บทที่ 9
แถวลำดับ
(Arrays)

โครงสร้างข้อมูลอย่างง่าย ไม่ว่าจะเป็น buil-in (Integer, Real, Boolean, Char) หรือ user-defined (เช่น ชนิด enumerated Day) ใช้เซลล์หน่วยความจำหนึ่งที่เพื่อเก็บตัวแปร การแก้ปัญหาของการเขียนโปรแกรม จำนวนมาก จะมีประสิทธิภาพมากขึ้น คือ เก็บหน่วยข้อมูลที่เกี่ยวข้องกัน รวมเข้าด้วยกัน ไม่ใช่เก็บหน่วยข้อมูลแต่ละตัวในตัวแปรแตกต่างกัน ตัวอย่างเช่น โปรแกรม ซึ่งประมวลผลคะแนนสอบสำหรับหนึ่งห้องเรียน จะเขียนง่ายขึ้น ถ้าคะแนนทั้งหมดเก็บไว้ด้วยกัน และสามารถประมวลผลเป็นหนึ่งกลุ่ม Pascal ยอมให้โปรแกรมเมอร์ จัดกลุ่ม หน่วยข้อมูลที่เกี่ยวข้องกัน เป็นโครงสร้างข้อมูลประกอบหนึ่งชุด ในบทนี้เราศึกษา โครงสร้างข้อมูล เช่นนี้ ได้แก่ แถวลำดับ

โครงสร้างข้อมูล หมายถึง การประกอบเข้าด้วยกันของหน่วยข้อมูลที่เกี่ยวข้องกันเก็บไว้ภายใต้ชื่อหนึ่งชื่อ (Data structure is a composite of related data items stored under the same name.)

9.1 โครงสร้างข้อมูลแถวลำดับ (The Array Data Structure)

แถวลำดับ หมายถึงโครงสร้างข้อมูล ซึ่งเก็บกลุ่มของหน่วยข้อมูลที่เป็นชนิดเดียวกัน (เช่น คะแนนสอบทั้งหมดของหนึ่งห้องเรียน) การใช้แถวลำดับ เราสามารถเกี่ยวข้องชื่อตัวแปรหนึ่งชื่อ (เช่น Scores) กับกลุ่มข้อมูลทั้งหมด (ดูรูป 9.1) เราสามารถอ้างถึงหน่วยข้อมูลแต่ละตัวในแถวลำดับได้ด้วย กระบวนการตั้งชื่อเหมือนกับที่เราใช้อธิบายครอบครัวและสมาชิกของครอบครัว

Pascal เก็บแถวลำดับในตำแหน่งหน่วยเก็บติดต่อกันในหน่วยความจำหลัก หน่วยข้อมูลหนึ่งตัวใช้เซลล์หน่วยความจำหนึ่งที่ เราสามารถกระทำการดำเนินการ เช่น การส่งแถวลำดับเป็นพารามิเตอร์ ไปยังกระบวนงานด้วยแถวลำดับทั้งหมด เราสามารถเข้าถึงหน่วยข้อมูลแต่ละตัวซึ่งเก็บในแถวลำดับ (เรียกว่าสมาชิกของแถวลำดับ) และประมวลผลมันคล้ายตัวแปรอย่างง่ายชนิดอื่นๆ ในหัวข้อนี้ เราอธิบายว่าจะประกาศแถวลำดับในโปรแกรม Pascal อย่างไร จะอ้างถึงสมาชิกแต่ละตัวของแถวลำดับอย่างไร

แถวลำดับ หมายถึง กลุ่มของหน่วยข้อมูล ซึ่งหน่วยข้อมูลทุกตัวต้องมีชนิดเหมือนกัน (Array is a collection of data items of the same type.)

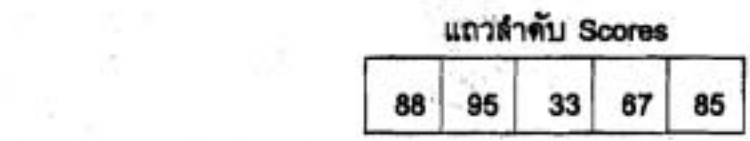
สมาชิกของแถวลำดับ หมายถึง หนึ่งหน่วยข้อมูล ซึ่งเป็นส่วนของแถวลำดับ (Array element is a data item that is part of an array.)

การประกาศแถวลำดับ (Declaring Arrays)

ปกติขั้นแรกเราอธิบายโครงสร้างของแถวลำดับ ในการประกาศชนิดแถวลำดับ จากนั้นเราสามารถจัดสรรหน่วยเก็บสำหรับแถวลำดับหนึ่งชุด หรือมากกว่าหนึ่งชุดของชนิดนั้น

```
                              8 elements    Element type
type                                ↓           ↓
  RealArray  =  array [1 . .8]  of Real; {array type declaration}
var
  X : RealArray;   {Allocate storage for array X}
```

ในตัวอย่างนี้ ชนิดแถวลำดับ, RealArray ถูกประกาศในการประกาศชนิด สัญลักษณ์ [1 . .8] บอก Pascal ให้จัดสรรเซลล์หน่วยความจำ 8 เซลล์ สำหรับตัวแปรใดๆ ที่มีชนิดเป็น RealArray เนื่องจากตัวแปร X ประกาศเป็นชนิด RealArray ดังนั้น X เป็นแถวลำดับมีสมาชิก 8 ตัว แต่ละตัวมีค่าเป็น Real



รูป 9.1 แถวลำดับ Scores ที่มีสมาชิกห้าตัว

โปรดจำไว้ว่า การประกาศชนิดแถวลำดับไม่ได้ทำให้คอมไพเลอร์ของ Pascal จัดสรรพื้นที่หน่วยเก็บในหน่วยความจำ ชนิดแถวลำดับเพียงแต่อธิบายโครงสร้างของแถวลำดับ ที่จริงเฉพาะตัวแปรเท่านั้นซึ่งเก็บสารสนเทศและต้องใช้หน่วยเก็บ Pascal จะไม่จัดสรรพื้นที่ หน่วยเก็บจนกว่าตัวแปรชนิดนี้จะถูกประกาศ

Syntax Display

การประกาศชนิดแถวลำดับ (Array Type Declaration)

Form :
```
type
array type = array [subscript type] of element type;
```

ตัวอย่าง :
```
type
    SmallArray = array [1. . 5] of Char;
```

มีความหมายดังนี้ : ไอเดนติไฟเออร์ array type อธิบายกลุ่มของสมาชิกแถวลำดับ สมาชิกแต่ละตัวสามารถเก็บข้อมูลชนิด element type

subscript type อาจจะเป็น standard ordinal types เช่น Boolean หรือ Char, ชนิด enumerated หรือ ชนิด subrange มีสมาชิกแถวลำดับหนึ่งตัวสำหรับแต่ละค่าใน subscript type ส่วน element type อธิบายชนิดของสมาชิกแต่ละตัวในแถวลำดับสมาชิกทุกตัวของแถวลำดับ มีชนิดเหมือนกัน

ข้อสังเกต 1 ชนิดมาตรฐาน Real และ Integer ใช้เป็น subscript type ไม่ได้ แต่ subrange ของ integer เป็น subscript type ได้

ข้อสังเกต 2 element type เป็นชนิดมาตรฐานหรือชนิดนิยามโดยผู้ใช้ ชนิดใดก็ได้

ข้อสังเกต 1 ใน syntax display กล่าวว่าชนิดมาตรฐาน Integer และ Real ไม่สามารถใช้เป็น subscript types เหตุผลที่ชนิด Integer ใช้ไม่ได้เพราะว่าแถวลำดับที่มีชนิดดรรชนีล่างเป็น Integer จะมีสมาชิกหนึ่งตัว สำหรับจำนวนเต็มแต่ละตัวในพิสัย –MaxInt-1 จนถึง MaxInt (เป็นแถวลำดับขนาดใหญ่มาก) ส่วนชนิด Real ใช้ไม่ได้เพราะว่ามันไม่ใช่ชนิดเชิงอันดับที่ (ordinal type)

ดรรชนีล่างของแถวลำดับ (Array Subscripts)

การประมวลผลข้อมูลซึ่งเก็บในแถวลำดับ เราต้องสามารถเข้าถึงสมาชิกแต่ละตัวของมันได้ เราใช้ชื่อแถวลำดับ (ตัวแปร) ตามด้วย ดรรชนีล่างของแถวลำดับ (บางครั้งเรียกว่า ดรรชนี (index) การทำสิ่งนี้ ดรรชนีล่างของแถวลำดับ อยู่ภายในวงเล็บ (brackets) การเลือกสมาชิกของแถวลำดับหนึ่งตัวสำหรับการประมวลผล ต้องเป็นกำหนดค่าใช้แทนกันได้ที่มีชนิดดรรชนีล่างระบุในการประกาศชนิดแถวลำดับ

ถ้า X เป็นตัวแปร ชนิด RealArray

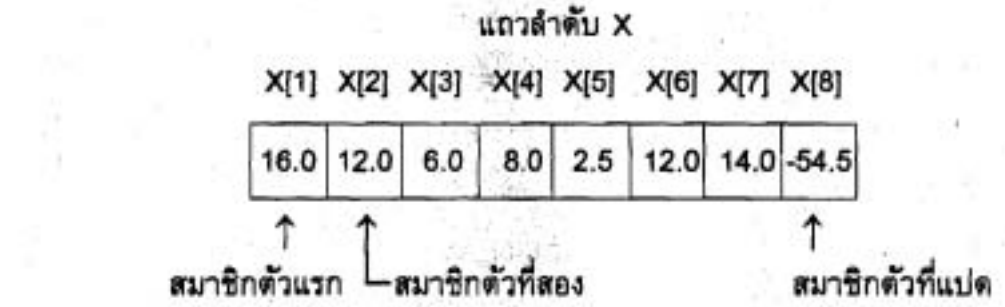
```
type
   RealArray = array [1 . . 8] of Real; {array type declaration}
var
   X : RealArray; {Allocation storage for array X}
```

ชนิดดรรชนีล่างคือ Integer พิสัยย่อย 1. . 8 ดังนั้นดรรชนีที่ใช้ได้ต้องเป็นจำนวนเต็มอยู่ในพิสัยย่อยนี้ เพราะฉะนั้นใช้ X[1] (อ่านว่า "X sub 1") เพื่ออ้างถึงสมาชิกตัวแรกของแถวลำดับ X

X[2] สำหรับสมาชิกตัวที่สอง และ X[8] สำหรับสมาชิกตัวที่แปด (ดูรูป 9.2) เราไม่สามารถใช้ X[0] หรือ X[9] เพื่ออ้างถึงสมาชิกในแถวลำดับ X (ทำไมใช้ไม่ได้)

เราเรียกตัวแปรซึ่งตามด้วยดรรชนีล่างอยู่ในวงเล็บ (เช่น X[1]) ว่าเป็นตัวแปรดรรชนีล่าง (subscript variable)

ตัวแปรดรรชนีล่างมีชนิดข้อมูลเหมือนกับสมาชิกแถวลำดับที่อ้างถึง เพราะว่าสมาชิกของแถวลำดับ X เป็นชนิด Real เราจึงสามารถจัดดำเนินการ X[1] เหมือนกับตัวแปร Real อื่นๆ โดยเฉพาะเราสามารถใช้ X[1] กับตัวดำเนินการคำนวณ ตัวดำเนินการสัมพันธ์ และตัวกำหนดค่าได้ เราสามารถส่ง X[1] เป็นตัวแปรไปยังกระบวนงาน Read (Ln) และ Write (Ln) และไปยังฟังก์ชัน built-in ของ Pascal ใดๆ ก็ตามซึ่งยอมรับพารามิเตอร์ ชนิด Real ได้



รูป 9.2 สมาชิกแปดตัวของแถวลำดับ X

ดรรชนีล่างของแถวลำดับ (ดรรชนี) หมายถึงค่าหรือนิพจน์อยู่ในวงเล็บตามหลังชื่อแถวลำดับซึ่งระบุว่าสมาชิกตัวไหนที่เข้าถึง (Array subscipt (index) is a value or expression enclosed in brackets after the array name, specifying which array element to access.)

ตัวแปรดรรชนีล่าง หมายถึง ตัวแปรซึ่งตามด้วยดรรชนีล่างอยู่ในวงเล็บ (Subscripted variable is a variable followed by a subscript in brackets.)

การเก็บหนึ่งค่าในสมาชิกแถวลำดับ เราเขียนข้อความสั่งกำหนดค่ามีรูปแบบดังนี้

```
subscripted variable := expression
```

การค้นคืน หรือ เข้าถึงค่าซึ่งเก็บในสมาชิกแถวลำดับ เพียงแค่เขียนตัวแปรดรรชนีล่าง ซึ่งสมนัยกันในนิพจน์

ตัวอย่าง 9.1 การเก็บและค้นคืนค่าในแถวลำดับ (Storing and Retrieving Values in an Array) ให้ X เป็นแถวลำดับในรูป 9.2 ข้อความสั่งบางคำสั่งซึ่งจัดดำเนินการสมาชิกของแถวลำดับนี้ แสดงในตาราง 9.1 ส่วน content ของแถวลำดับ X หลังจากการกระทำการข้อความสั่งเหล่านี้แล้วแสดงในรูป 9.3 โปรดสังเกตว่ามีเฉพาะสมาชิกแถวลำดับ X[3] และ X[4] เท่านั้น ที่ค่าเปลี่ยนแปลงไป เพราะว่ามีการกำหนดค่าใหม่ให้

ตาราง 9.1 การเก็บและค้นคืนค่าต่างๆ ในแถวลำดับ X

ข้อความสั่ง	คำอธิบาย
WriteLn (X[1])	แสดงผลค่าของ X[1], หรือ 16.0
X[4] := 25.0	เก็บค่า 25.0 ใน X[4]
Sum := X[1] + X[2]	เก็บผลบวกของ X[1] และ X[2] หรือ 28.0 ในตัวแปร Sum
Sum := Sum + X[3]	บวก X[3] กับ Sum Sum ตัวใหม่ คือ 34.0
.X[4] := X[4] + 1.0	บวก 1.0 กับ X[4] X[4] ตัวใหม่ คือ 26.0
X[3] := X[1] + X[2]	เก็บผลบวกของ X[1] กับ X[2] ใน X[3] เพราะฉะนั้น X[3] ตัวใหม่ คือ 28.0

แถวลำดับ x

X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]
16.0	12.0	28.0	26.0	2.5	12.0	14.0	-54.5

รูป 9.3 แถวลำดับ x หลังจากกระทำการข้อความสั่งต่างๆ ในตาราง 9.1

ดรรชนีล่างแต่ละตัวในตาราง 9.1 คือ สัญพจน์ (literal) อยู่ภายในวงเล็บ (ตัวอย่างเช่น X[4]) ในตัวอย่าง 9.2 จะแสดงให้เห็นว่า ดรรชนีล่างของแถวลำดับ อาจจะเป็นนิพจน์ซึ่งใช้แทนกันได้กำหนดค่า (assignment compatible) กับชนิดดรรชนีล่าง

ตัวอย่าง 9.2 การจัดดำเนินการสมาชิกแถวลำดับ (Manipulating Array Element)

ข้อความสั่ง ในตาราง 9.2 จัดดำเนินการสมาชิกในแถวลำดับ X (รูป 9.3) I เป็นตัวแปรชนิด Integer มีค่าเป็น 6 เมื่อ I มีค่าเท่ากับ 6, ตัวแปรดรรชนีล่าง X[I + 6] หมายถึงสมาชิก X[12] ซึ่งไม่มีในแถวลำดับ Standard Pascal จะแสดงผลข้อผิดพลาดเวลาดำเนินงาน Index expression out of bound ในทางตรงกันข้าม Turbo Pascal ไม่ตรวจสอบดรรชนีล่างของแถวลำดับว่าถูกต้องหรือไม่ เว้นแต่ว่าสามารถตรวจสอบ range ด้วยคำสั่งชี้แนะคอมไพเลอร์ {$R+} ถ้าสามารถตรวจสอบ range ได้ Turbo Pascal จะแสดงผลข้อความ Range Check error ดังนั้น เมื่อเราเขียนและทดสอบโปรแกรมด้วยแถวลำดับ จงใช้ enable range checking เสมอ

ตาราง 9.2 การใช้นิพจน์ดรรชนีล่างกับแถวลำดับ X (Using Subscript Expression with Array X)

ข้อความสั่ง	ผลลัพธ์
Write (8, X[8]	แสดงผล 8 และ –54.5 (ค่าของ X[8])
Write (I, X[I])	แสดงผล 6 และ 12.0 (ค่าของ X[6])
Write (X[I] + 1)	แสดงผล 13.0 (ค่าของ X[6] + 1)
Write (X[I] + I)	แสดงผล 18.0 (ค่าของ X[6] + 6)
Write (X[I + 1])	แสดงผล 14.0 (ค่าของ X[7])
Write (X[6 + 6])	illegal attempt to display X[12]
Write (X[2 * I – 4])	แสดงผล –54.5 (ค่าของ X[8])
Write (X[Trunc (X[5])])	แสดงผล 12.0 (ค่าของ X[2])
X[I] := X[I + 1]	กำหนด 14.0 (ค่าของ X[7]) ให้ X[6]
X[I –1] := X[I]	กำหนด 14.0 (ค่าใหม่ของ X[6] ให้ X[5]
X[I] – 1 := X[I-1]	illegal assignment statement

ข้อความสั่ง Writeบรรทัดที่แปดในตาราง 9.2 ใช้ X[5] เป็นอาร์กิวเมนต์ของฟังก์ชัน Trunc และ Trunc (X[5]) คือ 2.0 ดังนั้นค่าของ X[2] (ไม่ใช่ X[5] ถูกพิมพ์ ข้อผิดพลาดอาจเกิดขึ้นได้ ถ้าค่าของ Trunc (X[5]) อยู่นอก range ที่ใช้ได้ (1 ถึง 8)

ข้อความสั่งกำหนดค่าสามบรรทัดสุดท้าย ในตารางใช้ดรรชนีล่างแตกต่างกันสองแห่ง ข้อความสั่งแรก (บรรทัดที่เก้า) ทำสำเนาค่าของ X[7] ไปที่ X[6] (ดรรชนีล่าง I + 1 และ I) ข้อความสั่งกำหนดค่าบรรทัดที่สิบทำสำเนาค่าของ X[6] ไปที่ X[5] (ดรรชนีล่าง I และ I-1) ข้อความสั่งกำหนดค่าบรรทัดสุดท้าย (บรรทัดที่สิบเอ็ด) เกิดข้อผิดพลาดวากยสัมพันธ์ เพราะว่า X[I] – 1 เป็นนิพจน์ ไม่ใช่ตัวแปร

Syntax Display

การอ้างถึงสมาชิกแถวลำดับ (Array Element Reference)

Form : | array name [subscript] |

ตัวอย่าง : X[3 * I - 2]

มีความหมายดังนี้ : subscript ต้องเป็นนิพจน์ซึ่งใช้แทนกันได้กำหนดค่ากับชนิดดรรชนีล่าง ซึ่งกำหนดในการประกาศ array name ถ้านิพจน์เป็นแบบชนิดข้อมูลไม่ถูกต้อง จะตรวจพบข้อผิดพลาดวากยสัมพันธ์ Index type is not compatible with declaration ถ้าค่าของนิพจน์ไม่อยู่ใน range และ การตรวจสอบ range ทำได้โดยใช้คำสั่งชี้แนะ คอมไพเลอร์ {$R +} เกิด Range check error ระหว่างเวลาดำเนินงาน

แถวลำดับและชนิดดรรชนีล่างเพิ่มเติม (More Array and Subscript Types)

แถวลำดับประกาศไว้แล้วมีชนิดดรรชนีล่างซึ่งเป็น subranges ของจำนวนเต็ม และใช้เก็บค่าตัวเลข สิ่งนี้ไม่มีใน Pascal เพราะว่า ชนิดดรรชนีล่างเป็นชนิดเชิงอันดับที่ใดๆ ก็ได้ (ยกเว้น Integer) และสมาชิกแถวลำดับเป็นชนิดมาตรฐาน หรือชนิดซึ่งประกาศมาแล้ว ชนิดแถวลำดับที่แตกต่างจำนวนหนึ่งอธิบายในตาราง 9.3

จากตารางแสดงให้เห็นว่า แถวลำดับ Name มีสมาชิก 10 ตัว และสามารถเก็บตัวอักษรของชื่อคน แถวลำดับ Fahrenheit มีสมาชิก 21 ตัว เก็บอุณหภูมิฟาเรนไฮท์ ซึ่งสมมูลกับอุณหภูมิเซลเซียส แต่ละตัวใน range –10 ถึง +10 องศาเซลเซียส

ตัวอย่างเช่น Fahrenheit [0] เป็นอุณหภูมิฟาเรนไฮท์ 32.0 สมนัยกับ 0 องศาเซลเซียส

แถวลำดับ LetterCount และ LetterFound มีชนิดดรรชนีล่างเหมือนกัน (เป็นตัวอักษรตัวใหญ่) แต่ชนิดของสมาชิกแตกต่างกัน แถวลำดับ Answers มีสมาชิกเพียงสองตัวเท่านั้น ซึ่งมีค่าดรรชนีล่างเป็น True และ False

ตาราง 9.3 ชนิดแถวลำดับและการประยุกต์ใช้ (Some Array Types and Applications)

Array Type	Subscript Type	Application
Type NameArray = array [1 . . 10] of Char; var Name : NameArray;	integer subrange	Name [1] := 'A'; เก็บชื่อของคน โดยใช้อักขระ 10 ตัว
Type Temps = array [-10 . . 10] of Real; var Fahrenheit : Temps;	integer subrange	Fahrenheit [0] := 32.0; เก็บอุณหภูมิฟาเรนไฮท์ ซึ่งสมนัยกับ –10 ถึง 10 องศาเซลเซียส
Type Counters = array ['A' . . 'Z'] of Integer; var LetterCount : Counters;	character subrange	LetterCount ['A'] := 0; เก็บจำนวนครั้งของการเกิดอักษรตัวใหญ่แต่ละตัว
type Flags = array ['A' . . 'Z'] of Boolean; var LetterFound : Flags;	character subrange	LetterFound ['X'] := False; เก็บเซตของค่าแบบบูล ซึ่งบอกว่าตัวอักษรตัวใดมีหรือไม่มี
type BoolCounts = array [Boolean] of Integer; var Answers : BoolCounts;	Boolean	Answer [True] := 15; เก็บจำนวนของคำตอบถูกและคำตอบผิดในการทดสอบ

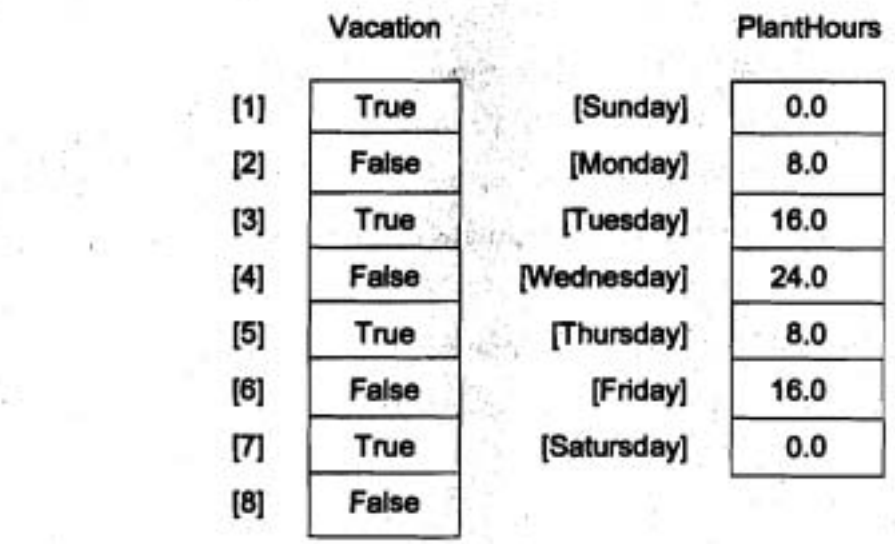
ตัวอย่าง 9.3

ส่วนการประกาศสำหรับโปรแกรมการดำเนินการเพาะปลูก แสดงไว้ข้างล่างนี้ การประกอบชนิดประกาศแบบชนิดข้อมูลอย่างง่าย สองชุด คือ EmpRange และ Day และชนิดแถวลำดับสองชุด คือ EmpArray และ DayArray ในการประกาศตัวแปรได้ประกาศแถวลำดับสองชุด คือ Vacation และ PlantHours

```
const
   NumEmp = 8;                {number of employees}
type
   EmpRange = 1. . NumEmp; {subscript range]
   EmpArray = array [EmpRange] of Boolean;
   Day =   (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
            Satursday);
   DayArray = array [Day] of Real;
var
   Vacation : EmpArray;
   PlantHours : DayArray;
```

พิจารณาการประกาศแถวลำดับชนิด EmpArray เป็นอันดับแรกชนิดดรรชนีล่าง คือ EmpRange (subrange 1. . NumEmp) ดังนั้น ชนิดดรรชนีล่างสำหรับแถวลำดับ ชนิด EmpArray (ได้แก่ Vacation คือ 1 . . 8) สมาชิกแต่ละตัวของแถวลำดับ Vacation เก็บค่าแบบบูล (ดูรูป 9.4) contents ของแถวลำดับชุดนี้แสดงว่าพนักงานคนไหนอยู่ระหว่างหยุดพักผ่อน (Vacation [1] เป็น True ถ้าพนักงาน 1 อยู่ระหว่างหยุดพักผ่อน) ถ้าพนักงาน 1, 3, 5 และ 7 อยู่ระหว่างหยุดพักผ่อนและพนักงานส่วนที่เหลือทำงานปกติ แถวลำดับ Vacation จะมีค่าดังที่แสดงในรูป 9.4

ต่อไปพิจารณาการประกาศสำหรับแถวลำดับชนิด DayArray เพราะว่าชนิดดรรชนีล่าง เป็นชนิด enumerated ชื่อ Day, ดรรชนีล่างสำหรับแถวลำดับชนิด DayArray คือค่าชนิด enumerate ได้แก่ Sunday, Monday และเรื่อยไป เพราะว่า PlantHours เป็นชนิด DayArray, สมาชิกแต่ละตัวของแถวลำดับ PlantHours เก็บค่าชนิด Real สมาชิกแถวลำดับ PlantHours [Sunday] แสดงจำนวนชั่วโมงของการเพาะปลูก ซึ่งได้ดำเนินการในวันอาทิตย์ของสัปดาห์ที่ผ่านมา



รูป 9.4 แถวลำดับ Vacation และ PlantHours

แถวลำดับในรูป 9.4 แสดงว่าการเพาะปลูก ปิดทำการในวันหยุดสุดสัปดาห์และดำเนินการ single shift ในวันจันทร์และวันพฤหัส (เปิดเวลา 8.0 นาฬิกา), double shift ในวันอังคาร และวันศุกร์ และ triple shift ในวันพุธ

เราอาจลบการประกาศค่าคงตัว NumEmp และแบบชนิดข้อมูล EmpRange เพียงแค่ประกาศชนิดแถวลำดับ EmpArray ดังนี้

```
type
   EmpArray = array [1 . . 8] of Boolean;
```

การประกาศตอนเริ่มต้นมีข้อดีสามข้อดังนี้ ข้อแรกการเปลี่ยนแปลงขนาดแถวลำดับ Vacation ทำได้ง่าย คือให้นิยามใหม่แก่ค่าคงตัว NumEmp เราเปลี่ยนขนาดแถวลำดับข้อที่สองค่าคงตัว NumEmp นำไปใช้อ้างถึงได้ใน body ของโปรแกรมข้อสุดท้าย แบบชนิด EmpRange สามารถใช้เป็น ชนิดไอเดนติไฟเออร์ที่ใดก็ได้ในโปรแกรม

.ชนิดแถวลำดับไม่มีชื่อ (Anonymous Array Type) ถึงแม้ว่าในตำราเล่มนี้เราจะไม่ทำเช่นนี้ แต่มันเป็นไปได้ที่จะประกาศแถวลำดับ โดยไม่มีการประกาศชนิดของมันเป็นอันดับแรก

การประกาศตัวแปร

```
var
   X : array [1 . .20] of Integer; {anonymous type array]
```

จัดสรรหน่วยเก็บ สำหรับแถวลำดับ X มีสมาชิก 20 ตัวมีค่าเป็นจำนวนเต็ม เนื่องจากไม่มีการประกาศชนิดของแถวลำดับชนิดของแถวลำดับ X จึงเรียกว่า ชนิดไม่มีชื่อ (anonymous type) เป็นการฝึกปฏิบัติเขียนโปรแกรมที่แย่ (bad programming practice) เรื่องการประกาศแถวลำดับ โดยชนิดไม่มีชื่อ

ชนิดไม่มีชื่อ หมายถึง ชนิดซึ่งไม่ปรากฏชื่อใช้ในการประกาศตัวแปร (Anonymous type is an unnamed type used in a variable declaration.)

หรือ

```
type
   ArrayType = array [1 . . 20] of Integer;
var
   X : ArrayType;
```

แบบฝึกหัด 9.1

1. กำหนดข้อความสั่ง Y := X3 และ Y := X[3] จงเขียนการประกาศชนิด และการประกาศตัวแปร เพื่อให้ข้อความสั่งเหล่านี้ถูกต้อง สมมติว่า y เป็นชนิด Real

2. ถ้าแถวลำดับชุดหนึ่งถูกประกาศให้มีสมาชิก 10 ตัว เราสามารถอ้างถึงสมาชิกแถวลำดับโดยใช้ ดรรชนีล่าง 1 ถึง 10 ได้เสมอหรือไม่

3. สำหรับการประกาศต่อไปนี้ มีเซลล์หน่วยความจำ จำนวนเท่าใด ถูกจองไว้สำหรับเก็บข้อมูล และข้อมูลซึ่งสามารถเก็บไว้ที่นี้ได้ ต้องเป็นชนิดอะไร หน่วยความจำถูกจัดสรรให้ หลังจากการประกาศชนิด หรือ หลังจากการประกาศตัวแปร

```
type
   IndexRange  = 1 . . 5;
   AnArray     = array [IndexRange] of Char;
var
   Grades      : AnArray;
```

4. จงอธิบายชนิดแถวลำดับข้างล่างนี้ และบอกว่าแถวลำดับแต่ละชนิดนั้นสามารถเก็บสมาชิกได้กี่ตัว

a) array [1 . . 20] of Char

b) array ['0' . . '9'] of Boolean

c) array [-5. . 5] of Real

d) array [Boolean] of Char

5. จงเขียนการประกาศตัวแปร และการประกาศชนิด สำหรับการอธิบายรายละเอียดของแถวลำดับ เฉพาะที่ถูกต้องเท่านั้น

a) ชนิดดรรชนีล่าง Boolean, ชนิดสมาชิก Real

b) ชนิดดรรชนีล่าง 'A'.. 'F', ชนิดสมาชิก Integer

c) ชนิดดรรชนีล่าง Char, ชนิดสมาชิก Boolean

d) ชนิดดรรชนีล่าง Integer, ชนิดสมาชิก Real

e) ชนิดดรรชนีล่าง Char, ชนิดสมาชิก Real

f) ชนิดดรรชนีล่าง Real, ชนิดสมาชิก Char

g) ชนิดดรรชนีล่าง Day (enumerated type), ชนิดสมาชิก Real

6. จงเขียนการประกาศชนิดแถวลำดับ สำหรับการแทนสิ่งต่อไปนี้

a) กลุ่มของห้อง (living room, dining room, kitchen, เป็นต้น) ซึ่งมีการกำหนดเนื้อที่

b) ระดับเกรดของโรงเรียนประถม (1 ถึง 6) ที่มีการกำหนดจำนวนของนักเรียนต่อเกรด

c) กลุ่มของสีด้วยค่าตัวอักษรที่กำหนดตามอักษรตัวแรก ของชื่อสี (ตัวอย่างเช่น 'B' สำหรับ Blue)

9.2 การเข้าถึงโดยลำดับกับสมาชิกแถวลำดับ (Sequential access to Array Elements)

โปรแกรมจำนวนมากต้องใช้สมาชิกทุกตัวของแถวลำดับเพื่อประมวลผลในลำดับ โดยเริ่มต้นด้วยสมาชิกตัวแรก การใส่ข้อมูลไว้ในแถวลำดับ การพิมพ์ contents ของมัน หรือกระทำการประมวลผลงานแบบลำดับอื่นๆ เราใช้ for ลูป ซึ่งตัวแปรควบคุมลูป (I) เป็น

ดรรชนีล่างของแถวลำดับด้วย (X[I]) การเพิ่มค่าของตัวแปรควบคุมลูป ครั้งละ 1 จะทำให้สมาชิกของแถวลำดับตัวถัดไปจะถูกประมวลผล

ตัวอย่าง 9.4 การเก็บข้อมูลในแถวลำดับ (Storing Data in an Array)

แถวลำดับ Cube ประกาศดังนี้ เก็บยกกำลังสาม (cube) ของจำนวนเต็ม 10 ตัวแรก (ตัวอย่างเช่น Cube[1] คือ 1, Cube[3] คือ 27, Cube[10] คือ 1000

```
type
   IndexRange  = 1. . 10;
   IntArray    = array [IndexRange] of Integer;
var
   Cube        : Intarray;   {array of cubes}
      I        : Integer;    {loop-control variable}
```

ข้อความสั่ง for

```
for I := 1 to 10 do
   Cube[I] := I * I * I
```

เก็บค่าไว้ในแถวลำดับเป็นดังนี้

แถวลำดับ Cube

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
1	8	27	64	125	216	343	512	729	1000

ตัวอย่าง 9.5 การอ่านและการแสดงผลแถวลำดับ

ข้อมูลต้องอ่านเข้าไปไว้ในแถวลำดับครั้งละหนึ่งสมาชิกเสมอ และแสดงผลครั้งละหนึ่งสมาชิกแถวลำดับ ในรูป 10.5

การประกาศ

```
const
   MaxItems    = 8;        {number of data items}
type
   IndexRange  = 1 . . MaxItems;
   RealArray   = array [IndexRange] of Real;
```

```
var
   X :  RealArray; {array of data}
   I  :  IndexRange; {loop-control variable}
```

จัดสรรหน่วยเก็บสำหรับแถวลำดับ X มีสมาชิกเป็น Real ด้วยดรรชนีล่างในพิสัย 1. . 8 การประมวลผล for ลูป สามชุดกับแถวลำดับ X และตัวแปรควบคุมลูป I ($1 \leq I \leq 8$) คือดรรชนีล่างของแถวลำดับในการลูปแต่ละครั้ง

```
forลูปชุดแรก
   for I := 1 to MaxItems do
      Read (X[I];
```

อ่านค่าข้อมูลหนึ่งตัวไว้ในสมาชิกแถวลำดับแต่ละตัว (ข้อมูลตัวที่หนึ่งเก็บใน X[1], ข้อมูลตัวที่สองเก็บใน X[2] เป็นต้น)

ข้อความสั่ง Read ถูกทำซ้ำสำหรับแต่ละค่าของ I จาก 1 ถึง 8 การทำซ้ำแต่ละครั้งทำให้ค่าข้อมูลตัวใหม่ถูกอ่านและเก็บใน X[I] ดรรชนีล่าง I กำหนดว่า สมาชิกแถวลำดับตัวใดจะรับค่าข้อมูลตัวถัดไป

Edit Window

```
{$R +}
program ShowDiff;
{
   Computes the average value of an array of data and prints the difference
   between each value and the average
}
   const
      MaxItems     = 8;    {number of data items}
   type
      IndexRange   = 1 . . MaxItems;
      RealArray    = array [IndexRange] of Real;
   var
      X :  RealArray;      {array of data}
```

```
    I  :  IndexRange;      {loop-control variable}
    Average,               {average value of data}
    Sum : Real;            {sum of the data}
begin {showDiff}
  {Enter the data.}
Write ('Enter', MaxItems : 1, ' numbers > ');

for I := 1 to MaxItems do
  Read (X[I]);

{Compute the average value.}
Sum := 0.0;          {Initialize Sum.}

for I := 1 to MaxItems do
  Sum := Sum + X[I];      {Add each element to Sum.}

Average := Sum / MaxItems; {Get average value.}
WriteLn ('The average value is ', Average : 3 : 1);
WriteLn;
{Display the difference between each item and the average.}
WriteLn ('Table of differences between X[I] and average');
WriteLn (' I ' : 4, 'X[I] ' : 8, 'Difference' : 14);

for I := 1 to MaxItems do
  WriteLn (I : 4, X[I] : 8 : 1, X[I] – Average : 14 : 1)

end. {ShowDiff}
```

Output Window

```
Enter 8 numbers > 16.0 12.0 6.0 8.0 2.5 12.0 14.0 –54.5
The average value is 2.0
```

Table of differences between X[I] and average

I	X[I]	Difference
1	16.0	14.0
2	12.0	10.0
3	6.0	4.0
4	8.0	6.0
5	2.5	0.5
6	12.0	10.0
7	14.0	12.0
8	-54.5	-56.5

รูป 9.5 ตารางของค่าแตกต่าง (Table of Differences)

for ลูปสุดท้าย

```
for I := 1 to MaxItems do
   WriteLn (I : 4, X[I] : 8.1, X[I] – Average : 14 1)
```

แสดงผลเป็นตารางแสดงสมาชิกของแถวลำดับแต่ละตัว X[I] และค่าแตกต่างระหว่างสมาชิกตัวนั้นกับค่าเฉลี่ย

X[I] – Average

for ลูปที่สอง

```
Sum := 0.0;              {Initialize Sum to zero.}
For I := 1 to MaxItems do
   Sum := Sum + X[I]; {Add each element to Sum.}
```

สะสมผลบวกของสมาชิกของแถวลำดับ X ทั้งหมดแปดตัวในตัวแปร Sum ทุกครั้งที่ทำซ้ำ for ลูป,, I มีค่าเพิ่มอีก 1

ดังนั้น ข้อความสั่ง

```
   Sum := Sum + X[I]; {Add each element to Sum.}
```

บวกสมาชิกตัวถัดไปของแถวลำดับ X กับ Sum การกระทำของส่วนโปรแกรมนี้ตามรอยในตาราง 10.4 สำหรับการทำซ้ำสามครั้งแรกของลูป

ตาราง 9.4 ตามรอยบางส่วนของ for ลูป (Partial Trace of for Loop)

Statement Part	I	X[I]	Sum	Effect
Sum := 0.0;			0.0	Initialize Sum
for I := 1 to MaxItems do	1	16.0		Initialize I to 1
Sum := Sum + X[I]			16.0	add X[1] to Sum
Increment and test I	2	12.0		2 ≤ 8 is true
Sum := Sum + X[I]				add X[2] to Sum
Increment and test I	3	6.0		3 ≤ 8 is true
Sum := Sum + x [I]			34.0	add X[3] to Sum

แบบฝึกหัด 9.2

1. ลำดับของข้อความสั่งต่อไปนี้ เปลี่ยนแปลง contents ของแถวลำดับ X ซึ่งแสดงในรูป 10.5 จงอธิบายว่าแต่ละข้อความสั่ง ทำอะไรกับแถวลำดับ และแสดง contents สุดท้ายของแถวลำดับ x หลังจากข้อความสั่งทั้งหมดถูกกระทำการ

```
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]
```

2. จงเขียนข้อความสั่งโปรแกรมเพื่อให้ทำการการดำเนินการข้างล่างนี้กับแถวลำดับ X ที่แสดงในรูป 9.5

a) แทนที่สมาชิกตัวที่สามด้วย 7.0

b) สำเนาสมาชิกในตำแหน่งที่ห้า ไว้ที่สมาชิกตัวแรก

c) ลบสมาชิกตัวแรกออกจากสมาชิกตัวที่สี่ และเก็บผลลัพธ์ในสมาชิกตัวที่ห้า

d) เพิ่มค่าของสมาชิกตัวที่หกด้วย 2

e) คำนวณผลบวกของสมาชิกห้าตัวแรก

f) คูณสมาชิกหกตัวแรก แต่ละตัวด้วย 2 และใส่ผลคูณแต่ละตัวในสมาชิกของแถวลำดับ AnswerArray

g) แสดงผลสมาชิกที่เป็นเลขคู่ ทั้งหมดบนหนึ่งบรรทัด

เขียนโปรแกรม

1. จงเขียนลูป เพื่อคำนวณผลคูณของสมาชิกแถวลำดับทุกตัวของเลขจำนวนจริง เขียนชนิดที่ถูกต้องสำหรับแถวลำดับนี้

9.3 แถวลำดับเป็นพารามิเตอร์และตัวถูกดำเนินการ (Arrays as Parameters and Operands)

ในหัวข้อนี้ เราจะแสดงให้เห็นว่าการส่งสมาชิกแถวลำดับแต่ละตัว และแถวลำดับทั้งหมดเป็นพารามิเตอร์ ทำอย่างไร จากนั้นจะอภิปราย การดำเนินการอีกชนิดหนึ่ง ซึ่งสามารถกระทำกับแถวลำดับทั้งหมด : การทำสำเนาแถวลำดับ

สมาชิกแถวลำดับเป็นพารามิเตอร์ (Array Elements as Parameters)

ในรูป 9.5 ตัวแปรดรรชนีล่าง X[I] เป็นพารามิเตอร์จริง สำหรับกระบวนงาน Read และ WriteLn ของ Pascal เราสามารถส่งสมาชิกแถวลำดับแต่ละตัวเป็นพารามิเตอร์จริงไปยังกระบวนงาน หรือฟังก์ชันซึ่งเราเขียนเอง ในกรณีนี้ พารามิเตอร์รูปนัย ต้องมีแบบชนิดข้อมูล เหมือนกับสมาชิกแถวลำดับ

ตัวอย่าง 9.6

กระบวนงาน Switch ในรูป 9.6 สับเปลี่ยน (exchange) ค่าของพารามิเตอร์ ชนิด Real สองตัวของมัน

```
procedure Switch (var P, Q (input / output) : Real);
{
  Exchanges the values of P and Q
  Pre  :  P and Q are assigned values.
```

```
    Post :  P has the value passed into Q and vice versa.
}
  var
    Temp : Real; {temporary variable for the exchange}
begin {Switch}
  Temp := P;
    P := Q;
    Q := Temp
end; {Switch}
```

รูป 9.6 กระบวนงาน Switch

ข้อความสั่งเรียกกระบวนงาน

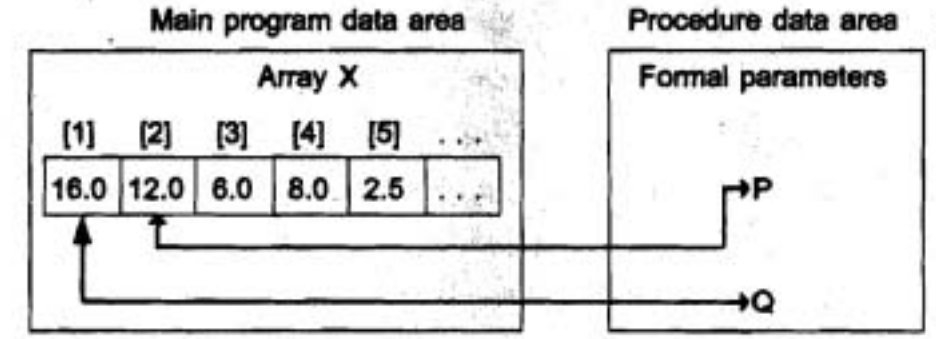
```
Switch (X[2], x [1])
```

ใช้กระบวนงานนี้สับเปลี่ยน contents ของสมาชิกสองตัวแรก (ชนิด Real) ของแถวลำดับ X จากรูป 10.5 พารามิเตอร์จริง X[2] สมนัยกันกับพารามิเตอร์รูปนัย P และพารามิเตอร์จริง X[1] สมนัยกันกับพารามิเตอร์รูปนัย Q ซึ่งการสมนัยกันนี้ แสดงในรูป 9.7 ถึงแม้ว่าเราสามารถส่งสมาชิกแถวลำดับเป็นพารามิเตอร์จริง ได้แต่เราไม่สามารถใช้สมาชิกแถวลำดับเป็นพารามิเตอร์รูปนัยได้

การประกาศกระบวนงาน

```
procedure Switch (var X[I], X[J] {input / output} : Real);
```

จะเกิดข้อผิดพลาดวากยสัมพันธ์ เพราะว่า ชื่อของพารามิเตอร์รูปนัยต้องเป็นไอเดนติไฟเออร์ของ Pascal



รูป 9.7 การสมนัยกันของพารามิเตอร์สำหรับ Switch (X[2], X[1])

การทำสำเนาแถวลำดับ (Copying an Array)

การดำเนินการแถวลำดับทั้งหมดที่ได้ศึกษาแล้ว เราประมวลผลสมาชิกแถวลำดับครั้งละหนึ่งตัว อย่างไรก็ตาม Pascal ยอมให้เราทำสำเนาค่าของแถวลำดับทั้งหมด โดยการใช้ตัวกำหนดค่า (assignment operator) การแสดงตัวอย่างให้เห็น กำหนดการประกาศดังนี้

```
Const
   MaxSize = 100;
type
   IndexRange   = 1 . . MaxSize;
   TestArray    = array [IndexRange] of Real;
var
   X, Y : TestArray;
```

ข้อความสั่งกำหนดค่า

```
   X : Y;   {copy array Y to array X}
```

ทำสำเนาค่าแต่ละตัวในแถวลำดับ Y ซึ่งสมนัยกันกับสมาชิกของแถวลำดับ X (ตัวอย่างเช่น Y[1] ทำสำเนาไปยัง X[1], Y[2] ไปยัง X[2] เช่นนี้เรื่อยไป) แถวลำดับจะถูกทำสำเนาได้ ก็ต่อเมื่อแถวลำดับซึ่งเกี่ยวข้องกันนั้นมีแบบชนิดข้อมูลเหมือนกัน

พารามิเตอร์แถวลำดับ (Array Parameters)

เมื่อเราเขียนส่วนจำเพาะซึ่งมีพารามิเตอร์แถวลำดับ ส่วนจำเพาะนั้นสามารถจัดดำเนินการกับสมาชิกแถวลำดับบางตัว หรือกับสมาชิกทั้งหมด ซึ่งสมนัยกันกับพารามิเตอร์แถวลำดับจริง (actual array parameter) ของมันได้

ตัวอย่าง 9.7 การเริ่มต้นของแถวลำดับ (Initializing an Array)

รูป 9.8 แสดงกระบวนงานซึ่งกำหนดค่าเริ่มต้นของสมาชิกทั้งหมดของแถวลำดับชนิด TestArray ให้เป็น InValve เราสามารถใช้ข้อความสั่งเรียกกระบวนงาน

FillArray (X, 0.0);

FillArray (Y, 1.0);

เพื่อใส่แถวลำดับ X และ Y ที่ได้ประกาศตอนต้น หลังจากข้อความสั่งเหล่านี้กระทำการ สมาชิกของแถวลำดับ X ทุกตัว จะเป็นศูนย์ และสมาชิกของแถวลำดับ Y ทุกตัวจะเป็นหนึ่งทุกตัว

```
procedure FillArray (var W {output} : TestArray;
                     InValue {input}  : Integer);
{
   Sets all elements of its array parameter to InValue.
   Pre  :  InValue is defined.
   Post :  W [I] = InValue, for 1 ≤ I ≤ MaxSize.
}
   var
      I : IndexRange; {array subsript and loop control}
begin {FillArray}
   for I := 1 to MaxSize do
      W [I] := InValue
end; {FillArray}
```

รูป 9.8 กระบวนงาน FillArray

ในกระบวนงาน FillArray, พารามิเตอร์แถวลำดับรูปนัยประกาศเป็นพารามิเตอร์ชนิด TestArray หัวเรื่องกระบวนงานข้างล่างนี้

```
procedure FillArray (var W : array [IndexRange] of Real;
                     InValue {input} : Integer);
```

ไม่ถูกต้อง เพราะว่า ชนิดของพารามิเตอร์แต่ละตัวต้องเป็นไอเดนติไฟเออร์

พารามิเตอร์แถวลำดับตัวแปรและ พารามิเตอร์แถวลำดับค่า (Variable and Value Array Parameters)

เมื่อแถวลำดับถูกส่งเป็นพารามิเตอร์ตัวแปร, Pascal ส่งเลขที่อยู่ (address) ของสมาชิกแถวลำดับจริงตัวแรกไปยังพื้นที่ข้อมูล (data area) ของกระบวนงาน เนื่องจากสมาชิกแถวลำดับเก็บในเซลล์หน่วยความจำติดต่อกัน แถวลำดับทั้งหมดของข้อมูลจึงถูกเข้าถึงได้

เมื่อแถวลำดับถูกส่งเป็นพารามิเตอร์ค่า การทำสำเนาเฉพาะที่ (local copy) ของแถวลำดับกระทำเมื่อกระบวนงานถูกเรียกแถวลำดับเฉพาะที่ (local array) ถูกเริ่มต้นแล้ว ดังนั้นมันเก็บค่าเดียวกับที่สมนัยกันของแถวลำดับจริง กระบวนงานจัดดำเนินการแถวลำดับเฉพาะที่ และการเปลี่ยนแปลงใดๆ ซึ่งกระทำกับแถวลำดับเฉพาะที่จะไม่เปลี่ยน contents ของแถวลำดับจริง

ตัวอย่าง 9.8 แสดงให้เห็นความแตกต่างเหล่านี้ สมมติว่ามีการประกาศ ดังนี้

```
const
   MaxSize = 5;
type
   IndexRange = 1 . . MaxSize;
   TestArray = array [IndexRange] of Real;
var
   X, Y, Z : TestArray;
```

ตัวอย่าง 9.8

ถึงแม้ว่ามันเป็นไปได้ที่จะใช้ข้อความสั่งกำหนดค่าหนึ่งคำสั่ง เพื่อทำสำเนาแถวลำดับหนึ่งชุดไปเป็นแถวลำดับอีกหนึ่ง

ข้อความสั่งกำหนดค่า

```
Z := X + Y {Ilegal addition of arrays}
```

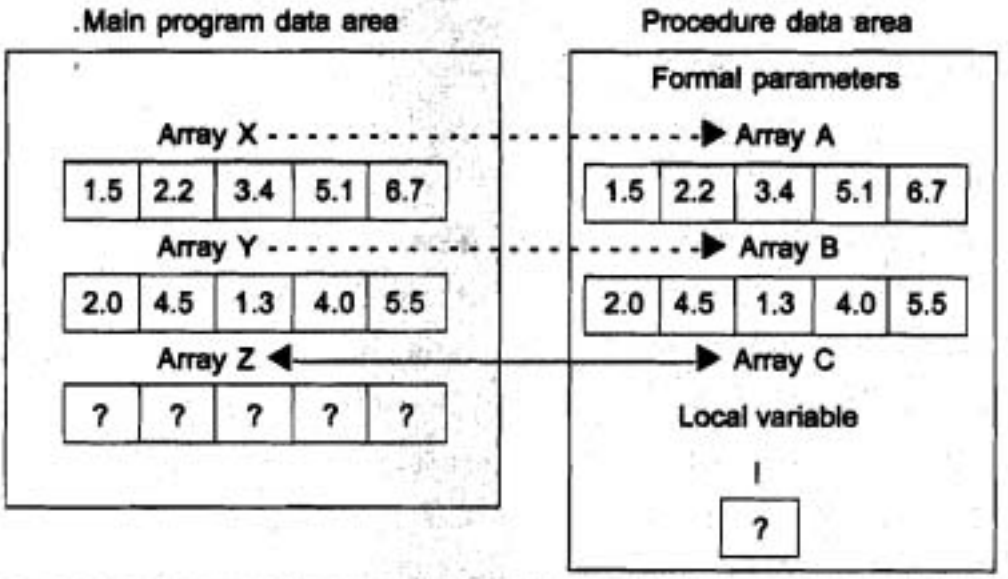
ไม่ถูกต้องเพราะว่าตัวถูกดำเนินการของตัวดำเนินการ + เป็นแถวลำดับไม่ได้ เราต้องใช้กระบวนงาน AddArray (รูป 10.9) เพื่อบวกแถวลำดับสองชุดชนิด TestArray การสมนัยกันของพารามิเตอร์สำหรับแถวลำดับถูกสร้างโดยข้อความสั่งเรียกกระบวนงาน

AddArray (X, Y, Z)

ซึ่งแสดงในรูป 10.10 แถวลำดับ A และ B ในพื้นที่ข้อมูลของกระบวนงาน เป็นการดำเนาเฉพาะที่ของแถวลำดับ X และ Y ลูกศรเส้นทึบ (solid arrow) หมายถึง เลขที่อยู่ของสมาชิกตัวแรกของแถวลำดับ Z ถูกเก็บในพารามิเตอร์ C ผลลัพธ์ของกระบวนงาน เก็บโดยตรงในแถวลำดับ Z หลังจากกระทำการของกระบวนงาน, Z[1] จะเป็นผลบวกของ X[1] กับ Y[1] หรือ 3.5; Z[2] จะเป็น 6.7 เช่นนี้เรื่อยไป แถวลำดับ X และ Y ไม่เปลี่ยนแปลง

```
procedure AddArray   (A, B {input} : TestArray;
                      var C {output} : TestArray);
{
   Stores the sum of A [I] and B [I] in C [I]
   Array elements with subscripts 1. . MaxSize are sumed, element by element.
   Pre  :  A [I] and B [I] (1 ≤ I ≤ MaxSize) are defined.
   Post :  C [I] := A [I] + B [I] (1 ≤ I ≤ MaxSize).
}
   var
      I : IndexRange; {loop control and array subscritp}
begin {AddArray}
   {Add corresponding elements of each array.}
   for I := 1 to MaxSize do
      C[I] := A[I] + B[I]
end; {AddArray}
```

รูป 9.9 กระบวนงาน AddArray



รูป 9.10 การสมนัยของพารามิเตอร์ สำหรับ AddArray (X, Y, Z)

สไตล์ของโปรแกรม (Program Style)

ประสิทธิภาพของพารามิเตอร์ตัวแปรกับการป้องกันของพารามิเตอร์ค่า (Efficiency of Variable Parameters versus Protection of Value Parameters) พารามิเตอร์ A และ B ในรูป 9.9 ประกาศเป็นพารามิเตอร์ค่าเพราะว่า มันเก็บค่าที่ส่งไปยังกระบวนงาน AddArray เท่านั้นและค่าของมันไม่ควรถูกเปลี่ยนแปลงโดย AddArray, Pascal ต้องสร้างสำเนาเฉพาะที่ของแถวลำดับสองชุดนี้ ทุกครั้งที่กระบวนงาน AddArray ถูกเรียกการทำสำเนาครั้งนี้ใช้เวลาเครื่องคอมพิวเตอร์ และเนื้อที่หน่วยความจำอย่างคุ้มค่า (valuable) ถ้าแถวลำดับที่กำลังถูกทำสำเนามีขนาดใหญ่มาก โปรแกรมอาจจบ (terminate) ด้วยข้อผิดพลาดเพราะว่าเนื้อที่หน่วยความจำทั้งหมดถูกใช้ไปแล้ว

เพื่อรักษาเวลาและเนื้อที่หน่วยความจำ, โปรแกรมเมอร์ที่มีประสบการณ์บางครั้งประกาศแถวลำดับซึ่งจะถูกใช้เฉพาะส่วนจำเพาะอินพุตเท่านั้นเป็นพารามิเตอร์ตัวแปร ไม่ใช่พารามิเตอร์ค่า สิ่งนี้หมายความว่า การสมนัยกันของแถวลำดับจริง ถูกจัดดำเนินการโดยตรงด้วยส่วนจำเพาะและไม่มีการป้องกันใดๆ จากการดัดแปรโดยไม่ตั้งใจ (accidental modification) ของส่วนจำเพาะ การเปลี่ยนแปลงใดๆ ซึ่งเกิดขึ้นกับแถวลำดับจริง จึงเป็น

ผลกระทบ (side effect) ที่ไม่พึงต้องการของการกระทำการของส่วนจำเพาะ ถ้าการสมนัยกันของแถวลำดับไปยังพารามิเตอร์ค่า การเปลี่ยนแปลงที่กระทำกับการทำสำเนาเฉพาะที่ไม่มีผลกระทบ (unaffected) เพื่อหลีกเลี่ยงผลกระทบ ประกาศพารามิเตอร์แถวลำดับใช้เฉพาะส่วนจำเพาะอินพุตให้เป็นพารามิเตอร์ค่าเท่านั้น เว้นแต่ว่าแถวลำดับมีขนาดใหญ่มาก (สมาชิกมีมากกว่า 500 ตัว)

ตัวอย่าง 9.9 การเปรียบเทียบแถวลำดับสองชุด (Comparing Two Arrays)

ฟังก์ชัน SameArray ในรูป 9.11 ตรวจดูว่า แถวลำดับสองชุด (ชนิด TestArray) เหมือนกันหรือไม่ เราพิจารณาแถวลำดับสองชุด เหมือนกันถ้าสมาชิกตัวแรกของแถวลำดับชุดหนึ่ง เหมือนกับสมาชิกตัวที่หนึ่งของแถวลำดับอีกชุดหนึ่ง สมาชิกตัวที่สองของชุดที่หนึ่งเหมือนกับสมาชิกตัวที่สองของอีกชุดหนึ่ง เช่นนี้เรื่อยไปเราสามารถบอกว่าแถวลำดับสองชุดนั้นไม่เหมือนกัน โดยการหาหนึ่งคู่ของสมาชิกที่ไม่เท่ากัน เพราะฉะนั้น while ลูป อาจถูกกระทำการ จากหนึ่งครั้ง (สมาชิกตัวแรกไม่เท่ากัน) จนถึง MaxSize–1 ครั้ง การออกจากลูปเกิดขึ้นเมื่อมีการพบหนึ่งคู่ของสมาชิกที่ไม่เท่ากัน หรือคู่สุดท้ายของการทดสอบ หลังจากออกจากลูป (After loop exit)

ข้อความสั่งกำหนดค่าแบบบูล

SameArray := (A [I] = B [I]) {Define result.}

นิยามผลลัพธ์ของฟังก์ชัน ถ้าการออกจากลูปเกิดขึ้นเพราะว่าคู่ของสมาชิกที่มีดรรชนีล่างเป็น I มีค่าไม่เท่ากัน ผลลัพธ์ของฟังก์ชัน เป็น False ถ้าการออกจากลูปเกิดขึ้นเพราะว่าพบสมาชิกคู่สุดท้าย (I = Maxsize) แล้ว, ผลลัพธ์ของฟังก์ชัน เป็น True ถ้าสมาชิกที่มีดรรชนีล่างเป็น I มีค่าเท่ากันและเป็น False ถ้าสมาชิกมีค่าไม่เท่ากันต่อไปเป็นตัวอย่างว่าเราจะใช้ฟังก์ชัน SameArray อย่างไร ข้อความสั่ง if

```
if SameArray (X, Y) then
   Z := X
else
   AddArray (X, Y, Z)
```

คือการทำสำเนา แถวลำดับ X ไปที่แถวลำดับ Z (เมื่อ X และ Y เหมือนกัน) หรือเก็บผลบวกของแถวลำดับ X และ Y ในแถวลำดับ Z (เมื่อ X และ Y ไม่เหมือนกัน)

เนื่องจากแถวลำดับมีสมาชิก MaxSize ตัว ข้อผิดพลาดร่วม คือการใช้เงื่อนไข

(I <= MaxSize) and (A [I] = B [I])

เป็นเงื่อนไข while ในรูป 10.11 ทำให้คู่สมาชิกทั้งหมดต้องถูกทดสอบโดยเงื่อนไข while ถ้าแถวลำดับเท่ากันเมื่อ I เท่ากับ MaxSize + 1, ส่วนแรกของเงื่อนไขนี้ ประเมินผลเป็น False แต่ส่วนที่สองยังคงถูกประเมินผลเว้นแต่ว่าจะมีการใช้การประเมินผลทางลัด (short – circuit evaluation)

สิ่งนี้นำไปสู่ Range check error เพราะว่าสมาชิกแถวลำดับ A [MaxSize + 1] ไม่มีจริง

กรณีศึกษาถัดไป แสดงให้เห็นสองวิธีที่แตกต่างกันสำหรับการเข้าถึงแถวลำดับ : การเข้าถึงโดยลำดับและการเข้าถึงโดยสุ่ม หลังจากนั้นจะอภิปรายความแตกต่างของสองวิธีนี้

กรณีศึกษา ปัญหาทำงบประมาณในบ้าน (Home Budget Problem)

จงเขียนโปรแกรมซึ่งเก็บร่องรอย (track) ค่าใช้จ่ายรายเดือน โดยการจำแนกประเภท โปรแกรมจะอ่านจำนวนค่าใช้จ่ายแต่ละประเภท บวกเลขตัวนี้กับผลรวมประเภทที่ถูกต้อง และพิมพ์ค่าใช้จ่ายรวมทั้งหมดแยกตามประเภท ข้อมูลอินพุตประกอบด้วยชนิดและจำนวนค่าใช้จ่ายแต่ละประเภทระหว่างเดือนที่ผ่านมา

วิเคราะห์ (Analysis)

ประเภทงบประมาณ ได้แก่ บันเทิง อาหาร เสื้อผ้า ค่าเช่า ค่าเล่าเรียน เงินประกัน และอื่นๆ โปรแกรม ต้องสะสมผลรวมแยกเป็น 7 ประเภท แต่ละชนิดเกี่ยวข้องกับสมาชิกของสมาชิกเจ็ดตัวของแถวลำดับ โปรแกรมอ่านค่าใช้จ่ายแต่ละประเภท ตรวจดูว่ามันเป็นค่าใช้จ่ายประเภทใด จากนั้นบวกค่าใช้จ่ายนี้กับสมาชิกแถวลำดับที่ถูกต้อง หลังจากประมวลผลค่าใช้จ่ายทั้งหมดแล้ว โปรแกรมพิมพ์ตารางซึ่งสร้างแต่ละประเภทและผลรวมสะสมทั้งหมดของมัน ในโปรแกรมซึ่งสะสมผลบวกนั้น ผลรวมทั้งหมดของแต่ละชนิด ต้องเริ่มต้นด้วยศูนย์

วิธีง่ายๆ คือ เราใช้แถวลำดับที่ดรรชนีล่างเป็น 1 ถึง 7 เพื่อเก็บตัวเลขงบประมาณ แต่โปรแกรมจะอ่านง่ายขึ้น ถ้าเราประกาศแบบชนิดข้อมูล BudgetCat และใช้แบบชนิดข้อมูลนี้เป็นชนิดดรรชนีล่างของแถวลำดับ เพื่อสนับสนุนสภาพมอดุลาร์ (modurarity) และง่ายต่อการพัฒนา เราถือว่าแบบชนิด BudgetCat เป็นแบบชนิดข้อมูลนามธรรม (abstract data type (ADT)) และประกาศมัน รวมทั้งตัวดำเนินการของมัน ReadBudgetCat และ WriteBudgetCat ในหน่วย BudgetCat ADT (ดูเขียนโปรแกรม แบบฝึกหัดข้อ 4 ที่ตอนท้ายของหัวข้อนี้) แบบชนิดข้อมูล BudgetCat แสดงถัดไป มีชนิดพิเศษเพิ่ม Done ซึ่งใช้เพื่อแสดงว่าถ้าใช้จ่ายทั้งหมดถูกประมวลผลแล้ว

ข้อมูลที่ต้องการ (Data Reguirements)

แบบชนิดข้อมูล

BudgetCat = (Entertainment, Food, Clothing, Rent, Tuition, Insurance, Miscellaneous, Done);

อินพุตของปัญหา (Problem Input)

ค่าใช้จ่ายแต่ละตัวและประเภทของมัน

เอาต์พุตของปัญหา (Problem Output)

แถวลำดับของผลรวมทั้งหมดของค่าใช้จ่าย (Budget)

ออกแบบ (Design)

อัลกอริทึมเริ่มต้น (Initial Algorithm)

1. เริ่มต้นให้ผลรวมทุกประเภททั้งหมดเป็นศูนย์ (Initialize all category totals to zero.)

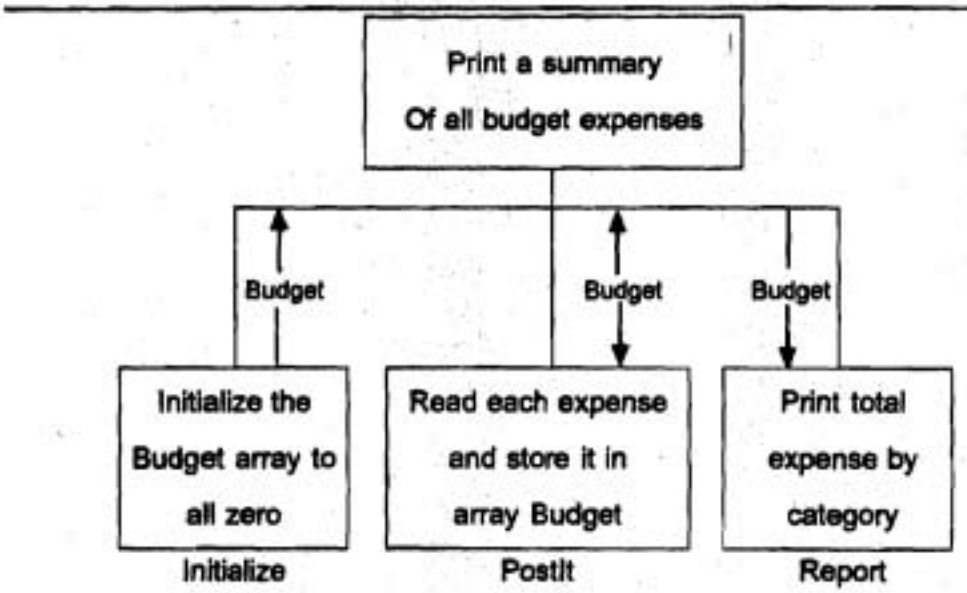
2. แสดงผลประเภทงบประมาณที่เป็นไปได้ (Display the possible budget categories.)

3. อ่านค่าใช้จ่ายแต่ละชนิด และบวกเลขตัวนี้กับผลรวมที่ถูกต้อง (Read each expenditure and add it to the appropriate total.)

4. พิมพ์ผลรวมสะสมทั้งหมด สำหรับแต่ละประเภท (Print the accumulated total for each category.)

แผนภูมิโครงสร้างและการแบ่งละเอียด (Structure Chart and Refinements)

แผนภูมิโครงสร้าง ในรูป 9.2 แสดงความสัมพันธ์ระหว่างขั้นที่ 1, 3 และ 4 ซึ่งจัดดำเนินการกับแถวลำดับ Budget กระบวนงาน Initialized และ PostIt เก็บสารสนเทศในแถวลำดับนี้ กระบวนงาน Report แสดงผลสารสนเทศ ขั้นที่ 2 ทำให้เกิดผลโดยกระบวนงาน DisplayMenu (ไม่ได้แสดงไว้) ซึ่งแสดงรายการของประเภทงบประมาณสำหรับใช้ประโยชน์ของผู้ใช้โปรแกรม



รูป 9.12 แผนภูมิโครงสร้างสำหรับปัญหาการทำงบประมาณบ้าน

การปฏิบัติให้เกิดผล (Implementation)

การลงรหัสโปรแกรมหลัก (Coding the Main Program) รูป 9.13 แสดงโปรแกรม โปรแกรมหลักประกอบด้วยการประกาศสำหรับแบบชนิดข้อมูล (BudgetArray) และแถวลำดับ Budget (ชนิด BudgetArray) แถวลำดับ Budget ปรากฏในรายการพารามิเตอร์ แต่ละตัว และถูกส่งระหว่าง แต่ละกระบวนงานกับโปรแกรมหลัก

ตัวแปรควบคุมลูป ชื่อ NextCat (ชนิด BudgetCat) ถูกประกาศเป็นตัวแปรเฉพาะที่ในแต่ละกระบวนงานในกระบวนงาน Initialize ข้อความสั่งกำหนดค่า

Budget [NextCat] := 0.0

ทำซ้ำหนึ่งครั้งสำหรับแต่ละค่าของ NextCat และกำหนดสมาชิกแต่ละตัวของ Bndget ให้เป็น 0 ในกระบวนงาน Report

ข้อความสั่ง

WriteBudgetCat (NextCat); {Display budget category.}

WriteLn (Budget [NextCat] : 15 : 2)

เรียกกระบวนงาน writeBudgetCat (จากหน่วย BudgetCatADT) เพื่อแสดงผล ชื่อประเภทงบประมาณและ WriteLn แสดงจำนวนรวมทั้งหมดของทุกประเภท

กระบวนงาน PostIt ต้องอ่านค่าใช้จ่ายแต่ละประเภทและบวกค่าใช้จ่ายนี้กับสมาชิกแถวลำดับที่ถูกต้อง ผลรวมทั้งหมดของค่าใช้จ่ายบันเทิงถูกสะสมใน Budget [Entertainment] ค่าใช้จ่ายอาหารถูกสะสมใน Budget [Food] เช่นนี้เรื่อยไป

การลงรหัสกระบวนงาน PostIt (Coding Procedure PostIt)

กระบวนงาน PostIt ที่แสดงในรูป 9.14 ใช้กระบวนงาน ReadBudgetCat (จากหน่วย BudgetCatADT) เพื่ออ่านประเภทงบประมาณกระบวนงาน PostIt เรียก ReadBudgetCat เพื่ออ่านประเภทถัดไปไว้ใน NextCat ตัว body ของ while ลูปถูกกระทำการสำหรับแต่ละค่าของ NextCat ซึ่งไม่ใช่ Done (เป็น sentinel)

ข้อความสั่งกำหนดค่า

Budget [NextCat] := Budget [NextCat] + Expense;

บวกค่าใช้จ่ายกับสมาชิกของแถวลำดับ Budget ซึ่งถูกเลือกโดย NextCat

---

```
{$R+}
program HomeBudget;
{Prints a summary of all expenses by budget category}
uses
   BudgetCatADT;     {Import type BudgetCat and ReadBudgetCat and
                      WriteBudgetCat.}
   type
      BudgetArray = array [BudgetCat] of Real; {array type}
   var
      Budget : BudgetArray; {output – array of totals}

procedure Initialize (var Budget {output} : BudgetArray);
{
   Initializes array Budget to all zeros.
   Pre  :  None.
   Post :  Each element of Budget is 0.0 .
```

```
}
   var
      NextCat : BudgetCat;      {loop-control variable}
                                {array subscript}
   begin    {Initialize}
      for   NextCat := Entertainment to Miscellaneous do
            Budget [NextCat] := 0
   end;

   procedure DisplayMenu;
   {Displays the budget categories}
      var
         NextCat : BudgetCat;
   begin
      WriteLn ('LIST OF BUDGET CATEGORIES : ');
      for NextCat : = Entertainment to Done do
         begin
            WriteBudgetCat (NextCat);
            WriteLn
         end {for}
   end; {DisplayMenu}

Procedure PostIt (var Budget {input / output] : BudgetArray;
{
   Read each expenditure amount and adds it to the appropriate element of
   array Budget.
   Pre  :  Each array element Budget [NextCat] is 0.0 .
   Post :  Each array element Budget [NextCat] is the sum of expense
           amounts for category NextCat.
```

```
}
begin {PostIt stub}
   WriteLn ('Procedure PostIt entered')
end; {PostIt stub}

Procedure Report (Budget {input} : BudgetArray);
{
   Prints the expenditures in each budget category.
   Pre  :  Array Budget is defined.
   Post :  Displays each budget category name and amount .
   Calls :  WriteBudgetCat
begin   {Report}
   WriteLn;
   WriteLn ('Category' : 15, 'Expenses' : 15);
   {Print each category name and the total.}
   for NextCat := Entertainment to Miscellaneous do
      begin
         Write BudgetCat (NextCat); {Display category.}
         WriteLn (Budget [NextCat] : 15 : 2)
      end {for}
end; {Report}
begin {HomeBudget}
   {Initialize array Budget tot all zeros.}
   Initialize (Budget);
   {Display list of budget categories.}
   DisplayMenu;
   {Read and process each expenditure.}
   PostIt (Budget);
```

```
    {Print the expenditures in each category.}
    Report (Budget)
end. {HomeBudget}
```

รูป 10.13 โปรแกรมทำงบประมาณบ้าน (Home Budget Program)

```
procedure PostIt (var Budget {input / output} : BudgetArray);
{
    Read each expenditure amount and adds it to the appropriate element
    of array Budget.
    Pre  :  Each array element Budget [NextCat] is 0.0 .
    Post :  Each array element Budget [NextCat] is the sum of expense
            amounts for category NextCat.
    Calls :  ReadBudgetCat
}
    var
        NextCat : BudgetCat; {next budget category}
        Expense : Real;          {expenditure amount}
begin {PostIt}
    {Read each budget category and expense and add it to Budget.}
    ReadBudgetCat (NextCat);
    while NextCat < > Done do
        {invariant :
            no prior value of NextCat is Done and Budget [NextCat] is the
            sum of all budget entries so for for each category NextCat.
        }
        begin
```

```
            Write (' Enter the expenditure amount $ ');
            ReadLn (Expense);
            Budget [NextCat] : Budget [NextCat] + Expense;
            WriteLn;
            ReadBudgetCat (NextCat)
        end {while}
end; {PostIt}
```

รูป 9.14 กระบวนงาน PostIt

```
LIST OF BUDGET CATEGORIES :
Entertainment
Fod
Clothing
Rent
Tuition
Insurance
Miscellaneous
Done

Enter the first letter of the category > C
Enter the expenditure amount $ 25.00

Enter the first letter of the category > M
Enter the expenditure amount $ 25.00

Enter the first letter of the category > C
Enter the expenditure amount $ 15.00
```

```
Enter the first letter of the category > E
Enter the expenditure amount $ 675.00
Enter the first letter of the category > D

Category          Expense
Entertainment      675.00
Fod                  0.00
Clothing            40.00
Rent                 0.00
Tuition              0.00
Insurance            0.00
Miscellaneous       25.00
```

รูป 9.15 Sample Run ของโปรแกรม HomeBudget

การทดสอบ (Testing)

ก่อนวิ่งโปรแกรม เราต้อง save และคอมไพล์ ไว้ในแฟ้มดิสก์ BUDGETCA.PAS ซึ่งประกอบด้วยหน่วย BudgetCatADT (ดูเขียนโปรแกรม แบบฝึกหัดข้อ 4 ที่ตอนท้ายของหัวข้อนี้)

Sample run ของโปรแกรม HomeBudget ในรูป 9.15 แสดงว่าข้อมูลอินพุตไม่มีการเรียงอันดับตามประเภทรายจ่าย เราควรจะทดสอบ (verify) ว่าประเภทของงบประมาณทั้งหมด ถ้าไม่มีรายจ่ายแต่อย่างใด ยังต้องเป็นศูนย์ และค่าของประเภทไม่ถูกต้อง จะทำให้โปรแกรมสิ้นสุดก่อนเวลาจบปกติ (terminate prematurely) โปรดสังเกตว่า การใส่อักษรตัวแรกเพียงหนึ่งตัวเท่านั้น พอเพียงสำหรับประเภทงบประมาณรายจ่าย แต่ละชนิด เพราะว่าทุกชนิดไม่มีการซ้ำกันเลย (budget category unique)

การเข้าถึงแถวลำดับโดยลำดับกับการเข้าถึงแถวลำดับโดยสุ่ม (Sequential Versus Random Access to Arrays)

โปรแกรมงบประมาณบ้าน แสดงให้เห็นวิธีร่วมสองอย่างของการเลือกสมาชิกแถวลำดับสำหรับการประมวลผล บ่อยครั้งที่เราจำเป็นต้องจัดดำเนินการกับสมาชิกทุกตัวของแถวลำดับ ในลักษณะเป็นรูปแบบอย่างเดียวกัน (ตัวอย่างเช่น เราต้องเริ่มต้นด้วยการให้สมาชิกทุกตัวเป็นศูนย์) ในสถานการณ์เช่นนี้ คือการประมวลผลกับสมาชิกแถวลำดับในลำดับ (เข้าถึงโดยลำดับ) เริ่มต้นด้วยสมาชิกตัวแรก และจบด้วยสมาชิกตัวสุดท้าย ในกระบวนงาน Initialize และ Report เรากระทำงานสำเร็จเข้าถึงโดยลำดับ โดยใช้ for ลูป ซึ่งมีตัวแปรควบคุมลูปเป็นดรรชนีล่างของแถวลำดับด้วย

ในกระบวนงาน PostIt อันดับซึ่งสมาชิกแถวลำดับจะถูกเข้าถึงขึ้นอยู่กับว่าอันดับของข้อมูล เสร็จสิ้นแล้ว ค่าซึ่งอ่านเข้าไปไว้ใน NextCat กำหนดสมาชิกตัวที่จะถูกเพิ่มค่าวิธีนี้เรียกว่า การเข้าถึงโดยสุ่ม (random access) เพราะว่าไม่สามารถทำนายอันดับได้

แบบฝึกหัด 9.3

1. เมื่อใดจึงเป็นวิธีที่ดีกว่าในการส่งแถวลำดับทั้งหมดของข้อมูล ไม่ใช่ส่งสมาชิกแต่ละตัวไปยังกระบวนงาน

2. เมื่อใด คือการทำสำเนาแถวลำดับทั้งหมด สำหรับแถวลำดับ ซึ่งเป็นพารามิเตอร์ของกระบวนงาน เกิดอะไรขึ้นกับการทำสำเนาหลังจากกระบวนงานกระทำการ

3. ในฟังก์ชัน Same Array (รูป 9.11) I จะมีค่าเป็นอะไร

เมื่อข้อความสั่ง

SameArray := (A [I] = B [I])

กระทำการ ถ้าแถวลำดับ A เท่ากับแถวลำดับ B, ถ้าสมาชิกตัวที่สามไม่จับคู่กัน, ถ้ามีเฉพาะสมาชิกตัวสุดท้ายเท่านั้นที่จับคู่กัน

เขียนโปรแกรม

1. จงเขียนกระบวนงาน ซึ่งกำหนดค่า True ให้สมาชิก I ของแถวลำดับเอาต์พุต ถ้า สมาชิก I เป็นแถวลำดับอินพุตชุดหนึ่ง ซึ่งมีค่าเหมือนกับ สมาชิก I ของแถวลำดับอินพุตอีกชุดหนึ่ง กรณีอื่นๆ กำหนดค่า False ถ้าแถวลำดับอินพุต มีชนิดดรรชนีล่างเป็น Index Type แถวลำดับเอาต์พุตควรมีชนิด เป็นดังนี้

```
type
    BoolArray = array [IndexType] of Boolean;
```

2. จงเขียนกระบวนงาน เพื่อทำสำเนาค่าแต่ละตัว ซึ่งเก็บในแถวลำดับหนึ่งชุดไปยังสมาชิกที่สมนัยกันของแถวลำดับอีกหนึ่งชุด ยกตัวอย่าง เช่น ถ้าแถวลำดับสองชุดนั้นคือ InArray และ OutArray ให้สำเนา In Array [1] ไป OutArray [1] จากนั้นสำเนา InArray [2] ไปยัง OutArray [2] เช่นนี้เรื่อยไป

3. จงเขียนแถวลำดับ ซึ่งสมนัยกับกระบวนงาน Switch ในรูป 9.6 ซึ่งไม่ต้องจัดสรรหน่วยความจำแถวลำดับชั่วคราวเฉพาะที่ ดังนั้น SwitchArray จึงกำหนดแถวลำดับอินพุต Aarray และ Barray ซึ่งจะสับเปลี่ยน Aarray [1] กับ Barray [1], Aarray [2] กับ Barray [2] เช่นนี้เรื่อยไป

9.4 การประมวลผลแถวลำดับย่อย (Subarray Processing)

โดยปกติ เราจะไม่ทราบก่อนการกระทำการโปรแกรมจริงๆ ว่าจะมีสมาชิกจำนวนกี่ตัวเก็บในแถวลำดับ ตัวอย่างเช่น อาจารย์ท่านหนึ่งประมวลผลคะแนนสอบของนักศึกษาในชั้นเรียนหนึ่งจำนวน 150 คน, นักศึกษาจำนวน 200 คน ในชั้นเรียนถัดไป เช่นนี้เป็นต้น เนื่องจากเราต้องประกาศขนาดของแถวลำดับก่อนการเริ่มต้น กระทำการโปรแกรม (ณ เวลาคอมไพล์โปรแกรม) เราจึงต้องจัดสรรเนื้อที่หน่วยเก็บให้พอเพียง เพื่อให้โปรแกรมสามารถประมวลผลแถวลำดับที่คาดว่าจะใหญ่สุด โดยไม่มีข้อผิดพลาด

เมื่อเราอ่านข้อมูลแถวลำดับไปไว้ในหน่วยความจำ เริ่มต้นการใส่แถวลำดับด้วยสมาชิกตัวแรก และให้แน่ใจว่าได้เก็บร่องรอย (track) ของจำนวนหน่วยข้อมูลที่เก็บจริงในแถวลำดับส่วนของแถวลำดับซึ่งเก็บข้อมูลจริงเรียก แถวลำดับย่อยเติมเต็ม (filled subarray)

ความยาว (length) ของแถวลำดับย่อยเติมเต็ม หมายถึงจำนวนหน่วยข้อมูลหรือจำนวนสมาชิกซึ่งเก็บจริงในแถวลำดับ

ตัวอย่าง 9.10

แถวลำดับ Scores ซึ่งประกาศข้างล่างนี้ สามารถเก็บชั้นเรียนที่มีนักศึกษาได้ถึง 250 คน สมาชิกของแถวลำดับแต่ละตัวมีค่าเป็นจำนวนเต็ม

```
const
    ClassSize = 250;        {maximum class size}
```

```
type
   ClassIndex = 1 . . Class Size; {subscript type of scoreArray}
   ScoreArray = array [ClassIndex] of Integer;
var
   Scores : ScoreArray; {array of exam scores}
   ClassLength : Integer; {length of filled subarray}
```

กระบวนงาน ReadScores ในรูป 9.16 อ่านคะแนนสอบ 250 ชุด และพิมพ์ข้อความเตือน (warning message) เมื่อแถวลำดับเต็มพารามิเตอร์เอาต์พุต ClassLength แทนความยาวของแถวลำดับเต็มเต็ม และเริ่มต้นให้เป็นศูนย์ ภายใน while ลูป ข้อความสั่ง

```
ClassLength := ClassLength + 1; {Increment ClassLength.}
Scores [ClassLength] : = TempScore; {save the score.}
```

เพิ่มค่า ClassLength และเก็บคะแนนซึ่งเพิ่งอ่านเข้ามา (ค่าของ TempScore) ไว้ในสมาชิกแถวลำดับตัวถัดไป หลังจากออกจากลูปค่าของ ClassLength คือความยาวของแถวลำดับย่อยเดิมเต็ม

ในการประมวลผลต่อมาของแถวลำดับ Scores เราใช้ตัวแปร ClassLength เพื่อจำกัดจำนวนของสมาชิกแถวลำดับที่จะประมวลผล เนื่องจากเฉพาะแถวลำดับย่อยที่มีดรรชนีล่าง 1 ถึง ClassLength เท่านั้น ซึ่งจะมีค่าเป็นข้อมูลที่มีความหมาย (meaningful data) สมาชิกแถวลำดับที่มีค่าดรรชนีล่างมากกว่า ClasssLength ไม่ควรถูกจัดดำเนินการ Class Length ถูกส่งเป็นพารามิเตอร์ไปยังกระบวนงานใดๆ ก็ได้ ซึ่งประมวลผลแถวลำดับย่อยเดิมเต็ม

```
procedure ReadScores (var Scores {output} : ScoreArray;
                      var ClassLength {output} : Integer);
{
   Read an array of exam scores (Scores) for a class of up to ClassSize
   students.
   Pre  :  None.
   Post :  The filled subarray is Scores [1. . ClassLength] and ClassLength
           is the number of values read (0 <= ClassLength <= ClassSize)
```

```
}
   const
      Sentinel = -1;        {sentinel score}
   var
      TempScore : Integer; {temporary storage for a score}

begin
   Write ('Enter next score after the prompt or enter');
   WriteLn (Sentinel : 1 ' to stop : ') I
   {Read each array element until done.}
   ClassLength := 0;   {initial class length}
   Write ('Score > ');
   ReadLn (TempScore);
   while (Temp Score < > Sentinel) and (ClassLength < Class Size) do

      {invariant :
         No prior value of TempScore is Sentinel,
         ClassLength <= ClassSize,
         and Scorees [1 . . ClassLength] is the filled subarray
      }
      begin
         ClassLength := ClassLength + 1; {Increment ClassLength.}
         Scores [ClassLength] := TempScore;
         Write ('Score > ') '
         ReadLn (TemScore)
      end; {while}

   {  assert : Sentinel was read or array is filled.  }
```

```
    if TempScore < > Sentinel then
        WriteLn ('Cannot store ', TempScore, ' – array is full.')
end; {Read Scores}
```

รูป 9.16 ส่วนการอ่านของแถวลำดับ (Reading Part of an Array)

แบบฝึกหัด 9.4

1. ในกระบวนงาน ReadScores มีอะไรเป็นตัวป้องกัน ผู้ใช้จากการใส่คะแนนสอบซึ่งมากกว่า ClassSize

2. กำหนดให้สามารถประมวลผลแถวลำดับย่อยได้ อะไรคือผลที่จะเกิดตามมาของการประกาศ ขอบเขตแถวลำดับขนาดใหญ่กว่าที่จำเป็น

3. ทำไมเราจึงอ่านค่าข้อมูลถัดไปไว้ใน TempScore แทนที่จะอ่านข้อมูลนั้นโดยตรงไว้ใน สมาชิกแถวลำดับถัดไป

เขียนโปรแกรม

1. จงเขียน while ลูป ในกระบวนงาน ReadScores ใหม่เป็น repeat-until loop แล้วอธิบายว่าทำไม while ลูป จึงดีกว่า และทำไมเราจึงไม่ใช้ for ลูป

9.5 การค้นหาและการเรียงลำดับของแถวลำดับ (Searching and Sorting an Array)

ในหัวข้อนี้จะอภิปรายปัญหาร่วมสองชนิดในการประมวลผลแถวลำดับ คือ การค้นหาแถวลำดับเพื่อหาตำแหน่งของค่าที่ต้องการ และการเรียงลำดับของแถวลำดับ เพื่อจัดเรียงใหม่ สมาชิกแถวลำดับในอันดับเชิงตัวเลข (numerical order) เช่น ตัวอย่างของการค้นแถวลำดับ เราต้องการค้นหาแถวลำดับเพื่อหาว่านักศึกษาคนไหน ได้คะแนนที่ระบุไว้ ตัวอย่างของการเรียงลำดับของแถวลำดับอาจเป็นการจัดเรียงใหม่ สมาชิกแถวลำดับ เพื่อให้ทั้งหมดเรียงลำดับคะแนนจากน้อยไปหามาก (increasing order) การเรียงลำดับเช่นนี้จะเป็นประโยชน์ ถ้าเราต้องการให้แสดงผล รายการเรียงลำดับคะแนนสอบ หรือถ้าเราจำเป็นต้องหาตำแหน่งของคะแนนที่แตกต่างกันในแถวลำดับ

การหาค่าเล็กที่สุดในแถวลำดับ (Finding the Smallest Value in an Array)

เริ่มต้นด้วยการแก้ปัญหาการค้นหาอย่างง่าย : การหาค่าเล็กที่สุดในแถวลำดับ

·อัลกอริทึมการค่าเล็กที่สุดในแถวลำดับ (Algorithm for Finding the Smallest Value in an Array)

1. สมมติว่าสมาชิกตัวแรก คือ ค่าที่เล็กที่สุด และเก็บดรรชนีล่าง (subscript) ของมันเป็นดรรชนีล่างของสมาชิกตัวที่มีค่าเล็กที่สุด

2. for สมาชิกแถวลำดับแต่ละตัว do

3. if สมาชิกตัวปัจจุบัน < ตัวที่เล็กที่สุด then

4. save ดรรชนีล่างของสมาชิกตัวปัจจุบัน เป็นดรรชนีล่างของตัวเล็กที่สุด

ฟังก์ชัน FindMin ในรูป 10.17 ปฏิบัติให้เกิดผลของอัลกอริทึมข้างต้นนี้ สำหรับแถวลำดับชนิด ScoreArray (ดูตัวอย่าง 10.10) ระหว่างการวนซ้ำแต่ละครั้งของลูป MinIndex คือดรรชนีล่างของสมาชิกตัวเล็กที่สุด และ W [MinIndex] คือค่าของมัน ฟังก์ชันนี้กลับคืนค่าสุดท้ายซึ่งกำหนดให้ MinIndex ซึ่งเป็นดรรชนีล่างของค่าที่เล็กที่สุดในแถวลำดับ พารา-มิเตอร์ StartIndex และ EndIndex นิยามขอบเขต (boundaries) ของแถวลำดับย่อย W [StartIndex .. EndIndex] ซึ่งค่าที่เล็กที่สุดของมันกำลังถูกพบ การส่งดรรชนีล่างเหล่านี้เป็นอาร์กิวเมนต์มีผลในฟังก์ชันทั่วไปมากกว่า

โปรดสังเกตว่า ฟังก์ชัน FindMin กลับคืนดรรชนีล่างของค่าที่เล็กที่สุด ไม่ใช่ค่าที่เล็กที่สุด สมมติว่า Next เป็นข้อมูลชนิด Integer และ ClassLength คือ จำนวนของสมาชิกแถวลำดับซึ่งเก็บข้อมูล ข้อความสั่งต่อไปนี้ แสดงผลค่าเล็กที่สุดในแถวลำดับ Scores (ชนิด ScoreArray)

```
IndexOfMin := FindMin (Scores, 1, ClassLength);
WriteLn ('The smallest exam score is ', Scores [IndexOfMin] : 1)
```

ข้อความสั่งหนึ่งคำสั่งข้างล่างนี้ ถึงแม้ว่าจะอ่านยาก แต่มีความหมายเหมือนกัน เราใช้ function designator เป็นนิพจน์ดรรชนีล่าง (subscript expression)

```
WriteLn ('The smallest exam score is ',
    Scores [FindMin (Scores, 1, ClassLength)] : 1)
```

```
Function FindMin (W {input} : ScoreArray;
   StartIndex, EndIndex {input} : ClassIndex) : Integer;
{
   Returns the subscript of the smallest element in the subarray W [StartIndex
   . . EndIndex] .
   Pre  :  StatIndex and EndIndex are defined and W [StartIndex .
           EndIndex] is part of the fill subarray.
   Post :  Returns K if W [K] <= W [I] for all I in subrange StartIndex.
           EndIndex. If the smallest value appears in twor or more elements,
           K is the subscript of the first.
}
   var
      MinIndex, {index of smallest so far}
      NextIndex : Integer; {index of current element}

·begin {FindMin}
   MinIndex := StartIndex; {Assume first entry is smallest.}
   for NextIndex := StartIndex + 1 to EndIndex do
      if  W [NextIndex] < W [MinIndex] then
         MinIndex := NextIndex; {Value at NextIndex is smallest.}

      {  assert :
         All elements are examined and MinIndex is the index of the smallest
         element.
      }
      FindMin := MinIndex {Define result.}
·end; {FindMin}
```

รูป 9.17 ฟังก์ชัน FindMin

การค้นหาแถวลำดับ (Searching an Array)

เราสามารถค้นหาแถวลำดับสำหรับคะแนนเฉพาะ โดยการเปรียบเทียบสมาชิกแถวลำดับแต่ละตัว เริ่มต้นจากตัวแรกกับคะแนนเป้าหมาย (target score) ค่าซึ่งเรากำลังค้นหาถ้ามีการจับคู่กัน (match occurs) แสดงว่าเราพบคะแนนเป้าหมายในแถวลำดับ และสามารถกลับคืน ดรรชนีล่างของมัน เป็นผลลัพธ์ของการค้นหา กรณีอื่นๆ เราค้นหาต่อไปจนกระทั่งเราพบการจับคู่กัน หรือทดสอบสมาชิกแถวลำดับทั้งหมดแล้ว แต่ไม่พบ

ความต้องการข้อมูลสำหรับฟังก์ชันการค้นหา

อินพุตพารามิเตอร์ (Input Parameters)

```
Scores       : ScoreArray    {array to be searched}
ClassLength  : ClassIndex    {number of elements in Scores}
Target       : Integer       {score being searched for}
```

ฟังก์ชันเอาต์พุต (Function Output)

ดรรชนีล่างของสมาชิกตัวแรกซึ่งเก็บค่า Target หรือศูนย์ถ้าไม่พบ Target

ตัวแปรของโปรแกรม (Program Variables)

```
Next      : Integer   {subscript of next score to test}
NoTarget  : Boolean   {program flag – true if target has not found in elements
                       tested so far}
```

อัลกอริทึมสำหรับฟังก์ชันการค้นหา (Algorithm for Search Function)

1. เริ่มต้นด้วยสมาชิกแถวลำดับตัวแรก
2. ให้ NoTarget เป็น True
3. while ไม่พบ target และยังมีสมาชิกแถวลำดับอีก do
   4. if สมาชิกตัวปัจจุบัน มีค่าเป็นคะแนนเป้าหมาย then
      ให้ NoTarget เป็น False else
      พยายามกับสมชิกตัวถัดไป
5. if ไม่พบ target then
   return 0
   else
   return ดรรชนีล่างของ target

ในขั้นที่ 3 while loop กระทำการจนกระทั่งมันพบสมาชิกแถวลำดับ ซึ่งมีค่าเป็นคะแนนเป้าหมาย หรือ มีการทดสอบสมาชิกแถวลำดับทั้งหมดแล้วแต่ไม่พบคะแนนเป้าหมาย

ขั้นที่ 4 เปรียบเทียบสมาชิกแถวลำดับตัวปัจจุบัน (ถูกเลือกโดยดรรชนีล่าง Next) กับคะแนนเป้าหมาย และถ้ามันจับคู่กันให้ NoTarget เป็น False ถ้ามันไม่จับคู่กันให้ดรรชนีล่าง Next มีค่าเพิ่มขึ้นอีก 1 หลังจากออกจากลูป ขั้นที่ 5 ข้อความสั่ง if นิยาม ผลลัพธ์ของฟังก์ชันเป็น 0 (ถ้าไม่พบเป้าหมาย) หรือนิยามเป็นค่าของ Next เมื่อพบการจับคู่กัน รูป 9.18 แสดงฟังก์ชัน Search

ตัวบ่งชี้โปรแกรม (program flag) NoTarget ควบคุมการทำซ้ำ ลูปและสื่อสารผลลัพธ์ ของลูปการค้นหากับข้อความสั่ง if ซึ่งตามหลังลูป NoTarget ถูกตั้งให้เป็น True ก่อนเข้าไปยังลูปการค้นหา และตั้งใหม่ (reset) ให้เป็น False ทันทีที่สมาชิกทดสอบจับคู่กับเป้าหมาย ถ้าไม่มีสมาชิกแถวลำดับตัวใดเลยจับคู่กับเป้าหมาย เมื่อการค้นแถวลำดับทั้งหมดกระทำแล้ว NoTarget จะเป็นจริง ข้อความสั่ง if กลับคืน 0 เมื่อ NoTarget เป็นจริง กรณีอื่นๆ มันกลับคืนค่าของดรรชนีล่าง Next และเป็นเหตุให้ NoTarget เปลี่ยนเป็น False

ลูปยืนยง (loop invariant) คือ

{invariant :

Target was not found in subarray Scores [1. . Next-1]

and Next is <= ClassLength + 1 .

}

สิ่งนี้หมายความว่า ค่าแต่ละค่าของ Next (เริ่มต้นด้วย 1) ไม่พบเป้าหมายในสมาชิกแถวลำดับ ที่มีดรรชนีล่างน้อยกว่า Next เพราะว่า Next <= ClassLength + 1 ต้องเป็นจริง หลังจากออกจากลูป Scores [ClassLength] คือสมาชิแถวลำดับตัวสุดท้าย ซึ่งจะถูกเปรียบเทียบกับเป้าหมาย

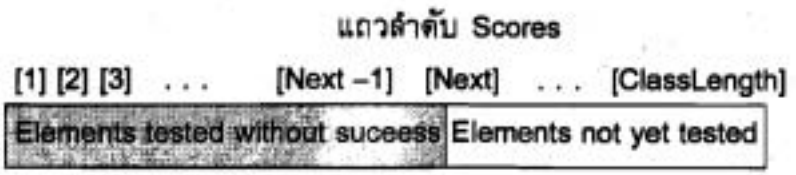
```
function Search  (Scores : ScoreArray;
                  ClassLength : ClassIndex;
                  Target : Integer) : Integer;
{
   Searches for Target in array Scores.
```

```
Pre  :  1 <= ClassLength <= ClassSize and
        Scores [1 . . ClassLength] is the filled subarray.
Post :  Returns the subscript to Terget if found;
        Otherwise, return zero.
var
   Next : Integer;        {index of the current score}
   NoTarget : Boolean;    {program flag – true if Target has not been
                           found in elements tested so far}

begin {Search}
   {Compare each element of scores to Target until done.}
   Next : = 1;            {start with the first score.}
   NoTarget : = True;     {Target is not found.}
   While NoTarget and (Next <= ClassLength) do

      {invariant :
         Target was not found in subarray Scores [1 . . Next –1]
         and Next is ≤ ClassLength + 1.
      }
      if Scores [Next] = Target then
         NoTarget := False {Target is Found.}
      else
         Next := Next + 1; {Advance to next score.}

   {assert :
      Target was found or all elements tested without success.
   }
   if  NoTarget then
```

```
        Search : = 0 {Target was not found.}
    else
        Search := Next {Target found at Scores [Next].}
end; {Search}
```

รูป 9.18 ฟังก์ชัน Search

ในค้าร่างแถวลำดับ Scores ข้างล่างนี้



สมาชิกในส่วนซ้ายมือที่แรเงาของแถวลำดับ หมายถึงสมาชิกต่างๆ ซึ่งได้ทดสอบแล้ว และสมาชิกซึ่งมีดรรชนีล่างเป็น Next กำลังจะถูกทดสอบในการวนซ้ำลูป ปัจจุบัน ถ้า Scores [Next] ไม่จับคู่กับเป้าหมาย ส่วนทางซ้ายมือซึ่งแรเงาของแถวลำดับจะโตขึ้นอีกหนึ่งสมาชิก และค่าของ next จะเพิ่มขึ้นอีก 1

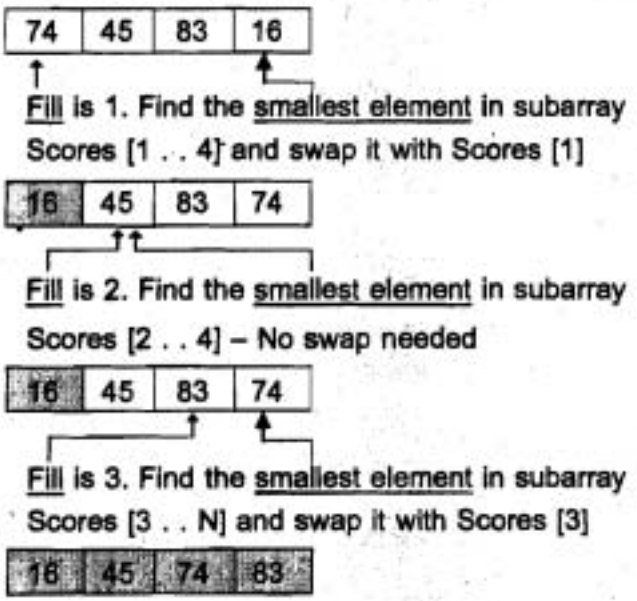
ตัวยืนยง (invariant) เป็นจริงก่อนการวนซ้ำครั้งแรก (Next มีค่าเท่ากับ 1) เพราะว่าไม่มีสมาชิกแถวลำดับซึ่งอยู่ก่อนสมาชิกที่มีดรรชนีล่างเป็น 1 ถ้าสมาชิกตัวปัจจุบันจับคู่กับเป้าหมายจะออกจากลูป ซึ่งไม่เปลี่ยนแปลงค่าของ Next ดังนั้นตัวยืนยงยังคงเป็นจริง ถ้า Next มีค่าเป็น ClassLength + 1, สมาชิกแถวลำดับทั้งหมด มีการทดสอบแล้ว โดยไม่ประสบผลสำเร็จ และการออกจากลูปเกิดขึ้น

การเรียงลำดับของแถวลำดับ (Sorting an Array)

โปรแกรมจำนวนมากกระทำการอย่างมีประสิทธิภาพมากขึ้น ถ้าข้อมูล ซึ่งมันประมวลผลมีการเรียงลำดับแล้ว ก่อนการเริ่มต้นประมวลผล ตัวอย่างเช่น โปรแกรมประมวลผลเช็ค กระทำการอย่างรวดเร็วมากขึ้น ถ้าเช็คทั้งหมดมีการเรียงอันดับหมายเลขบัญชี โปรแกรมอื่นๆ ให้เอาต์พุตที่สามารถเข้าใจได้มากกว่า ถ้าสารสนเทศนั้นมีการเรียงลำดับก่อนจะถูกแสดงผล ตัวอย่างเช่น มหาวิทยาลัยต้องการให้ผู้สอนออกรายงานเกรดโดยเรียงตามลำดับของรหัสประจำตัวของนักศึกษา ในหัวข้อนี้ เราจะอธิบายอัลกอริทึมการเรียงลำดับอย่างง่าย ชนิดหนึ่ง จากที่มีอยู่จำนวนมากซึ่งมีการศึกษา โดยนักวิทยาศาสตร์คอมพิวเตอร์ (computer seientists)

การเรียงลำดับแบบเลือก (selection sort) หมายถึงอัลกอริทึมการเรียงลำดับโดยสัญชาติญาน การกระทำการเรียงลำดับแบบเลือกของแถวลำดับที่มีสมาชิก N ตัว (ดรรชนีล่าง 1 . . N) เราหาสมาชิกตัวที่เล็กที่สุดในแถวลำดับ และสลับที่สมาชิกตัวที่เล็กที่สุดกับสมาชิกตัวที่มีดรรชนีล่างเป็น 1 นั่นคือ เอาสมาชิกตัวที่เล็กที่สุดมาวางที่ตำแหน่งที่ 1 จากนั้นเราหาสมาชิกตัวที่เล็กที่สุดที่เหลืออยู่ในแถวลำดับย่อยกับดรรชนีล่าง 2 . . N และสลับที่มันกับสมาชิกตัวที่มีดรรชนีล่างเป็น 2 นั่นคือ เราใส่สมาชิกตัวที่เล็กเป็นอันดับสองที่ตำแหน่งที่ 2 จากนั้นเราหาสมาชิกตัวที่เล็กที่สุดที่เหลืออยู่ ในแถวลำดับย่อยกับดรรชนีล่าง 3 . . N และสลับที่มันกับสมาชิกที่มีดรรชนีล่างเป็น 3 เช่นนี้เรื่อยไป

.รูป 9.19 ตามรอยการดำเนินการของอัลกอริทึมการเรียงลำดับแบบเลือกบนแถวลำดับย่อย ที่มีสมาชิก 4 ตัว แถวลำดับชุดแรกที่แสดงให้เห็น คือ แถวลำดับเริ่มต้น จากนั้นเราแสดงแต่ละขั้นตอน คือ สมาชิกตัวที่เล็กที่สุดถัดไป ย้ายไปยังตำแหน่งถูกต้องของมัน แถวลำดับย่อยที่แรเงาไว้ แทนส่วนของแถวลำดับแต่ละชุด ซึ่งถูกเรียงลำดับแล้ว โปรดสังเกตว่ามีการสับเปลี่ยนอย่างมากที่สุด N-1 ครั้ง ในการเรียงลำดับของแถวลำดับที่มีสมาชิก N ตัว



รูป 9.19 ตามรอยการเรียงลำดับแบบเลือก

อัลกอริทึมสำหรับการเรียงลำดับแบบเลือก (Algorithm for Selection Sort)

1. for Fill : = 1 to N-1 do
   2. Find the position of the smallest element in subarray Scores [Fill . . N].
   3. If Fill is not the position of the smallest element then
      4. Exchange the smallest element with the one at position Fill.

เราสามารถใช้ฟังก์ชัน FindMin (ดูรูป 9.17) เพื่อให้กระทำขั้นตอนที่ 2 กระบวนงาน SelectSort ในรูป 9.20 ปฏิบัติให้เกิดผลของอัลกอริทึมการเรียงลำดับแบบเลือก (Selection sort) สำหรับแถวลำดับ Scores ที่มีสมาชิก ClassLength ตัว ตัวแปรเฉพาะที่ IndexOfMin เก็บดรรชนีล่างของคะแนนสอบที่น้อยที่สุดที่พบในแถวลำดับย่อยปัจจุบัน ที่ตอนจบของแต่ละ pass เราเรียกกระบวนงาน Switch (ดูรูป 9.6) เพื่อสับเปลี่ยนสมาชิกตัวที่มีดรรชนีล่างเป็น IndexOfMin กับสมาชิกตัวที่มีดรรชนีล่างเป็น Fill เมื่อ IndexOfMin และ Fill มีค่าไม่เท่ากัน หลังจากการกระทำการของกระบวนงาน SelectionSort แล้ว คะแนนสอบจะเรียงอันดับจากน้อยไปหามาก เราต้องดัดแปร (modify) กระบวนงาน Switch เพื่อให้ยอมรับพารามิเตอร์ชนิด Integer

ตัวยืนยงลูป สำหรับลูปภายนอก (outer loop)

{invariant :

The elements in Scores [1 . . Fill –1] are in their proper place and Fill <= ClassLength.

}

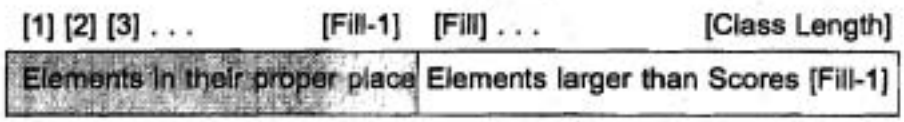
```
procedure SelectSort  (var Scores {input / output} : ScoreArray;
                       ClassLength {input} : Integer);
{
   Sorts the data in array Scores.
   Pre  :  1 <= ClassLensth <= ClassSize and the filled subarray is Scores
           [1 . . ClassLength] .
   Post :  the values in Scores [1 . . ClassLength] are in increasing order.
   Calls : Procedures Switch and FindMin
```

```
}
   var
      Fill,   {index of element to contain next smallest score}
      IndexOfMin : Integer; {index of next smallest element}

begin {SelectionSort}
   for Fill := 1 to ClassLength –1 do
      begin
         {invariant :
            The elements in Scores [1 . . Fill –1] are in their proper place and
            Fill <= ClassLength.
         }

         {Find smallest score in Scores [Fill . . ClassLength] .}
         IndexOfMin := FindMin (Scores, Fill, ClassLength);
         {Exchange elements at Fill and IndexOfMin.}
         if IndexOfMin < > Fill then
            Switch (Scores [IndexOfMin], Scores [Fill])
      end {for Fill}
end; {SelectSort}
```

รูป 9.20 กระบวนงาน Selection Sort

สรุปความก้าวหน้าของการเรียงลำดับแบบเลือก แถวลำดับย่อยซึ่งสมาชิกของมันอยู่ในตำแหน่งถูกต้องแสดงในส่วนแรเงาของแถวลำดับ Scores ในร่างข้างล่างนี้ สมาชิกส่วนที่เหลือยังไม่อยู่ในตำแหน่ง และทั้งหมดมีค่าใหญ่กว่า Scores[Fill – 1]

แถวลำดับ Scores



ระหว่างแต่ละ pass ส่วนของแถวลำดับซึ่งแรเงาจะเติบโตทีละหนึ่งตัว และ Fill จะเพิ่มเพื่อสะท้อนสิ่งนี้ เมื่อ Fill มีค่าเท่ากับ ClassLength สมาชิก ClassLength –1 ตัวแรกจะอยู่ในตำแหน่งถูกต้อง ดังนั้น Scores [ClassLength] จึงต้องอยู่ในตำแหน่งถูกต้องของมันด้วย

สไตล์ของโปรแกรม (Program Style)

การสร้างและการใช้ส่วนจำเพาะอเนกประสงค์ (Creating and Using General Purpose Modules)

ฟังก์ชัน FindMin เป็นส่วนจำเพาะอเนกประสงค์ ซึ่งหาสมาชิกตัวเล็กที่สุดในแถวลำดับย่อยใดๆ ของพารามิเตอร์แถวลำดับของมัน การเขียน FindMin ในวิธีนี้ทำให้เราสามารถใช้มันในกระบวนงาน SelectSort ทำให้การลงรหัส SelectSort ง่ายขึ้น

แบบฝึกหัด 9.5

1. สำหรับฟังก์ชัน Search ในรูป 9.8 จะเกิดอะไรขึ้น ถ้า
   a) คะแนนสอบของนักศึกษาคนสุดท้ายจับคู่กับเป้าหมาย
   b) มีคะแนนสอบหลายชุดจับคู่กับเป้าหมาย
2. จงตามรอยการกระทำการของการเรียงลำดับแบบเลือกของรายการข้อมูลสอง ชุดข้างล่างนี้

   10 55 34 56 76 5 5 15 25 35 45 45
3. เราจะดัดแปร (modify) อัลกอริทึมการเรียงลำดับแบบเลือกอย่างไร เพื่อให้ได้คะแนนเรียงอันดับจากมากไปหาน้อย (คะแนนตัวแรกมีค่ามากที่สุด)

เขียนโปรแกรม (Programming)

1. จงเขียนกระบวนงาน เพื่อหาสมาชิกตัวเล็กที่สุดถัดไปของแถวลำดับ สมมติว่าไม่มีสมาชิกที่ซ้ำกันในแถวลำดับ และกลับคืน startIndex ถ้าแถวลำดับมีหน่วยข้อมูลเพียงหนึ่งตัวเท่านั้น
2. อีกวิธีหนึ่งของการกระทำ การเรียกลำดับแบบเลือกคือใส่ค่ามากที่สุดในตำแหน่ง N, ค่ามากที่สุดตัวถัดไป ในตำแหน่ง N-1 เช่นนี้เรื่อยไป จงเขียนกระบวนงานเวอร์ชันนี้
3. อีกเทคนิคหนึ่ง สำหรับการทำให้เกิดผลของการค้นหาแถวลำดับ โดยไม่ต้องมีตัวปงชี้โปรแกรม (program flag) คือใช้ while ลูป ซึ่งเพิ่มค่า increment Next ตราบใดที่

ทั้งคู่ของข้อความสั่งต่อไปนี้เป็นจริง : เป้าหมายไม่จับคู่กับสมาชิกตัวปัจจุบัน และ Next มีค่าน้อยกว่า ClassLength หลังจากออกจากลูป สมาชิกตัวที่ตำแหน่ง Next สามารถทดสอบใหม่ เพื่อหาผลลัพธ์ของฟังก์ชัน ถ้าสมาชิกตัวนี้ จับคู่กับเป้าหมาย ผลลัพธ์ คือ Next กรณีอื่นๆ ผลลัพธ์เป็น 0 จงเขียน body ของฟังก์ชัน

9.6 แถวลำดับที่มีสมาชิกและตรรชนีล่างเป็นชนิด Char

แถวลำดับจำนวนมาก ข้อมูลมีค่าเป็นตัวเลข เช่น คะแนนสอบ หรือจำนวนชั่วโมงทำงาน แถวลำดับซึ่งจะแนะนำในหัวข้อนี้ เก็บข้อมูลอักขระ (character data) นอกเหนือจากการเก็บข้อมูลอักขระในแถวลำดับแล้ว เราสามารถใช้อักขระเป็นตรรชนีล่างแถวลำดับได้

แถวลำดับของอักขระ กับตัวแปรสายอักขระ (Arrays of Characters Versus String Variableds)

Turbo Pascal ยอมให้เราใช้แถวลำดับของอักขระ หรือตัวแปรสายอักขระ เพื่อเก็บกลุ่มของอักขระในหน่วยความจำ การประกาศต่อไปนี้ จัดสรรหน่วยเก็บสำหรับแถวลำดับ FirstName และตัวแปรสายอักขระ LastName ซึ่งแต่ละตัวใช้หน่วยเก็บ สำหรับอักขระ 20 ตัว

```
Const
   StringSize = 20;
type
   IndexRange = 1 . . StringSize;
   CharArray = array [IndexRange] of char;
   String20 = string [StringSize];
var
   FirstName : CharArray; {array of 20 characters}
   LastName : String20; {string of up to 20 characters}
```

เราประมวลผลแถวลำดับ FirstName เหมือนกับแถวลำดับอื่นๆ ใน Pascal การอ่านข้อมูลไว้ในแถวลำดับ FirstName หรือแสดงผล contents ของมัน จำเป็นต้องใช้ลูป

การประกาศชนิด String20 แสดงว่าตัวแปรชนิดนี้ (ตัวอย่างเช่น LastName) สามารถเก็บอักขระได้มากถึง 20 ตัว แทนที่จะเป็นค่ามากสุดโดยปริยาย (default maximum)

สำหรับตัวแปรสายอักขระ ซึ่งเป็นอักขระ 255 ตัว เราสามารถทำงานกับตัวแปรสายอักขระ LastName ง่ายกว่าแถวลำดับ First Name ด้วยเหตุผล หลายข้อ ดังนี้

1) เราสามารถใช้กระบวนงาน Read หรือ ReadLn และ Write หรือ WriteLn ของ Pascal เพื่ออ่านค่าข้อมูล ไว้ใน LastName หรือแสดงผลค่าใน LastName

2) เราสามารถเปรียบเทียบ LastName กับค่าสายอักขระ หรือตัวแปรสายอักขระอีกหนึ่งตัว โดยใช้ตัวดำเนินการสัมพันธ์ (relational operators) ตัวอย่างเช่น

LastName = 'ZZZZZ' เป็นจริง ถ้า LastName ประกอบด้วยอักขระ Z ห้าตัว

LastName < 'ZZZZZ' เป็นจริง ถ้า LastName ประกอบด้วยสายอักขระ Z น้อยกว่าห้าตัว

3) เราสามารถใช้ข้อความสั่ง กำหนดค่าเช่น

LastName := ' Washington'

เพื่อเก็บค่าสายอักขระ ใน LastName

ความยาวของตัวแปรสายอักขระ LastName เป็นพลวัต (dynamic) หมายถึง เปลี่ยนแปลงได้ และถูกนิยามเป็นจำนวนอักขระ ซึ่งเก็บไว้ (เช่นในการกำหนดค่าข้างต้น มีค่าเท่ากับ 10) ฟังก์ชัน designator Length (LastName) กลับคืนความยาวของ LastName

เราสามารถอ้างถึงอักขระแต่ละตัวใน FirstName และ LastName ได้ทั้งคู่ ตัวอย่าง เช่น FirstName [1] และ LastName [1] เข้าถึงอักขระตัวแรกของแถวลำดับ และ สายอักขระตามลำดับ ตัวแปรดรรชนีล่าง LastName [Length (LastName)] เข้าถึงอักขระ ตัวสุดท้ายที่เก็บในสายอักขระ LastName ในบทที่ 12 จะอธิบายการประมวลผลสายอักขระในรายละเอียดเพิ่มขึ้น

Syntax Display

ฟังก์ชัน Length (สำหรับตัวแปรสายอักขระ)

Form : | Length (string) |

ตัวอย่าง : Length (Name)

มีความหมายดังนี้ : ฟังก์ชัน Length กลับคืนเลขจำนวนเต็มระบุ ถึงจำนวนอักขระปัจจุบันที่เก็บใน string

แถวลำดับกับดรรชนีล่างชนิด Char (Arrays with Type Char Subscripts)

บ่อยครั้งที่เราสามารถใช้แถวลำดับกับดรรชนีล่างชนิด Char (หรือ subrange ของ Char) เพื่อทำให้โปรแกรม ซึ่งประมวลผลตัวอักขระง่ายขึ้น ตัวอย่าง 9.11 และกรณีศึกษาซึ่งตามมาใช้แถวลำดับซึ่งดรรชนีล่างของมันเป็นอักษรตัวใหญ่ (uppercase letters)

.ตัวอย่าง 9.11

โปรแกรมในรูป 9.21 แสดงผล จำนวนการเกิดของอักขระแต่ละตัวในหนึ่งบรรทัดของข้อความ เราใช้แถวลำดับ LetterCount (ดรรชนีล่างชนิด 'A' . . 'Z') เพื่อเก็บจำนวนการเกิดของตัวอักษรแต่ละตัว (ตัวอย่างเช่น LetterCount ['A'] หมายถึงการเกิดของตัวอักษร A) กระบวนงาน CountLetter อ่านอักขระข้อมูล แต่ละตัวไว้ใน Ch และเพิ่มค่าสมาชิกของแถวลำดับ LetterCount ซึ่งเลือกโดยอักขระข้อมูล Ch ฟังก์ชัน UpCase (ดูหัวข้อ 7.4) เปลี่ยนตัวอักษรที่อ่านแต่ละตัวให้เป็นตัวใหญ่ ดังนั้นตัวอักษร t และ T ทั้งคู่ นับตัวอักษร T เพิ่มขึ้น กระบวนงาน FillCountArray คล้ายกับ FillArray (ดูรูป 9.8) กระบวนงาน PrintCount แสดงผลเป็นตารางให้เห็นอักษรแต่ละตัวและจำนวนการเกิดของอักษรแต่ละตัว

```
Edit Window
{$R+}
program Concordance;
{
   Finds and prints the number of occurrences of each letter. The case of
   each character is immaterial. Letters with counts of zero are not displayed.
}
   type
      Letter = 'A' . . 'Z';
      CountArray = array [Letter] of Integer;
   var
      Letter Count : CountArray; {output – array of counts}

   {Insert procedure FillCountArray here.}
```

```
procedure CountLetters (var LetterCount {output} : CountArray);
{
   Counts the number of occurrences of each letter.
   Pre  :  LetterCount is initialized to all zeros.
   Post :  Reads next data line and LetterCount (Ch) is the number of
           occurences of letter Ch in the line.
}
   var
      Ch : Char;              {each data character}

begin {CountLetters}
   WriteLn ('Type in a data line :');
   while not EOLN do
      begin
         Read (Ch);           {Get next character.}
         Ch := UpCase (Ch); {Convert to uppercase.}
         If (Cn >= 'A') and (Ch <= 'Z' then
            LetterCount [Ch] := LetterCount [Ch] +1
      end; {while}
   ReadLn    {skip the <eoln>.}
end; {CountLetters}
procedure PrintCount (LetterCount {input} : CountArray);
{
   Prints counts of letters.
   Pre  :  LetterCount is initialized.
   Post :  Displays each letter and its count if non-zero.
}
   var
      NextChar : Letter : {loop control and subscript}
```

```
    begin {PrintCount}
      WriteLn;
      WriteLn ('Letter', ' Occurences' : 16);
      for NextChar := 'A' to 'Z' do
        if  LetterCount [NextChar] > 0 then
          WriteLn (NextChar : 6, LetterCount [NextChar] : 16)
    end; {PrintCount}

begin {Concordance}
    {Initialize (LetterCount, 0);
    FillCountArray (LetterCount, 0);
    {Count the letters in a line.}
      CountLetters (LetterCount);
      {Print counts of letters that are in the line.}
      PrintCount (LetterCount)
end. {Concordance}
```

Output Window

```
Type in a data line :
This is it !
Letter Occurrences
   H          1
   I          3
   S          2
   T          2
```

รูป 9.21 การนับจำนวนตัวอักษรในหนึ่งบรรทัด

กรณีศึกษา ปัญหาการสร้างรหัสลับ (Cryptogram Generator Problem)

รหัสลับ หมายถึง ข้อความเข้ารหัสประกอบจากการแทนที่อักขระรหัสให้กับตัวอักษรแต่ละตัวของข้อความต้นฉบับ (A cryptogram is a coded message formed by substituting a code character for each letter to the original message.)

คอมพิวเตอร์ สามารถโปรแกรมเพื่อให้สร้างข้อความลงรหัสเหล่านี้

ปัญหา (Problem)

สร้างโปรแกรมรหัสลับ ซึ่งการแทนที่ถูกกระทำเป็นแบบเดียวกันตลอดทั้งข้อความต้นฉบับ – ตัวอย่างเช่น อักษร A ทุกตัวถูกแทนด้วย S, อักษร B ทุกตัวถูกแทนด้วย P เช่นนี้เรื่อยไป เครื่องหมายวรรคตอน (รวมทั้งอักขระว่างระหว่างคำ) ยังคงเหมือนเดิม

วิเคราะห์ (Analysis)

โปรแกรมต้องตรวจอักขระแต่ละตัวในข้อความและแทนที่อักขระแต่ละตัวซึ่งเป็นตัวอักษรด้วยสัญลักษณ์รหัสของมัน (its code symbol) เราเก็บสัญลักษณ์รหัสในแถวลำดับ Code ซึ่งมีดรรชนีล่างเป็น 'A' . . ' Z ' และ สมาชิกเป็นชนิด Char ตัวอักขระซึ่งเก็บใน Code ['A'] จะเก็บสัญลักษณ์สำหรับตัวอักษร 'A' สิ่งนี้จะทำให้เราสามารถค้นดูสัญลักษณ์รหัสสำหรับตัวอักษรได้ง่าย โดยใช้ตัวอักษรเป็นดรรชนี ของแถวลำดับ Code ในตัวอย่างแถวลำดับ Code ซึ่งแสดงต่อไปนี้ ตัวอักษรแต่ละตัวถูกแทนที่ด้วยตัวอักษรถัดไปในพยัญชนะ และตัวอักษร Z ถูกแทนที่ด้วย A

แถวลำดับ Code

['A']	['B']	['C']	['D']		['Y']	['Z ']
B	C	D	E	. . .	Z	A

ข้อมูลที่ต้องการ (Data Reguirements)

Problem Inputs

·Code : array [Letter] of Char {array of code symbol}

Each message character

Problem Outputs

Each character of the cryptogram

ออกแบบ (Design)

Initial Algorithm

1. Read in the code symbol for each letter.

2. Read each message character and display the cryptogram.

Algorithm Refinements and the Structure Chart

จากแผนภูมิโครงสร้าง (รูป 9.22) จะเห็นว่ากระบวนงาน ReadCode กระทำขั้นที่ 1 และกระบวนงาน Encrypt กระทำขั้นที่ 2 ข้อมูลที่ต้องการและอัลกอริทึมสำหรับกระบวนงานเหล่านี้เป็นดังนี้

ตัวแปรเฉพาะที่สำหรับ ReadCode

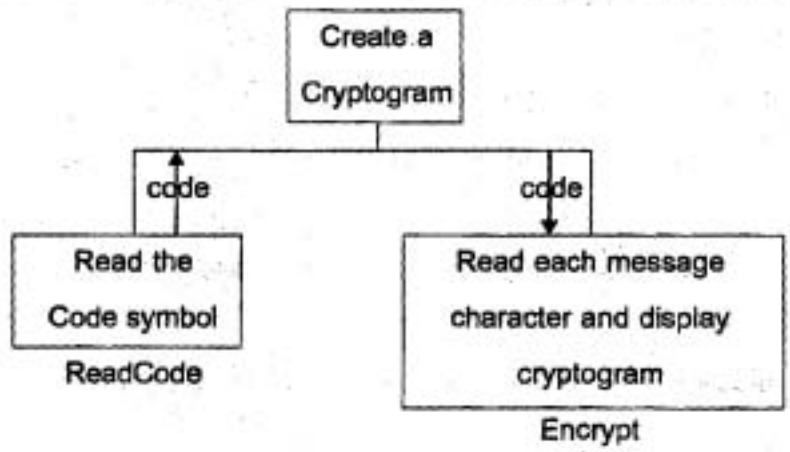
NextLetter : 'A ' . . 'Z ' {Loop – control variable and array subscript}

อัลกอริทึมสำหรับ ReadCode

1. Display the alphabet.

2. for each letter do

   3. Read in the code symbol and store it in array Code.



รูป 9.22 แผนภูมิโครงสร้างสำหรับ Cryptogram Generator

ตัวแปรเฉพาะที่สำหรับ Encrypt

Ch : Char {each message character}

อัลกอริทึมสำหรับ Encrypt

1. while there are more message characters do

```
begin
   2. Read the next message character.
   3. Display the message character of its code symbol.
end
```

การทำให้เกิดผล (Implementation)

โปรแกรมในรูป 9.23 สมมติว่า ตัวอักษรใหญ่เป็นอักขระติดต่อกัน เช่นที่มันอยู่ในชุดอักขระ ASCII

```
Edit Window
{$R+}
program Cryptogram;
{Generates cryptognams corresponding to input messages}
   type
      Letter = 'A' . . 'Z';
      CodeArray = array [Letter] of Char;
   var
      Code : CodeArray; {input – array of code symbols}

   procedure ReadCode (var Code {output} : CodeArray);
   {
      Read in the code symbol for each character.
      Pre  :  None.
      Post :  26 data values are read into array Code.
   }
      var
         NextLetter : Letter; {each letter}
   begin {ReadCode}
```

```
    WriteLn ('First specify the code.');
    WriteLn ('Enter a code symbol under each letter.');
    WriteLn ('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
    {Read each code symbol into array Code.}
    for NextLetter := ' A ' to ' Z ' do
       Read (Code [NextLetter]);
    ReadLn;        {skip the <eoln>.}
    WriteLn;       {skip a line.}
end; {ReadCode}

procedure Encrypt (Code {input} : CodeArray);
{
    Reads each character and prints it or its code symbol.
    Pre  :  Array Code is defined.
    Post :  Read a data line. A code symbol was printed for each letter;
            each nonletter was printed
}
    var
       Ch : Char; {input – each message character}
begin {Encrypt}
    WriteLn ('Enter each character of your message : ');
    while not EOLN do
       begin
          Read (Ch);
          Ch := UpCase (Ch); {Convert to uppercase.}
       If (Ch >= 'A') and (Ch <= 'Z') then
          Write (Code [Ch] {Print code symbol.}
       else
```

```
            Write (Ch)     {Print nonletter}
          end; {while}
        ReadLn                {skip the <eoln>.}
      end; {Endcrypt}

begin {Cryptogram}
  {Read in the code symbol for each letter.}
  ReadCode (Code);

  {Read each character and print if or its code symbol.}
  Encryt (Code)
end. {Cryptegram}
```

Output Window

```
First specify the code.
Enter a code symbol under each letter.
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA

Enter each character of your message :
A ting      me! #
B UJOZ    PDF! #
```

รูป 9.23 Cryptogram Gencrator

การทดสอบ (Testing)

ในการวิ่งตัวอย่างที่แล้ว สัญลักษณ์รหัสสำหรับตัวอักษรแต่ละตัวใส่เข้าไป โดยตรงได้ตัวอักษรตัวนั้น และอ่านโดยโปรซ์เดอร์ ReadCode การวิ่งตัวอย่างจบด้วยสอง

บรรทัดของเอาต์พุต บรรทัดแรกประกอบด้วย ข้อความ บรรทัดที่สอง ประกอบด้วย รหัสลับ สำหรับการทดสอบอย่างง่าย พยายามใช้ตัวอักษรแต่ละตัว เป็นสัญลักษณ์ รหัสของมันเอง ในกรณีเช่นนี้ ทั้งสองบรรทัดควรจะเหมือนกัน จงมั่นใจว่าโปรแกรมเข้ารหัส อักษรตัวเล็กได้เช่นเดียวกับอักษรตัวใหญ่ โปรแกรมจะไม่เปลี่ยนแปลงตัวอักขระที่ไม่ใช่ตัวอักษร

แบบฝึกหัด 9.7

1. จะมีอะไรเปลี่ยนแปลงกับ cryptogram generator บ้างถ้ากำหนดรหัสเหมือนกัน และมันควรจะถอดรหัส ไม่ใช่ เข้ารหัสลับ

เขียนโปรแกรม

1. จงเปลี่ยนแปลงกับโปรแกรม cryptogram เพื่อให้เข้ารหัส (encode) อักขระว่างและสัญลักษณ์วรรคตอน , ; : ? ! . ข้อแนะนำ ชนิดตรรชนีล่างเป็น Char

2. โปรแกรม cryptogram generator จะทำงานไม่ถูกต้อง ถ้าผู้ใช้ใส่สัญลักษณ์รหัสเหมือนกันมากกว่าหนึ่งครั้งจงดัดแปร ReadCode เพื่อให้ข้อผิดพลาดเช่นนี้บังคับผู้ใช้ให้ใส่ใหม่ (reenter) รหัสที่ไม่กำกวม

9.7 การแก้จุดบกพร่องโปรแกรมด้วยแถวลำดับ (Debugging Programs with Arrays)

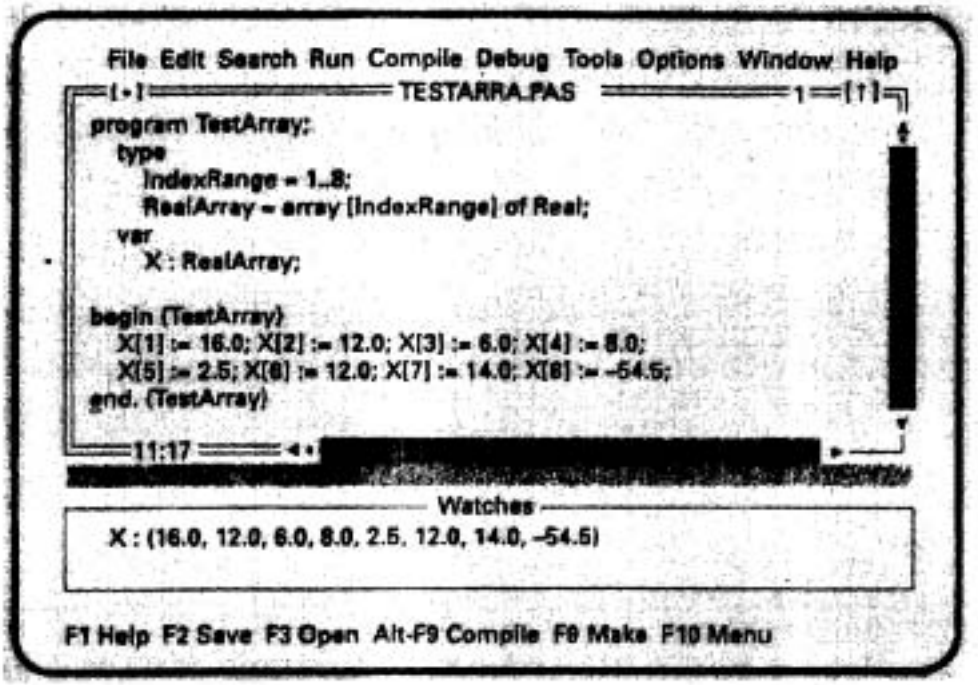
เมื่อแก้จุดบกพร่องโปรแกรม (หรือส่วนจำเพาะ) ซึ่งประมวลผลแถวลำดับ วิธีที่ดีที่สุด คือ ทดสอบโปรแกรมบนแถวลำดับที่มีจำนวนสมาชิกน้อย ถ้าค่าคงตัว (constants) นำมาใช้ในการประกาศขนาดของแถวลำดับ ค่าคงตัวเหล่านี้ควรมีค่าน้อย (small values) หลังจากโปรแกรมของเราไม่มีข้อผิดพลาดใดๆ จึงค่อยเปลี่ยนค่าคงตัวให้เป็นค่าปกติของมัน โปรดจำไว้ว่าเพื่อให้สามารถตรวจสอบ range จะใช้ตัวชี้แนะคอมไพเลอร์ {$R+}

ถ้าเราใช้ตัวแปรหรือนิพจน์เป็นดรรชนีล่างแถวลำดับ จงใช้ Watch window เพื่อสังเกตค่าดรรชนีล่าง ขณะกระทำการโปรแกรม เราสามารถใส่สมาชิกแถวลำดับ (ตัวแปรดรรชนีล่าง) หรือแถวลำดับทั้งหมดใน Watch window ค่าของสมาชิกแถวลำดับใน Watch window กำหนดโดยค่าดรรชนีล่างปัจจุบัน

ถ้า X เป็นแถวลำดับซึ่งแสดงในรูป 9.5 และ X[I] อยู่ใน Watch window ค่าที่แสดงผลสำหรับ X[I] คือ 16.0 เมื่อ I มีค่าเป็น 1 เมื่อ I มีค่าเป็น 2 ค่าที่แสดงผลของ X[I] คือ 12.0 ตัวชี้แนะคอมไพเลอร์ {$R+} ควรจะใช้เพื่อให้มั่นใจว่า X[I] คือการอ้างถึงแถวลำดับที่ถูกต้อง

ถ้าเราใส่ชื่อแถวลำดับใน Watch window แถวลำดับทั้งหมดจะถูกแสดงผลด้วยค่าสมาชิกแถวลำดับทั้งหมดแต่ละตัว คั่นด้วย comma และอยู่ภายในวงเล็บ ถ้า X เป็นแถวลำดับแสดงในรูป 9.5 และไอเดนติไฟเออร์ X ถูกใส่ใน Watch window หน้าต่างของ Watch window จะเหมือนรูป 9.24 ค่าแปลกๆ อาจจะแสดงสำหรับสมาชิกแถวลำดับซึ่งไม่ได้กำหนดค่าไว้

ถ้าแถวลำดับมีขนาดใหญ่มาก ไม่พอดีกับ Watch window ใช้ปุ่ม F6 เพื่อย้ายไปที่ Watch window จากนั้นใช้ปุ่ม left arrow หรือ
right arrow (หรือ mouse และ horizontal scroll bar ของ Watch window เพื่อกราดตรวจ (scan) ตลอดแถวลำดับ



รูป 9.24

อีกทางเลือกหนึ่งที่เราสามารถแสดงส่วนของแถวลำดับใน Watch window โดยการใส่ตัวแปรดรรชนีล่าง ตามด้วย comma และ repeat count เป็น นิพจน์ Watch ตัวอย่าง เช่น Watch expression

X[3], 4

ทำให้บรรทัดข้างล่างนี้ แสดงใน Watch window :

X[3], 4 : 6.0, 8.0, 2.5, 12.0

บรรทัดนี้แทนค่าของสมาชิกแถวลำดับสี่ตัว : X[3], X[4], X[5], และ X[6] ในทำนองเดียวกัน ตัวแปรดรรชนีล่างตามด้วย repeat count อาจจำมาใช้เพื่อแสดงส่วนของสายอักขระ

สำหรับลูป ซึ่งประมวลผลสมาชิกแถวลำดับแต่ละตัวเราอาจชอบใส่จุดพัก (breakpoint) ก่อนและหลังลูปมากกว่าตามรอย (trace) ข้อความสั่งแต่ละคำสั่งใน body ของลูป ซึ่งอ่านข้อมูล ไว้ในแถวลำดับ : มันมีสาระเล็กน้อยในการดูแต่ละค่าที่กำลังเก็บ และจุดพักจะทำให้เราเปรียบเทียบ contents ของแถวลำดับก่อนและหลังการกระทำการลูป

9.8 ข้อผิดพลาดร่วมของการเขียนโปรแกรม (Common Programming Errors)

ข้อผิดพลาดร่วมส่วนใหญ่ ในการใช้แถวลำดับ คือ นิพจน์ดรรชนีล่าง ซึ่งค่าของมันออกนอกพิสัยที่กำหนดไว้ระหว่างกระทำการโปรแกรม ถ้าเหตุการณ์เกิดขึ้น เมื่อไม่มีการตรวจสอบ range (default ใน Turbo Pascal) ตำแหน่งหน่วยเก็บภายนอกแถวลำดับหรือคำสั่งโปรแกรมอาจดัดแปรโดยที่เราไม่ทราบสำหรับเหตุผลนี้ มันจึงสำคัญมากที่เราต้องตรวจสอบพิสัย โดยใช้ตัวชี้แนะคอมไพเลอร์ {$R+} เมื่อทดสอบและแก้จุดบกพร่องโปรแกรม เราควรใส่ range checking enable ระหว่างวิ่งโปรแกรมปกติ เว้นแต่ว่าความเร็วการกระทำการโปรแกรมของเราเป็นเรื่องวิกฤติ

ข้อผิดพลาดของพิสัยดรรชนีล่าง (subscript range errors) เกิดขึ้นโดยนิพจน์ดรรชนีล่างไม่ถูกต้อง ถ้าดรรชนีล่างเป็นตัวแปรควบคุมลูปและตัวเพิ่มค่า ก่อนการวนซ้ำของแต่ละลูป ข้อผิดพลาดพิสัยดรรชนีล่าง อาจให้ผลลัพธ์จาก การวนลูปไม่รู้จบ (nonterminating loops) หรือ ลูปซึ่งกระทำการเพิ่มมากกว่าที่ต้องการหนึ่งครั้ง

ข้อผิดพลาดของพิสัยดรรชนีล่าง ส่วนใหญ่ คือ ค่าดรรชนีล่างที่ขอบเขตของลูป ถ้าค่าเหล่านี้อยู่ในพิสัย ดูเหมือนว่าการอ้างดรรชนีล่างอื่นๆ ทั้งหมดในลูปอยู่ในพิสัย

แบบชนิดข้อมูลทั้งหมดของ Pascal ต้องมั่นใจว่า ไม่มีชนิดที่ไม่ต้องกัน (no type inconsistencies) ชนิดของดรรชนีล่างและชนิดของสมาชิกซึ่งใช้ในการอ้างแถวลำดับทั้งหมดต้องสมนัยกับที่ระบุไว้ในการประกาศชนิดของแถวลำดับ

ในทำนองเดียวกัน แถวลำดับสองชุดซึ่งใช้ในข้อความสั่ง ทำสำเนาแถวลำดับหรือเป็นพารามิเตอร์ซึ่งสมนัยกันต้องเป็นชนิดเหมือนกัน โปรดจำไว้ว่า ให้ใช้เฉพาะไอเดนติไฟเออร์ ไม่มีดรรชนีล่าง เป็นพารามิเตอร์แถวลำดับรูปนัย (formal array parameters) และให้ระบุชนิดของพารามิเตอร์แถวลำดับทั้งหมด โดยใช้ไอเดนติไฟเออร์

ข้อสรุปของตัวสร้างใหม่ของ Pascal (Summary of New Pascal Constructs)

Construct	Effect
Array Declavation	
type	แบบชนิดข้อมูล IntArray
IndexRange = 1 . . 10;	อธิบายแถวลำดับที่มีสมาชิก
IntArray = array [IndexRange]	ชนิด Integer จำนวน 10 ตัว
of Interger;	Cube และ Count เป็น
var	แถวลำดับที่มีโครงสร้างนี้
Cube, Count : IntArray;	
Array References	
for I := 1 to 10 do	save I ยกกำลังสามใน
Cube [I] := I * I * I;	สมาชิกตัวที่ I ของแถวลำดับ Cube
If Cube [5] > 100 then	แสดงผล cube สองตัวแรก
Write (Cube [1], Cube [2])	ถ้า Cube [3] มีค่ามากกว่า 100
Array Copy	ทำสำเนา contents ของแถวลำดับ
Count := Cube	Cube ไปยังแถวลำดับ Count

แบบฝึกหัด Quick – Check

1. โครงสร้างข้อมูล คืออะไร

2. แบบชนิดข้อมูลมาตรฐาน ชนิดใด ไม่สามารถใช้เป็นชนิด

a) ดรรชนีล่างแถวลำดับ b) ชนิดสมาชิกของแถวลำดับ

3. ค่าของข้อมูลชนิดแตกต่างกันสามารถเก็บในแถวลำดับ ได้หรือไม่

4. ถ้าแถวลำดับถูกประกาศให้มีสมาชิก 10 ตัว โปรแกรมจำเป็นต้องใช้ทั้ง 10 ตัวหรือไม่

5. a) เมื่อใดจึงจะสามารถใช้ตัวกำหนดค่า (assignment operator) กับตัวถูกดำเนินการแถวลำดับได้

b) เมื่อใดจึงจะสามารถใช้ตัวดำเนินการเท่ากัน (equality operator) กับตัวดำเนินการแถวลำดับได้

6. สองวิธีของการเข้าถึงแถวลำดับคือ .....
และ.....

7. ลูปชนิดใด ซึ่งยอมให้เราเข้าถึงสมาชิกของแถวลำดับในลักษณะเรียงอันดับแบบ.....

8. แถวลำดับย่อยเติมเต็ม (filled subarray) คืออะไร ความยาวของมัน (its length) คืออะไร มีความสัมพันธ์อะไรระหว่าง สมาชิกตัวสุดท้ายในแถวลำดับย่อยเติมเต็มกับความยาวของมัน

9. จงอธิบายว่าทำไมพารามิเตอร์ตัวแปร จึงใช้หน่วยความจำอย่างมีประสิทธิภาพมากกว่า เมื่อส่ง (passing) แถวลำดับไปยังกระบวนงาน

10. จงประกาศตัวแปร Name และ Age ซึ่งใช้เก็บชื่อของพนักงาน (เป็นสายอักขระ) และอายุ (จำนวนเต็ม) ในแถวลำดับคนละชุด สมมติว่ามีชื่อมากที่สุด 50 ชื่อและอายุที่จำเป็นต้องเก็บ

คำถามทบทวน (Review Questions)

1. จงหาข้อผิดพลาด (error) ในส่วนรหัสข้างล่างนี้ และข้อผิดพลาดนี้จะตรวจพบเมื่อใด

```
program Test;
  type
```

```
    AnArray = array [1 . . 8] of Integer :
  var
    X :   AnArray;
    I : Integer;
  begin {Test}
    for I := 1 to 9 do
      X[I] := I
  end. {Test}
```

2. จงประกาศแถวลำดับของ reals ชื่อ Week ซึ่งสามารถอ้างถึง โดยใช้วันใดๆ ก็ได้ของสัปดาห์เป็นดรรชนีล่าง เมื่อ Sunday คือดรรชนีล่างตัวแรก

3. จงบอกว่าข้อความสั่งกำหนดค่าชุดใดในส่วนรหัส ข้างล่างนี้ ไม่ถูกต้อง สำหรับข้อความสั่งไม่ถูกต้องแต่ละชุด จงอธิบายว่าเป็นข้อผิดพลาดอะไร และจะตรวจพบเมื่อใด

```
program Errors;
  type
    AnArray = arrau [1. . 8] of Real;
  var
    X, Y : AnArray;
      I : Integer;
begin
  I := 1;
  X [I] := 7.329;
  X [1 . . 8] := 9.25;
  X [I] : = -23.5;
  X := y;
  X [1 . . 5] := y [1 . . 5]
end;
```

4. วิธีร่วมสองวิธีของการเลือกสมาชิกแถวลำดับสำหรับการประมวลผลคืออะไร

5. พารามิเตอร์จะแตกต่างกันอย่างไร สำหรับกระบวนงานสองชุด ซึ่งจัดดำเนินการ (manipulate) แถวลำดับ ถ้าชุดหนึ่งประมวลผลแถวลำดับย่อยและอีกชุดหนึ่งไม่ใช่

6. พารามิเตอร์สำหรับกระบวนงาน คือ แถวลำดับสองชุด (ชนิด RealArray) และจำนวนเต็มซึ่งแทนความยาวของแถวลำดับกระบวนงาน ทำสำเนาแถวลำดับชุดแรกในรายการพารามิเตอร์ ไปยังแถวลำดับอื่น ในอันดับย้อนกลับ (reverse order) จงเขียนกระบวนงาน

7. ข้อดีสามข้อของการใช้สายอักขระ คืออะไร

8. อะไรคือเหตุผลที่ถูกต้องสำหรับการไม่ส่งแถวลำดับซึ่งเป็นอินพุตไปยังกระบวนงาน เป็นพารามิเตอร์ค่า

9. มีการสับเปลี่ยนกี่ครั้งที่ต้องใช้เพื่อเรียงลำดับรายการจำนวนเต็ม ข้างล่างนี้ โดยใช้การเรียงลำดับแบบเลือก มีการเปรียบเทียบกี่ครั้ง

20 30 40 25 60 80

10. ประสิทธิภาพของการเรียงลำดับแบบเลือก ในสัญกรณ์ Big-0 คืออะไร ทำไม

การเขียนโปรแกรม (Programming Projects)

1. จงเขียนโปรแกรมสำหรับปัญหาต่อไปนี้ เราได้รับแฟ้มซึ่งประกอบด้วยกลุ่มของคะแนน (ชนิด Integer) สำหรับการสอบไล่วิชาคอมพิวเตอร์ ต้องการให้คำนวณค่าเฉลี่ยของคะแนนเหล่านี้ และกำหนดเกรดให้กับนักศึกษาแต่ละคน ซึ่งเป็นไปตามกฎดังนี้

ถ้าคะแนนสอบของนักศึกษาอยู่ภายใน 10 คะแนน (สูงหรือต่ำ) ของคะแนนเฉลี่ย เกรดคือ S (Satisfactory) ถ้าคะแนนสอบของนักศึกษามากกว่าค่าเฉลี่ยตั้งแต่ 10 คะแนนขึ้นไป กำหนดเกรดเป็น O (Outstanding) ถ้าคะแนนสอบของนักศึกษาน้อยกว่าค่าเฉลี่ยตั้งแต่ 10 คะแนนขึ้นไป กำหนดเกรดเป็น U (Unsatisfactory)

เอาต์พุตของโปรแกรมเป็นรายการสองสดมภ์ซึ่งมีเลเบลกำกับแสดงคะแนนและเกรดที่สมนัยกัน **ข้อแนะนำ** โปรแกรมของเราควรมีฟังก์ชันและกระบวนงานซึ่งสมนัยกับหัวเรื่องของฟังก์ชันและกระบวนงานต่อไปนี้

```
procedure ReadStuData (var RawScores {input} : Text;
                       var Score {output}    : ArrayType;
                       var Count {output}    : Integer;
                       var TooMany {output}  : Boolean);
{
   Reads exam scores from file RawScores into array Scores. Count
   contains number of students read.
```

```
    TooMany is set to True if RawScores contains more than Classize
    scores.
}

procedure PrintGrade (OneScore {input}: Integer;
                      Average {input} : Real);
{Prints student grade after comparing OneScore to Average}

function Mean (Score : ArrayType; Count : Interger) : Real;
{Computer average of Count student scores}

procedure PrintTable (Score {input} : ArrayType;
                      Count {input} : Integer);
{
    Print a table showing each student's score and grade on a separate
    line.
    Calls : PrintGrade.
}
```

2. เขียนโปรแกรมข้อ 1 ใหม่ (Redo Programming Project1) สมมติว่าแต่ละบรรทัดของแฟ้ม RawScores ประกอบด้วยรหัสนักศึกษา (เป็น Integer) และคะแนนสอบ ให้จัดสรรแถวลำดับสามชุดสำหรับเก็บรหัสนักศึกษา, คะแนนสอบ และเกรด

จงดัดแปรกระบวนงาน ReadStuData ให้อ่านรหัสนักศึกษา และคะแนนจากบรรทัดที่ I ไว้ในสมาชิกแถวลำดับ ID[I] และ Score[I] ตามลำดับ จงเขียนกระบวนงานชุดใหม่ชื่อ AssignGrades ซึ่งกำหนดค่าให้กับแถวลำดับ Grades โดยขึ้นอยู่กับคะแนนสอบ จงดัดแปรกระบวนงาน PrintTabel ให้แสดงผลเป็นตารางที่มีสามสดมภ์ มีหัวเรื่องดังนี้

ID Score Grade

3. จงเขียนโปรแกรมอ่านหน่วยข้อมูล N ตัว ไว้ในแถวลำดับสองชุด ชื่อ X และ Y ขนาด 20 ตัว จากนั้นเก็บผลคูณของสมาชิกของ X และ Y ที่สมนัยกันในแถวลำดับชุดที่สาม

ชื่อ Z ขนาด 20 ตัวเช่นกัน พิมพ์ตารางที่มีสามสดมภ์ แสดงแถวลำดับ X, Y และ Z จากนั้นคำนวณและพิมพ์รากที่สองของผลบวกของหน่วยข้อมูลในแถวลำดับ Z ให้นักศึกษาสมมติข้อมูลเอง โดยให้ N มีค่าน้อยกว่า 20 ตัว

4. ผลลัพธ์ของการสอบข้อสอบชนิดตอบถูกหรือผิด (a true-false exam) ของวิชาคอมพิวเตอร์ ได้มีการลงรหัสเป็นอินพุตของโปรแกรม สารสนเทศของนักศึกษาแต่ละคนประกอบด้วยรหัสประจำตัวของนักศึกษา และคำตอบของนักศึกษากับคำถามถูก-ผิดจำนวน 10 ข้อ ข้อมูลจึงเป็นดังนี้

Student Identification	Answer String
0080	FTTFTFTTFT
0340	FTFTFTTTFF
0341	FTTFTTTTTT
0401	TTFFTFFTTT
0462	TTFTTTFFTF
0463	TTTTTTTTTT
0464	FTFFTFFTFT
0512	TFTFTFTFTF
0618	TTTFFTTFTF
0619	FFFFFFFFFF
0687	TFTTFTTFTF
0700	FTFFTTFFFT
0712	FTFTFTFTFT
0837	TFTFTTFTFT

จงเขียนโปรแกรม ขั้นแรกอ่าน สายอักขระคำตอบแทนคำตอบที่ถูกต้อง 10 ข้อ (ใช้ FTFFTFFTFT เป็นข้อมูล) ขั้นต่อไปอ่านข้อมูลของนักศึกษาแต่ละคนไว้ในแถวลำดับสองชุด จากนั้นคำนวณและเก็บจำนวนคำตอบถูกของนักศึกษาแต่ละคนในแถวลำดับอีกชุดหนึ่งที่สมาชิกสมนัยกัน ตรวจหาคะแนนดีที่สุด ให้ชื่อว่า Best จากนั้นพิมพ์ตารางที่มีสามสดมภ์แสดงรหัสประจำตัวนักศึกษา คะแนนสอบ และเกรดของนักศึกษาแต่ละคน สำหรับการคิดเกรดกำหนดเกณฑ์ไว้ดังนี้

เกรด A ได้คะแนนสอบเท่ากับ Best หรือ Best-1

เกรด C ได้คะแนนสอบเท่ากับ Best-2 หรือ Best-3

กรณีอื่นๆ ได้เกรด F

5. จำนวนเฉพาะ (prime number) หมายถึงเลขจำนวนเต็มบวกใดๆ ก็ตามซึ่งมีค่ามากกว่า 1 และมีเฉพาะเลข 1 กับตัวมันเองเท่านั้นที่หารมันลงตัว จงเขียนโปรแกรมคำนวนหาจำนวนเฉพาะทั้งหมดที่มีค่าน้อย 2000 วิธีหนึ่งในการก่อกำเนิด (gemerate) จำนวนเฉพาะคือสร้างแถวลำดับของค่าแบบบูลีน (Boolean values) ซึ่งเป็นจริง (true) สำหรับจำนวนเฉพาะทั้งหมด ส่วนกรณีอื่นๆ เป็นเท็จ เริ่มต้นตั้งให้สมาชิกทั้งหมดของแถวลำดับมีค่าเป็นจริง จากนั้นสำหรับเลขทุกตัวจาก 2 ถึง 2000 ตั้งดรรชนีตำแหน่งของแถวลำดับ (array location indexed) ด้วยเลขที่มีตัวประกอบ (แต่ไม่ใช่เลขตัวมันเอง) ให้เป็นเท็จ (false) เมื่อทำเสร็จแล้ว เอาต์พุตเลขทั้งหมดซึ่งตำแหน่งแถวลำดับของมันเป็นจริง เลขเหล่านี้จะเป็นจำนวนเฉพาะ