บทที่ 12
เซตและสายอักขระ

12.1 แบบชนิดข้อมูลเซต

12.2 ตัวดำเนินการเซต

12.3 สายอักขระความยาวแปรได้

12.4 อธิบายการประมวลผลสายอักขระ

กรณีศึกษา : Text Editor

12.5 ข้อผิดพลาดร่วมของการเขียนโปรแกรม

ในบทนี้จะแนะนำแบบชนิดข้อมูลเชิงโครงสร้างตัวใหม่ ซึ่งผู้ใช้ให้นิยาม ได้แก่ เซต เซตมีประโยชน์สำหรับการตรวจดูว่า ตัวแปรมีค่าซึ่งอยู่ในรายการของค่าต่างๆ หรือไม่ ในบทนี้ เราจะเรียนรู้ว่าเซตคืออะไร มีตัวดำเนินการอะไรบ้าง ซึ่งใช้กับเซต และจะใช้เซตในโปรแกรมอย่างไร

หัวข้อที่สองในบทนี้คือแบบชนิดข้อมูลสายอักขระ สายอักขระใช้ในโปรแกรมซึ่งประมวลผลข้อมูลข้อความ (text data) ได้แก่ ตัวประมวลผลคำ (word processor) ตัวบรรณาธิกรข้อความ (text editors) และงานประยุกต์ประมวลผลข้อมูลทางธุรกิจ (business data-processing applications)

Turbo Pascal จัดหาแบบชนิดข้อมูลสายอักขระพลวัต (dynamic string data type) รวมทั้งฟังก์ชันและกระบวนงานหลายชุด ซึ่งให้เราใช้เพื่อทำงานกับตัวแปรสายอักขระ (string variables) ในหัวข้อ 9.7 ได้แนะนำแบบชนิดข้อมูลสายอักขระของ Turbo Pascal ไปแล้ว และเปรียบเทียบมันกับแถวลำดับของตัวอักขระ ในบทนี้เราจะอภิปรายแบบชนิดข้อมูลสายอักขระของ Turbo Pascal และตัวดำเนินการสำหรับตัวแปรสายอักขระอย่างครบถ้วน

12.1 แบบชนิดข้อมูลเซต (Set Data Type)

เซตเป็นตัวแปรเชิงโครงสร้างซึ่งประกอบด้วย รายการของเลขจำนวนเต็ม ตัวอักขระ หรือค่าชนิดแจงนับ

เซตคล้ายกับแถวลำดับตรงที่มันสามารถรวมกลุ่มของสมาชิกอย่างง่าย แต่สิ่งที่ไม่เหมือนกับแถวลำดับคือ สมาชิกของเซตไม่ต้องเรียงอันดับ

เซต หมายถึง รายการแบบไม่มีอันดับของสมาชิกซึ่งเป็นค่าของชนิดเชิงอันดับที่เหมือนกัน (Set is an unrdered list of elements that are values of the same ordinal type.)

ในวิชาคณิตศาสตร์ เซตถูกแทนด้วยรายการของสมาชิกเซต สมาชิกเซตต้องอยู่ในวงเล็บปีกกา ตัวอย่างเช่น เซต {1, 3, 5, 7, 9} เป็นเซตของเลขจำนวนเต็มคี่ จาก 1 ถึง 9 ใน Pascal สมาชิกเซตอยู่ในวงเล็บก้ามปู ดังนั้น เซตข้างต้นใน Pascal ต้องเขียนดังนี้

[1, 3, 5, 7, 9]

เนื่องจากสมาชิกของเซตไม่ต้องเรียงอันดับเพราะฉะนั้น เซต [9, 5, 7, 1, 3] จึงสมมูลกับเซต [1, 3, 5, 7, 9]

สมาชิกเซต หมายถึง ค่าในเซต (Set element (set member) is a value in a set.)

การประกาศเซต (Declaring Sets)

เราประกาศเซต หรือตัวแปรเซต เหมือนกับที่เราประกาศชนิดโครงสร้างอื่นๆ ขั้นแรกประกาศแบบชนิดข้อมูลเซต หลังจากนั้นประกาศตัวแปรของชนิดนั้น

ตัวอย่าง 12.1

การประกาศ

```
type
   DigiSet = set of 0..9; {set type of integer elements}
var
   Odds, Evens, Middle, Mixed : DigitSet; {4 sets}
```

ประกาศชนิดเซต ชื่อ DigitSet และตัวแปรเซตสี่ตัว ได้แก่ Odds, Evens, Middle, และ Mixed

ตัวแปรเซตแต่ละตัวเป็นชนิด Digitset ประกอบด้วยสมาชิกระหว่างศูนย์ตัว ถึง 10 ตัวเลือกจากเลขจำนวนเต็มในพิสัยย่อย 0..9 ถึงแม้ว่าเนื้อที่หน่วยความจำจะถูกจัดสรรสำหรับสี่เซต แต่ขณะนี้ contents ของมันยังไม่ถูกนิยาม (undefined) การทำงานกับเซตเราต้องให้นิยามมัน โดยใช้การกำหนดค่าเซต (set assignment) เช่น ที่แสดง

Syntax Display

การประกาศชนิดเซต (Set Type Declaration)

Form :
```
type
    set type = set of base type;
```

ตัวอย่าง

```
type
    LetterSet = set of 'A' .. 'Z';
```

มีความหมายดังนี้ : ไอเดนติไฟเออร์ set type ถูกนิยามมีค่าต่างๆ ซึ่งกำหนดใน base type ตัวแปรซึ่งจะถูกประกาศเป็นชนิด set type หมายถึงเซตซึ่งสมาชิกของมันถูกเลือกจากค่าต่างๆ ใน base type และ base type ต้องเป็นชนิดเชิงอันดับที่ (ordinal type)

ข้อสังเกต การทำให้เกิดผลส่วนใหญ่มีข้อจำกัดที่จำนวนของค่าต่างๆ ใน base type ของเซต Turbo Pascal จำกัดเลขจำนวนนี้ของค่าต่างๆ ในแบบชนิดข้อมูล Char (256) ยอมให้เราประกาศ set of Char เป็นชนิดเซต

ตามข้อจำกัดนี้ เราจึงไม่สามารถใช้แบบชนิดข้อมูล Integer เป็น base type แต่เราสามารถใช้พิสัยย่อยของชนิด Integer ที่มีค่าได้มากที่สุด 256 ค่า

การกำหนดค่าเซตและสัญพจน์เซต (Set Assignment and Set Literals)

ข้อความสั่งกำหนดค่าเซตใส่ค่าต่างๆ ในหนึ่งเซต

ข้อความสั่ง

```
Odds := [1, 3, 7, 9];
Evens := [0, 2, 4, 6, 8]
```

กำหนดค่าต่างๆ ให้กับตัวแปรเซตสองตัว ซึ่งประกาศในตัวอย่าง 12.1

ข้อความสั่งกำหนดค่าแต่ละคำสั่งกำหนด (assigns) สัญพจน์เซตให้กับตัวแปรเซต

ในที่นี้ สัญพจน์เซต หมายถึง รายการของค่าต่างๆ จาก set base type อยู่ภายในวงเล็บก้ามปู หลังจากการกำหนดค่าเหล่านี้แล้ว เซต Odds ประกอบด้วยเซตของเลขโดดคี่ในพิสัยย่อย 0 ถึง 9 และเซต Evens ประกอบด้วยเซตของเลขโดดคู่ ในพิสัยเดียวกัน เราสามารถใช้สองเซตนี้เพื่อตรวจดูว่าตัวแปรมี content เป็นเลขคี่หรือเลขคู่หรือไม่

สัญพจน์เซต ['0' .. '9' , '+' , '-', 'E' , '.'] หมายถึง เซตของตัวอักขระซึ่งสามารถปรากฏในเลขจำนวนจริง เซตนี้มีสมาชิก 14 ตัว การใช้สัญพจน์พิสัยย่อย '0' .. '9' เพื่อแสดงถึงตัวอักขระเลขโดด 10 ตัว สะดวกกว่าการเขียนรายการอักขระเลขโดดแต่ละตัวแยกจากกัน 10 ตัว

สัญพจน์เซต หมายถึง รายการของค่าต่างๆ อยู่ในวงเล็บก้ามปู (Set literal is a list of values enclosed in brackets)

Syntax Display

สัญพจน์เซต (Set literal)

Form : [list-of-elements]

ตัวอย่าง ['+' , '-' , '*' , '/' , '<' , '>' , '=']

มีความหมายดังนี้ : เซตถูกนิยามสมาชิกของมันคือ list-of-elements อยู่ในวงเล็บก้ามปู สมาชิกของเซตต้องมีชนิดเชิงอันดับที่เหมือนกัน หรือชนิดเชิงอันดับที่แทนกันได้ (นิยามในหัวข้อ 7.6) เครื่องหมาย comma ใช้คั่นสมาชิกแต่ละตัวใน list-of-elements กลุ่มของสมาชิกติดต่อกันอาจกำหนดด้วยสัญกรณ์พิสัยย่อย (ตัวอย่างเช่น minvalue..maxvalue) เมื่อ minvalue และ maxvalue เป็นนิพจน์ใช้แทนกันได้ (type-compatible expressions) และ minvalue มีค่าน้อยกว่า หรือเท่ากับ maxvalue)

ข้อสังเกต ใน Turbo Pascal, Ord(minvalue) ต้อง $\geq$ 0 และ Ord (maxvalue) ต้อง $\leq$ 255

Syntax Display

การกำหนดค่าเซต (set Assingment)

Form : set var := set expression

ตัวอย่าง

Uppercase := ['A' .. 'Z'] {set of uppercase letters} มีความหมายดังนี้ ตัวแปร set var ถูกนิยามเป็นเซตซึ่งสมาชิกของมันกำหนดโดย set expression

ในที่นี้ set expression อาจจะเป็นสัญพจน์ หรือตัวแปรเซตอีกตัวหนึ่ง อีกทางเลือกหนึ่ง set expression อาจระบุการจัดดำเนินการของสองเซตหรือมากกว่าสองเซต โดยใช้ตัวดำเนินการเซต (set operators)

base type ของ set var และ set expression ต้องเป็นชนิดใช้แทนกันได้ และสมาชิกทุกตัวใน set expression ต้องอยู่ใน base type ของ set var

เซตว่างและเอกภพสัมพัทธ์ (Empty Set and Universal Set)

สัญพจน์เซตพิเศษสองตัวที่เกี่ยวข้องกับชนิดเซตแต่ละตัวได้แก่เซตว่างและเอกภพสัมพัทธ์

เซตว่าง หมายถึง เซตที่ไม่มีสมาชิกใดๆ ใช้สัญลักษณ์คู่ของวงเล็บก้ามปู (Empty set is a set having zero elements denoted by [ ])

การสร้างเซตว่าง เราต้องใช้การกำหนดค่า เช่น

Middle := [ ]

เซตว่างแตกต่างจากเซตไม่ถูกนิยาม (undefined set)

เซตไม่ถูกนิยาม หมายถึง ตัวแปรเซตที่ถูกประกาศแต่ไม่มีการกำหนดค่าใดๆ (Undefined set is a set variable that is declared but it not assigned any values.)

เอกภพสัมพัทธ์ หมายถึง เซตซึ่งประกอบด้วยค่าทั้งหมดใน base type สำหรับชนิดเซตเฉพาะหนึ่งเซต (Universal set is a set containing all possible values of a set.)

ตัวอย่างเช่น เซตชนิด DigiSet ซึ่งมี base type เป็นพิสัยย่อย 0..9 ดังนั้นเอกภพสัมพัทธ์ จะแทนด้วย [0..9] การกำหนดค่า

Mixed := [0..9]

นิยามตัวแปรเซต Mixed เป็นเอกภพสัมพัทธ์

บ่อยครั้งที่เราเริ่มต้นให้ตัวแปรเซตเป็นเซตว่าง หรือเอกภพสัมพัทธ์ ก่อนใช้เซตนั้น

เซตกับค่าชนิดแจงนับ (Sets with Enumerated Type Values)

เราสามารถให้นิยามเซตด้วยค่าที่เลือกมาจากชนิดข้อมูลแจงนับของเราเอง

ตัวอย่าง 15.2

ในการประกาศข้างล่างนี้ เราประกาศและใช้ชนิดแจงนับ (Cars) เป็น base type สำหรับชนิดเซต (CarSet) ต่อไป เราประกาศตัวแปรเซตสามตัวที่มีชนิดเป็น CarSet

```
type
   Cars =  (Dodge, Ford, Lincoln, Cadillac, Fiesta, Pontiac, Convette, Buick,
            Chevrolet, Mercury, Mastang);
   CarSet = set of Cars;
var
   Avis, Hertz, Merger : CarSet;
```

การกำหนดค่า

```
   Avis := [Dodge, Lincoln, Fiesta];
   Hertz := [Dodge .. Cadillac, Mercury];
   Merger := [Dodge .. Mustang]
```

ให้นิยามเซต Avis ประกอบด้วยรายการสมาชิกสามตัว เซต Hertz ประกอบด้วยสมาชิกห้าตัว และเซต Merger ซึ่งเป็นเอกภพสัมพัทธ์ ของชนิด CarSet

เราอาจวางตัวแปรเซตไว้ทางขวามือของข้อความสั่งกำหนดค่าได้ จัดให้ตัวแปรเซตทั้งคู่มีชนิด base ใช้แทนกันได้ ข้อความสั่งถัดไป การกำหนดค่า เปลี่ยนแปลงค่าของเซต Merger

```
Merger := Hertz
```

แบบฝึกหัด 12.1 Self-Check

1. เซตต่อไปนี้ชุดใดถูกต้อง ชุดใดไม่ถูกต้อง เซตที่ถูกต้องสมาชิกของมันคืออะไร

   a) [1..3, 1..5]

   b) ['1', '3', '1' .. '5']

   c) [1, 3, '1' .. '5']

   d) ['1', '3', 'A' .. 'C']

   e) [1, 10, 500 .. 501]

2. จงเขียนเซตซึ่งประกอบด้วยอักขระพิเศษ ใช้ใน Pascal สำหรับสัญลักษณ์กำกับวรรคตอน หรือสัญลักษณ์ตัวดำเนินการ

12.2 ตัวดำเนินการเซต (Set Operators)

การดำเนินการหลายอย่าง อาจกระทำบนเซต ในหัวข้อนี้ เราอธิบายตัวดำเนินการใหม่หนึ่งตัว คือ in และแสดงให้เห็นว่าจะใช้ตัวดำเนินการที่คุ้นเคยอื่นๆ กับเซตอย่างไร

การทดสอบสำหรับภาวะสมาชิกของเซต (Testing for Set Membership)

เราใช้ส่วนจำเพาะค้นหา (ดูรูป 9.18) เพื่อตรวจดูว่าค่าเฉพาะ (a particular value) อยู่ในแถวลำดับหรือไม่ และถ้าอยู่ ตำแหน่งของมัน

Pascal จัดหาตัวดำเนินการภาวะสมาชิกของเซต ได้แก่ in เพื่อตรวจดูว่าค่าเฉพาะนั้นเป็นสมาชิกของเซตหรือไม่

นิพจน์ที่มีตัวดำเนินการ in จะกลับคืนค่าแบบบูล เราสามารถใช้ตัวดำเนินการ in กับเซตแทนที่จะเป็นเงื่อนไขประกอบ (compound condition)

ตัวอย่างเช่น เราเขียนเงื่อนไข

(Ch = '.') or (Ch = '?') or (Ch = ';') or (Ch = '!')

เขียนรวบรัดกว่าเป็น

Ch in ['.', '?'. ';', '!']

เงื่อนไขทั้งคู่เป็นจริง ถ้า Ch เป็นอักขระตัวใดตัวหนึ่งในเซต

ตัวอย่าง 12.3

บางโปรแกรมต้องการให้เราตรวจดูว่าตัวแปรอักขระเป็นตัวสระ อักษรตัวพิมพ์เล็ก หรืออักษรตัวพิมพ์ใหญ่ หรือไม่

เราสามารถใช้เซตสามชุดซึ่งจะได้นิยามถัดไปช่วยในการทำสิ่งนี้

```
type
   CharSet = set of Char; {set type of Char elements}

var
   Vowels, Uppercase, lowercase : CharSet; {3 sets}
   NextChar : Char; {a data character}

begin
   Vowels := ['A', 'a', 'E', 'e', 'I' 'i', 'O', 'o', 'U', 'u',];
```

```
Uppercase := ['A'..'Z'];
Lowercase := ['a'..'z']
```

ลูปข้างล่างนี้ทำซ้ำจนกระทั่ง NextChar เป็นตัวอักษร ข้อความสั่ง if แสดงข้อความเกี่ยวกับ NextChar ซึ่งขึ้นอยู่กับ content ของมัน

```
repeat
   Write ('Enter a letter> ');
   ReadLn (NextChar);

   {Display message about NextChar.}
   if NextChar in Vowdls then
      WriteLn (NextChar, ' is a vowel')
   else if NextChar in Uppercase then
      WriteLn (NextChar, ' is an uppercase consonant')
   else if NextChar in Lowercase then
      WriteLn (NextChar, ' is an lowercase consonant')
   else
      WriteLn (NextChar, ' is not a letter')
until NextChar in ['A'..'Z', 'a'..'z']
```

โปรดระวังเมื่อส่วนเติมเต็มนิพจน์ประกอบด้วย ตัวดำเนินการ สภาวะสมาชิกเซต ส่วนเติมเต็มของเงื่อนไข until คือ

```
not (NextChar in ['A'..'Z', 'a'..'z']) {not a letter}
```

ข้อผิดพลาดร่วมคือการเขียนนิพจน์ข้างต้นนี้เป็น

```
NextChar not in 'A'..'Z', 'a'..'z'] {invalid expression}
```

Syntax Display

ตัวดำเนินการภาวะสมาชิกของเซต in (Set Membership Operator in)

Form : `element in [list-of-elements]`

ตัวอย่าง

```
NextChar in ['+', '-', '*', '/', '<', '>', ',', '=']
```

มีความหมายดังนี้ :

ตัวดำเนินการภาวะสมาชิกของเซต in อธิบายเงื่อนไขซึ่งประเมินผลเป็น True เมื่อ element อยู่ใน list-of-elements กรณีอื่นๆ เงื่อนไขประเมินผลเป็น False แบบชนิดข้อมูลของ element ต้องใช้แทนกันได้กับสมาชิกเซต ตัวดำเนินการ in มีการทำก่อน (precedence) เท่ากับตัวดำเนินการสัมพันธ์ (relational operators)

ส่วนรวม ส่วนร่วม และผลต่างของเซต (Set Union, Intersection, and Difference)

ถ้าเราได้ศึกษาเรื่องเซตในวิชาคณิตศาสตร์มาแล้ว จะทราบว่าการดำเนินการทั้งสามชนิดนี้ ใช้กระทำร่วมกันบนเซต :

ส่วนรวม, ส่วนร่วม และผลต่าง

การดำเนินการเหล่านี้ จัดดำเนินการสองเซตเพื่อให้เป็นเซตที่สามใน Pascal เรากำหนดการดำเนินการเหล่านี้โดยใช้ตัวดำเนินการ + (set union), * (set intersection), และ - (set difference)

ส่วนรวม (union) ของสองเซต (ตัวดำเนินการเซต +) หมายถึง เซตของสมาชิกซึ่งอยู่ในเซตใดเซตหนึ่ง หรืออยู่ในทั้งสองเซต

ตัวอย่างเช่น

[1, 3, 4] + [1, 2, 4] คือ [1, 2, 3, 4]

[1, 3] + [2, 4] คือ [1, 2, 3, 4]

['A', 'C', 'F'] + ['B', 'C' 'D', 'F'] คือ ['A', 'B' 'C', 'D', 'F']

['A', 'C', 'F'] + ['A', 'C', 'D', 'F'] คือ ['A', 'C', 'D', 'F']

ส่วนร่วม (intersection) ของสองเซต (ตัวดำเนินการเซต *) หมายถึงเซตของสมาชิกซึ่งต้องอยู่ในทั้งสองเซต

ตัวอย่างเช่น

[1, 3, 4] * [1, 2, 4] คือ [1, 4]

[1, 3] * [2, 4] คือ [ ]

['A', 'C', 'F'] * ['B', 'C', 'D', 'F'] คือ ['C', 'F']

['A', 'C', 'F'] * ['A', 'C', 'D', 'F'] คือ ['A', 'C', 'F']

ผลต่างของเซต A และเซต B (ตัวดำเนินการเซต -) หมายถึง เซตของสมาชิกซึ่งอยู่ในเซต A แต่ไม่อยู่ในเซต B

ตัวอย่างเช่น

[1, 3, 4] - [1, 2, 4] คือ [3]

[1, 3] - [2, 4] คือ [1, 3]

['A', 'C', 'F'] - ['B', 'C' 'D', 'F'] คือ ['A']

['A', 'C', 'F'] - ['A', 'C', 'D', 'F'] คือ [ ]

['A', 'C', 'D', 'F'] - ['A', 'C', 'F'] คือ ['D']

ตัวดำเนินการเซต - ไม่เป็นการสลับที่ (commutative) สิ่งนี้หมายความว่า A-B และ B-A จะมีค่าแตกต่างกันส่วนตัวดำเนินการเซต + และ * ทั้งคู่เป็นตัวดำเนินการสลับที่

ตัวดำเนินการสลับที่ หมายถึงตัวดำเนินการซึ่งให้ผลลัพธ์เหมือนกันโดยไม่สนใจอันดับของตัวถูกดำเนินการของมัน (Commutative operator is an operator that produces the same resault regardless of the order of its operands.)

ในนิพจน์ Pascal, ตัวดำเนินการ +, * และ - กำหนดการดำเนินการคำนวณเมื่อตัวถูกดำเนินการของมันเป็นชนิด Real หรือ Integer และตัวดำเนินการทั้งสามตัวนี้กำหนดการดำเนินการบนเซต เมื่อตัวถูกดำเนินการของมันเป็นเซต

เมื่อตัวดำเนินการหนึ่งตัวมีความหมายมากกว่าหนึ่งความหมาย เงื่อนไขนี้เรียกว่าภาระเกินของตัวดำเนินการ

ภาระเกินของตัวดำเนินการ หมายถึง การให้ตัวดำเนินการหนึ่งตัวมีมากกว่าหนึ่งความหมาย (Opertor overloading is giving an operator more than one meaning.)

กฎการทำก่อนสำหรับตัวดำเนินการ +, * และ - เท่ากัน โดยไม่สนใจการนำไปใช้ (ดูตาราง 4.5) เมื่อมีข้อสงสัยให้ใช้วงเล็บเล็กเพื่อกำหนดอันดับของการประเมินผล

บ่อยครั้งที่สมาชิกใหม่หนึ่งตัวต้องนำไปใส่ในเซตมีจริง (existing set) การใส่เช่นนี้กระทำสำเร็จโดยการประกอบส่วนรวมของเซตมีจริงกับเซตหนึ่งหน่วย ซึ่งเป็นเซตที่มีเฉพาะสมาชิกตัวใหม่เท่านั้น

ตัวอย่าง

[1, 3, 4, 5] + [2] คือ [1, 2, 3, 4, 5]

ในที่นี้เซต [2] คือเซตหนึ่งหน่วย

เซตหนึ่งหน่วย หมายถึง เซตซึ่งมีสมาชิกหนึ่งตัวอยู่ในวงเล็บก้ามปู (Unit set is a set containing a single element enclosed in brackets.)

หลีกเลี่ยงข้อผิดพลาดร่วมของการลืมใส่วงเล็บก้ามปูล้อมรอบเซตหนึ่งหน่วย ตัวอย่างเช่น นิพจน์

[1, 3, 4, 5] + 2 {ตัวถูกดำเนินการตัวที่สองไม่ใช่เซต}

Avis + Cadillac {ตัวถูกดำเนินการตัวที่สองไม่ใช่เซต}

ทั้งสองชุดไม่ถูกต้อง (invalid) เพราะว่ามีตัวถูกดำเนินการหนึ่งตัวเป็นเซต และตัวถูกดำเนินการอีกตัวหนึ่งเป็นค่าคงตัว (constant) ดูตัวอย่าง 12.2 เช่นเดียวกัน นิพจน์

[Avis] + [Cadillac] {ตัวถูกดำเนินการตัวที่หนึ่งไม่ใช่เซต}

ไม่ถูกต้องเพราะว่าวงเล็บก้ามปูปิดล้อม Avis ไม่จำเป็นต้องมี เพราะว่า Avis เป็นเซตอยู่แล้ว

ตัวดำเนินการสัมพันธ์บนเซต (Set Relational Operators)

ตัวดำเนินการสัมพันธ์ =, <>, <= และ >= ใช้เปรียบเทียบเซตสองชุด เซตสองชุดซึ่งนำมาเปรียบเทียบกันต้องมีชนิดฐาน (base type) เหมือนกัน ผลลัพธ์ของการเปรียบเทียบคือค่าแบบบูล (Boolean value)

ตัวดำเนินการ = และ <> ทดสอบว่าเซตสองชุดมีสมาชิกเหมือนกันหรือไม่

ตัวอย่างเช่น

[1,3] = [1,3] คือ Ture [1,3] <> [1,3] คือ False

[1,3] = [2,4] คือ False [1,3] <> [2,4] คือ True

[1,3] = [3,1] คือ True [1,3] <> [3,1] คือ False

[ ] = [1] คือ False [ ] <> [1] คือ True

จะเห็นว่าบรรทัดเหนือบรรทัดสุดท้ายรายการเรียงอันดับของสมาชิกในเซตไม่สำคัญ ([1,3] และ [3,1] แทนเซตเดียวกัน)

อย่างไรก็ตาม ปกติ รายการสมาชิกของเซตจะเรียงลำดับเชิงอันดับที่จากน้อยไปหามาก (increasing ordinal sequence)

ตัวดำเนินการสัมพันธ์ <= และ >= กำหนดภาวะความสัมพันธ์เซตย่อย (subset) และซูเปอร์เซต (superset) ดังนี้

เซต A เป็นเซตย่อยของเซต B (A <= B) ถ้าสมาชิกทุกตัวของ A เป็นสมาชิกของเซต B :

[1,3] <= [1,2,3,4] เป็นจริง

[1,3] <= [1,3] เป็นจริง

[1,2,3,4] <= [1,3] เป็นเท็จ

[1,3] <= [ ] เป็นเท็จ

[ ] <= [1,3] เป็นจริง

เซต A เป็นซูเปอร์เซตของเซต B (A >= B) ถ้าสมาชิกทุกตัวของ B เป็นสมาชิกของ A :

[1,3] >= [1,2,3,4] เป็นเท็จ

[1,3] >= [1,3] เป็นจริง

[1,2,3,4] >= [1,3] เป็นจริง

[1,3] >= [ ] เป็นจริง

[ ] >= [1,3] เป็นเท็จ

ตัวดำเนินการเซต สรุปไว้ในตาราง 12.1

ตาราง 12.1 ตัวดำเนินการเซต

Operator	Meaning	Example
+	Set union	['A'] + ['B'] คือ ['A', 'B']
-	Set difference	['A', 'B'] - ['A'] คือ ['B']
*	Set intersection	]'A', 'B'] * ['A'] คือ ['A']
=	Set equality	['A', 'B'] = ['B', 'A'] คือ จริง
<>	Set inequality	['A', 'B'] <> ['A'] คือ จริง
<=	Subset	['A', 'B'] <= ['A] คือ เท็จ
>=	Superset	['A', 'B'] >= ['B'] คือ จริง

การอ่านและการเขียนเซต (Reading and Writing Sets)

คล้ายกับโครงสร้างข้อมูลอื่นๆ ส่วนใหญ่ คือ เซตไม่สามารถเป็นพารามิเตอร์ของกระบวนงาน Read หรือ Write มาตรฐานหน่วยข้อมูลซึ่งจะเก็บในเซต ต้องถูกอ่านทีละตัวและใส่ในเซตว่างตั้งแต่แรก โดยใช้ตัวดำเนินการส่วนรวมเซต

ตัวอย่าง 12.4 การอ่านเซต (Reading Sets)

กระบวนงาน ReadSet ในรูป 12.1 อ่านหนึ่งประโยคซึ่งจบด้วย . (period) และใส่รูปแบบตัวพิมพ์ใหญ่ ของอักษรแต่ละตัวในเซต ซึ่งแทนด้วยพารามิเตอร์ Letters (ชนิดเซต CahrSet)

ข้อความสั่ง

```
Letters := [ ]; {initialize letters.}
```

เริ่มต้นให้เซต Letters เป็นเซตว่าง และข้อความสั่ง if

```
if NextChar in ['A'..'Z'] then
   Letters := Letters + [NextChar]; {Insert a letter.}
```

ใส่อักษรตัวพิมพ์ใหญ่ตัวใหม่แต่ละตัวในเซต Letters

```
procedure ReadSet (var Letters {output} : CharSet);
{
   Reads a sentence terminated by a period and stores the uppercase form
   of each letter in Letters.
   Pre  :  None.
   Post :  Returns through Letters all the letters read before the period.
}
   const
      Sentinel = '.',      {sentinel character}
   var
      NextChar : Char; {next input character}

begin    {ReadSet}
   Letter := [  ];         {initialize Letters.}
   WriteLn ('Enter a sentence ending with symbol ', Sentinel);
   Read (NextChar);
   while NextChar <> Sentinel do
```

```
    {invariant :
       No prior value of NextChar is the sentinel and Letters contains each
       uppercase letter read so far.
    }
    begin
       NextChar := Upcase(NextChar); {Convert to uppercase.}
       if NextChar in ['A'..'Z'] then
          Letters := Letters + [NextChar]; {Insert a letter.}
       Read (NextChar)
    end {while}

  {assert : Last character read was the sentinel.}
end; {ReadSet}
```

รูป 12.1 กระบวนงาน ReadSet

ตัวอย่าง 12.5 การเขียนเซต (Writing Set)

การพิมพ์เซต เราต้องทดสอบว่าทุกค่าในชนิดฐานเป็นสมาชิกเซตหรือไม่ เฉพาะค่าที่เป็นสมาชิกเท่านั้นจึงจะพิมพ์กระบวนงาน PrintSet ในรูป 12.2 พิมพ์อักษรตัวพิมพ์ใหญ่ในเซตซึ่งแทนด้วยพารามิเตอร์ Letters ของมัน ถ้าเราเรียก PrintSet ที่มี ['A', 'C', 'Z'] เป็นพารามิเตอร์ของมัน PrintSet จะแสดงผลเป็น {A, C, Z}

```
procedure PrintSet (Letters {input} : CharSet);
{
   Prints the uppercase letters in set Letters.
   Pre  :  Letters is defined.
   Post :  Each uppercase letter in Letter is displayed.
}
```

```
    var
        NextLetter : Char; {loop-control variable}

begin {PrintSet}
    Write ('{');
    for NextChar := 'A' to 'Z' do
        if NextLetter in Letters then
            Write (NextLetter, ', '); {Print a set member.}
    WriteLn ('}')
end; {PrintSet}
```

รูป 12.2 กระบวนงาน PrintSet

แบบฝึกหัด 12.2 Self-Check

1. กำหนดให้ A เป็นเซต [1, 3, 5, 7], B เป็นเซต [2, 4, 6] และ C เป็นเซต [1, 2, 3] จงประเมินผลนิพจน์เซตข้างล่างนี้

   a) A + (B - C)

   b) A + (B * C)

   c) A + B + C

   d) (C - A) <= B

   e) [ ] <= A * B * C

   f) A + B <> [1..7]

   g) C + (A - C)

   h) C - (A - B)

   i) (C - A) - B

   j) A - C - [5, 7] = [ ]

   k) 2 in A

   l) 2 in A + B

2. กำหนดให้ A และ B เป็นเซตสองชุด จงเขียนนิพจน์แบบบูล เพื่อตรวจดูว่า A เป็นเซตย่อยแท้ (proper subset) ของ B หรือไม่ A เป็นเซตย่อยแท้ของ B ถ้าสมาชิกทุกตัวของ A อยู่ใน B และมีสมาชิกหนึ่งตัวใน B ที่ไม่ได้อยู่ใน A

เขียนโปรแกรม

1. จงดัดแปรกระบวนงาน PrintSet ให้พิมพ์เซตของชนิด DigitSet

12.3 สายอักขระความยาวแปรได้ (Variable - Length Strings)

ส่วนที่เหลือของบทนี้ อธิบายการประมวลผลของสายอักขระ (character strings) ซึ่งใช้เก็บข้อมูลตันฉบับ (textual data) สำหรับงานประยุกต์มากมาย ตัวอย่างเช่น ข้อความของหนังสือเล่มนี้ เขียนบนตัวประมวลผลคำ (word processor) และการประกอบหน้ากระดาษของมัน ใช้ระบบการจัดพิมพ์ด้วยคอมพิวเตอร์แบบตั้งโต๊ะ (desktop publishing system) และตั้งค่าชนิดโดยใช้คอมพิวเตอร์

โปรแกรมซึ่งทำงานกับสายอักขระเช่นการสร้าง junk mail ส่วนตัว สร้าง mailing labels และผู้รอบรู้เรื่องภาษา ใช้คอมพิวเตอร์เพื่อวิเคราะห์งานวรรณกรรม

ถ้าเราเคยใช้ตัวประมวลผลคำ เราจะคุ้นเคยกับชนิดของการดำเนินการซึ่งกระทำบนข้อมูลสายอักขระ ตัวอย่างเช่น บ่อยครั้งที่เราต้องการใส่อักขระหนึ่งตัวหรือมากกว่าหนึ่งตัวไปในข้อมูลสายอักขระมีจริง (existing data) การลบ (delete) ส่วนของสายอักขระ, การเขียนทับ (overwrite) หรือการแทนที่ (replace) สายอักขระย่อยชุดหนึ่ง ของสายอักขระหนึ่งชุดด้วยสายอักขระย่อยอีกชุดหนึ่ง ค้นหาสายอักขระย่อยเป้าหมาย หรือการเชื่อม (join) สายอักขระสองชุดเข้าด้วยกัน เพื่อให้เป็นสายอักขระหนึ่งชุดที่มีความยาวมากกว่าในหัวข้อนี้เราเรียนรู้ว่าจะกระทำการดำเนินการเหล่านี้โดยใช้กระบวนงานหรือฟังก์ชันในตัวของ Turbo Pascal อย่างไร (ไม่มีใน standard Pascal)

สายอักขระว่าง (The Null String)

ความยาวของตัวแปรสายอักขระเป็นพลวัต (dynamic) และถูกกำหนดโดยข้อมูลซึ่งเก็บไว้ในสายอักขระนั้น ความยาวนี้ไม่สามารถยาวมากกว่าที่ประกาศมากที่สุดสำหรับตัวแปรนั้น

บางครั้งเราจำเป็นต้องเริ่มต้นตัวแปรสายอักขระ ให้เป็นสายอักขระที่ไม่มีตัวอักขระใดๆ หรือสายอักขระว่าง

ถ้า Name เป็นตัวแปรสายอักขระ ข้อความสั่ง

Name := ' ';

WriteLn ('Legnth is ', Length (Name) : 1)

กำหนดสายอักขระว่างให้ Name และเรียกฟังก์ชัน length ให้แสดงข้อความ

Length is 0

สายอักขระว่าง หมายถึง สายอักขระที่ไม่มีตัวอักขระใดๆ (Null string is a string with zero characters.)

การเปลี่ยนสายอักขระให้เป็นเลข (Converting a String to a Number)

Turbo Pascal มีกระบวนงานในตัวชื่อ Val ซึ่งใช้สำหรับเปลี่ยนสายอักขระให้เป็นเลข สายอักขระที่ต้องการให้มีการเปลี่ยนแปลงนี้ ต้องเป็นสายอักขระตัวเลข

สายอักขระตัวเลข หมายถึง สายอักขระซึ่งมันไปตามกฎวากยสัมพันธ์สำหรับตัวเลข Pascal ที่ถูกต้อง ตัวอย่างเช่น

'123' , '0.12E5' เป็นต้น

สายอักขระตัวเลข หมายถึง สายอักขระซึ่ง content ของมัน เป็นตัวเลข (Numeric string is a string whose contents are a number.)

สมมติว่า IntName และ Error เป็นตัวแปรชนิด Integer ข้อความสั่งเรียกกระบวนงาน

Val ('1234', IntNum, Error);

ทำให้ค่าจำนวนเต็ม 1234 เก็บใน IntNum และ 0 เก็บใน Error กระบวนงาน Val กลับคืน การชี้ข้อผิดพลาดผ่านพารามิเตอร์ตัวที่สามของมัน (ตัวแปร Error) Val กลับคืนค่า 0 เมื่อไม่มีข้อผิดพลาดใดๆ

ข้อความสั่งเรียกกระบวนงาน

Val ('12#34%', IntNum, Error);

กลับคืนค่า 3 ไว้ที่ Error แสดงว่าตัวอักขระในตำแหน่งที่ 3 ไม่ใช่ตัวเลข เมื่อมีข้อผิดพลาดเกิดขึ้น การเปลี่ยนจะไม่ถูกกระทำ ดังนั้นค่าของ IntNum จึงไม่ถูกนิยาม (undefined)

ตาราง 12.2 แสดงรายการผลลัพธ์ของการเรียกกระบวน Val หลายชุด สมมติว่า ReadNum เป็นตัวแปรชนิด Real และ IntNum และ Error เป็นตัวแปรชนิด Integer ชนิดของพารามิเตอร์ตัวที่สอง กำหนดว่า ค่า Integer หรือค่า Real จะกลับคืน เช่นที่แสดงให้

เห็นในสองบรรทัดสุดท้าย ตัวอักขระว่าง (blanks) จะปรากฏในสายอักขระตัวเลขที่จะถูกเปลี่ยนไม่ได้

ตาราง 12.2 การใช้กระบวนงาน Val

Call	Values Returned
Val ('-3507', IntNum, Error)	IntNum คือ -3507 Error คือ 0
Val ('-3507', RealNum, Error	RealNum คือ -3507.0 Error คือ 0
Val ('1.23E3', RealNum, Error)	RealNum คือ 1230.0 Error คือ 0
Val ('1.23E 3', RealNum, Error)	RealNum คือ undefined Error คือ 6
Val (' 1.2E3', RealNum, Error)	RealNum คือ undefined Error คือ 1

ตัวอย่าง 12.6

ส่วนของโปรแกรม

```
repeat
   Write ('Enter an integer value> ');
   ReadLn (NumStr);
   Val (NumStr, InNum, Error)
unit Error = 0;
```

เก็บค่าจำนวนเต็มใน IntNum ถ้าสายอักขระตัวเลขอ่านไว้ใน NumStr (ตัวแปรสายอักขระ) กระบวนงาน Val กลับคืน ค่าตัวเลขของมันใน IntNum (ชนิด Integer) ถ้าสายอักขระตัวเลขถูกอ่าน, กระบวนงาน Val กลับคืนค่าไม่ใช่ศูนย์ใน Error (ชนิด Integer) และลูปถูกทำซ้ำๆ กัน

รหัสเช่นนี้สมเหตุสมผลกับอินพุตตัวเลขให้โปรแกรมถ้าผู้ใช้โปรแกรมใส่เลขจำนวนเต็มไม่ถูกต้อง ลูปนี้ป้องกันข้อผิดพลาด Invalid numeric format ถ้าไม่ใช้ลูปโปรแกรมจะเลิกกลางคัน (abort)

Syntax Display

กระบวนงาน Val (Val Procedure)

กำหนดให้ Year เป็นอักขระสี่ตัวซึ่งแทนปี ค.ศ. (ตำแหน่งที่ 9-12) ถ้า contents ของ Date คือ 'Jan 25, 2004' content ของสายอักขระความยาวแปรได้ Month, Day และ Year จะเป็น 'Jan' , '25' และ '2004' ตามลำดับ

ตัวอย่าง 12.8

กระบวนงาน printWords ในรูป 12.3 แสดงผลแต่ละคำที่พบในพารามิเตอร์ Sentence ของมัน บนบรรทัดแยกจากกัน สมมติว่ามีตัวว่างหนึ่งที่เสมอระหว่างคำ

```
procedure PrintWords (Sentence {input} : string);
{
   Displays each word of a sentence on a separate line.
   Pre  :  Variable-length string Sentence is defined.
   Post :  Each word in Sentence is displayed on a separate line.
}
   const
      WordSparator = ' ' ;

   var
      Word : string;     {each word}
      SentLen,           {length of Sentence}
      First,             {first character in each word}
      Next : Integer;    {position of next character}

begin   {PrintWords}
   {Display each word of sentence on a separate line.}
   First := 1;           {First word starts at position 1.}
   SentLen := Length (Sentence);
   for Next := 1 to SentLen do
```

```
        begin
            if (Sentence [Next] = WordSeparator then
                begin
                    {Get word.}
                    Word := Copy (Sentence, First, Next - First);
                    WriteLn (Word);
                    First := Next + 1
                end {if}
        end;  {for}

        {Display last word.}
        Word := Copy (Sentence, First, SentLen - First + 1);
        WriteLn (Word)
    end; {PrintWords}
```

รูป 12.3 กระบวนงาน PrintWords

ตัวแปร First ชี้ที่เริ่มต้นคำปัจจุบันเสมอ และเริ่มต้นให้มีค่าเท่ากับ 1 ระหว่างการกระทำการแต่ละครั้งของ for ลูป

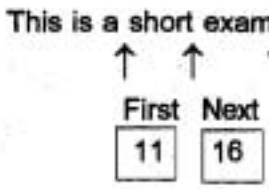
นิพจน์แบบบูล

```
Sentence [Next] = WordSeparator
```

ทดสอบว่า อักขระตัวถัดไปคือตัวว่าง (blank) หรือไม่ ถ้าใช่ สายอักขระย่อยที่ตำแหน่ง First จนถึง Next-1 ใน Sentence ถูกสำเนาไปไว้ที่ Word โดยข้อความสั่ง

```
Word := Copy (Sentence, First, Next - First);
```

ค่าของ First และ Next แสดงให้เห็นข้างล่างนี้ ก่อนคำที่สี่ของสายอักขระซึ่งเก็บใน entence จะถูกแสดงผล ค่าของ Next - First เท่ากับ 5 ดังนั้นสายอักขระย่อยที่แสดงคือ short



หลังจากพิมพ์แต่ละคำแล้ว First ถูกตั้งใหม่ให้เป็น Next + 1 ซึ่งเป็นตำแหน่งของอักขระตัวแรกของคำถัดไป หลังจากออกจากลูป

ข้อความสั่ง

```
Word := Copy (Sentence, First, SentLn - First + 1);
```

เก็บคำสุดท้ายของ Sentence ใน Word สำหรับประโยคนี้ First เท่ากับ 17 และค่าของพารามิเตอร์ตัวที่สามคือ (23 - 17 + 1) คำสุดท้ายที่แสดงคือ example

Syntax Display

ฟังก์ชัน Copy (Copy function)

Form : `Copy (source, index, size)`

ตัวอย่าง

```
Copy ('Mr. John Doe', 5, 4)
```

มีความหมายดังนี้ : ฟังก์ชันกลับคืน สายอักขระย่อยของ source ที่ตำแหน่ง index และประกอบด้วยอักขระ size ตัวพารามิเตอร์ source เป็นตัวแปร หรือค่าของสายอักขระ index และ size ต้องเป็นชนิด Integer

ข้อสังเกต ถ้า index มีขนาดใหญ่กว่าความยาวของ source สายอักขระว่างจะถูกกลับคืน ถ้า size ที่กำหนดมีตัวอักขระมากกว่าตัวอักขระที่เหลืออยู่ ตามหลังตำแหน่ง index จะมีเฉพาะตัวอักขระที่เหลืออยู่ของสายอักขระถูกกลับคืน

การต่อกันของสายอักขระ (Concatenating Strings)

ฟังก์ชัน concat คือการต่อกัน หรือรวมกัน ของสายอักขระ เพื่อให้เป็นสายอักขระใหม่หนึ่งชุด

การต่อกันของสายอักขระ หมายถึง การรวมสายอักขระเข้าด้วยกัน เพื่อให้เป็นสายอักขระชุดใหม่ (Concatenating string is joining strings together to form a new string.)

ตัวอย่าง 12.9

ข้อความสั่งข้างล่างนี้ ต่ออาร์กิวเมนต์สายอักขระของมันเข้าด้วยกัน และเก็บผลลัพธ์สายอักขระใน Name สำหรับ content ของสายอักขระ (สัญลักษณ์ □ หมายถึง ตัวว่าง)

Title	First	Last
Ms. □	Bo □	Peep

ข้อความสั่ง

```
Name := Concat (Title, Last);
```

เก็บสายอักขระ 'Ms. Peep' ใน Name

ข้อความสั่ง

```
Name := Concat (Title, First, Last);
```

เก็บสายอักขระ 'Ms. Bo Peep' ใน Name

ข้อความสั่ง

```
Name := Concat (Last, ',', First, Title);
```

เก็บสายอักขระ 'Peep, Bo Ms.' ใน Name

ตัวดำเนินการ + สามารถใช้ต่อสายอักขระได้เช่นกัน

ตัวอย่างข้อความสั่งกำหนดค่าก่อนหน้า เขียนได้ดังนี้

```
Name := Last + ',' + First + Title;
```

เมื่อตัวถูกดำเนินการเป็นสายอักขระ, Turbo Pascal ให้ความหมายตัวดำเนินการ + เป็น "concatenate" แทนที่จะเป็น "add" หรือ "union"

ตัวอย่าง 12.10

ฟังก์ชัน Reverse ในรูป 12.4 ใช้ฟังก์ชัน Concat เพื่อย้อนกลับ (reverse) สายอักขระซึ่งส่งไปยังสายอักขระอาร์กิวเมนต์ InString หลังจากการเริ่มต้นให้เป็นสายอักขระว่าง, TempString เก็บสายอักขระซึ่งกำลังจะประกอบเข้าด้วยกัน ตัวอักขระถูกนำมาครั้งละหนึ่งตัวจาก InString เริ่มจากอักขระตัวสุดท้าย และรวมเข้าด้วยกัน โดยต่อที่ตอนท้ายของ TempString ดังนั้นอักขระตัวแรกของ InString จึงเป็นอักขระตัวสุดท้ายที่รวมเข้าด้วยกันกับ TempString ตาราง 12.3 ตามรอยการกระทำการของฟังก์ชันนี้ เมื่อ InString คือ 'Turbo'

ตาราง 12.3 การตามรอย for ลูป เมื่อ InString คือ 'Turbo'

I	InString (I)	TempString
5	'o'	'o'
4	'b'	'ob'
3	'r'	'obr'
2	'u'	'obru'
1	'T'	'obruT'

```
function Reverse (InString : string) : string;
{  Reverses the string stored in InString  }

  var
    I      :  Integer;      {loop-control variable}
    TempString : string   {temporary reversed string}

begin {Reverse}
  TempString := ' ';        {Initialize TempString}

  for I := Length (InString) downto 1 do
    TempString := Concat (TempString, InString [I]);

  Reverse := TempString {Define result.}
end;  {Reverse}
```

รูป 12.4 ฟังก์ชันสำหรับการย้อนลำดับสายอักขระ

Syntax Display

ฟังก์ชัน Concat

Form : | Concat (string list) |

ตัวอย่าง

Concat ('Bo' , 'Diddly')

มีความหมายดังนี้ : อาร์กิวเมนต์สายอักขระ ใน string list รวมเข้าด้วยกันเรียงตามอันดับที่แสดงในรายการเป็นหนึ่งสายอักขระชุดใหม่

ข้อสังเกต ถ้าสายอักขระผลลัพธ์มีความยาวมากกว่า 255 ตัวอักขระ มันจะตัดทิ้งตัวอักขระหลังอักขระตัวที่ 255

การค้นหาสายอักขระ (String Search)

เมื่อประมวลผลข้อมูลสายอักขระ บ่อยครั้งเราจำเป็นต้องหาตำแหน่งของสายอักขระย่อยเฉพาะ ตัวอย่างเช่น เราต้องการทราบว่า สายอักขระ 'and ' มีอยู่ในประโยคหรือไม่ ถ้ามีอยู่ที่ไหน ถ้า Target เป็นสายอักขระความยาวเท่ากับสี่ มี content เป็น 'and ' ข้อความสั่ง

PosAnd := Pos (Target, Sentence)

กำหนดค่าตำแหน่งเริ่มต้น ของการเกิดครั้งแรกของ 'and ' ในสายอักขระ Sentence ให้ PosAnd

ถ้าสายอักขระ 'Birds and bee fly all day.' เก็บใน Sentence ค่าที่กำหนดให้ PosAnd คือ 7

ถ้าไม่มีสายอักขระ 'and ' อยู่ใน Sentence ฟังก์ชัน Pos กลับคืนค่าศูนย์

ตัวอย่าง 12.11

คอมไพเลอร์สามารถตรวจดูรูปแบบของข้อความสั่งจำนวนมาก โดยการตรวจสอบว่า ข้อความสั่งเริ่มต้นด้วยคำสงวนหรือไม่ ถ้ามีตัวว่างนำหน้า ให้ลบออกจาก Statement และถ้า Target คือสายอักขระความยาวเท่ากับสี่มี content เป็น 'for ' เงื่อนไข

Pos (Target, Statement) = 1

จะเป็นจริง เมื่อ Statement คือข้อความสั่ง for

คอมไพเลอร์สามารถตัดทอน (extract) สมาชิกเชิงวากยสัมพันธ์ของแต่ละข้อความสั่งได้ ตัวอย่างเช่น ข้อความสั่ง for มีรูปแบบวากยสัมพันธ์ดังนี้

for counter := initial to final do statement

ข้อความสั่งสองคำสั่งแรกข้างล่างนี้ใช้ฟังก์ชัน Pos เพื่อหาตำแหน่งของสายอักขระ 'for ' (contents ของ Target1) และ ':=' (contents ของ Target2)

ข้อความสั่ง if ทำสำเนาสายอักขระซึ่งอยู่ระหว่างสัญลักษณ์เหล่านี้ไปไว้ยังสายอักขระ Counter

```
PosFor := Pos (Target1, Statement);
PosAssign := Pos (Target2, Statement);
if (PosFor > 0) and (PosAssign > Posfor) then
   Counter := Copy  (Statement, PosFor + 4,
                          PosAssign - PosFor - 4)
```

เนื่องจากสายอักขระ 'for ' มีอักขระสี่ตัว ตำแหน่งเริ่มต้นของ Counter อยู่ที่ตำแหน่ง Posfor + 4 จำนวนของตัวอักขระใน Counter คำนวณโดยนิพจน์ PosAssign - PosFor - 4

ถ้าสายอักขระ 'for ID := 1 to N do X := X + 1' เก็บใน Statement ดังนั้น PosFor จะเป็น 1, PosAssign เท่ากับ 8 และ content ของ Counter คือสายอักขระ 'ID' (ความยาวเท่ากับ (8 - 1 - 4 เท่ากับ 3)

PosFor	PosAssign	Counter
1	8	ID □

Syntax Display

ฟังก์ชัน Pos

Form : Pos (pattern, source)

ตัวอย่าง

```
Pos ('you', 'Me/you')
```

มีความหมายดังนี้ : สายอักขระ source ถูกตรวจจากซ้ายไปขวาเพื่อหาตำแหน่งของการเกิดครั้งแรกของสายอักขระย่อย pattern

ถ้า pattern ถูกพบ, ค่ากลับคืนคือตำแหน่งของอักขระตัวแรกของ pattern ที่พบใน source กรณีอื่นๆ ค่ากลับคืนคือศูนย์

กระบวนงาน Delete และ Insert (Procedure Delete and Insert)

นอกจากฟังก์ชันจัดดำเนินการสายอักขระแล้ว Turbo Pascal ยังมีกระบวนงานเพื่อใส่และลบทิ้งสายอักขระย่อยด้วย

ตัวอย่าง 12.12

สมมติว่า Sentence ประกอบด้วยสายอักขระ

'This is the example'

ก่อนการเรียกกระบวนงานครั้งแรก ข้อความสั่งเรียกกระบวนงาน

```
Delete (Sentence, 1, 5)
```

ลบทิ้งอักขระห้าตัวแรกออกจากสายอักขระ Sentence, เริ่มต้นจากตำแหน่งที่ 1 ดังนั้น content ใหม่ของ Sentence กลายเป็น

' is the example'

ถ้า Target คือสายอักขระความยาวเท่ากับสี่ มี content เป็น 'the' ข้อความสั่งเรียกกระบวนงาน

```
Delete (Sentence, Pos (Target, Sentence), 4);
```

ลบการเกิดครั้งแรกของสายอักขระ 'the' ออกจาก Sentence ดังนั้น content ใหม่ของ Sentence กลายเป็น

'is example'

สุดท้ายข้อความสั่ง

```
PosTarg := Pos (Target, Sentence);
if PosTarg > 0 then
    Delete (Sentence, PosTage, Length (Target))
```

ลบการเกิดครั้งแรกของสายอักขระ Target ออกจาก Sentence จัดให้พบ Target ถ้า Target คือสายอักขระ 'ex' content ใหม่ของ Sentence กลายเป็น

'is ample'

ตัวอย่าง 12.13

สมมติว่า contents ของ Sentence คือสายอักขระ 'is the stuff?'

และ contents ของ NewString คือ 'Where'

ข้อความสั่งเรียกกระบวนงาน

Insert (NewString, Sentence, 1)

ใส่สายอักขระ 'Where' ที่ตำแหน่งที่ 1 ของสายอักขระ Sentence

ทำให้เปลี่ยนแปลง contents ของ Sentence เป็น

'Where is the stuff?'

สมมติว่า contents ของ Target คือ 'stuff' และ contents ของ NewString คือ '*#%!'

ข้อความสั่ง

```
PosStuff := Pos (Target, Sentence);
if PosStuff > 0 then
    Insert (NewString, Sentence, PosStuff)
```

ใส่สายอักขระ '*#%!' ข้างหน้าสายอักขระ 'stuff' ใน sentence ดังนั้น contents ใหม่ของ Sentence กลายเป็น

'Where is the '*#%! stuff'

ตัวอย่าง 12.14

กระบวนงาน Replace ในรูป 12.15 แทนที่สายอักขระเป้าหมายเฉพาะ (Target) ในสายอักขระต้นฉบับ (Source) ด้วยสายอักขระใหม่ (Pattern) มันใช้ฟังก์ชัน Pos เพื่อหาตำแหน่ง Target, กระบวนงาน Delete เพื่อลบออก และกระบวนงาน Insert เพื่อใส่ Pattern ในตำแหน่งของ Target มีการแสดงข้อความผิดพลาดถ้าไม่พบ Target

Syntax Display

กระบวนงาน Delete

Form : `Delete (source, index, size)`

ตัวอย่าง

```
Delete ('He**110', 3, 2)
```

มีความหมายดังนี้ : ตัวอักขระถัดไปจำนวน size ตัวถูกลบออกจากสายอักขระ source เริ่มต้นด้วยอักขระตัวที่อยู่ตำแหน่ง index พารามิเตอร์ source ต้องเป็นสายอักขระ และ size, index ต้องเป็นชนิด Integer

ข้อสังเกต

ถ้า index มีค่ามากกว่า Length (source) จะไม่มีตัวอักขระใดๆ ถูกลบทิ้ง ถ้า size ที่กำหนดมีตัวอักขระมากกว่าตัวอักขระที่เหลืออยู่ ส่วนที่เหลืออยู่ของสายอักขระจะถูกลบทิ้ง เริ่มต้นด้วยตำแหน่ง index

Syntax Display

กระบวนงาน Insert

Form : | Insert (pattern, destination, index) |

ตัวอย่าง

Insert ('bb', 'Bully', 3)

มีความหมายดังนี้ : สายอักขระ pattern ใส่ก่อนตัวอักขระปัจจุบันในตำแหน่ง index พารามิเตอร์ pattern และ destination ต้องเป็นสายอักขระ และ index ต้องเป็นชนิด Integer

ข้อสังเกต

ถ้าสายอักขระผลลัพธ์มีความยาวมากกว่า 255 ตัวอักขระ มันจะถูกตัดทิ้ง (truncated) หลังอักขระตัวที่ 255

```
procedure Replace (Target, Pattern {input} : string;
                   var Source {input/output} : string);
{
  Replaces first string Target in Source with Pattern if found.
  Pre  :  Target, Pattern, and Source are defined.
  Post :  Source is modified.
}
  var
    PostTarg : Integer;  {position of Target}
```

```
begin {Replace}
   PostTarg := Pos (Target, Source)  {Locate Target.}
   if PostTarg > 0 then
      begin
         Delete ( Source, PosTarg, Length (Target));
         Insert (Pattern, Source, PosTarg)
      end
   else
      WriteLn ('No replacement – ', Target,' not found.')
end; {Replace}
```

รูปที่ 12.5 กระบวนงาน Replace

แบบฝึกหัด 12.3

1. จงคำนวณหาผลลัพธ์ของการเรียกกระบวนงานและ function designator ข้างล่างนี้ สมมติว่าตัวแปรสายอักขระเป็นชนิด string [20] และ contents เริ่มต้นของ Temp1 คือ 'Abra' และ 'contents ของ Temp2 คือ cadabra'

   a) Magic := Concat (Temp1, Temp2)

   b) Length (Magic)

   c) HisMagic := Copy (Magic, 1, 8)

   d) Delete (HisMagic, 4, 3)

   e) Insert (Temp1, HisMagic, 3)

   f) Pos (Temp2, Magic)

   g) Pos (Temp1, Magic)

   h) Val ('1.234', RealNum, Error)

   i) Str (1.234 : 3 : 1, RealStr)

   j) Val (Temp1, RealNum, Error)

   k) Str (0.00345 : 3 : 1, RealStr)

2. ให้ Source, Target และ Destin เป็นตัวแปรสามตัวชนิด string ที่มีความยาวเท่ากับ 20 สมมติว่า Source เริ่มต้นด้วยนามสกุล (last name) ของพนักงาน ตามด้วย comma หนึ่งตัว และเว้นว่างหนึ่งที่ และจบด้วยชื่อของพนักงาน (first name) ตัวอย่างเช่น

last name, first name

จงใช้ฟังก์ชัน Pos และ Copy เพื่อเก็บชื่อของพนักงานใน Destin และเก็บนามสกุลใน Target

เขียนโปรแกรม

1. จงเขียนฟังก์ชันซึ่งจะสร้าง palindrome จากสายอักขระใดๆ ก็ตาม Palindrome หมายถึง สายอักขระซึ่งอ่านจากซ้ายไปขวาหรืออ่านจากขวาไปซ้ายจะเหมือนกัน (A palindrome is a string that reads the same backwords and forwards.)

ตัวอย่างเช่น กำหนดให้สายอักขระ 'abc' ฟังก์ชันของเราจะกลับคืนสายอักขระ 'abccba'

2. จงเขียนกระบวนงานซึ่งอ่านเลขจำนวนเต็มหนึ่งค่าในพิสัยที่นิยาม (กระบวนงานอินพุต) ไว้ในสายอักขระ จากนั้นเปลี่ยน (converts) ให้เป็นเลขจำนวนเต็ม และถ้ามีข้อผิดพลาดใดๆ ให้ใส่จำนวนเต็มค่าใหม่ กระบวนงานนี้ควรกลับคืนเลขจำนวนเต็มอยู่ในพิสัย (an in-range integer)

12.4 อธิบายการประมวลผลสายอักขระ (String Processing Illustrated)

เราได้ใช้ตัวบรรณาธิการข้อความ (text editor) เพื่อสร้างและตรวจแก้โปรแกรม Pascal มาแล้ว โปรแกรมที่ทันสมัยนี้ใช้คำสั่งงานพิเศษเพื่อย้ายเคอร์เซอร์ (cursor) ไปรอบจอภาพ และกำหนดการดำเนินการตรวจแก้ ถึงแม้ว่าเราจะไม่สามารถพัฒนาตัวบรรณาธิการเช่นนี้ได้ แต่เราสามารถเขียนโปรแกรมบรรณาธิการซึ่งทันสมัยน้อยกว่าได้

กรณีศึกษา ตัวบรรณาธิการข้อความ (Text Editor)

1) ปัญหา (Problem)

การออกแบบและการเขียนตัวบรรณาธิการ (editor) เพื่อกระทำการดำเนินการตรวจแก้ (editing operations) บางอย่างบนบรรทัดของข้อความ ตัวบรรณาธิการควรจะสามารถหาตำแหน่งของสายอักขระเป้าหมายเฉพาะ ลบทิ้งสายอักขระย่อย ใส่สายอักขระย่อยที่ตำแหน่งกำหนดให้ และแทนที่สายอักขระย่อยชุดหนึ่งด้วยสายอักขระย่อยอีกชุดหนึ่ง

2) วิเคราะห์ (Analysis)

เราสามารถใช้ฟังก์ชันและกระบวนงานจัดดำเนินการสายอักขระของ Turbo Pascal เพื่อกระทำการดำเนินการตรวจแก้ที่เกี่ยวข้องกันได้โดยง่าย เราจะเขียนโปรแกรมใส่สายอักขระหนึ่งชุด จากนั้นประมวลผลสายอักขระโดยตรงด้วยคำสั่งงานตรวจแก้

ความต้องการข้อมูล (Data Requirements)

อินพุตปัญหา (Problem Inputs)

```
Source   : string   {the source string}
Command  : Char     {each edit command}
```

เอาต์พุตปัญหา (Problem output)

```
Source   : string   {modified source string}
```

3) ออกแบบ (Design)

อัลกอริทึมเริ่มต้น (Initial Algorithm)

1. อ่านสายอักขระที่จะถูกตรวจแก้ไว้ใน Source
   (Read the string to be edited into source)
2. repeat
   3. อ่านคำสั่งงานตรวจแก้ (Read an edit command)
   4. กระทำการดำเนินการตรวจแก้ (Perform and edit operation)

   until done

การแบ่งละเอียดและโครงสร้างโปรแกรม (Refinements and Program Structure)

ขั้นที่ 4 ถูกกระทำด้วยกระบวนงาน DoEdit ซึ่งเรียกตัวดำเนินการที่เหมาะสมเพื่ออ่านสายอักขระข้อมูลชุดใดก็ตาม และให้กระทำการดำเนินการที่ต้องการ ส่วนของผังโครงสร้างสำหรับตัวบรรณาธิกรข้อความแสดงในรูป 15.6 ตัวแปรเฉพาะที่และอัลกอริทึมสำหรับกระบวนงาน DoEdit เป็นดังนี้

ตัวแปรเฉพาะที่ (Local Variables)

```
OldStr  : string   {substring to be found, replaced, or deleted}
NewStr  : string   {substring to be inserted}
Index   : Integer  {index to the string Source}
```

อัลกอริทึมสำหรับ DoEdit

1. case Command of

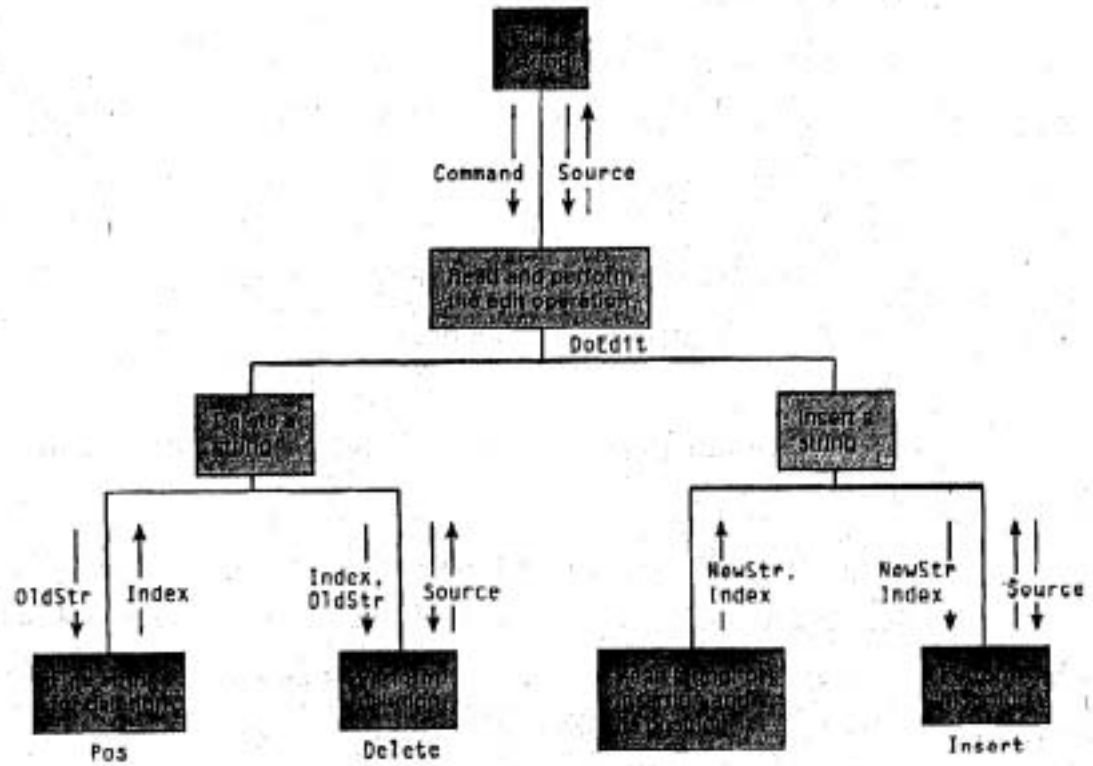
   'D' : อ่านสายอักขระย่อยซึ่งจะถูกลบทิ้ง และลบมันทิ้ง

   'I' : อ่านสายอักขระย่อยซึ่งจะถูกใส่และตำแหน่งของมัน และใส่มัน

   'F' : อ่านสายอักขระย่อยซึ่งจะถูกค้นหา และพิมพ์ตำแหน่งของมันถ้าพบมัน

   'R' : อ่านสายอักขระย่อยซึ่งจะถูกแทนที่ และแทนที่มันด้วยสายอักขระย่อยชุดใหม่

   end {case}



รูป 12.6 ผังโครงสร้างสำหรับโปรแกรมบรรณาธิกรข้อความ

4) การทำให้เกิดผล (Implementation)

โปรแกรมที่ครบบริบูรณ์แสดงให้เห็นในรูป 12.7 พร้อมทั้งการวิ่งตัวอย่าง (sample run)

```
program TextEdit;
{Performs text editing operations on a source string}

  const
    Sentined = 'Q';        {sentinel command}

  var
    Source : string;       {the string being edited}
    Command : Char;        {each edit command}

  {Insert procedure Repoace.}  {See Fig 12.5}
  procedure DoEdit (Command {input} : Char;
                    var Source {input/output} : string);
  }
    Performs the edit operation specified by Command.
    Pre  :  Command and Source are defined.
    Post :  One or more data strings are read and Source is modified if
            Command is 'D', 'I', 'F', or 'R'. If Command is 'Q', a message
            is displayed; otherwise, nothing is done.
  }
    var
      NewStr, OldStr : string;    {work strings}
      Index : Integer;            {index to string Source}
```

```
begin {DoEdit}
   {Perform the operation.}
   case Command of
      'D' :  begin {Delete}
                Write ('Delete what string ? ');
                ReadLn (OldStr);
                Index := Pos (OldStr, Source);
                if Index > 0 then
                   Delete (Source, Index, Length (OldStr))
                else
                   begin
                      Write (OldStr);
                      WriteLn (' not found')
                   end  {if}
             end;  {Delete}
      'I' :  begin {Insert}
                Write ('Insert what string? ');
                ReadLn (NewStr);
                Write ('At what position ? ');
                ReadLn (Index);
                Insert (NewStr, Source, Index)
             end; {Insert}
      'F' :  begin {Find}
                Write ('Find what string? ');
                ReadLn (OldStr);
                Index := Pos (OldStr, Source);
                if Index > 0 then
                   begin
```

```
                    Write (OldStr);
                    WriteLn (' found of position', Index : 3)
                  end
                else
                  begin
                    Write (OldStr);
                    WriteLn (' not found')
                  end {if}
              end;  {Find}
          'R' :  begin {Replace}
                    Write ('Replace old string? ');
                    ReadLn (OldStr);
                    Write ('With new string? ');
                    ReadLn (NewStr);
                    Replace (OldStr, NewStr, Source)
                 end;  {Replace}
          'G' :  WriteLn ('Quitting text editor.')
        else
          WriteLn ('Invalid edit caharacter')
        end {case}
    end; {DoEdit}

begin {TextEdit}
  {Read in the string to be edited.}
  WriteLn ('Enter the source string : ');
  ReadLn (Source);

  {Perform each edit operation until done.}
```

```
    repeat
        {Get the operation symbol.}
        WriteLn;
        Write ('Enter D(Delete), I(Insert),');
        Write ('F(Find), R(Replace), Q(Quit)> ');
        ReadLn (Command);
        Command := UpCase (Command) {Convert to uppercase}

        {Perform operation.}
        DoEdit (Command, Source);

        {Display latest string.}
        Write ('New source : ');
        WriteLn (Source)
    until Command = Sentinel
end.  {TextEdit}
```

Output Window

```
Enter the source string :
Mary had a cute little lamb.

Enter D(Delete), I(Insert), F(Find), R(Replace), Q(Quit) > f
Find what string? cute
cute found of position 12
New source : Mary had a cute little lamb.

Enter D(Delete), I(Inster), F(Find), R(Replace), Q(Quite) > i
Insert what string ? very
At what position ? 12
```

```
New Source : Mary had a very cute little lamb.
Enter D(Delete), I(Insert), F(Find), R(Replace), Q(Quit) > R
Replace old string ? lamb.
With new string ? lamb chop
New source : Mary had a very cute little lamb chop.

Enter D(Delete), I(Insert), F(Find), R(Replace), Q(Quite) > D
Delete what string ? very cute little
New source :          had a lamb chop.

Enter D(Delete), I(Insert), F(Find), R(Replace), Q(Quit) > q
Quitting text editor.
New source : Mary had a lamb chop.
```

รูป 12.7 โปรแกรมบรรณาธิกรข้อความและการวิ่งตัวอย่าง (Text Editor Program and Sample Run)

12.5 ข้อผิดพลาดร่วมของการเขียนโปรแกรม (Common Programming Errors)

ข้อผิดพลาดในการใช้เซต (Errors in Using Sets)

ข้อควรจำ ตัวแปรเซต คล้ายกับตัวแปรใดๆ ก็ตาม คือต้องถูกเริ่มต้นก่อนจึงจะนำไปจัดดำเนินการ เราอาจหลงไปสมมติว่าเซตว่างและจากนั้นเริ่มต้นประมวลผลโดยที่ไม่เริ่มต้นทำให้มันเป็นเซตว่าง, [ ], ตลอดการกำหนดค่าอย่างชัดแจ้ง

ตัวดำเนินการของ Pascal หลายตัวใช้ได้กับเซต ความหมายของตัวดำเนินการเปลี่ยนไปเมื่อตัวถูกดำเนินการของมันเป็นเซต แทนที่จะเป็นตัวเลข โปรดจำไว้ว่า จงใช้เซตหนึ่งหน่วย (unitset) (เซตที่มีสมาชิกหนึ่งตัว) เมื่อเราใส่หรือลบทิ้ง สมาชิกของเซต

การดำเนินการส่วนรวมของเซต (set union operation) ในนิพจน์

```
['A', 'E', 'O', 'U'] + 'I'     {incorrect set union}
```

ไม่ถูกต้อง และการเขียนใหม่เป็น

```
['A', 'E', 'O', 'U'] + ['I']   {corrcet set union}
```

เนื่องจากเซต ไม่สามารถเป็นตัวถูกกระทำของกระบวนงาน Read หรือ Write มาตรฐาน เราจึงต้องอ่านสมาชิกของเซตทีละหนึ่งตัว และใส่มันในเซตว่างเริ่มต้น โดยใช้ตัวดำเนินการยูเนียนของเซต การพิมพ์เซต เราต้องทดสอบแต่ละค่าในชนิดฐานของเซต สำหรับภาวะสมาชิกของเซต เฉพาะค่าเหล่านี้เท่านั้นในเซตที่จะพิมพ์ได้

ข้อผิดพลาดในการใช้สายอักขระ (Errors in Using Strings)

โปรดจำไว้ว่า ตัวดำเนินการสายอักขระจัดโดย Turbo Pascal และไม่มีให้ใช้ใน standard Pascal

โปรดตรวจสอบอย่างรอบคอบถึงการเรียงอันดับ (order) ของพารามิเตอร์ เมื่อใช้ตัวดำเนินการในตัว (built-in operators)

สำหรับฟังก์ชัน Pos และกระบวนงาน Insert สายอักขระต้นฉบับเป็นพารามิเตอร์ตัวที่สอง ไม่ใช่เป็นตัวแรก

เนื่องจากสายอักขระ ที่จริงคือแถวลำดับของตัวอักขระ จงใช้ตัวชี้แนะคอมไพเลอร์ (compiler disective) {$R+} เพื่อตรวจจับข้อผิดพลาดการตรวจสอบพิสัย (Range check errors)

ข้อสรุปตัวสร้างใหม่ของ Pascal (Summary of New Pascal Construct)

Construct	Effect
Set type Declaration	
type DigitSet = set of 0 . . 9;	ประกาศเซตชนิด DigitSet ซึ่งชนิดฐานของมันคือเซตของเลขโดดจาก 0 ถึง 9
var Digits, Primes : DigitSet;	Digits และ Primes เป็นตัวแปรเซตมีชนิดเป็น DigitSet
Set Assignment	
Digits := [ ];	Digits คือ เซตว่าง
Primes := [2, 3, 5] + [7];	Primes คือ เซต [2, 3, 5, 7]
Digits := Digits + [1 . . 3];	Digits คือ เซต [1, 2, 3]
Digtis := [0 . . 9] - [1, 3, 5, 7, 9];	Digits คือ เซต [0, 2, 4, 6, 8]
Digits := [1, 3, 5, 7, 9] * Primes	Digits คือ เซต [3, 5, 7]

Construct	Effect
Set Relations	
Primes <= Digits	True ถ้า Primes เป็นเซตย่อยของ Digits
Primes >= [ ]	เป็น True เสมอ
Prime <> [ ]	True ถ้า Prime มีสมาชิก
[1, 2, 3] = [3, 2, 1]	True เพราะว่าเซตเป็นรายการแบบไม่มีอันดับ (unordeved)
String Declaration	

const
  Capacity = 10;
type
  StringType = string [Capacity];
var
  FirstName, LastName, TempName
  : Stringtype;
	FirstName, LastName และ TempName เป็นสายอักขระความยาวแปรได้ (ความจุของสายอักขระเท่ากับ 10 ตัวอักขระ

String Assignment	
FirstName := 'Daffy';	เก็บ 'Daffy' ใน FirstName
LastName := 'Duck';	เก็บ 'Duck' ใน lastName
TempName := LastName;	และ TempName
String Copy	
TempName := Copy (FirstName, 1, 3);	ทำสำเนา 'Daf' ไว้ที่ TempName
String Concatenation	
TempName := Concat (FirstName, LastName);	เก็บ 'DaffyDuck' ใน TempName
TempName := FirstName + LastName;	
String Search	
Pos ('Du', FirstName);	กลับคืน 0 ('Du' not found)
Pos ('Du', LastName);	1 ('Du' พบที่ 1)
Pos ('Du', TempName);	กลับคืน 6 ('Du' พบที่ 6)

Construct	Effect
String Deletion	
Delete (TempName, 7, 2);	เปลี่ยน TempName เป็น 'DaffyDk'
String Insection	
Insert ('uc' TempName, 7);	เปลี่ยน TempName เป็น 'DaffyDuck'

แบบฝึกหัด Quick-Check

1. เอกภพสัมพัทธ์ (universal set) คืออะไร
2. การดำเนินการชุดใดที่มีสมาชิกมากที่สุด :

a) set union, an intersection หรือ a difference การดำเนินการเหล่านี้ชุดใดไม่ใช่การสลับที่ (commutative)

3. เราสามารถมีเซตซึ่งมีชนิดฐาน (base type) เป็นชนิด Integer หรือ Char ได้หรือไม่ จงอธิบาย ชนิดพิสัยย่อย ที่มี host type เป็น Integer หรือ Char
4. มีความแตกต่างอย่างไรหรือไม่กับอันดับสมาชิกของเซตที่ใส่เข้าไป และมีความแตกต่างอย่างไรหรือไม่ ถ้าสมาชิกที่ใส่มีมากกว่าหนึ่งครั้งในเซตเดิม
5. กำหนดให้เซต Set1 คือ [1..3] จงบอก contents ของเซตต่อไปนี้

a) Set2 := Set1 + [4, 5, 6];

b) Set3 := Set1 - Set2;

c) Set4 := Set3 + [4, 7];

d) Set5 := Set4 + [4, 6];

e) Set6 := Set5 * Set2;

ตัวอย่างข้อสอบชุดที่ 1

ข้อ 1

1.1 จงพิจารณาเลขต่อไปนี้

275	3,475	7.4	.1475	6000
6E3	275.0	0.001	27365982	0
10E4	0.074E3	0.1E999	275.	0.0620
-741E-1	0E0	25.3E+01	-74.1	0.0

a) มีเลขอะไรบ้างซึ่งเป็นจำนวนเต็ม (integers) ที่ถูกต้องใน Pascal

b) มีเลขอะไรบ้างซึ่งเป็น invalid integer ใน Pascal

c) มีเลขอะไรบ้างซึ่งเป็นจำนวนจริง (real numbers) ที่ถูกต้องใน Pascal

d) มีเลขอะไรบ้างซึ่งเป็น invalid real number ใน Pascal

e) มีค่าใดบ้างซึ่งเป็นค่าเหมือนกันใน Pascal

f) มีค่าใดบ้างซึ่งถูกต้องใน Pascal แต่ไม่สามารถปฏิบัติงานให้เกิดผลได้ ให้อธิบายและยกเหตุผลประกอบด้วย

1.2 ชื่อต่อไปนี้มีชื่อใดบ้างซึ่งเป็น identifier ที่ถูกต้อง และมีชื่อใดบ้างซึ่งเป็น identifier ไม่ถูกต้อง ให้อธิบายว่าทำไมชื่อนั้นจึงใช้ไม่ได้

ITEM_2	ufo	TO-MORROW	averylongnameindeed		
IN-PUT	array	PC49	McDougall	X99999	$100
Function	case	AND	program	boolean	writeln
A-number		_counter	ID#	Item2	

ข้อ 2

2.1 สมมติว่า x, y และ z เป็นตัวแปรชนิดจำนวนเต็มหลังจากการกระทำการลำดับของข้อความสั่งข้างล่างนี้แล้ว ค่าสุดท้ายของตัวแปร x, y และ z คืออะไรให้แสดงทุกขั้นตอน

```
x := 50 ;
y := 340 ;
z := x + y - 190 ;
x := 17 ;
```

```
y  := x + z ;
z  := x + 200 ;
z  := x + z - 200
```

2.2 จงใส่เครื่องหมายวงเล็บกำกับนิพจน์ข้างล่างนี้เพื่อให้มีความชัดเจน จากนั้นคำนวณหาค่าของนิพจน์แต่ละชุด

a) 6.75 - 12.3 / 3

b) 6 * 11 - 42 div 5

c) 175 mod 15 div 3 * 65

d) 13 + 7 * 5 - 4 * 5 div 2

e) 11 mod 4 div 2 < > 0

f) ('A' >= 'Z') or ('9' >= '8') and ('A' < 'I')

ข้อ 3

3.1 จงอธิบายการทำงานของกระบวนงานข้างล่างนี้

```
procedure puzzle (N : nonnegative integer) ;
   begin
      repeat
         write (N mod 10) ;
         N := N div 10
      until N = 0
   end;
```

เมื่อกำหนดให้พารามิเตอร์รูปนัย n มีค่าเท่ากับ 649 เอาต์พุตของโปรแกรมนี้คืออะไร

3.2 a) จงเขียนโปรแกรมรับอินพุตเป็นเลขสองจำนวนจากคีย์บอร์ด เก็บค่าที่ตัวแปรชื่อ speed1 และ speed2 ตามลำดับ จากนั้นให้ตรวจสอบว่า เลขตัวใดมีค่ามากกว่าให้เก็บไว้ที่ตัวแปรชื่อ highspeed เอาต์พุต พิมพ์เลขทั้งสามจำนวนโดยให้มีทศนิยมสองตำแหน่ง

b) จงเขียนกระบวนงานชื่อ findlarger รับเลขสองจำนวน ชื่อ first และ second ตามลำดับมี data type เป็น real จากนั้นตรวจสอบว่าเลขตัวใดมีค่ามากกว่าให้เก็บไว้ที่ larger ซึ่งเป็น variable formal parameter

ข้อ 4

4.1 จากโปรแกรมข้างล่างนี้ จงอธิบายการทำงานและเอาต์พุตของโปรแกรมคืออะไร

```
program exam_41 (output);
   var a, b, k : integer ;
   function f (x : integer) : integer ;
      begin
         f := 3 * x + 2
      end ;
   begin
      a := 0;
      for k := 1 to 5 do a := a + k ;
      b := f (a) ;
      writeln (b)
   end.
```

4.2 จากโปรแกรมข้างล่างนี้ จงอธิบายการทำงานและเอาต์พุตของโปรแกรมคืออะไร

```
program exam_42 (output) ;
   const morgansubrange = 1 .. 10 ;
   var k , m : integer ;
      a : array [morgansubrange] of char;
   begin
      a := 'ABCDEFGHIJ' ;
      for k := 1 to 10 do
         begin
            m := k mod 2;
            if m = 1 then write (a[k])
         end ;
      writen ('xyz')
   end.
```

ข้อ 5

5.1 จงเขียนโปรแกรมอ่านเลขจำนวนเต็มบวกหนึ่งจำนวน แล้วตรวจสอบว่าถ้าเป็นเลขหนึ่งหลัก พิมพ์คำว่า one-degit, ถ้าเป็นเลขสองหลัก พิมพ์คำว่า two-digit, ถ้าเป็นเลขสามหลัก พิมพ์คำว่า three-digit, กรณีอื่นๆ ให้พิมพ์คำว่า invalid-data

a) ให้ใช้คำสั่ง if

b) ใช้คำสั่ง case

5.2 จากโปรแกรมข้างล่างนี้ จงอธิบายการทำงาน และเอาต์พุตของโปรแกรมคืออะไร

```
program exam_52 (output, tfile) ;
var   ch    :  char ;
      tfile  :  text ;
begin
      assigh (tfile, 'my.txt') ;
      rewrite (tfile) ;
      writeln (tfile,   'HERE IS A LINE.') ;
      close (tfile) ;
      reset (tfile) ;
      while not eof (tfife) do
         begin
            read (tfile, ch) ;
            if ch = ' ' then write ('*')
            else write (ch)
         end ;
      writeln
end.
```

ตัวอย่างข้อสอบชุดที่ 2

ข้อ 1

1.1 จงพิจารณาเลขต่อไปนี้

2.06	3,475	59.01	.1475	$123
1.000	275.0	-12	27365982	0
10E4	12.3E-02	0.1E999	275.	123
-741E-1	1E4	25.3E+01	-1.27E-03	0.0

a) มีเลขอะไรบ้างซึ่งเป็นจำนวนเต็ม (integers) ที่ถูกต้องใน Pascal

b) มีเลขอะไรบ้างซึ่งเป็น invalid integer ใน Pascal

c) มีเลขอะไรบ้างซึ่งเป็นจำนวนจริง (real numbers) ที่ถูกต้องใน Pascal

d) มีเลขอะไรบ้างซึ่งเป็น invalid real number ใน Pascal

e) มีค่าใดบ้างซึ่งเป็นค่าเหมือนกันใน Pascal

1.2 ชื่อต่อไปนี้มีชื่อใดบ้างซึ่งเป็น identifier ที่ถูกต้อง และมีชื่อใดบ้างซึ่งเป็น identifier ไม่ถูกต้อง ให้อธิบายว่าทำไมชื่อนั้นจึงใช้ไม่ได้

A	Example1	TO-MORROW	averylongnameindeed		
IN-PUT	array	PC49	McDougall	X99999	$100
Output	1x	end	program	boolean	writeln
A-number	_counter	ID#	Item2		

1.3 จงเขียนนิพจน์ Pascal ที่ถูกต้องจากสูตรข้างล่างนี้

$7 + A(B + C^2)$

$A - B + \sqrt{C(D + E)}$

1.4 ให้ A = 1, B = 2 และ C = 3 จงประเมินผลนิพจน์ข้างล่างนี้

((A > B) OR (B > C)) AND (C > A)

NOT (A = B) AND (C > A)

ข้อ 2

2.1 จงศึกษาโปรแกรมข้างล่างนี้ จากนั้นอธิบายและแสดงเอาต์พุต

```
Program final_exam(output) ;
var a, b, c, d : integer ;
  procedure mystery ;
  var x, y : integer ;
  begin
    x := 10 ;
    y := 12 ;
    a := x - y ;
    b := x + y ;
  end ;
  begin
    mystery ;
    c := a + b ;
    d := a - b * c ;
    writeln(d)
  end.
```

2.2 จงอธิบายโปรแกรมข้างล่างนี้พร้อมทั้งยกตัวอย่างประกอบด้วย

```
Program example_selectcase(input, output);
uses crt ;
var n : integer ;
procedure proc 1 ;
begin
  writeln('one digit number')
end;
procedure proc2;
begin
```

```
    writeln('two digit number')
end;
procedure proc3;
begin
    writeln('three digit number')
end;
begin
clrscr ;
    write('enter a number from 1 to 999 :');
    readln (n);
    case n of
      1..9     :  proc1 ;
      10..99   :  proc2 ;
      100..999 :  proc3 ;
    end
end.
```

ข้อ 3

3.1 จงอธิบายรหัสข้างล่างนี้ และเอาต์พุตของโปรแกรมนี้คืออะไร

```
type
    days = (mon, tue, wed, thr, fri, sat, sun) ;
var
    today : days ;
begin
    writeln(ord(mon)) ;
    writeln(ord(thr)) ;
    today := friday ;
    today := pred(today) ;
    if today = thr then writeln('Hello') ;
```

3.2 อธิบายรหัสข้างล่างนี้ และเอาต์พุตของโปรแกรมนี้คืออะไร

```
type
   multi_array = array[1..15, 1..20] of integer ;
var
   j, k : integer ;
   A : multi_array ;
begin
   for j := 1 to 15 do
      for k := 1 to 15 do
         A[j, k] := j + k ;
end;
```

ข้อ 4

จงเขียนโปรแกรมโดยใช้ข้อความสั่ง for เปลี่ยนอุณหภูมิจากองศาฟาเรนไฮต์ (fahrenheit) ให้เป็นองศาเซลเซียส (celsius) จากช่วงอุณหภูมิ - 100 ถึง 300 องศาจากนั้นเอาต์พุตให้พิมพ์ทั้งสองค่า โดยใช้สูตรข้างล่างนี้

$$C = 32 + (9/5) F$$

ข้อ 5

จงเขียนผังงาน (flowchart) และโปรแกรมอ่านเลขจำนวนเต็มบวก สามจำนวนเรียงลำดับจากน้อยไปหามาก ซึ่งแทนความยาวด้านสามด้านของสามเหลี่ยมหนึ่งรูป แล้วตรวจสอบว่าทั้งสามด้านนี้

a) ไม่สามารถสร้างเป็นรูปสามเหลี่ยม

b) สามารถสร้างเป็นรูปสามเหลี่ยมด้านเท่า

c) สามารถสร้างเป็นรูปสามเหลี่ยมหน้าจั่ว

หรือ d) สามารถสร้างเป็นรูปสามเหลี่ยมด้านไม่เท่า

จากนั้นคำนวณและพิมพ์พื้นที่ของรูปสามเหลี่ยมโดยใช้สูตรข้างล่างนี้

$$\text{area} = \sqrt{s(s - a)(s - b)(s - c)}\ ,\ s = \frac{a + b + c}{2}$$

เมื่อ a, b, c คือด้านของรูปสามเหลี่ยม

ข้อแนะนำ เส้นตรง 3 เส้น จะสร้างเป็นรูปสามเหลี่ยมได้ก็ต่อเมื่อ เส้นที่ยาวที่สุดต้องน้อยกว่าผลบวกของความยาวของสองเส้นที่สั้นกว่า

ตัวอย่างข้อสอบชุดที่ 3

ข้อ 1

1.1 จงบอกว่าชื่อต่อไปนี้เป็นไอเดนติไฟเออร์ (identifiers) ประเภทใด

a) Pascal reserved word b) Standard identifier

c) Valid identifer หรือ d) Invalid identifier

end .....

Rate .....

123XYZ .....

Start .....

ThisIsAlongOne .....

Bill .....

const .....

X=Z .....

Sue's .....

1.2 จงบอกว่าค่าสัญพจน์ (literal values) ต่อไปนี้ถูกต้องหรือไม่ถูกต้องใน Pascal ถ้าไม่ถูกต้องให้บอกเหตุผล จากนั้นให้ระบุว่าค่าสัญพจน์ที่ถูกต้องแต่ละตัวนั้นมีแบบชนิดข้อมูล (data type) ชื่ออะไร

'XYZ' .....

'' .....

$25.123 .....

1.15E3 .....

-9999 .....

12345 .....

"x" .....

'x' .....

True .....

'True' .....

ข้อ 2

2.1 ข้อความสั่งกำหนดค่า (assignment statements) ต่อไปนี้ถูกต้องหรือไม่ ถ้าไม่ถูกต้องให้อธิบายเหตุผล ถ้าข้อใดถูกต้องให้บอกผลลัพธ์ของการกำหนดค่าที่ถูกต้อง สมมติว่า R เป็นข้อมูลชนิด Real, B เป็นข้อมูลชนิด Boolean, C เป็นข้อมูลชนิด Char, และ S เป็นข้อมูลชนิด String

a) R := 3.5 + 5.0 .....

b) C := 'My name' .....

c) S := Your name .....

d) B := Boolean .....

e) R := 2 * 5 .....

2.2 จงอธิบายและแสดงรูปแบบของบรรทัดผลลัพธ์ เมื่อ X มีค่าเท่ากับ 3.456 และ N มีค่าเท่ากับ 890 จากข้อความสั่งข้างล่างนี้

```
WriteLn ('Some numbers are ' , X : 3 : 1 , '*' , X : 6 : 3 , N : 4 , '*' , N : 1) ;
```

2.3 จงวาดรูปต้นไม้การประเมินผล (Evaluation tree) ของนิพจน์ข้างล่างนี้

```
Z - (A + B div 2) + W * Y
```

ข้อ 3

3.1 a) จงอธิบายการทำงานของกระบวนงานข้างล่างนี้ และเอาต์พุตคืออะไร

```
procedure Nonsense ;
begin {Nonsense}
   WriteLn ('******') ;
   WriteLn ('*    *') ;
   WriteLn ('******') ;
end ; {Nonsense}
```

b) จงอธิบายตัวโปรแกรม (program body) ข้างล่างนี้ และเอาต์พุตคืออะไร

```
begin
   Nonsense ;
   Nonsense ;
   Nonsense ;
end.
```

3.2 a) จากข้อความสั่ง case ข้างล่างนี้ จงเปลี่ยนเป็นข้อความสั่ง if ซึ่งมีความหมายเหมือนกัน (are equivalent)

```
case X = Y of
   True : WriteLn ('Equal') ;
   False : WriteLn ('unequal) ;
end {case}
```

b) จงเขียนข้อความสั่ง case ซึ่งสมนัย (corresponds to) กับข้อความสั่ง nested-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')
```

ข้อ 4

4.1 จงตรวจสอบและแก้ไขข้อผิดพลาดวากยสัมพันธ์ (syntax errors) ในข้อความสั่งข้างล่างนี้ หลังจากนั้นให้เขียน ผังงาน (flowchart) ที่ถูกต้อง

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

4.2 จงเขียนโปรแกรมอ่านค่าของอินพุต N จากนั้นพิมพ์แถวของตัวเลขจำนวน N แถว มีรูปแบบดังนี้

แถวที่หนึ่ง 1 2 3 4 ...N แถวที่สอง 2 3 4 5 ...N + 1

แถวที่สาม 3 4 5 6 ...N + 2 เช่นนี้เรื่อยไป **ตัวอย่าง เช่น** สำหรับค่าอินพุตเท่ากับ 5 พิมพ์

```
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

5.1 จงศึกษาโปรแกรมข้างล่างนี้จากนั้นอธิบายเรื่องสโคป (scope) ของไอเดนติไฟเออร์ทุกตัวในโปรแกรม และแสดงผลของผลลัพธ์โปรแกรม

```
program ScopeRules ;
  var
    W, X, Y : real ;
procedure Change (var X {input/out} : Real) ;
  var
    W, Z : Real ;
begin
  W := 35.0 ;
  X := 6.0 ;
  Y := Y + 1.0 ;
  Z := 3.0 ;
  WriteLn ('W' : 5, 'X' : 5, 'Y' : 5, 'Z' : 5) ;
  WriteLn (W : 5 : 1 , X : 5 : 1 , Y : 5 : 1 , Z : 5 : 1 , 'in Change')
end;
begin
  W := 5.5 ;
  X := 2.0 ;
  Y := 3.0 ;
  Change (W) ;
  WriteLn (W : 5 : 1 , X : 5 : 1 , Y : 5 : 1 , 'in ScopeRules' : 19)
end.
```

5.2 จงศึกษากระบวนงานข้างล่างนี้จากนั้นอธิบายการกระทำการเรียก WhatDo (4) และเอาต์พุตของโปรแกรมคืออะไร

```
procedure WhatDo (I : Integer) ;
begin
  if I > 1 then
    begin
      Write (I : 2) ;
      WhatDo (I - 1) ;
      Write (I : 2)
    end
  end ; {WhatDo}
```

ตัวอย่างข้อสอบชุดที่ 4

ข้อ 1

1.1 จงบอกว่าชื่อต่อไปนี้เป็นไอเดนติไฟเออร์ (identifiers) ประเภทใด

a) Pascal reserved word b) Standard identifier

c) Valid identifer หรือ d) Invalid identifier

7up .....

repeat .....

123XYZ .....

Lima, Ohio .....

program .....

ListOfEmployee .....

Bill .....

maxint .....

X * Y .....

output .....

1.2 จงหาค่าของนิพจน์ข้างล่างนี้

a) 4 * 11 MOD (trunc (trunc (8.9) / sqrt (16))) + ord ('A')

b) ord ('a') - abs (-11.2) + sqrt (round (15.51))

ข้อ 2

2.1 จงหาข้อผิดพลาดในโปรแกรมข้างล่างนี้และแก้ไขให้ถูกต้อง

```
program Errors (output) ;
(* there are eleven errors. *
var
   Day     :  char ;
   Percent :  real
   A , B   ;  int ;
begin (Program)
```

```
    Day  =  'M' ;
    Percentate := 72 / 10 ;
    A  := 5 ;
    B  := A * 3.2 ;
    Writln (A , B : 20) ;
    Writeln (Day : 10 : 2) ;
    Writeln (A + B : 8 , Percent : 8)
  end
```

2.2 จงอธิบายและแสดงเอาต์พุต ของส่วนโปรแกรมข้างล่างนี้

```
Indent   := 1 ;
for Ch   := 'A' to 'Z' DO
   begin
      Writeln (Ch : Indent) ;
      Indent := Indent + 1
   end ;
```

ข้อ 3

3.1 จงอธิบายการทำงานของโปรแกรมข้างล่างนี้ และแสดงเอาต์พุตที่ถูกต้อง

```
program ExerciseNine (output) ;
var
   A  :  integer ;
procedure Sub1 (A : integer) ;
   begin
      A  := 20 ;
      WriteLn (A)
   end;
procedure Sub2 (var A : integer) ;
   begin
      A  := 30 ;
```

```
    WriteLn (A)
  end ;
begin
  A := 10 ;
  WriteLn (A) ;
  Sub1 (A) ;
  WriteLn (A) ;
  Sub2 (A) ;
  WriteLn (A)
end.
```

3.2 กำหนดให้ Score เป็นตัวแปรชนิด integer มีค่าเป็น 0, 1, 2, 3, ..., 10 จากข้อความสั่ง case ข้างล่างนี้ จงเขียนใหม่โดยใช้ข้อความสั่ง nested- if ซึ่งมีความหมายเหมือนกัน (are equivalent)

```
case Score of
   10, 9 : Grade := 'A' ;
    8,  7 : Grade := 'B' ;
    6, 5 : Grade := 'C' ;
    4, 3, 2, 1, 0 : Grade := 'F' ;
end ; { of case Score }
```

ข้อ 4

4.1 กำหนดให้ x เป็นแถวลำดับหนึ่งมิติมีสมาชิก 8 ตัวดังนี้

16.0 12.0 6.0 8.0 2.5 12.0 14.0 -54.5

จงอธิบายว่าแต่ละข้อความสั่ง (statement) ทำอะไรได้ผลลัพธ์อะไรและหลังจากข้อความสั่งทั้งหมดถูกกระทำการแล้ว สมาชิกทุกตัวของแถวลำดับ x มีค่าเป็นอะไร

```
    i   := 3 ;
  x[i]  := x[i] + 10.0 ;
x[i - 1]  := x[2 * i - 1] ;
x[i + 1]  := x[2 * i] + x[2 * i + 1] ;
```

```
for i  := 5 to 7 do
x[i]   := x[i + 1] ;
for i  := 3 downto 1 do
x[i + 1] := x[i]
```

4.2 กำหนดให้ A เป็นแถวลำดับหนึ่งมิติ มีสมาชิกแปดตัว มีค่าดังนี้

16.0 12.0 6.0 8.0 2.5 12.0 14.0 -54.5

จงเขียนข้อความสั่งให้ทำสิ่งต่อไปนี้

a) แทนที่สมาชิกตัวที่สามด้วย 7.0

b) สำเนาสมาชิกตัวที่ห้าไว้ในสมาชิกตำแหน่งที่หนึ่ง

c) ลบสมาชิกตัวที่หนึ่งออกจากสมาชิกตัวที่สี่แล้วเก็บผลลัพธ์ไว้ในสมาชิกตัวที่ห้า

d) เพิ่มค่าสมาชิกตัวที่หกด้วย 8

e) จงหาผลบวกของสมาชิกห้าตัวแรก

f) จงคูณสมาชิกแต่ละตัวของสมาชิกหกตัวแรกด้วย 2 แล้วแทนผลคูณแต่ละตัวในสมาชิกของแถวลำดับ AnswerArray

g) แสดงผล (display) สมาชิกทุกตัวที่มีตำแหน่งเป็นเลขคู่ บนหนึ่งบรรทัด

ข้อ 5

จงเขียนกระบวนงานชื่อ FillCountArray สร้างแถวลำดับหนึ่งมิติชื่อ LetterCount ซึ่งมีสมาชิกทั้งหมด 26 ตัว มีค่าเป็นเลขจำนวนเต็ม เริ่มต้นให้ทุกตัวมีค่าเท่ากับศูนย์และดรรชนี (index) เป็นข้อมูลชนิด subrange 'A' .. 'Z' วาดรูปประกอบด้วย

ตัวอย่างข้อสอบชุดที่ 5

ข้อ 1

จงศึกษาส่วนโปรแกรมข้างล่างนี้แล้วตอบคำถามตอนท้าย

```
{incorrect if statement}
if x > Y then
begin
x := x - 10.0
writeLn ('X Bigger')
end
else
writeLn('XSmaller')
writeLn ('Y is ' , Y)
```

(a) จงบอกที่ผิดในส่วนโปรแกรมและอธิบายว่าทำไมจึงผิด

(b) จงใส่เครื่องหมาย cemicolons (;) ในตำแหน่งที่ถูกต้องเพื่อไม่ให้เกิดข้อผิดพลาดวากยสัมพันธ์และย่อหน้าตามข้อตกลงเพื่อให้ทำให้โปรแกรมอ่านง่ายจากนั้นเขียนส่วนโปรแกรมชุดถูกต้อง

(c) ถ้าเราลบทิ้งคู่ begin และ end จงอธิบายว่าทำไมส่วนโปรแกรมชุดนี้จึงคอมไพล์ไม่ผ่าน

(d) ถ้าเราใส่คู่ begin และ end ปิดล้อมสองบรรทัดสุดท้ายในส่วนโปรแกรมซึ่งแก้ไขถูกต้องแล้วในข้อ (a) จงอธิบายผลลัพธ์ที่เกิดขึ้น

ข้อ 2

2.1 พิสัยย่อย (subranges) ต่อไปนี้ชุดใดถูกต้องและชุดใดไม่ถูกต้องให้อธิบาย

(a) 1 .. MaxInt

(b) -5.0 .. 5.0

(c) 0 .. '9'

(d) 'ACE' .. 'HAT'

(e) -15 .. 15

2.2 การประกาศชนิดแจงนับ (enumerated type) ต่อไปนี้ไม่ถูกต้องเพราะเหตุใดให้อธิบายและแก้ไขให้ถูกต้อง

```
type Prime = (2, 3, 5, 7, 9, 11, 13);
```

2.3 จงพิจารณาการประกาศชนิดแจงนับต่อไปนี้

```
type (Class = (Frosh, Soph, Jr, Sr);
```

จากนั้นจงหาค่าของ

(a) Ord(Succ(Pred(Soph)))

(b) Pred(pred(Jr))

ข้อ 3

3.1 จงแสดงผลลัพธ์ของข้อความสั่ง ข้างล่างนี้

```
for NextCh := 'A' to 'Z' do
    Write(NextCh) ;
```

3.2 จงอธิบายความหมายของนิพจน์แบบบูล (boolean expression) ข้างล่างนี้ จากนั้นให้เขียนส่วนเติมเต็ม (complement) ของมัน

```
(Ch >= 'a') and (Ch <= 'z')
```

3.3 (a) จงอธิบายทุกบรรทัดของฟังก์ชันข้างล่างนี้ และฟังก์ชันนี้คำนวณค่าอะไร

```
function Hypot (X, Y : Real) : Real ;
  begin {Hypot}
    Hypot := Sqrt (Sqr(X) + Sqr(Y))
  End; {Hypot}
```

(b) จงเขียนข้อความสั่งซึ่งเรียกฟังก์ชัน Hypot มีอาร์กิวเมนต์เป็น A และ B เก็บผลลัพธ์ของฟังก์ชันในตัวแปร C

ข้อ 4

(a) จากโปรแกรมข้างล่างนี้ จงวาดรูปพื้นที่ข้อมูลของโปรแกรม Show และพื้นที่ข้อมูลของกระบวนงาน SumDiff เพื่อแสดงค่าปัจจุบันของอาร์กิวเมนต์และพารามิเตอร์ทุกตัวในบรรทัดที่ 2, ... บรรทัดที่ 6

(b) จงแสดงเอาต์พุตของโปรแกรมในรูปตารางของค่า X, Y, Z และ W

```
program Show;
  var
    W, X, Y, Z : Integer ;
    procedure SumDiff (Num1, Num2 : Integer ;
                       var Num3, Num4 : Integer);
      begin {SumDiff}
        Num3 := Num1 + Num2 ;
        Num4 := Num1 - Num2 ;
      end; {SumDiff}
  begin {Show}
    X := 9; Y := 7; Z := 3;  W := 5;
    Writeln ('       X,    Y  Z    W ') ; WriteLn;
    WriteLn('Linel :' , X : 4,Y : 4, Z : 4,W :4):
บรรทัดที่ 2  SumDiff(X, Y, Z, W);
    WriteLn('Line2:' , X : 4, Y : 4, Z : 4, W : 4);
บรรทัดที่ 3  SumDiff(Y, X, Z, W);
    WriteLn('Line3:', X : 4, Y : 4, Z : 4, W : 4);
บรรทัดที่ 4  SumDiff(Z, W, Y, X);
    WriteLn(Line4 :', X : 4, Y : 4, Z : 4, W : 4);
บรรทัดที่ 5  SumDiff(Z, Z, X, Y);
    WriteLn('Line5:', X : 4, Y : 4 , Z : 4, W : 4);
บรรทัดที่ 6  SumDiff(Y, Y, Y, W);
    WriteLn('Line6:', X : 4, Y : 4, Y : 4, Z : 4, W : 4);
  end. {Show}
```

ข้อ 5

จงเขียนกระบวนงานสามชุด ชุดที่หนึ่งสำหรับวาดรูปวงกลม (circle) ชุดที่สองวาดรูปสี่เหลี่ยม (square) และชุดที่สามวาดรูปสามเหลี่ยม (triangle) จากนั้นเขียนโปรแกรมหลักอ่านข้อมูลตัวอักษรหนึ่งตัวซึ่งอาจเป็นตัว C หรือ S หรือ T แล้วเรียกกระบวนงานให้วาดรูปที่มีความหมายตรงกัน