอริทึม ซึ่งเลือกหนึ่งการกระทำของการกระทำต่างๆ เรียกว่า ขั้นตอนการตัดสินใจ (Decision step is an algorithm step that selects one of several actions.)

อัลกอริทึมในปัญหาบัญชีเงินเดือนต่อไปนี้ ประกอบด้วยขั้นตอนการตัดสินใจ ซึ่งลงรหัสด้วยข้อความสั่ง if ของ Pascal เพื่อคำนวณเงินเดือนรวม และเงินเดือนสุทธิของพนักงานหนึ่งคน

กรณีศึกษา ปัญหาบัญชีเงินเดือน

1. ปัญหา (Problem)

บริษัทที่เราทำงานอยู่จ่ายเงินให้กับพนักงานหนึ่งเท่าครึ่ง สำหรับเวลาทำงานทั้งหมดส่วนที่เกิน 40 ชั่วโมง ต่อสัปดาห์ พนักงานซึ่งมีเงินรายรับมากกว่า $100.00 ต่อสัปดาห์ ต้องจ่ายค่าธรรมเนียมสหภาพ $25 ต่อสัปดาห์ จงเขียนโปรแกรมคำนวณเงินได้รวม (gross pay) และเงินได้สุทธิ (net pay) ของพนักงาน

2. วิเคราะห์ (Analysis)

ขั้นแรกเราต้องอ่านจำนวนชั่วโมงทำงานและอัตราค่าจ้างรายชั่วโมงของพนักงาน (อินพุตของปัญหา) หลังจากอ่านข้อมูลเหล่านี้แล้วคำนวณเงินได้รวม พนักงานซึ่งทำงาน 40 ชั่วโมง หรือน้อยกว่า 40 ชั่วโมง ให้จ่ายเงินตามอัตราเดียวกัน สำหรับจำนวนชั่วโมงทำงานทั้งหมด ส่วนพนักงานซึ่งทำงานมากกว่า 40 ชั่วโมงขึ้นไป ได้รับเงินอัตราหนึ่งสำหรับ 40 ชั่วโมงแรก และ 1.5 เท่าเป็นอัตราสำหรับชั่วโมงทำงานนอกเวลา จากนั้นคำนวณเงินได้สุทธิ โดยลบเงินค่าธรรมเนียมสหภาพจากเงินได้รวม

```
Date Reguirements
Problem Constants
MaxNoOvertime = 40.0 {maximum hours without over time pay}
Bonus Rate = 1.5     {time and a half for over time}
MaxNoDues = 100.00   {maximum salary with out union dues}
Dues = 25.00         {union dues}
Problem input
   Hours : Real      {hours worked}
   Rate : Real       {hourly rate}
```

Problem output

Gross : Real {gross pay}

Net : Real {net pay}

Relevant Formulag

grosspay = hours worked x hourly rate

gross pay = 40 x hourly rate + 1.5 x overtime hours x hourly rate

Netpay = gross pay - union dues

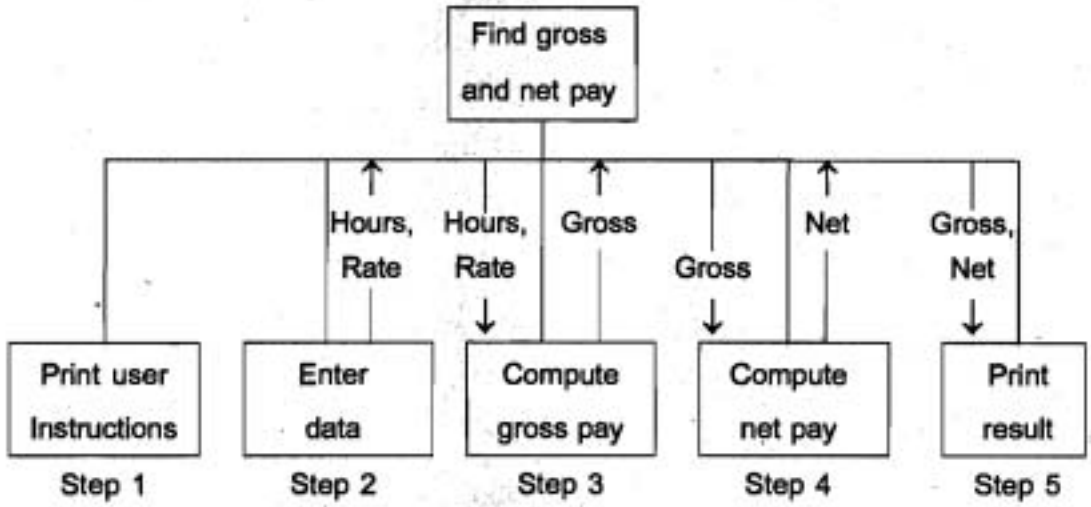
gross pay มี 2 สูตร : สูตรแรกสำหรับพนักงานซึ่งไม่มีเงินตอบแทนล่วงเวลา สูตรที่สองสำหรับพนักงานซึ่งมีเงินล่วงเวลา

3. ออกแบบ (Besign)

ผังโครงสร้าง (รูป 4.11) สำหรับปัญหาข้อนี้ ประกอบด้วยกระแสข้อมูลของสารสนเทศ แสดงให้เห็นอินพุต และเอาต์พุตของขั้นตอนอัลกอริทึมแต่ละชุด ผังโครงสร้างซึ่งแสดงขั้นที่ 2 Enter data จัดค่าสำหรับ Hours และ Rate และเป็นเอาต์พุตของมัน (ลูกศรกระแสข้อมูลชี้ขึ้น) ในทำนองเดียวกัน ขั้น Compute gross pay ใช้ Hours และ Rate เป็นอินพุต (ลูกศรกระแสข้อมูลชี้ลง) และได้ Gross เป็นเอาต์พุต

อัลกอริทึมเริ่มต้น

1. แสดง user instructions
2. ใส่ hours worked และ hourly rate
3. คำนวณ gross pay
4. คำนวณ net pay
5. Display gross และ net pay



รูป 4.11 ผังโครงสร้างสำหรับ Payroll Problem

การแบ่งละเอียดอัลกอริทึม (Algorithm Refinements)

แบ่งละเอียดอัลกอริทึมขั้นที่ 3 และ 4 เป็นขั้นตอนการตัดสินใจ

Step 3 Refinement

3.1 if no overtime then

3.2 Compute gross pay withont overtime pay

else

3.3 Compute gross pay with overtime pay

Step 4 Refinement

4.1 if no union dues then

4.2 Net gets Gross

else

4.3 Net gets Gross - Dues

ขั้นตอนตัดสินใจเหล่านี้ เขียนด้วยรหัสเทียม (pseudocode) ซึ่งเป็นการผสมกันระหว่างวลีภาษาอังกฤษและตัวสร้าง Pascal เพื่อใช้อธิบายขั้นอัลกอริทึม รหัสเทียมใช้การย่อหน้า และคำสงวน if, then และ else เพื่อแสดงโครงสร้างเชิงตรรกะของแต่ละขั้นตอน

ขั้นตอนตัดสินใจแต่ละขั้นมีเงื่อนไข (ตามหลัง if) ซึ่งสามารถเขียนเป็นภาษาอังกฤษ หรือ Pascal และสามารถใช้เลขลีบเนื่องให้กับ True task และ False task

4. การปฏิบัติให้เกิดผล (Implementation)

โปรแกรม (รูป 4.12) เริ่มต้นด้วยการเรียกโปรซีเดอร์ InstructPay เพื่อแสดงผลคำสั่งผู้ใช้ (สิบบรรทัดแรกของเอาต์พุตโปรแกรม) หลังจากอ่านข้อมูลอินพุตแล้ว ข้อความสั่ง if

```
{Compute gross pay}
   if Hours < =   MaxNoOvertime then
      Gross :=    Hours * Rate          {no overtime pay}
else
      Gross : =   MaxOvertime * Rate + BonusRate * (Hours - MaxNo
                  Overtime) * Rate;     {overtime}
```

คือการ implement ขั้นที่ 3 คอมเมนต์แรกทางขวามือฝังตัว (embedded) ในข้อความสั่ง if เครื่องหมาย semicolon ในบรรทัดสุดท้ายคั่นข้อความสั่ง if ออกจากข้อความสั่ง if ถัดไป

5. การทดสอบ (Testing)

การทดสอบโปรแกรมนี้ วิ่งโปรแกรมกับเซตข้อมูลหลายชุด ซึ่งจะให้ผลลัพธ์ทุกกลุ่มของความเป็นไปได้ทั้งหมดสำหรับสองเงื่อนไขของข้อความสั่ง if (True / True, True / False, False / True, False / False) ตัวอย่างเช่น ถ้าต้องการได้ค่าเงื่อนไข True / True จำนวนชั่วโมงทำงานควรน้อยกว่า 40 ชั่วโมง และ gross pay ควรน้อยกว่า $100.00 การทดสอบโปรแกรมด้วยชุดข้อมูล ซึ่งจำนวนชั่วโมงทำงานเท่ากับ 40 ชั่วโมง และ gross pay เท่ากับ $ 100.00 พอดีต้องทดสอบด้วยเช่นกัน

Edit Window

```
Program Payroll;
{
   Computes and prints gross pay and net pay given an hourly rate and
   number of hours worked. Overtime pay is included in the gross pay
   computation. The employee 's net pay includes a possible deduction }
   for union dues.
}
const
   MaxNoOver time = 40 ;        {maximum hours without over time pay}
   Bonus Rate = 1.5 ;           {time and a half for overtime}
   Max No Due 3 = 100.00 ;      {maximum salary without union dues}
   Dues = 25.00 ;               {union dues}
Var
   Hours, Rate,                 {inputs - hours worked, hourly rate}
   Gross, Net : Real;           {ont puts - gross pay, net pay}
Procedure InstructPay ;
{Displays user instructions}
begin {InstructPay}
   WriteLn ('This program computes gross and net pay.') ;
   WriteLn ('Employees who work more than' ; MaxNoOvertime : 1) ;
   WriteLn ('hours receive overtime pay for the excess hours.') ;
   WriteLn ('Union dues of $ ' , Dues : 4 : 2, ' are deducted') ;
   WriteLn ('for an employee who earns more than $ ' , MaxNoDues : 4 : 2);
   WriteLn ;
   WriteLn (' Enter hours worked and hourly rate');
   WriteLn ('on separate lines after the prompts.');
   WriteLn ('Press Enter after typing each number.');
   WriteLn ;
end; {Instruct Pay}
```

```
begin {Payroll}
   InstructPay ; {Display user instructions}

   {Enter Hours and Rate.}
   Write ('Hours worked > ');
   ReadLn (Hours) ;
   Write ('Hourly rate > ');
   ReadLn (Rate) ;

   {Compute gross pay.}
   if Hours <= MaxNoOvertime then {No overtime pay}
      Gross : = Hours * Rate
   else
      Gross : = MaxNoOvertime * Rate +
                BonusRate * (Hours - MaxNoOvertime) * Rate; {overtime}

   {Compute net pay}
   if Gross <= MaxNoDues then
      Net := Gross            {Deduct no dues.}
   else
      Net := Gross - Dues ; {Deduct union dues.}

   {Print Gross and Net.}
   WriteLn ('Gross pay is $', Gross : 4 : 2);
   WriteLn ('Net pay is $', Net : 4 : 2)
end. {Payroll}
```

Output Window

```
This program computes gross and net pay.
Employees who work more than 40 hours receive overtime pay for the
excess hours.
Union dues of $25.00 are deducted for an employee who earns more than
$100.00

Enter hours worked and hourly rate
on separate lines after the prompts.
Press Enter after typing each number.

Hours worked > 40.0
Hourly rate > 5.0
Gross pay is $200.00
Net pay is $175.00
```

รูป 4.12 โปรแกรมสำหรับ Payroll Problem

สไตล์โปรแกรม

การใช้ค่าคงตัว เพื่อส่งเสริมการอ่านง่ายและการบำรุงรักษาโปรแกรมค่าคงตัว MaxNoOvertime และ BonusRate ที่ปรากฏในข้อความสั่ง if ในรูป 4.12 เราเพียงแค่แทนด้วยสัญพจน์ 40 และ 1.5 โดยตรง

ในข้อความสั่ง if :

```
{Compute gross pay.}
if Hours <= 40 then
   Gross :=  Hours * Rate  {no over time pay}
else
   Gross :=  40 * Rate +
             1.5 * (Hours - 40) * Rate;     {overtime pay}
```

อย่างไรก็ตาม การใช้ค่าคงตัวซึ่งไม่ใช่สัญพจน์ มีข้อดี 2 ข้อคือ ข้อหนึ่ง ข้อความสั่ง if เดิมเข้าใจง่ายกว่า เพราะว่ามันมีคำอธิบายชื่อ MaxNoOvertime และ BonusRate อยู่แล้ว ไม่ใช่เป็นเลข ซึ่งไม่มีความหมายใดๆ ข้อที่สอง การเขียนโปรแกรมด้วยค่าคงตัว บำรุงรักษาง่ายกว่าการเขียนเป็นสัญพจน์ ตัวอย่างเช่น การเปลี่ยนโบนัส สำหรับ overtime ให้เป็นสองเท่า ไม่ใช่หนึ่งเท่าครึ่ง เราจำเป็นต้องเปลี่ยนเฉพาะการประกาศค่าคงตัว Bonus Rate เป็น 2.00 เท่านั้นแต่ถ้าเราใส่สัญพจน์ 1.5 โดยตรงในข้อความสั่ง if เราจำเป็นต้องเปลี่ยนข้อความสั่ง if และข้อความสั่งอื่นๆ ซึ่งมีการจัดดำเนินการกับสัญพจน์ตัวนั้น

โปรดสังเกตว่าค่าคงตัวจะปรากฏในข้อความสั่ง WriteLn ในกระบวนงาน InstructPay ด้วยเป็นการทำให้อย่างบริบูรณ์ เมื่ออ้างถึงค่าคงตัวโปรแกรมในตัวกระบวนงาน

แบบฝึกหัด 4.6 (Self - Check)

1. จงเปลี่ยนการแบ่งละเอียดของขั้นที่ 4 ของ payroll problem เพื่อให้ใช้ขั้นตอนการตัดสินใจที่มีหนึ่งทางเลือก

ข้อแนะนำ กำหนด Gross ให้กับ Net ก่อนขั้นตอนตัดสินใจและใช้ขั้นตอนตัดสินใจเพื่อเปลี่ยนแปลง Net เมื่อมีความจำเป็น

2. จงบอกเอาต์พุตของโปรแกรม payroll เมื่อ

a) Hours เท่ากับ 30.0, Rate เท่ากับ 5.00

b) Hours เท่ากับ 20.0, Rate เท่ากับ 4.00

c) Hours เท่ากับ 50.0, Rate เท่ากับ 2.00

d) Hours เท่ากับ 50.0, Rate เท่ากับ 6.00

เขียนโปรแกรม

1. จงดัดแปรโปรแกรม payroll problem เพื่อให้หักลด union dues 10% สำหรับเงินเดือนรวม (gross salary) ที่มากกว่า $100.00 และกรณีอื่นๆ ให้หักลด union dues 5% และหักลด 3% เป็น city wage tax สำหรับพนักงานทุกคน

4.7 การตามรอยอัลกอริทึมด้วยมือ (Hand - Tracing an Algorithm)

ขั้นวิกฤตในการออกแบบอัลกอริทึม คือการทวนสอบความถูกต้องของอัลกอริทึม ก่อนที่เราจะใช้เวลามากมายลงรหัสเวลาที่ใช้เพิ่มเพียงไม่กี่นาทีบ่อยครั้ง ทำให้ประหยัดเวลานับเป็นชั่วโมงๆ ของการลงรหัส และเวลาทดสอบโปรแกรม

ตามรอยด้วยมือหรือการตรวจปรู๊ฟ หมายถึง การจำลองแบบทำทีละขั้นตอนของอัลกอริทึมบนกระดาษ

Hand trace (desk check) is a step-by-step simulation. ดูว่าคอมพิวเตอร์จะกระทำการอัลกอริทึมนั้นอย่างไร

ขณะนี้เราจะตามรอยการกระทำการของอัลกอริทึมการแบ่งละเอียดสำหรับการแก้ปัญหา payroll problem ในหัวข้อ 4.6

Refined Algorithm

1. Display user instructions
2. Enter hours worked and hourly rate
3. Compute gross pay
   3.1 if no overtime then
       3.2 Compute gross pay without overtime pay
   else
       3.3 Compute gross pay with overtime pay
4. Compute net pay
   4.1 if no union dues then
       4.2 Net gets Gross
   else
       4.3 Net gets Gross - Dues
5. Display gross and net pay

ตาราง 4.7 คือ hand trace ของขั้นที่ 2 จนถึงขั้นที่ 5 ของอัลกอริทึมแต่ละขั้นตอน คือรายการทางซ้ายมือในอันดับของการกระทำการและผลของแต่ละขั้นตอนกำหนดในสถมภ์สุดท้าย ถ้าขั้นตอนนั้นเปลี่ยนแปลงค่าของตัวแปร ในตารางจะแสดงค่าใหม่ ถ้าไม่มีค่าใหม่แสดง ตัวแปรยังคงเก็บค่าเดิมไว้

ตัวอย่างเช่น ตารางแสดงให้เห็นขั้นตอน

2. Enter hours worked and hourly rate เก็บค่าข้อมูล 30.0 และ 10.00 ในตัวแปร Hours และ Rate ตามลำดับ : Gross และ Net ยังคง undefined (แสดงด้วยเครื่องหมาย ? ในแถวแรกของตาราง)

ตาราง 4,7 ตามรอยอัลกอริทึม สำหรับ Payroll Problem

Algorithm Step	Hours	Rate	Gross	Net	Effect
	?	?	?	?	
2. Enter hours and rate	30.0	10.0			Read the drta
3.1 if no over time					Hours < = 40
then					is true
3.2 Gross gets Hours * Rate			300.0		Compute gross
					with no over time
4.1 if no union dues then					Gross <= 100.00
					is false
4.3 Net gets Gross - Dues				275.0	Deduct union
					dues
5. Display Gross and Net					Display 300.00
					and 275.00

การตามรอยในตาราง 4.7 แสดงให้เห็นว่า 300.0 และ 275.0 เก็บใน Gross และ Net ตามลำดับ และแสดงผลการทวนสอบว่าอัลกอริทึมถูกต้อง เราต้องเลือกข้อมูลอื่นๆ ซึ่งทำให้สองเงื่อนไขประเมินผลกันกลุ่มที่แตกต่างของค่าของมัน เนื่องจากมีสองเงื่อนไข และแต่ละเงื่อนไขมีค่าเป็นไปได้สองค่า (True และ False) จึงมีการจัดกลุ่ม 2 คูณ 2 หรือ 4 กลุ่มที่แตกต่างกัน การตามรอยของอัลกอริทึมจบลงเมื่อแสดงให้เห็นแล้วว่าอัลกอริทึมทำงาน (works) ทุก combinations

นอกเหนือจากสี่กรณีซึ่งอภิปรายไปแล้วเราควรทวนสอบว่าอัลกอริทึมทำงานได้ถูกต้อง สำหรับข้อมูลไม่ปกติ (unusual data) ตัวอย่างเช่น จะเกิดอะไรขึ้นถ้า Hours มีค่าเท่ากับ 40.0 (ค่าของ MaxNoOvertime) หรือถ้า Gross มีค่าเท่ากับ 100.0 (ค่าของ MaxNo Due) อัลกอริทึมยังคงให้ผลลัพธ์ถูกต้องหรือไม่ เพื่อให้การตามรอยครบถ้วนบริบูรณ์เราควรแสดงให้เห็นว่าอัลกอริทึมจัดกระทำสถานการณ์เหล่านี้ได้อย่างถูกต้อง

โดยระมัดระวังเมื่อตามรอยแต่ละกรณี ทำให้มั่นใจว่าการกระทำการอัลกอริทึมเป็นเหมือนกับที่คอมพิวเตอร์กระทำการ บ่อยครั้งโปรแกรมเมอร์ สมมติเองว่าขั้นตอนเฉพาะจะกระทำการเช่นนั้น และไม่ทดสอบเงื่อนไขแต่ละกรณีและไม่ตามรอยแต่ละขั้นตอนอย่างชัดแจ้ง การตามรอยในวิธีนี้มีค่าเล็กน้อย

แบบฝึกหัด 4.7 (Self - Check)

1. จงจัดข้อมูลตัวอย่าง ซึ่งทำให้เงื่อนไขแรกในปัญหา payroll เป็นเท็จ และเงื่อนไขที่สองเป็นจริง และให้ตามรอยการกระทำการสำหรับข้อมูลเหล่านี้

2. ถ้า Hours = MaxHours และ Gross = MaxNoDues ขั้นตอนการกำหนดค่าชุดใดในอัลกอริทึม จะถูกกระทำจงตามรอย

4.8 ข้อความสั่ง Nested if และการตัดสินใจหลายทางเลือก (Nested if Statements and Multiple - Alternative Decisions)

เรามีข้อความสั่ง if เพื่อลงรหัสการตัดสินใจที่มีหนึ่งทางเลือก หรือสองทางเลือกมาแล้ว ขณะนี้เราจะใช้ข้อความสั่ง nested if (ข้อความสั่ง if หนึ่งคำสั่งอยู่ภายในข้อความสั่ง if อีกหนึ่งคำสั่ง) เพื่อลงรหัสการตัดสินใจแบบหลายทางเลือก

ข้อความสั่ง if ซ้อนใน หมายถึง ข้อความสั่ง if หนึ่งคำสั่ง ซึ่งมีข้อความสั่ง if อีกหนึ่งคำสั่งเป็นงานจริงของมันหรืองานเท็จของมัน (Nested if statement is an if statement with another if statement as its true task or its false task.)

ตัวอย่าง 4.11

ข้อความสั่ง nested if ข้างล่างนี้มีสามทางเลือก มันเพิ่มหนึ่งในตัวแปรสามตัว (NumPos, NumNeg หรือ NumZero) ด้วยเลข 1 ทั้งนี้ขึ้นอยู่กับว่า X มีค่ามากกว่าศูนย์, น้อยกว่าศูนย์หรือเท่ากับศูนย์ ตามลำดับ

กล่องสองกล่องแสดงโครงสร้างเชิงตรรกะ ของข้อความสั่ง nested - if

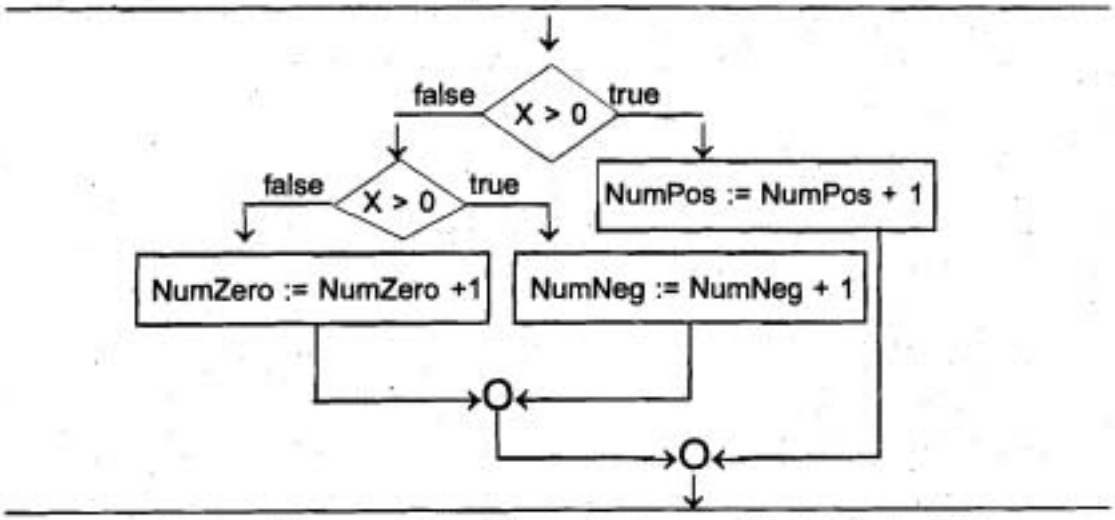
ข้อความสั่ง if ชุดที่สอง คือ งานที่เป็นเท็จ (ตามหลัง else) ของข้อความสั่ง if ชุดแรก

{increment NumPos, Num Neg, or NumZero depending on X}

```
if X. > 0 then
   NumPos := NumPos + 1
else
   If X < 0 then
      NumNeg := NumNeg + 1
   else {x = 0}
      NumZero := NumZero + 1
```

การกระทำการของข้อความสั่ง nested if กระทำดังนี้ : เงื่อนไขชุดแรก (X > 0) ถูกทดสอบ ถ้ามันเป็นจริง NumPos มีค่าเพิ่มขึ้นอีกหนึ่งและส่วนที่เหลือทั้งหมดหลัง else จะถูกข้าม

ถ้าเงื่อนไขชุดแรกเป็นเท็จ เงื่อนไขชุดที่สอง (X < 0) จะถูกทดสอบถ้ามันเป็นจริง, NumNeg จะมีค่าเพิ่มขึ้นอีกหนึ่ง



รูป 4.12 (a) ผังงานแสดงข้อความสั่ง nested - if สามทางเลือกของตัวอย่าง 4.11

กรณีอื่นๆ NumZero มีค่าเพิ่มขึ้นอีกหนึ่ง สิ่งที่สำคัญ คือเงื่อนไขชุดที่สอง จะถูกทดสอบเฉพาะเมื่อเงื่อนไขชุดแรกเป็นเท็จเท่านั้น

ตาราง 4.8 ตามรอยการกระทำการของข้อความสั่งนี้

เมื่อ X เท่ากับ -7 เพราะว่า X > 0 เป็นเท็จ, เงื่อนไขชุดที่สอง (X < 0) จึงถูกทดสอบ

ตาราง 4.8 ตามรอยข้อความสั่ง if ในตัวอย่าง 4.11 สำหรับ X = -7

Statement Part	Effect
If X > 0 then	-7 > 0 is false
If X > 0 then	-7 < 0 is true
NumNeg := NumNeg +1	add 1 to NumNeg

การเปรียบเทียบ Nested if และลำดับของ ifs (Comparison of Nested if and Seguence of ifs)

โปรแกรมเมอร์หัดใหม่บางครั้งชอบใช้ลำดับของข้อความสั่ง if มากกว่า ข้อความสั่ง nested if หนึ่งคำสั่ง

ข้อความสั่ง nested if ของตัวอย่าง 4.11 สมบูลเชิงตรรกะกับลำดับของข้อความสั่ง if ข้างล่างนี้

```
if  X > 0 then
    NumPos := NumPos + 1 ;
If X < 0 then
    Num Neg := Num Neg + 1 ;
If X = 0 then
    Num Zero := NumZero + 1
```

จะเห็นว่าข้อความสั่ง nested if อ่านง่ายกว่าและมีประสิทธิภาพมากกว่า อย่างไรก็ตาม เนื่องจากลำดับไม่ได้แสดงให้เห็นชัดเจนว่ามีเพียงหนึ่งข้อความสั่งกำหนดค่าเท่านั้นของข้อความสั่งกำหนดค่าสามชุด ซึ่งถูกกระทำการสำหรับ x เฉพาะหนึ่งค่าเหมือนเช่นที่ nested if ทำ

สำหรับเรื่องประสิทธิภาพ ข้อความสั่ง nested if กระทำการได้เร็วกว่า เมื่อ X เป็นค่าบวก เพราะว่าเงื่อนไขแรก (X > 0) เป็นจริง ดังนั้นส่วนของข้อความสั่ง if หลัง else แรกจะถูกข้าม ในทางตรงกันข้ามเงื่อนไขสามชุดทั้งหมดถูกทดสอบเสมอในลำดับของข้อความสั่ง if

เมื่อ X เป็นค่าลบ เงื่อนไขสองชุดถูกทดสอบใน nested if แต่เมื่อเปรียบเทียบกับลำดับของข้อความสั่ง if จะมีการทดสอบเงื่อนไขสามชุด

กฎของ Pascal สำหรับการจับคู่ else กับ if (Pascal Rule Lor Matching else with if)

ใช้การย่อหน้าเพื่อแสดงโครงสร้างเชิงตรรกะของข้อความสั่ง nested if การย่อหน้าในตัวอย่าง 4.11 แสดงข้อความสั่ง if - then - else ชัดเจน ว่าข้อความสั่ง if - then - else ชุดที่สองเป็นงานเท็จ (false task) ของชุดที่หนึ่ง คอมไพเลอร์ของ Pascal ไม่สนใจการย่อหน้าของโปรแกรมแต่มีกฎของมันเองใช้สำหรับจับคู่ else แต่ละตัวกับ if ซึ่งสมนัยกับมัน "Pascal จับคู่ else แต่ละตัวกับ if ตัวซึ่งอยู่หน้าตัวใกล้ที่สุด ซึ่งยังไม่ได้จับคู่กับ else ตัวใด" กฎนี้คล้ายกับกฎการจับคู่ วงเล็บเปิดและวงเล็บปิดในนิพจน์ นั่นคือ if คือวงเล็บเปิด และ else คือวงเล็บปิด

ตัวอย่าง 4.12

ในข้อความสั่ง nested if ข้างล่างนี้

{if - then - else as true task of an if - then}

```
if X > 0 then
    if Y > X then
        WriteLn ('Y > X > o')
    else
        WriteLn (' (X > 0) and (Y < = x)')
```

Pascal จับคู่ else กับ if ตัวที่สอง เพราะว่า Pascal แปลข้อความสั่งนี้เป็นข้อความสั่ง if - then ซึ่งงานจริง (ตามหลัง then) เป็นข้อความสั่ง if - then - else

ถ้าเราต้องการให้ else จับคู่กับ if ตัวแรก ไม่ใช่ if ตัวที่สอง เราต้องใส่ begin - end ปิดล้อมข้อความสั่ง if - then ชุดใน ดังนี้

```
{if - then as true task of if - then - else}
if X > 0 then
    begin
        if Y > X then
            WriteLn ('Y > X > 0')
    end
else
    WriteLn ('X < = 0')
```

Pascal แปลข้อความสั่งนี้เป็นข้อความสั่ง if - then - else เมื่อ true task คือ ข้อความสั่ง if -then

รูปแบบการตัดสินใจแบบหลายทางเลือกของ Nested if (Multiple - Alternative Decision Form of Nested if)

ข้อความสั่ง nested if จะมีความซับซ้อนมาก ถ้ามีมากกว่าสามทางเลือกและการย่อหน้าไม่ต้องกัน จึงอาจเป็นเรื่องยากในการกำหนดโครงสร้างเชิงตรรกะของข้อความสั่ง if ในสถานการณ์คล้ายตัวอย่าง 4.11 ซึ่งงานเท็จ (false task) แต่ละชุด ยกเว้นชุดสุดท้ายตามด้วย ข้อความสั่ง if - then - else เราสามารถลงรหัส nested if เป็นการตัดสินใจแบบหลายทางเลือก ซึ่งจะได้อธิบายต่อไป

การตัดสินใจแบบหลายทางเลือก หมายถึง ข้อความสั่ง nested if หนึ่งชุดซึ่งงานเท็จแต่ละชุด (ยกเว้นชุดสุดท้าย) เป็นข้อความสั่ง if - then - else (Multiple - alternative decision is a nested if statement which each false task (except possibly the last) is an if - then - else statement.)

Syntax Display

Multiple - Alternative Decision

Form :

```
if  condition, then
    statement1
else  if condition 2 then
    statemant2
else  if condition 3 then
    statement3
    •
    •
    •
else  if condition n then
    statementn
else
    statemento
```

ตัวอย่าง

```
{increment NumPos, NumNeg, NumZero depending on x}
  if x > 0 then
    Num Pos := Num Pos + 1
  else if x < 0 then
    Num Neg := Num Neg + 1
  else {x = 0}
    Num Zero := NumZero + 1
```

มีความหมายดังนี้ conditions ในการตัดสินใจหลายทางเลือก ถูกประเมินผลตามลำดับ จนกระทั่งมีเงื่อนไขที่เป็นจริง ถ้า condition เป็นจริง ข้อความสั่งที่ตามหลังมันจะถูกกระทำการและส่วนที่เหลือของการตัดสินใจหลายทางเลือก จะถูกข้าม (skipped) ถ้า condi-

tion เป็นเท็จ ข้อความสั่งที่ตามหลังจะถูกข้าม และเงื่อนไขถัดไปจะถูกทดสอบ ถ้าเงื่อนไขทั้งหมดเป็นเท็จ statement ซึ่งตามหลัง else ตัวสุดท้ายจะถูกกระทำการ

ข้อสังเกต ไม่จำเป็นต้องมี else statement ในกรณีนี้ถ้าเงื่อนไขทั้งหมดเป็นเท็จ ไม่มีอะไรเกิดขึ้น

ตัวอย่าง 4.13

สมมติว่าเราต้องการกำหนดเกรดเป็นตัวอักษรซึ่งขึ้นอยู่กับคะแนนสอบข้างล่างนี้

Exam scose	Grade Assigned
90 and above	A
80-89	B
70-79	C
60-69	D
below 60	F

การตัดสินใจหลายทางเลือกต่อไปนี้ แสดงผลเกรดเป็นตัวอักษร กำหนดตามข้อมูลในตารางนี้ สำหรับคะแนนสอบ 85 เงื่อนไขแรกที่เป็นจริงคือ Score > = 80 ดังนั้นแสดงผลเป็น B และการควบคุมโปรแกรมจะส่งไปยังข้อความสั่ง WriteLn หลังการตัดสินใจหลายทางเลือก

```
if  Score > = 90 then
    Write ('A')
else  if Score > = 80 then
    Write ('B')
else  if Score > = 70 then
    Write ('C')
else if Score > = 60 then
    Write ('D')
else
    Write ('F');
WriteLn (' is the exam grade - score is ', Score)
```

สไตล์ของโปรแกรม (Program Style)

การเขียนการตัดสินใจหลายทางเลือก (Writing a Multiple - Alternative Decision)

ในการตัดสินใจหลายทางเลือก คำสงวน else if และเงื่อนไขถัดไปเขียนให้อยู่ในบรรทัดเดียวกัน คำ else ทั้งหมดให้เขียนตรงกัน และแต่ละงานให้ย่อหน้าใต้เงื่อนไขซึ่งควบคุมการกระทำการของมัน

การเรียงอันดับของเงื่อนไขต่าง ๆ ในการตัดสินใจหลายทางเลือก (Order of Conditions in a Multiple - Alternative Decision)

เมื่อมีเงื่อนไขมากกว่าหนึ่งชุดในการตัดสินใจหลายทางเลือกเป็นจริงเฉพาะงานซึ่งตามหลังเงื่อนไขจริงชุดแรกเท่านั้นที่ถูกกระทำการเพราะฉะนั้นการเรียงอันดับของเงื่อนไขจึงมีผลกับผลลัพธ์ (outcome)

การตัดสินใจหลายทางเลือก ซึ่งกำหนดเกรดข้างล่างนี้เขียนไม่ถูกต้อง คะแนนสอบผ่านทั้งหมด (60 คะแนนหรือมากกว่าขึ้นไป) ถูกจัดให้เป็นเกรด D เพราะว่าเงื่อนไขแรกเป็นจริง และส่วนที่เหลือถูกข้าม ดังนั้น การเขียนข้อความสั่งที่ถูกต้องเงื่อนไขข้อจำกัดมากที่สุด (Score > = 90) ควรจะมาเป็นอันดับแรก

```
{incorrect grade assignment}
if Score > = 60 then
   Write ('D')
else if Score > = 70 then
   Write ('C')
else if Score > = 80 then
   Write ('B')
else if Score > = 90 then
   Write ('A")
else
   Write ('F")
```

ตัวอย่าง 4.14

เราอาจใช้การตัดสินใจหลายทางเลือกเพื่อ implement ตารางการตัดสินใจ ซึ่งอธิบายพิสัยหลายชุดของค่าต่างๆ สำหรับตัวแปรเฉพาะหนึ่งตัว และผลลัพธ์ (outcome)

ของแต่ละพิสัย ตัวอย่างเช่น สมมติว่าเราเป็นนักบัญชี จัดวางระบบบัญชีเงินเดือน ตามตาราง 4.9 ซึ่งแสดงพิสัยที่แตกต่างกันห้าระดับสำหรับเงินเดือนมากจนถึง $15,000.00 แต่ละบรรทัดในตารางแสดงภาษีขั้นต่ำ (สดมภ์ที่ 2) และร้อยละของภาษี (สดมภ์ที่ 3) สำหรับพิสัยเงินเดือนหนึ่งระดับ (สดมภ์ที่ 1) กำหนดเงินเดือนของบุคคลหนึ่งคน จงคำนวณภาษีโดยการบอกภาษีขั้นต่ำกับผลคูณของ ร้อยละภาษีกับเงินเดือนเฉพาะส่วนที่เกินเงินเดือนขั้นต่ำของพิสัยช่วงนั้น ตัวอย่างเช่น บรรทัดที่สองของตารางกำหนดว่า ภาษีของเงินเดือน $2,000.00 เท่ากับ $225.00 บวกกับ 16% ของเงินเดือนส่วนที่เกิน $1,500.00 (นั่นคือ 16% ของ $500.00 เท่ากับ $80.00) เพราะฉะนั้นภาษีทั้งหมดเท่ากับ $225.00 บวก $80.00 หรือเท่ากับ $305.00

ตาราง 4.9 ตารางการตัดสินใจสำหรับ ตัวอย่าง 4.14

Salary Range	Base Tax	Percentage of Excess
0.00 - 1499.99	0.00	15%
1500.00 - 2999.99	225.00	16%
3000.00 - 4999.99	465.00	18%
5000.00 - 7999.99	825.00	20%
8000.00 - 15,000.00	1425.00	25%

ตาราง 4.10 ตามรอยข้อความสั่ง if ในรูป 4.13 สำหรับเงินเดือนเท่ากับ $2000.00

Statement Part	Salavy	Tax	Effect
	2000.0	?	
if Salary < 0.0			2000.0 < 0.0 is false
else if Salary < 1500.00			2000.0 < 1500.0 is false
else if Salaey < 3000.00			2000.00 < 3000.0 is true
Tax := (Salary - 1500.00)			Evaluates to 500.00
* 0.16			Evaluates to 80.00
+ 225.00		305.00	Evaluates to 305.00

```
if    Salary < 0.00 then
      WriteLn ('Error! Negative salary $ ' , Salary : 10 : 2)
else  if Salary < 1500.00 then       {first range}
      Tax := 0.15 * Salary
else  if Salary < 3000.00 then       {second range}
      Tax := (Salary - 1500.00) * 0.16 + 225.00
else  if Salary < 5000.00 then       {third range}
      Tax := (Salary - 3000.00) * 0.18 + 0.18 + 465.00
else  if Salary < 8000.00 then       {fourth range}
      Tax := (Salary - 5000.00) * 0.20 + 825.00
else  if Salary < = 15000.00 then    {fifth range}
      Tax := (Salary - 8000.00) * 0.25 + 1425.00
else
      WriteLn ('Error! Too large salary $', Salary : 10 : 2)
```

รูป 4.13 ข้อความสั่ง if สำหรับตาราง 4.9

สไตล์ของโปรแกรม (Program style)

การตรวจสอบความถูกต้องค่าของตัวแปร

ถ้าเราตูความสมเหตุสมผล (validating) ค่าของตัวแปร ก่อนใช้ตัวแปรในการคำนวณ เราสามารถหลีกเลี่ยงการประมวลผลข้อมูลไม่ถูกต้อง หรือข้อมูลไม่มีความหมายแทนที่จะคำนวณภาษีไม่ถูกต้อง ข้อความสั่ง if ในรูป 4.13 พิมพ์ข้อความระบุความผิดพลาด (error message) ถ้าค่าของ Salary อยู่ภายนอกพิสัยที่เขียนในตาราง (0.0 ถึง 15000.00) เงื่อนไขชุดแรกตรวจพบเงินเดือนเป็นค่าลบ ข้อความระบุความผิดพลาดจะแสดงผลถ้า Salary มีค่าน้อยกว่าศูนย์ทุกเงื่อนไขประเมินผลเป็น False ทั้งหมด ถ้า Salary มีค่ามากกว่า 15000.00 ดังนั้นงานหลัง else แสดงผลข้อความระบุความผิดพลาด

ข้อความสั่ง if ที่มีตัวแปรมากกว่าหนึ่งตัว (Nested if Statements with More Than One Variable)

ตัวอย่างของเราส่วนใหญ่ ใช้ข้อความสั่ง nested if เพื่อทดสอบค่าของหนึ่งตัวแปร เพื่อเขียนข้อความสั่ง nested if แต่ละชุดเป็นการตัดสินใจแบบหลายทางเลือก

เมื่อมีตัวแปรหลายตัวเกี่ยวข้องกันเราไม่สามารถใช้การตัดสินใจแบบหลายทางเลือกได้ ตัวอย่าง 4.15 มีสถานการณ์ ซึ่งเราสามารถใช้ข้อความสั่ง nested if เป็นตัวกรอง (filter) เพื่อเลือกข้อมูลซึ่งเป็นไปตามเกณฑ์ที่แตกต่างกันหลายชุด

ตัวอย่าง 4.15

กระทรวงกลาโหม ต้องการโปรแกรมซึ่งจำแนกผู้ชายโสดอายุระหว่าง 18 ปี ถึง 26 ปี วิธีหนึ่งคือใช้การซ้อนในของข้อความสั่ง if - then ซึ่งมีเงื่อนไขทดสอบเกณฑ์ถัดไปเฉพาะเมื่อเกณฑ์ก่อนหน้านั้นทั้งหมดผ่าน ข้อความสั่ง if - then ซ้อนในต่อไปนี้ สมมติว่าตัวแปรทั้งหมดมีค่าเริ่มต้น และตัวแปรแบบบูล Single มีเซตก่อนหน้าแสดงว่าแต่ละคนเป็นโสด (Single เป็น True) หรือไม่ ข้อควรจำเงื่อนไข Single ควรเขียนเป็น Single = True

ข้อความสั่ง WriteLn จะกระทำการก็ต่อเมื่อเงื่อนไขทั้งหมดเป็นจริงเท่านั้น

```
{Display message if all criteria are met.}
if  Single then
   if  Gender = 'M' then
      if  (Age > = 18) and (Age < = 26) then
         WriteLn ('Current person satisfies the criteria'.)
```

อีกวิธีหนึ่งในการแก้ปัญหานี้ คือเขียนนิพจน์แบบบูล ซึ่งรวมเงื่อนไขแต่ละชุดทั้งหมดเข้าด้วยกัน ซึ่งต้องเป็นจริงและใช้ตัวดำเนินการ and นิพจน์ชุดนี้อยู่ทางขวามือของการกำหนดค่าแบบบูล ขัดความสั่ง if ตามด้วยการกำหนดค่าแบบบูล แสดงผลข้อความที่เหมาะสม โดยขึ้นอยู่กับค่าที่กำหนดให้

```
AllMet
   {Set AllMet to True if all criteria are met.}
   AllMet :=  Single and (Gender = 'M') and
                  (Age > = 18) and (Age L <= 26);
   {Display the result of the filtering operation.}
```

```
if  AllMet then
    WriteLn ('Current person satisfies the criteria.')
else
    WriteLn ('All criteria are not satisfied.')
```

ตัวอย่าง 4.16

ในการถกเถียงกันเกี่ยวกับการเลือกวิทยาลัยเข้าเรียนของเรา พ่อแม่บอกว่าให้เราสมัครโรงเรียนซึ่งเราเลือกเป็นอันดับหนึ่ง ถ้าคะแนน SAT ของเราสูงกว่า 1300 และตลอดภาคฤดูร้อน เราทำงานได้เงินมากกว่า $2000 ถ้าคะแนน SAT ไม่มากกว่า 1300 แต่เรายังทำงานได้เงินมากกว่า $2000 พ่อแม่แนะนำว่าให้เราสมัครโรงเรียนเก่าของพ่อแม่ และให้พักในหอพักของวิทยาลัย ถ้าเราไม่สามารถได้เงิน $2000 พ่อแม่อยากให้เราเดินทางไป-กลับ เรียนวิทยาลัยท้องถิ่น ข้อความสั่ง nested if ข้างล่างนี้คือข้อสรุปกระบวนการตัดสินใจนี้

```
if  Earnings > 2000.00     then
    if  SAT > 1300 then
        WriteLn ('Apply to first - choice college')
    else
        WriteLn ('Apply to parents alma mater')
else
    WriteLn ('Apply to local college')
```

ในที่นี้ WriteLn ชุดแรกกระทำการเมื่อเงื่อนไขทั้งคู่เป็น True

WriteLn ชุดที่สอง กระทำการเมื่อเงื่อนไขแรกเป็น True และเงื่อนไขที่สองเป็น False

WriteLn ชุดที่สาม กระทำการเมื่อเงื่อนไขแรกเป็น False

เราสามารถใช้การตัดสินใจแบบหลายทางเลือก ข้างล่างนี้เพื่อ implement การตัดสินใจครั้งนี้

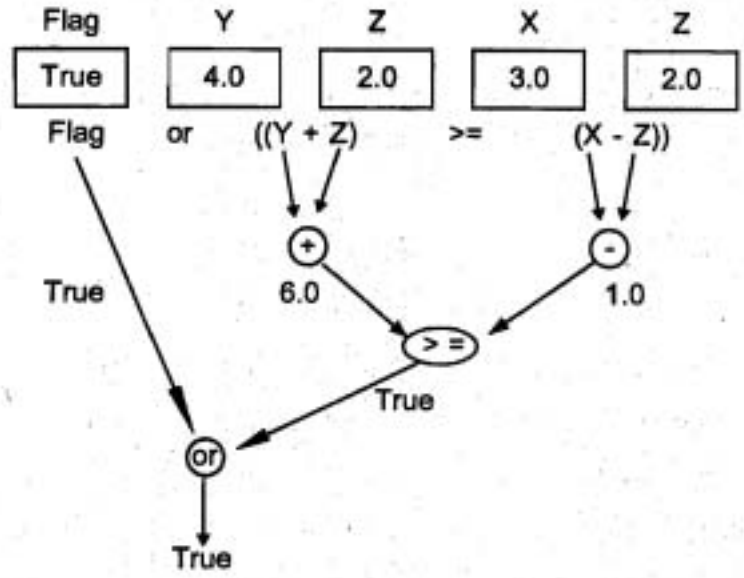
```
if    (Earnings > 2000.00) and (SAT > 1300) then
      WriteLn ('Apply to first - choice college')
else  if Earnings > 2000.00 then
      WriteLn ('Apply to parents alma mater')
else
      WriteLn ('Apply to local college')
```

ในที่นี้ WriteLn ชุดแรกกระทำการเมื่อเราทำงานภาคฤดูร้อนและคะแนน SAT เป็นที่น่าพอใจ เนื่องจากเงื่อนไขหลัง else if ทดสอบเฉพาะเมื่อเงื่อนไขแรกเป็นเท็จ นั่นหมายความว่าการทำงานเป็นที่พอใจแต่คะแนน SAT ไม่ผ่าน โปรดสังเกตว่าไม่จำเป็นต้องทดสอบค่า SAT ในเงื่อนไขนี้

WriteLn ชุดที่สามกระทำการ เมื่อการทำงานภาคฤดูร้อนไม่ได้เงินเพียงพอ

การประเมินผลวิธีลัดของนิพจน์แบบบูล (Short - Circuit Evaluation of Boolean Expressions)

เมื่อประเมินผลแบบบูล คอมไพเลอร์บางตัวทำงาน โดยใช้เทคนิคเรียกว่า การประเมินผลวิธีลัด คอมไพเลอร์ใช้เทคนิคนี้หยุดการประเมินผล นิพจน์แบบบูล ทันทีที่ค่าของมันสามารถกำหนดได้ ตัวอย่างเช่น ถ้าค่าของ Flag เป็น True นิพจน์ ในรูป 4.14 ต้องถูกประเมินผลให้เป็น True โดยไม่ต้องสนใจค่าของนิพจน์ภายในวงเล็บ ที่ตามหลัง or (นั่นคือ True of (...) ต้องเป็นจริงเสมอ) ด้วยเหตุนี้จึงไม่จำเป็นต้องประเมินผลนิพจน์ในวงเล็บที่ตามหลัง or เมื่อ Flag เป็น True ในทำนองเดียวกัน เราสามารถแสดงให้เห็นว่า False and (...) ต้องเป็น False เสมอ ดังนั้นจึงไม่จำเป็นต้องประเมินผลนิพจน์ในวงเล็บซึ่งตามหลังตัวดำเนินการ and โดยปริยาย Turbo Pascal ใช้การประเมินผลวิธีลัดของนิพจน์แบบบูลแต่ Standard Pascal ไม่ใช้



รูป 4.14 ต้นไม้การประเมินผล สำหรับนิพจน์ Flag or ((Y + Z) >= (x - Z))

การประเมินผลวิธีลัด หมายถึง การประเมินผลของนิพจน์แบบบูลมากเท่าที่จำเป็นเท่านั้น เพื่อหาค่าของมัน (Short - circuit evaluation is the evaluating only as much of a Boolean expression as is necessary to determine its value.)

ตัวอย่าง 4.17

.ถ้า X เป็นศูนย์เงื่อนไข if

if (X < > 0.0) and (Y / X > 5.0) then

เป็น False เพราะว่า (X < > 0.0) เป็น False ดังนั้น

False and (...) ต้องเป็น False

ด้วยเหตุนี้ จึงไม่จำเป็นต้องประเมินผล (Y / X > 5.0) เมื่อ X เป็นศูนย์ ถ้านิพจน์นี้ถูกประเมินผลอย่างไรก็ตาม การหารด้วยศูนย์ จะเกิดข้อผิดพลาดเวลาดำเนินการ (run-time error) เพราะว่าตัวหาร X เป็นศูนย์

เพื่อป้องกันข้อผิดพลาดเวลาดำเนินงานเช่นนี้ สำหรับโปรแกรมที่คอมไพล์ด้วยคอมไพเลอร์ Pascal และไม่ได้ใช้การประเมินผลนิพจน์แบบบูลวิธีลัด เงื่อนไข if ควรแปงเป็นดังนี้

```
if  (X < > 0.0) then
   if (Y / X > 5.0) then
```

เงื่อนไขแรกคุม (guards) เงื่อนไขที่สองและป้องกันเงื่อนไขที่สอง จากการประเมินผลเมื่อ X เป็นศูนย์ ผลลัพธ์ของการประเมินผลเงื่อนไขเหล่านี้ เหมือนกับวิธีลัดหรือการประเมินผลแบบบริบูรณ์ให้ระวังการประเมินผลวิธีลัด และหลักเลี่ยงการเขียนนิพจน์แบบบูลซึ่งปฏิบัติตามนั้น ถ้าเราใส่คอมเมนต์พิเศษ {$B+} ในโปรแกรมคอมไพเลอร์ของ Turbo Pascal จะสร้างรหัสซึ่งจะประเมินผลนิพจน์แบบบูลทั้งหมดอย่างบริบูรณ์ คอมเมนต์ซึ่งขึ้นต้นด้วยสัญลักษณ์ {$ เรียกว่า ตัวชี้แนะคอมไพเลอร์ (compiler directive) เพราะว่ามันจัดหาคำสั่งต่างๆ ให้คอมไพเลอร์

ตัวชี้แนะคอมไพเลอร์ {$B+} ต้องอยู่ก่อนนิพจน์แบบบูลชุดแรก ซึ่งต้องการให้ประเมินผลอย่างบริบูรณ์ เมื่อทำงานแล้วการประเมินผลแบบบูลอย่างบริบูรณ์ ยังคงมีผลอยู่จนกระทั่งพบคอมเมนต์ {$B-}

Syntax Display

ตัวชี้แนะคอมไพเลอร์ การประเมินผลแบบบูล (Boolean Evaluation Compiler Directive)

Form : | {$B-} หรือ {$B+} |

Default : {$B-}

มีความหมายดังนี้สถานะโดยปริยาย (defaulf state) Turbo Pascal ใช้การประเมินผลแบบบูลวิธีลัด ตัวชี้แนะคอมไพเลอร์ {$B+} ทำให้คอมไพเลอร์ สร้างรหัสสำหรับการประเมินผลอย่างบริบูรณ์ของตัวถูกดำเนินการทุกตัวของนิพจน์แบบบูล

โปรดสังเกตว่า การเขียนนิพจน์แบบบูลซึ่งอาจไม่ประสบผลสำเร็จ เมื่อใช้การประเมินผลอย่างบริบูรณ์ แทนที่จะเป็นการประเมินผลวิธีลัด ซึ่งไม่ใช่เป็นการฝึกปฏิบัติของการเขียนโปรแกรมที่ดี

แบบฝึกหัด 4.8 (Self - Check)

1. จงตามรอย (Trace) การกระทำการของข้อความสั่ง nested if ในตัวอย่าง 4.14 เมื่อ Salary คือ 13500.00

2. ในตัวอย่าง 4.16 มีการเปรียบเทียบกี่ครั้งที่ต้องใช้กระทำการข้อความความสั่ง if ชุดแรก และมีการเปรียบเทียบกี่ครั้ง เพื่อกระทำการ if ชุดที่สองให้บอกว่าข้อความสั่ง if ชุดใดมีประสิทธิภาพมากกว่า

3. จงประเมินผลนิพจน์ข้างล่างนี้ โดยใช้และไม่ใช้การประเมินผลวิธีลัด ถ้า X คือ 6 และ Y คือ 7

a) (X > 5) and (Y div X < = 10)

b) (X < = 10) or (X / (Y - 7) > 3)

เขียนโปรแกรม

1. จงเขียนข้อความสั่ง if ของตัวอย่าง 4.13 ใหม่ โดยใช้เฉพาะตัวดำเนินการสัมพันธ์ < ในทุกเงื่อนไข และในขั้นแรกให้ทดสอบการล้มเหลว การทดสอบเกรด

2. จง implement ตารางการตัดสินใจต่อไปนี้ โดยใช้ข้อความสั่ง nested if สมมติว่าค่าเฉลี่ยเกรด (grade point average) อยู่ภายในพิสัย 0.0 ถึง 4.0

Grade Point Average	Transcript Message
0.00 - 0.99	Failed semester - registration suspended
1.00 - 1.99	On probation for next semester
2.00 - 2.99	(No message)
3.00 - 3.49	Dean's list for semester
3.50 - 4.00	Highest honors for semester

3. จงเขียนโปรแกรมเพื่อโยนลูกเต๋าสองลูกแล้วใช้ตัวสร้างเลขสุ่ม (random number generator) เพื่อให้เกิดค่าที่เป็นไปได้ 6 ค่าของการโยนแต่ละครั้ง และพิมพ์ You Win! ถ้าโยนได้เลขรวมกันเท่ากัน 7 หรือ 11 พิมพ์ Snake Eyes! ถ้าโยนได้ 2 กรณีอื่นๆ พิมพ์ Try Again. (ข้อแนะนำ ใช้ Random (6) + 1 เพื่อกำหนดการโยนลูกเต๋าหนึ่งลูก)

4.9 ข้อความสั่ง case (The case Statement)

นอกเหนือจากข้อความสั่ง if แล้ว ใน Pascal เราสามารถใช้ข้อความสั่ง case เพื่อเลือกหนึ่งทางเลือกจากหลายทางเลือกข้อความสั่ง case มีประโยชน์โดยเฉพาะเมื่อการเลือกขึ้นอยู่กับค่าของหนึ่งตัวแปรหรือนิพจน์อย่างง่าย (เรียกว่า case selector) ตัวเลือก case ต้องเป็นข้อมูลชนิด ordinal data type หรือแบบชนิดข้อมูล ซึ่งค่าของมันทั้งหมดมีอันดับการเขียนรายการได้ แบบชนิดข้อมูล Integer, Boolean และ Char เป็นชนิด ordinal type แต่ Real และ String ไม่ใช่ข้อมูลชนิด ordinal type

แบบชนิดข้อมูล or dinal หมายถึง แบบชนิดข้อมูล ซึ่งมีเซตจำกัดของค่าต่างๆ ที่สามารถเรียงอันดับรายการจากตัวแรกจนถึงตัวสุดท้ายได้เสมอ (Ordinal data type is a data type having a finite set of values that can always be listed in order from the first to the last.)

ตัวอย่าง 4.18

ข้อความสั่ง case

```
case  MomOrDad of

   'M', 'm' : WriteLn ('Hello Mom - Happy Mother' 's Day');

   'D', 'd' : WriteLn ('Hello Dad - Happy Father' 's Day')

end; {case}
```

มีความหมายเหมือนกับข้อความสั่ง if ข้างล่างนี้เมื่อตัวอักขระ ซึ่งเก็บใน MomOrDad เป็นหนึ่งในรายการตัวอักษรสี่ตัว (M, m, D, d)

```
if  (MomOrDad = 'M') or (MomOrDad = 'm') then
   WriteLn ('Hello Mom - Happy Mother ' ' s Day')
else if (MomOrDad = 'D') or (MomOrDad = 'd' then
   WriteLn ('Hello Dad - Happy Father ' ' s Day')
```

การแสดงผลข้อความโดยข้อความสั่ง case ขึ้นอยู่กับค่าของตัวเลือก case MomOrDad (ชนิด Char) ถ้าตัวเลือก case มีค่าเป็น 'M' หรือ 'm' ข้อความชุดแรกถูกแสดงผล ถ้าตัวเลือก case มีค่าเป็น 'D' หรือ 'd' ข้อความชุดที่สองถูกแสดงผล รายชื่อ 'M', 'm' และ 'D', 'd' เรียกว่า case labels

ตัวอย่าง 4.19

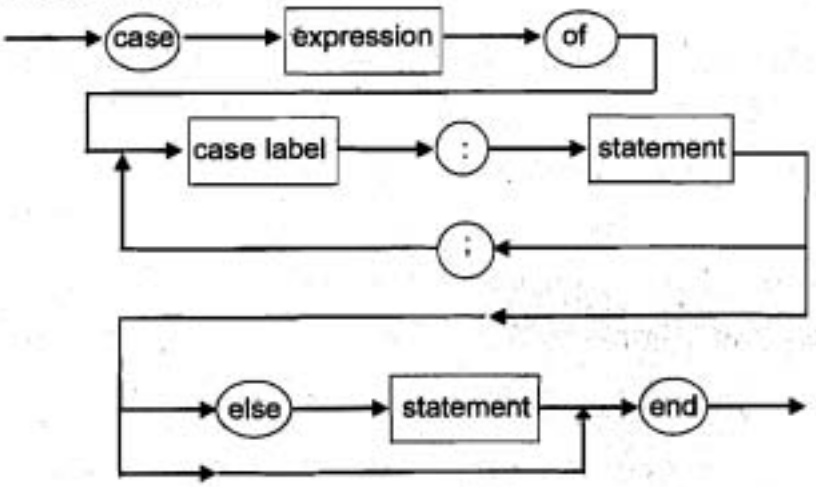
ข้อความสั่ง case ข้างล่างนี้คำนวณเงินได้รวม (gross pay) ของพนักงานหนึ่งคน ซึ่งทำงานได้เงินในหนึ่งวัน เมื่อค่าของ DayNumber แสดงว่าวันนั้นเป็นวันเสาร์ (DayNumber คือ 7) วันอาทิตย์ (DayNumber คือ 1) หรือเป็นวันทำงานปกติ (DayNumber คือ 2 ถ้า 6) หรือไม่ สำหรับวันหยุดคนงานได้รับค่าจ้างเป็นหนึ่งเท่าครึ่งของวันทำงานปกติ ทั้งนี้ค่าของ DayNumber, DailyRate และ Hours ต้องถูกนิยามก่อนการกระทำข้อความสั่ง case

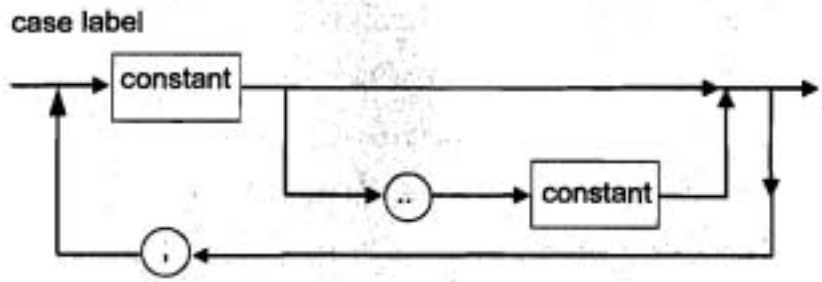
```
{Compute gross pay for a particular day}
case DayNumber of
1, 7            : Gross := Hours * 1.5 * DailyRate ;
2, 3, 4, 5, 6   : Gross := Hours * DailyRate
end : {case}
```

ข้อผิดพลาดร่วมอย่างหนึ่ง คือ การใช้สายอักขระ เช่น 'Saturday' หรือ 'Sunday' เป็น case label สิ่งนี้เป็นเหตุให้เกิดข้อผิดพลาดวากยสัมพันธ์ ordinal expression expected โปรดจำไว้ว่า เฉพาะค่าเชิงอันดับที่ (Ordinal values) เท่านั้น (ได้แก่ single characters, integers หรือ booleam values) ที่เป็น case label เราอธิบายข้อความสั่ง case ต่อไปและแผนภาพของมันในรูป 4.15

---

case statement





รูป 4.15 แผนภาพวากยสัมพันธ์ สำหรับข้อความสั่ง case และ case label

Syntax Display

Case Statement

Form :

```
case selector of
   label1,: statement1 ;
   label2 : statement2 ;
          :
   labeln : statementn
else
   statement*
end {case}
```

ตัวอย่าง

```
case N of
   1, 2  :  begin
               Write (' 1, 2') ;
               WriteLn ('Buckle my shoe')
            end ; {1,2}
   3, 4  :  WriteLn ('3, 4, shut the door') ;
   5, 6  :  WriteLn ('5, 6, Pick up sticks')
else
   WriteLn (N : 1, ' is out of range')
end; {case}
```

มีความหมายดังนี้ นิพจน์ selector ถูกประเมินผลและเปรียบเทียบกับ label แต่ละตัว ซึ่งอาจจะเป็นรายการของค่าที่เป็นไปได้หนึ่งค่า หรือมากกว่าหนึ่งค่าสำหรับ selector และจะมีเพียงหนึ่งข้อความสั่งเท่านั้น ที่ถูกกระทำการ ถ้าค่าของ selector อยู่ในรายการ label$_i$ จากนั้น statement$_i$ จะถูกกระทำการและหลังจากนั้นการควบคุมส่งไปยังข้อความสั่งแรกหลัง end {case} ข้อความสั่งแต่ละชุดอาจเป็นข้อความสั่ง Pascal หนึ่งคำสั่ง หรือข้อความสั่งประกอบหนึ่งคำสั่ง

ข้อสังเกต

(1) ถ้าค่าของ selector ไม่อยู่ในรายการของ label ชุดใดเลยไม่มีข้อความสั่งชุดใดถูกกระทำการ ยกเว้น ส่วนที่เป็น else clause เท่านั้น ถ้ามี else clause (statement$_n$) จะถูกกระทำการ สำหรับ standard Pascal ไม่มีส่วน else clause จึงเกิดข้อผิดพลาดเวลาดำเนินงาน (run-time error) เมื่อค่าของ selector ไม่จับคู่กับ label ใดๆ

(2) ค่า selector หนึ่งค่าจะปรากฏอย่างมากที่สุดหนึ่ง label

(3) ชนิดของค่า selector แต่ละตัวต้องสมนัยกับชนิดของนิพจน์ selector

(4) แบบชนิดข้อมูลเชิงดันดับที่ (any ordinal data type) จึงจะเป็นชนิดของ selector ได้

ถ้าไม่มี action ใดๆ ถูกกระทำสำหรับ case label นั้นเครื่องหมาย semicolon จะอยู่ถัดจากเครื่องหมาย colon แต่ละ statement ยกเว้น statement สุดท้ายต้องตามด้วยหนึ่ง semicolon ข้อความสั่งสุดท้ายให้ตามด้วย end

else Clause

จะเกิดอะไรขึ้นถ้าผู้ใช้โปรแกรมใส่หนึ่งค่า ซึ่งไม่อยู่ในรายการใน case label สำหรับ standard Pascal จะเกิดข้อผิดพลาดเวลาดำเนินงานแต่ใน Turbo Pascal ไม่เกิดอะไรขึ้น และข้อความสั่งถัดไปซึ่งตามหลัง end {case} จะถูกกระทำการ Turbo Pascal อนุญาตให้ใช้ else clause ซึ่งจะแสดงข้อความระบุข้อผิดพลาด หรือ การกระทำที่ถูกต้องถ้า case selector มีค่าข้อมูลซึ่งไม่คาดคิด ในตัวอย่างถัดไปของข้อความสั่ง case ส่วนที่เป็น else clause แสดงผลข้อความระบุความผิดพลาด invalid day number ถ้าค่าซึ่งไม่คาดคิดเก็บใน DayNumber ใน standard Pascal ไม่มีการใช้ else clause

```
{Compute gross pay for a particular day.}
case DayNumber of
   1, 7          : Gross := Hours * 1.5 * DailyRate;
   2, 3, 4, 5, 6 : Gross := Hours * DailyRate
else
   WriteLn ('invalid day number.')
End {case}
```

สัญกรณ์พิสัยย่อยใน case Labels (Subrange Notation in case Labels)

ใน Turbo Pascal ไม่เหมือน standard Pascal ตรงที่เราสามารถแสดงรายการพิสัยของค่าต่างๆ ใน case label โดยใช้สัญกรณ์พิสัยย่อย ในข้อความสั่ง case ถัดไปแสดงให้เห็นว่าเราไม่จำเป็นต้องแสดงรายการค่าที่สืบเนื่องทุกตัว จาก 2 ถึง 6 ใน label ชุดที่สอง เราแสดงรายการด้วยค่าเริ่มต้น และค่าสุดท้าย คั่นด้วยสองจุด (เราจะอภิปรายแบบชนิดข้อมูล subrange ในบทที่ 7)

```
{Compute gross pay for a particurar day.}
case DayNumber of
   1, 7 : Gross := Hours * 1.5 * DailyRate;
   2..6 : Gross : = Hours * DailyRate
end {case}
```

สัญกรณ์พิสัยย่อย ใช้ min..max เพื่อแสดงพิสัยเชิงอันดับที่จาก min ถึง max (Subrane notation wsing min..max to indicate the ordinal range from min through max.)

การเฝ้าระวังข้อความสั่ง case (Guarding a case statement)

ใน Standard Pascal บ่อยครั้งที่โปรแกรมเมอร์เฝ้าระวัง ข้อความสั่ง case ด้วยข้อความสั่ง if เพื่อป้องกันการเกิดข้อผิดพลาด case expression out of range ในส่วนของโปรแกรมถัดไปข้อความสั่ง if เฝ้าระวัง ข้อความสั่ง case ซึ่งซ้อนในอยู่ใน if ข้อความสั่ง case จะกระทำการ เฉพาะเมื่อค่าของ DayNumber อยู่ในพิสัย 1 ถ้า 7 เท่านั้นตามต้องการ และพิมพ์ข้อความระบุข้อผิดพลาด เมื่อ DayNumber ไม่ถูกต้อง โปรดสังเกตว่าไม่มีเครื่องหมาย semicolon ตามหลัง end {case}

```
{Compute gross pay for a particular day.}
if (DayNumber > = 1) and (DayNumber < = 7) then
   case DayNumber of
      1,7 : Gross := Hours * 1.5 * DailyRate;
      2,3,4,5,6 : Gross := Hours * DailyRate
   end {case}
else
   WriteLn ('invalid day number.')
```

การเปรียบเทียบ Nested if และ case (Comparison of Nested if and case)

.ข้อความสั่ง nested if ซึ่งมีลักษณะทั่วไปมากกว่าข้อความสั่ง case และสามารถใช้ implement การตัดสินใจแบบหลายทางเลือก ชนิดใดๆ ก็ได้ แต่ข้อความสั่ง case อ่านง่ายกว่า และอาจนำมาใช้เมื่อใดก็ตามเมื่อปฏิบัติงานโปรดจำไว้ว่าค่าชนิด Real หรือ string นำมาใช้เป็น case label ไม่ได้

ให้ใช้ข้อความสั่ง case เมื่อ case label แต่ละชุดประกอบด้วย ขนาดที่สมเหตุสมผลรายการของค่าต่างๆ (10 หรือน้อยกว่า 10) เมื่อจำนวนของค่าต่างๆ ใน case selector มีขนาดใหญ่ หรือมีช่องว่างมาก (large gaps) ในค่าต่างๆ เหล่านี้ ให้ใช้ข้อความสั่ง nested if

สไตล์ของโปรแกรม (Program Style)

ตัวเลือก case ชนิดบูลีน (Type Boolean case Selectors)

ถึงแม้ว่าข้อความสั่ง case อาจมี case selectors เป็นชนิด Boolean ได้ แต่สิ่งนี้ไม่ค่อยจะเกิดขึ้น ข้อความสั่ง case และข้อความสั่ง if ข้างล่างนี้ ทั้งสองชุดสมมูลกัน (equivalent) แต่น่าจะใช้ข้อความสั่ง if มากกว่า

```
case x = y of                    |  if x = y then
   True : WriteLn ('Equal');     |     WriteLn ('Equal')
   False : WriteLn ('Unequal');  |  else
end {case}                       |     WriteLn ('Unequal')
```

แบบฝึกหัด 4.9 Self - Check

1. จงเขียนข้อความสั่ง if ซึ่งสมนัยกับข้อความสั่ง case ต่อไปนี้

```
case X of
   2      : WriteLn ('Snake Eyes ! ')
   7, 11 : WriteLn ('Win!')
else
   WriteLn ('Try again.')
end {case}
```

2. จงเขียนข้อความสั่ง case ซึ่งสมนัยกับข้อความสั่ง if ข้างล่างนี้

```
if  (Grade > = 'A') and (Grade < = 'C') then
   WriteLn ('Passing')
else if (Grade = 'D') or (Grade = 'F') then
   WriteLn ('No credit')
else
   WriteLn ('Invalid grade')
```

3. ข้อความสั่ง nested if ในหัวข้อ 4.8 สามารถเขียนใหม่โดยใช้ข้อความสั่ง case ได้หรือไม่ ถ้าไม่ได้ ให้อธิบาย

เขียนโปรแกรม (Programming)

1. จงเขียนข้อความสั่ง case ให้พิมพ์ข้อความเพื่อแสดงว่า NextCh (type Char) เป็นสัญลักษณ์ของตัวดำเนินการ (+, -, = , <, >), เครื่องหมายกำกับวรรคตอน (comma, Semicolon, parenthesis, brace, bracket) หรือเลขโดด (digit) หรือไม่ ข้อความสั่งนี้ให้พิมพ์การเลือกชนิดด้วย

2. จงเขียนข้อความสั่ง nested if ซึ่งสมมูลกับข้อความสั่ง case ซึ่งอธิบายในแบบฝึกหัดข้อ (1)

3. ให้เฝ้าระวัง (guard) ข้อความสั่ง case ซึ่งอธิบายในแบบฝึกหัดข้อ (1)

4.10 ข้อผิดพลาดร่วมของการเขียนโปรแกรม (Common Programming Errors)

ตัวดำเนินการแบบบูล and, or และ not ใช้ได้เฉพาะกับนิพจน์แบบบูลเท่านั้น ในนิพจน์

```
Flag and (X = Y)
```

ในที่นี้ ตัวแปร Flag ต้องเป็นชนิด Boolean ตัวอย่างนี้นิพจน์จะไม่ถูกต้อง (invalid) ถ้าไม่มีวงเล็บกำกับ ยกเว้นเมื่อ X และ Y เป็นชนิด Boolean ด้วย เมื่อไม่มีวงเล็บกำกับ Flag and X จะถูกประเมินผลเป็นอันดับแรก เพราะว่า ตัวดำเนินการ and มีการทำก่อน (precedence) สูงกว่า =

โปรดระวัง เมื่อใช้เครื่องหมาย semicolons ภายในข้อความสั่ง if เราใช้เครื่องหมาย semicolons เพื่อคั่น ข้อความสั่งต่างๆ ของข้อความสั่งประกอบภายในข้อความสั่ง if เท่านั้น และใส่ semicolon หนึ่งตัวหลังข้อความสั่ง if เมื่อมีข้อความสั่งอื่นๆ ตามหลัง ต้องไม่ใส่ semicolon ก่อนหรือหลังคำสงวน then หรือ else ในข้อความสั่ง if ถ้าเราใส่ semicolon ก่อนคำว่า else คอมไพเลอร์จะจบข้อความสั่ง if นั้น และสั่งที่ไม่ถูกต้องคือ else เริ่มต้นข้อความสั่งชุดใหม่

อย่าลืมใส่ begin และ end ของข้อความสั่งประกอบเมื่อใช้เป็น true task หรือ false task ถ้าคู่ begin - end หายไป ข้อความสั่งแรกหนึ่งคำสั่งเท่านั้น จะเป็นส่วนของหนึ่ง task ซึ่งอาจนำไปสู่ข้อผิดพลาดวากยสัมพันธ์ ในตัวอย่างข้างล่างนี้ การไม่ใส่คู่ begin and ปิดล้อมงานจริง (true task) ทำให้คอมไพเลอร์เข้าใจว่า semicolon หลังข้อความสั่งกำหนดค่าเป็นตัวจบข้อความสั่ง if ผลลัพธ์คือเราได้ ข้อผิดพลาดวากยสัมพันธ์ ; expected เมื่อพบคำว่า else หลังจากนั้นถ้าเราใส่ semicolon ที่ตอนจบของข้อความสั่ง WriteLn ชุดแรก เราจะได้ข้อผิดพลาดวากยสัมพันธ์ error in statement เพราะว่าข้อความสั่งขึ้นต้นด้วยคำว่า else ไม่ได้

```
{if with missing begin - end}
if X > 0 then
   Sum := Sum + X ;
   WriteLn ('X is pasitive')
else
   WriteLn ('X is not positive)
```

เมื่อเขียนข้อความสั่ง nested if ให้พยายามเลือก เงื่อนไขต่าง ๆ ในวิธีซึ่งเราสามารถใช้รูปแบบหลายทางเลือก ดังที่แสดงในหัวข้อ 4.8

ถ้ามีเงื่อนไขมากกว่าหนึ่งเงื่อนไขอาจเป็นจริง ณ เวลาเดียวกันให้ใส่เงื่อนไขที่มีข้อจำกัดมากที่สุด เป็นอันดับแรก

โปรดจำไว้ว่า คอมไพเลอร์ Pascal จับคู่ else แต่ละตัวกับ if ซึ่งยังไม่คู่ ซึ่งอยู่ใกล้ที่สุด ถ้าไม่ระวัง เราอาจได้คู่ซึ่งไม่คาดคิดไว้ ในขณะที่อาจจะไม่มีข้อผิดพลาดวากยสัมพันธ์ แต่จะมีผลต่อผลลัพธ์ (outcome)

ในข้อความสั่ง case ต้องมั่นใจว่า case selector และ labels เป็นข้อมูลชนิดเชิงอันดับที่ (ordinal type) เหมือนกันได้แก่ เป็น Integer, Char หรือ Boolean แต่ไม่ใช่ Real หรือ String

โปรดจำไว้ว่า รายการของค่าเชิงอันดับที่ หรือ พิสัยย่อย(ใช้ได้เฉพาะ ใน Turbo Pascal) อาจใช้เป็น case labels และต้องไม่มีค่าใด ซึ่งปรากฏใน case label มากกว่าหนึ่ง case label

ให้ใช้ else clause เมื่อพิมพ์ข้อความเตือน (warning message) ถ้าการประเมินผล selector ได้ค่าซึ่งไม่อยู่ในรายการใดๆ ของ case labels ใน standard Pascal บ่อยครั้งเป็นเรื่องฉลาดที่เฝ้าระวัง case ด้วยข้อความสั่ง if อย่าลืมจบข้อความสั่ง case ด้วย end {case} และไม่มีการจับคู่ begin

ข้อสรุปตัวสร้างใหม่ของ Pascal (Summary of New Pascal Constructs)

Construct	Effect
Compond Statement begin {group} Write ('Enter X > '); ReadLn (X); Positive := Abs (X); Root := Sqrt (Positive) end {group}	ข้อความสั่งต่างๆ ในข้อความสั่งประกอบถูกกระทำการตามลำดับ

Construct	Effect
If Statement	
One Alternative	
If X < > 0.0 then Product := Product *X	คูณ Product ด้วย X เฉพาะเมื่อ X ไม่ใช่ค่าศูนย์
Two Alternatives	

If X > = 0.0 then
  WriteLn ('X, Positive')
else
  WriteLn ('X, negative')
	ถ้า X มีค่ามากกว่าหรือเท่ากับ 0.0 แสดงผล X แล้วตามด้วย คำว่า positive กรณีอื่นๆ แสดงผล X แล้วตามด้วยคำว่า negative

Several Alternatives	

If X < 0.0 then
  begin
  WriteLn ('negative');
  Sign := '-'
  end
else if X = 0.0 then
  begin
  WriteLn ('zero');
  Sign := '0'
  end
else
  begin
  WriteLn ('positive');
  Sign := '+'
  end
	หนึ่งข้อความในสามข้อความจะถูกพิมพ์ทั้งนี้ขึ้นอยู่กับว่า X มีค่าเป็นลบ บวก หรือ ศูนย์
	
	Sign (type Char) เก็บเครื่องหมายของ X

Construct	Effect
Case Statement	

Case NextCh of
'A', 'a' : WriteLn ('Excellent');
'B' ,'b' : WriteLn ('Good');
'C', 'c' : WriteLn ('O.K');
'D', 'd', 'F', 'f' :
begin
Write ('Poor, student');
WritLn ('on probation')
end
else
WriteLn ('Bad value')
end {case}
	แสดงผลหนึ่งข้อความของห้าข้อความ ขึ้นอยู่กับค่าของ NextChar (type Char) ถ้า NextChar คือ 'D', 'd', 'F', 'f' นักศึกษาอยู่ในขั้นทดลองเรียน (probation) ถ้า NextChar ไม่มีในรายการของ case labels จะแสดงผลข้อความ
	Bad value

แบบฝึกหัด Quick - Check

1. ข้อความสั่ง if หมายถึง ข้อความสั่งควบคุมสำหรับ.....

.....

2. ข้อความสั่งประกอบ (Compound statement) หมายถึงอะไร

3. ข้อความสั่ง case บ่อยครั้งถูกนำมาใช้แทนข้อความสั่งอะไร

4. ทำไมข้อความสั่ง ข้างล่างนี้ จึงล้มเหลว (fail) ในการ implement Pascal บางเวอร์ชัน แต่ไม่ล้มเหลวลงในเวอร์ชั่นอื่นๆ สมมติว่า I และ J เป็นตัวแปรชนิด Integer

```
if (I > 0) and (J div I = 0) then
   WriteLn ('I is a factor of J')
```

5. ตัวดำเนินการสัมพันธ์ < > หมายถึงอะไร

6. ตัวดำเนินการอะไรที่มีการทำก่อนสูงสุด (highest precedonce) จงบอกชื่อตัวดำเนินการ สี่ชื่อซึ่งมีการทำก่อนต่ำที่สุด

7. แผนภาพวากยสัมพันธ์ (Syntax diagram) คืออะไร มีวัตถุประสงค์อะไร

8. จงแก้ไขข้อผิดพลาดวากยสัมพันธ์ ในข้อความสั่งข้างล่างนี้

```
if  X > 25.0 then
    begin
    Y := X - 25.0 ;
else
    Y := X
end {if}
```

9. จงหาค่าซึ่งกำหนดให้ Fee ด้วยข้อความสั่ง if แต่ละชุด ข้างล่างนี้ เมื่อ Speed มีค่าเท่ากับ 75 ข้อความสั่ง if ชุดซ้ายมือ หรือชุดขวามือ ชุดใดถูกต้อง

```
if  speed > 35 then
    Fee := 20.00
else If Speed > 50 then
    Fee := 40.00
else if Speed > 75 then
    Fee := 60.00
```

```
if Speed > 75 then
    Fee := 60.00
else if Speed > 50 then
    Fee : = 40.00
else if Speed > 35 then
    Fee : = 20.00
```

10. เอาต์พุต บรรทัดใด ถูกแสดงผลด้วยข้อความสั่งข้างล่างนี้ เมื่อ Grade มีค่าเท่ากับ 'I' เมื่อ Grade มีค่าเท่ากับ 'B', เมื่อ Grade มีค่าเท่ากับ 'b'

```
Case Grade of
    'A' : Points := 4;
    'B' : Points := 3;
    'C' : Points := 2;
    'D' : Points := 1;
    'E', 'I', 'W' : Points := 0
else
    Write ('Bad grade- ')
end {case}
```

```
If ('A' < = Grade) and (Grade < = 'D') then
   Write ('Passed, points earned = ', Points)
else
   WriteLn ('No points earned')
```

11. จงอธิบายความแตกต่างระหว่างข้อความสั่ง nested if ทางซ้ายมือกับข้อความสั่ง if ทางขวามือ สำหรับแต่ละชุด

จงบอกค่าสุดท้ายของ X ถ้าค่าเริ่มต้นของ X เท่ากับ 1

```
if  X > = 1 then                 |   if  X > = 1 then
   X := x + 1                    |      X:= x + 1 ;
else if X > = 0 then             |   if  X > = 0 then
   X := x + 2                    |      X := X + 2
```

คำถามทบทวน (Review Questions)

1. จงอธิบายว่าทำไมตัวดำเนินการสัมพันธ์จึงต้องอยู่ในวงเล็บ เมื่อใช้ตัวดำเนินการ and หรือ or

2. ตัวดำเนินการสัมพันธ์ แตกต่างจากตัวดำเนินการแบบบูลอย่างไร

3. การประเมินผลแบบบูลวิธีลัด (short - circuit Boolean evaluation) คืออะไร ทำไมจึงไม่ควรนำมาใช้

4. จงตามรอย (trace) ส่วนของโปรแกรมข้างล่างนี้ และแสดงให้เห็นว่ากระบวนงานชุดใดจะถูกเรียกถ้าค่าของข้อมูลที่ใส่คือ 27.34

```
WriteLn ('Enter a temperature > ');
ReadLn (Temp);
If Temp > 32.0 then
   NotFreezing
else
   Icc Forming
```

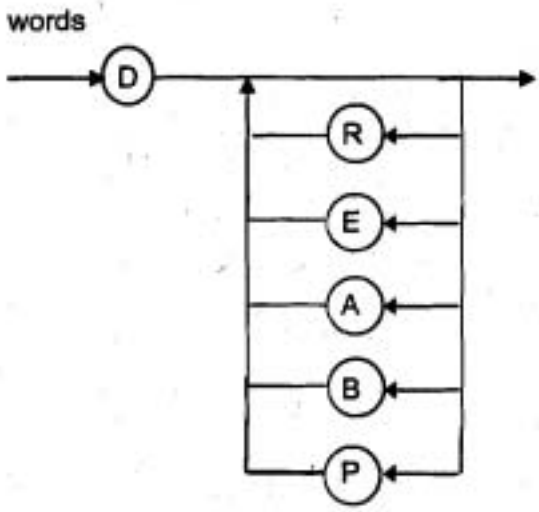
5. จงเขียนข้อความสั่ง nested if เพื่อให้แสดงผลข้อความซึ่งชี้ระดับการศึกษาของนักเรียนซึ่งขึ้นอยู่กับจำนวนปีของการเล่าเรียนในโรงเรียน : 0 - None, 1 ปี ถึง 5 ปี Elemen-

tary School, 6 ปี ถึง 8 ปี - Middle School, 9 ปีถึง 12 ปี - High School, มากกว่า 12 ปี - College พิมพ์ข้อความเพื่อแสดงว่า ข้อมูลผิด (bad data) ด้วยเช่นกัน

6. ทำคำถามข้อ 5 ใหม่ โดยใช้ข้อความสั่ง case

7. กำหนดแผนภาพวากยสัมพันธ์ข้างล่างนี้ คำชุดใดต่อไปนี้ถูกต้อง

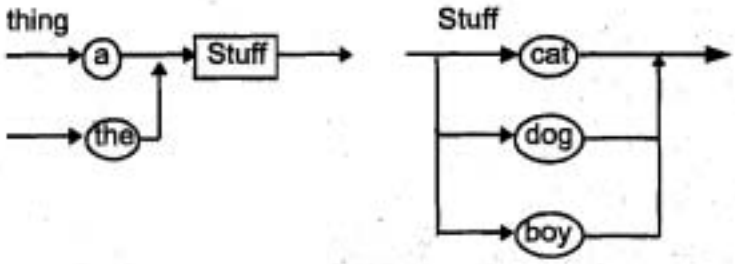
pear, bread, drear, deaden, dad, drab



8. จงเขียนการเฝ้าระวังข้อความสั่ง case เพื่อเลือกการดำเนินการ ซึ่งขึ้นอยู่กับค่าของ Inventory ดังนี้

เพิ่มค่า TotalPaper ด้วย PaperOrder ถ้า Inventory เท่ากับ 'B' หรือ 'C' ; เพิ่มค่า TotalRibbon ด้วย 1 ถ้า Inventory เท่ากับ 'E', 'F' หรือ 'D'; เพิ่มค่า TotalLabel ด้วย LabelOrder ถ้า Inventory เท่ากับ 'A' หรือ 'x' และไม่ต้องทำอะไร ถ้า Inventory เท่ากับ 'M'

9. จงเขียน หกคู่ ของคำซึ่งเป็นไปตามแผนภาพวากยสัมพันธ์ สำหรับ thing



โครงงานเขียนโปรแกรม (Programming Projects)

1. จงเขียนกระบวนงานสามชุด วาดรูปวงกลม (circle) สี่เหลี่ยมจตุรัส (Square) และสามเหลี่ยม (triangle) จากนั้นเขียนโปรแกรมอ่านตัวอักษรหนึ่งตัว ซึ่งอาจเป็น C, S หรือ T และขึ้นอยู่กับตัวอักษรที่เลือกแล้ววาดรูปหนึ่งรูป เป็นวงกลม, สี่เหลี่ยมจตุรัส หรือสามเหลี่ยม

2. จงเขียนโปรแกรมอ่านคำสี่คำ และแสดงผลโดยเรียงอันดับตามตัวอักษรจากน้อยไปหามาก และเรียงอันดับตามตัวอักษรจากมากไปหาน้อย

3. จงเขียนโปรแกรม อ่าน หมายเลขห้องเรียน ความจุของห้องเรียน และจำนวนนักศึกษาซึ่งลงทะเบียน จากนั้นพิมพ์ บรรทัดเอาต์พุตแสดงให้เห็น หมายเลขห้องเรียน ความจุของห้องเรียน จำนวนเก้าอี้ที่มีในห้อง จำนวนเก้าอี้ที่ว่าง และข้อความแสดงว่าชั้นเรียนเต็มหรือไม่แล้วเรียกกระบวนงานให้แสดงผลหัวเรื่อง ข้างล่างนี้ก่อนบรรทัดเอาต์พุต

Room Capacity Enrollment Empty seats Filled / not Filled

แสดงผลแต่ละส่วนของบรรทัดเอาต์พุต ใต้หัวเรื่องที่ถูกต้อง ทดสอบโปรแกรมด้วยข้อมูล ห้องเรียนต่อไปนี้

Room	Capacity	Enrollment
426	25	25
327	18	14
420	20	15
317	100	90

4. จงเขียนโปรแกรม คำนวณภาษีรัฐ (state tax) ซึ่งพนักงานจะต้องชำระเพิ่ม กำหนดดังนี้ รัฐเก็บภาษี 4% ของรายได้สุทธิ (net income) การคำนวณรายได้สุทธิให้เอา $500 ลบออกจากรายได้รวม (gross income) ของพนักงานแต่ละคน ตัวโปรแกรมอ่านรายได้รวมจำนวนพนักงานและภาษีซึ่งได้หักลบไปเรียบร้อยแล้ว จากนั้น คำนวณ ภาษีจริง และพิมพ์ค่าแตกต่างระหว่าง ภาษีที่คำนวณได้กับภาษีซึ่งได้เก็บไปแล้ว ด้วยข้อความดังนี้

'SEND CHECK' or ' REFUND' ทั้งนี้ขึ้นอยู่กับค่าของความแตกต่างว่าเป็นค่าบวก หรือ ค่าลบ

5. จงเขียนโปแกรมคำนวณจำนวนวัน (1 ถึง 366) ในหนึ่งปีสำหรับ วันเดือนปี ซึ่งกำหนดให้เป็น ข้อมูลอินพุต ตัวอย่างเช่น วันที่ 1 มกราคม 1994 คิดเป็น 1 วัน, วันที่ 31 ธันวาคม 1993 คิดเป็น 365 วัน, วันที่ 31 ธันวาคม 1996 คิดเป็น 366 วัน เพราะว่าปี 1996 เป็นปีอธิกวาร (leap year) ปีใดซึ่งเป็นปีอธิกวาร จะหารด้วย 4 ลงตัว ยกเว้น ปีใดก็ตามที่หารด้วย 100 ลงตัวเป็นปีอธิกวาร ก็ต่อเมื่อ มันหารด้วย 400 ลงตัวด้วย โปรแกรมของนักศึกษารับ วัน เดือน ปี เป็นเลขจำนวนเต็ม