บทที่ 8
แฟ้มข้อความ
(Text Files)

โปรแกรมตัวอย่างทั้งหมดที่ได้นำเสนอมาแล้วเป็นแบบเชิงโต้ตอบ

โปรแกรมเชิงโต้ตอบ หมายถึง โปรแกรมซึ่งอ่านข้อมูลทั้งหมดจากคีย์บอร์ด และแสดงผลเอาต์พุตทั้งหมดบนจอภาพ (An interactive program is a program that reads all data from the keyboard and displays all output on the screen.)

อินพุตและเอาต์พุตเชิงโต้ตอบดีสำหรับโปรแกรมซึ่งประมวลผลข้อมูลปริมาณน้อย แต่กระบวนงานนี้ ประสิทธิผลจะน้อยลง สำหรับโปรแกรมซึ่งต้องประมวลผลข้อมูลปริมาณมาก การปรับปรุง เพื่อให้มีประสิทธิภาพดีขึ้น เราสามารถใช้ แฟ้มข้อมูล สำหรับเก็บอินพุตและเอาต์พุตของโปรแกรม Pascal สามารถประมวลผลแฟ้มข้อมูล สองชนิด ได้แก่ แฟ้มข้อความ และแฟ้มฐานสอง (text files and binary files) ในบทนี้อธิบายแฟ้มข้อความ

กระบวนงานอินพุตและเอาต์พุตที่เราใช้สำหรับโปรแกรมเชิงโต้ตอบทำงานได้กับแฟ้มข้อความ ในบทนี้เราจะอธิบายกระบวนงานใหม่หนึ่งชุด คือ Read ซึ่งเป็นประโยชน์โดยเฉพาะกับการอ่านข้อมูลจากแฟ้มข้อความ นอกเหนือจากนี้อธิบายกระบวนงานใหม่และฟังก์ชันใหม่หลายชุด สำหรับแฟ้มข้อความ

8.1 แฟ้มข้อความ แฟ้มข้อมูล และแฟ้มเอาต์พุต (Text Files, Data Files, and Output Files)

แฟ้มข้อความ หมายถึงกลุ่มของอักขระแต่ละตัวเก็บภายใต้ชื่อเดียวกันบนดิสก์ เราสามารถเก็บ (save) ข้อมูลทั้งหมด ซึ่งจะต้องประมวลผล โดยโปรแกรมในแฟ้มข้อความก่อนวิ่งโปรแกรมนั้น หลังจากนั้นเราสั่งโปรแกรมให้อ่านข้อมูล ของมันจากแฟ้มข้อความไม่ใช่อ่านข้อมูลจากคีย์บอร์ด

แฟ้มข้อความ หมายถึง แฟ้มดิสก์ ซึ่งเก็บกลุ่มของตัวอักขระ (A text file is a disk file that contains a collection of characters.)

แฟ้มข้อมูล (หรือ**แฟ้มอินพุต**) หมายถึง แฟ้มซึ่งประกอบด้วยข้อมูลอินพุต สำหรับโปรแกรม (A data file (or input file) is a file that contains the input data for a program.)

แฟ้มข้อมูล อาจจะเป็นแฟ้มข้อความ หรือแฟ้มฐานสอง ข้อดีประการหนึ่งของแฟ้มข้อมูล คือ เราสามารถเข้าถึง (access) ได้โดยใช้โปรแกรมรรรณาธิการ (หรือโปรแกรมประมวลผลคำ) (editor program (or word procesor) และแก้ไขข้อผิดพลาดได้ ก่อนแฟ้มข้อมูลนั้น จะถูกประมวลผลโดยโปรแกรม ข้อดีอีกหนึ่งข้อ คือ แฟ้มข้อมูล สามารถอ่านแล้วอ่านซ้ำได้อีกโดยโปรแกรม คุณสมบัตินี้ทำให้สะดวกในการแก้จุดบกพร่อง เพราะว่าโปรแกรมของเราสามารถอ่านข้อมูลของมันจากแฟ้มข้อมูลเดียวกันทุกครั้งที่วิ่งโปรแกรมในโปรแกรมเชิงโต้ตอบ เราต้องใส่หน่วยข้อมูล (data item) แต่ละตัวใหม่ทุกครั้งระหว่างการวิ่งโปรแกรมแต่ละครั้ง

เราสามารถสั่งโปรแกรมให้เขียน (write) ผลลัพธ์ของมันในแฟ้มข้อความแทนที่จะเป็นบนจอภาพได้ด้วย ทำให้เราได้เวอร์ชันถาวรของผลลัพธ์โปรแกรมบนดิสก์ แฟ้มดิสก์นั้นสามารถพิมพ์ (printed) หรือใช้เป็นแฟ้มข้อมูล สำหรับโปรแกรมที่สอง ซึ่งจะประมวลผลต่อไป

แฟ้มเอาต์พุต หมายถึง แฟ้มซึ่งประกอบด้วยผลลัพธ์ของโปรแกรม (An output file is a file that contains a program's results.)

การจัดระเบียบสารสนเทศในแฟ้มข้อความ (Organization of Information in a Text file)

เราสามารถสร้าง (create) และเก็บ (save) แฟ้มข้อความโดยใช้ editor อักขระแต่ละตัวซึ่งเราพิมพ์ (type) อยู่ชั่วคราวในหน่วยความจำหลักและแสดงผลบนจอภาพ เมื่อเรา

save แฟ้มสารสนเทศ ซึ่งแสดงบนจอภาพถูกเก็บ (store) เป็นกระแสของอักขระ (a stream of characters)

แฟ้มข้อความ ในรูป 8.1 ประกอบด้วยอักขระทั้งหมด 45 ตัว (หรือไบต์) ได้แก่ อักขระเลขโดด จุด อักขระว่าง ตัวอักษร และอักขระพิเศษสองตัว คือ <eoln> และ <eof>

<eoln> อักขระ end–of–line เก็บในแฟ้มข้อความทุกครั้งที่เรากดปุ่ม Enter บนคีย์บอร์ด

<eof> อักขระ end–of–file ใส่อัตโนมัติหลังอักขระตัวสุดท้ายของแฟ้มข้อความเมื่อเรา save แฟ้มข้อความ

แฟ้มข้อความหนึ่งแฟ้มมีอักขระ <eoln> หลายตัวได้แต่อักขระ <eof> มีได้หนึ่งตัวเท่านั้น

159000.00 Johnny Jones <eoln> 25000.00 Sally Smythe <eoln> <eof>

รูป 8.1 แฟ้มข้อความ คือ กระแสของตัวอักขระ

ถึงแม้ว่าแฟ้มข้อความเก็บเชิงกายภาพ (physically stored) บนดิสก์เป็นหนึ่งกระแสความยาวของตัวอักขระ มันถูกจัดระเบียบเชิงตรรกะ (logically organized) เป็นลำดับของตัวอักขระแยกจากกันเป็นบรรทัดด้วย อักขระ <eoln> ถ้าเราใช้ editor ของ Turbo Pascal เพื่อเข้าถึงแฟ้มข้อความในรูป 8.1 มันจะดูคล้ายสารสนเทศในรูป 8.2 ยกเว้นเราจะมองไม่เห็นอักขระ <eoln> และ <eol>

เราสามารถดูแฟ้มข้อความโดยใช้ editor ของ Turbo Pascal หรือเราสามารถแสดงผล (display) แฟ้มข้อความบนจอภาพ โดยใช้คำสั่งงานของ MS-DOS ดังนี้

15900.00 Johnny Jones <eoln>
25000.00 Sally Smythe <eoln> <eof>

รูป 8.2 การจัดระเบียบเชิงตรรกะของแฟ้มข้อความในรูป 8.1

```
> TYPE filename
```

เมื่อ filename เป็นชื่อของแฟ้มข้อความ แต่ละบรรทัดของแฟ้มจะแสดงผลบนบรรทัดแยกจากกันของจอภาพ

คำสั่งงาน MS – DOS

```
> PRINT Filename
```

พิมพ์แฟ้ม Filename

การอ่านข้อมูลจากแฟ้มข้อความ (Reading Data From a Text File)

เราสามารถใช้กระบวนงาน ReadLn ของ Pascal เพื่ออ่านข้อมูลจากแฟ้มข้อความ จงพิจารณาโปรแกรมซึ่งประกอบด้วยการประกาศ ตัวแปร ข้างล่างนี้

```
var
   Salary : Real;          {input – a salary}
   Name : String ;         {input – a name}
   Separator : Char ;      {input – blank between Salary and Name}
   MyData :   Text ;       {a text file used as a data file}
```

บรรทัดสุดท้ายประกาศ MyData เป็นตัวแปรชนิด Text ซึ่งเป็นแบบชนิดข้อมูลที่ให้นิยามมาแล้ว (predefined data type) สำหรับแฟ้มข้อความ

ข้อความสั่ง

```
ReadLn (MyData, Salary, Separator, Name)
```

มีพารามิเตอร์สี่ตัว ได้แก่ ตัวแปรแฟ้ม Text, ตัวแปร Real, ตัวแปร Char และตัวแปร String คำสั่งนี้อ่านหน่วยข้อมูล (data items) สามตัวจากแฟ้มชื่อ My Data เก็บหน่วยข้อมูลตัวแรกใน Salary ตัวที่สองใน Separator และตัวที่สามใน Name ถ้า MyData แทนแฟ้มข้อความในรูป 8.2 และกำลังอ่านบรรทัดแรก ข้อความสั่งนี้เก็บ 15900.00 ใน Salary อักขระว่างใน Separator และสายอักขระ 'Johnny Jones' ใน Name

Pascal รู้ได้อย่างไรว่าเป็นการอ่านข้อมูลจากแฟ้มข้อความแทนที่จะเป็นคีย์บอร์ด สิ่งนี้กำหนดโดยพารามิเตอร์ตัวแรกในการเรียก ReadLn ถ้าพารามิเตอร์ตัวแรกเป็นตัวแปร

ชนิด Text, Turbo Pascal อ่านข้อมูลจากแฟ้มที่แสดงด้วยตัวแปรนั้น กรณีอื่นๆ Turbo Pascal อ่านข้อมูลจากคีย์บอร์ด

ReadLn ประมวลผลแต่ละบรรทัดของแฟ้มข้อความคล้ายกับ บรรทัดข้อมูลที่เข้าทางคีย์บอร์ด บรรทัดแรก ของแฟ้มระบุด้วย ตัวแปร MyData เริ่มต้นด้วยลำดับของอักขระเลขโดด ที่มีจุดทศนิยม เมื่อ ReadLn กำลังอ่านข้อมูลไว้ในตัวแปรชนิด Real มันข้าม (skip) ตัวว่างนำหน้า และอ่านอักขระเลขโดดทั้งหมดจนถึงตัวว่างถัดไป หรือ <eoln> ด้วยเหตุนี้ มันเก็บเลขจำนวนจริง 15900.00 ใน Salary อักขระถัดไปที่ถูกอ่านคือ อักขระว่างระหว่างเลขโดด 0 และตัวอักษร J ตัวว่างนี้จึงเก็บใน Separator

ใน Turbo Pascal (ไม่ใช่ Standard Pascal) เมื่อ ReadLn กำลังอ่านข้อมูลไว้ในตัวแปรชนิด String มันอ่านอักขระทั้งหมดตลอดทั้งอักขระ <eoln> ถัดไป และเก็บทั้งหมดแต่ไม่รวมอักขระ <eoln> ในตัวแปร String เพราะฉะนั้น ReadLn เก็บตัวอักขระจาก 1 จนถึง 5 (ได้แก่สายอักขระ "Johnny Jones") ใน Name

อักขระบางตัวบนบรรทัดข้อมูล อาจถูกประมวลผลโดย ReadLn แต่ไม่ถูกเก็บ ข้อความสั่ง

ReadLn (MyData, Salary)

อ่านและเก็บเลขถัดไปในหนึ่งบรรทัดของแฟ้ม MyData ไว้ใน Salary จากนั้นมันข้าม (Skips) อักขระส่วนที่เหลือบนบรรทัดข้อมูล (จนถึง <eoln> ถัดไป) สิ่งนี้ไม่ถือเป็นข้อผิดพลาดการดำเนินการ ReadLn ถัดไปจะเริ่มต้นอ่านข้อมูล จากบรรทัดข้อมูลถัดไปของแฟ้ม MyData

กระบวนงาน Read (Procedure Read)

กระบวนงานอินพุต ตัวที่สองคือ Read ซึ่งอ่านข้อมูลกระบวนงานจากคีย์บอร์ด หรือจากแฟ้ม แต่ไม่เหมือนกับ ReadLn เพราะว่า Read ประมวลผลเฉพาะตัวอักขระข้อมูลมากเท่าที่จำเป็น เพื่อเป็นไปตามรายการอินพุตของมันเท่านั้น

ตัวอย่าง 8.1

ข้อความสั่ง

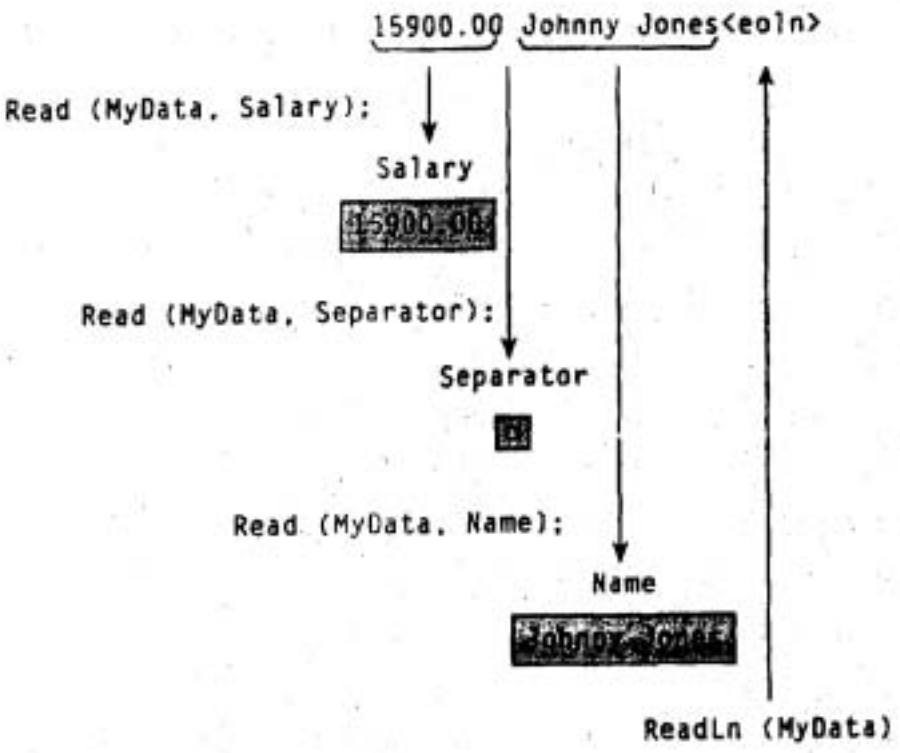
```
.Read (MyData, Salary) ;        {Read Salary From MyData}
Read (MyData, Separator) ;      {Read the blank character}
Read (MyData, Name) ;           {Read Name From MyData}
ReadLn (MyData)                 {Skip past <eoln> character in MyData}
```

ซึ่งมีผลเหมือนกับ

ReadLn (MyData, Salary, Separator, Name)

ข้อความสั่ง Read บรรทัดแรกอ่านอักขระซึ่งประกอบขึ้น เป็นเลขจำนวนจริงไว้ใน Salary หยุดที่อักขระว่างข้อความสั่ง Read บรรทัดที่สองอ่านอักขระว่างไว้ใน Separator ข้อความสั่ง Read บรรทัดที่สาม อ่านอักขระที่เหลือทั้งหมด จนถึงอักขระ <eoln> ถัดไปไว้ใน Name

ข้อความสั่ง ReadLn ข้ามอักขระ <eoln> ไปเริ่มต้นบรรทัดข้อมูลถัดไป รูป 8.3 แสดงให้เห็นผลของการกระทำการข้อความสั่งเหล่านี้ เมื่อบรรทัดที่หนึ่งของแฟ้ม MyData กำลังถูกอ่าน อักขระว่างแสดงด้วยสัญลักษณ์ □ ใน Separator



รูป 8.3 ผลของการอ่านบรรทัดแรกของแฟ้ม MyData

ทบทวน Read และ ReadLn (Review of Read and ReadLn)

จำนวนของตัวอักขระ ซึ่งอ่านโดย Read หรือ ReadLn ขึ้นอยู่กับชนิดของตัวแปรที่รับข้มูล เมื่ออ่านข้อมูล ไว้ในตัวแปรชนิด Char นั้น อ่านหนึ่งตัวอักษร เมื่ออ่านข้อมูลไว้ในตัวแปรตัวเลข โปรแกรมข้ามตัวว่างนำหน้า (leading blanks) อักขระควบคุม (Control Characters) หรืออักขระ <eoln> จนกระทั่งมันพบอักขระซึ่งไม่ใช่ตัวว่าง อักขระควบคุม หรือ <eoln> อักขระตัวแรกต้องเป็นเครื่องหมาย หรือเลขโดด ถ้าไม่ใช่จะเกิดข้อผิดพลาดการกระทำการเป็น Non-digit found while reading number และโปรแกรมจะหยุด ถ้าอักขระตัวแรก คือ เครื่องหมายหรือเลขโดด โปรแกรมจะอ่านอักขระเลขโดดอย่างต่อเนื่อง (รวมทั้งจุดทศนิยมสำหรับข้อมูลชนิด Real) จนกระทั่งมันพบตัวว่าง อักขระควบคุม หรือ <eoln> เมื่ออ่านข้อมูลไว้ในตัวแปร String ตัวอักขระทั้งหมดจนถึงแต่ไม่รวมอักขระ <eoln> ถัดไปถูกอ่านและเก็บ (Stored)

สำหรับกระบวนงาน ReadLn อย่างเดียว ตัวอักขระ <eoln> ถูกอ่านด้วยแต่ไม่ถูกเก็บ

Syntax Display

กระบวนงาน Read และ ReadLn

Form :
```
Read (infite, input list)
ReadLn (infile, input list)
```

ตัวอย่าง : Read (Indata, X, Y, Z)

ReadLn (Indata, Ch 1, Ch 2)

มีความหมายดังนี้ : ลำดับของอักขระถูกอ่านจากแฟ้ม Indata (ชนิด Text) ไว้ในตัวแปรซึ่งระบุใน input list ชนิดของตัวแปรแต่ละตัวใน input list ต้องเป็น Boolean, Char, Integer, Real, String หรือ subrange ของ Char หรือ Integer ถ้าแบบชนิดข้อมูลของตัวแปรเป็น Char เฉพาะอักขระหนึ่งตัวเท่านั้นถูกอ่านไว้ในตัวแปรนั้น

ถ้าแบบชนิดข้อมูลของตัวแปร เป็น Interger หรือ Real, อักขระว่างนำหน้าใดๆ หรืออักขระ <eoln> จะถูกข้ามและลำดับของอักขระเลข (รวมทั้งจุดทศนิยมสำหรับ Real) ถูกอ่านและเก็บในตัวแปรนั้น

ถ้าแบบชนิดข้อมูลเป็น String อักขระทั้งหมดจนถึง <eoln> ถัดไปถูกอ่านหลังจาก input list สำหรับ Read ครบแล้ว อักขระถัดไปที่จะถูกอ่าน คือ ตัวซึ่งอยู่ตามหลังอักขระตัวท้ายสุด ซึ่งอ่านไปแล้ว

ถ้ากระบวนงาน ReadLn ถูกเรียก อักขระตัวถัดไปที่จะถูกอ่าน คือตัวซึ่งตามหลังอักขระ <eoln> ถัดไป ถ้าไม่มี infile หน่วยข้อมูลจะถูกอ่านจากคีย์บอร์ด

ตาราง 8.1 แสดงรายการตัวอย่างข้อความสั่ง Read และ ReadLn หลายชุดและผลลัพธ์ สมมติว่า x เป็นชนิด Real, N เป็นชนิด Integer, C เป็นชนิด Char, S เป็นชนิด String และอักขระตัวแรกซึ่งจะถูกประมวลผล คือ อักขระตัวแรกในแฟ้มซึ่งแสดงทางขวามือ อักขระตัวถัดไปซึ่งจะถูกอ่าน หลังจากการดำเนินการ Read หรือ ReadLn แต่ละชุด แสดงด้วยตัวเน้น

ตาราง 8.1 ผลของ Read และ ReadLn

Statement and Effect	Next Character to Be Read
Read (Indata, X, N, C) X คือ 1234.56 N คือ 789 C คือ ' '	1234.56 ☐ 789 ☐ **A**345.67<eoln> W<eoln> <eof>
ReadLn (Indata, X, N, C) X คือ 1234.56 N คือ 789 C คือ ' '	1234.56 ☐ 789 ☐ A345.67 <eoln> **W**<eoln> <eof>
Read (Indata, X, C, N) X คือ 1234.56 C คือ ' ' N คือ 789	1234.56 ☐ 789 ☐ A345.67 <eoln> W<eoln> <eof>

Statement and Effect	Next Character to Be Read
ReadLn (Indata, X, C, N) X คือ 1234.56 C คือ ' ' N คือ 789	1234.56 □ 789 □ A345.67 <eoln> **W**<eoln> <eof>
Read (InData, C, X, N) C คือ '1' X คือ 234.56 N คือ 789	1234.56 □ 789 □ A345.67 <eoln> W<eoln> <eof>
ReadLn (Indata, C, X, N) C คือ '1' X คือ 234.56 N คือ 789	1234.56 □ 789 □ A345.67 <eoln> **W**<eoln> <eof>
ReadLn (InData, X, N) ; Read (InData, C) X คือ 1234.56 N คือ 789 C คือ 'W'	1234.56 □ 789 □ A345.67 <eoln> W **<eoln>** <eof>
ReadLn (InData, X) ; ReadLn (InData, C) ; X คือ 1234.56 C คือ 'W'	1234.56 □ 789 □ A345.67 <eoln> W <eoln> **<eof>**
ReadLn (InData, S) ; Read (InData, C) S คือ '1234.56 789 A 345.67' C คือ 'W'	1234.56 □ 789 □ A 345.67 <eoln> W **<eoln>** <eof>

โปรดสังเกตว่า ในตาราง 8.1 การดำเนินการสี่ชุดแรก ตัวแปร C ประกอบด้วยอักขระว่าง ในทุกกรณีนี้ C ตามหลัง X หรือ N ในรายการอินพุต เพราะฉะนั้น ลำดับของอักขระเลขจนถึงอักขระว่าง (แต่ไม่รวมอักขระว่าง) ถูกอ่านไว้ใน X หรือ N และอักขระว่างถูกอ่านไว้ใน C

การเขียนผลลัพธ์ไว้ในแฟ้มข้อความ (Writing Results to a Text File)

ใช้กระบวนงาน Write และ WriteLn เพื่อเขียนผลลัพธ์ไว้ในแฟ้มข้อความในวิธีเดียวกับที่เราเขียนไว้บนจอภาพ (screen) ถ้าพารามิเตอร์ ตัวแรกในการเรียก Write หรือ WriteLn เป็นตัวแปรชนิด Text, อักขระเอาต์พุตจะส่งไปยังแฟ้มซึ่งระบุโดยชื่อตัวแปรนั้น ไม่ใช่ส่งไปจอภาพ เมื่อใดก็ตามที่ใช้ WriteLn อักขระ <eolen> จะใส่ไว้ในแฟ้มหลังอักขระเอาต์พุตตัวสุดท้าย

โปรแกรมเขียนค่าตัวเลขไว้ในแฟ้มเป็นลำดับของอักขระเลขโดด อักขระลบ (-) อยู่หน้าเลขลบ (negative number) อักขระจุดทศนิยม (.) ใส่ในค่าชนิด Real เราสามารถใช้ข้อกำหนดรูปแบบ (fomat specifications) เมื่อเขียนไว้ในแฟ้มข้อความ (ดูหัวข้อ 2.8)

Syntax Display

กระบวนงาน Write และ WriteLn

Form :

```
Write (outfile, output list)
WriteLn (outfile, output list)
```

ตัวอย่าง : Write (MyResult, Salary)

WriteLn (MyResult, Hours : 3 :1, '$', Salary : 4 : 2)

มีความหมายดังนี้ : อักขระที่กำหนดโดย output list ถูกเขียนไว้ตอนท้ายของแฟ้ม outfile ชนิดของนิพจน์แต่ละตัวใน output list ต้องเป็น Boolean, Char, Integer, Real, String หรือ subrange ของ Char หรือ Integer

ถ้านิพจน์เป็นชนิด Char, อักขระหนึ่งตัวถูกเขียนไปไว้ที่แฟ้ม outfile กรณีอื่นๆ อักขระหลายตัวอาจถูกเขียน ข้อกำหนดรูปแบบอาจรวมอยู่ใน Output list เมื่อ WriteLn ถูกเรียกอักขระ <eoln> จะเขียนไว้ใน outfile หลังอักขระตัวสุดท้ายที่ระบุโดย output list ถ้าไม่มี outfile, ผลลัพธ์เอาต์พุตถูกเขียนไว้บนจอภาพ

คีย์บอร์ดและจอภาพเป็นแฟ้มข้อความ (The Keyboard and Screen as Text Files)

ในโปรแกรมเชิงโต้ตอบ Pascal ถือว่าข้อมูลส่งเขาที่คีย์บอร์ดราวกับว่ามันถูกอ่านจากแฟ้มระบบ Input ข้อมูลเหล่านี้เก็บชั่วคราวในบัฟเฟอร์อินพุต (input buffer) ซึ่งเป็นเนื้อที่หน่วยเก็บจนกระทั่งเรากดปุ่ม Enter จากจุดนี้เราสามารถตรวจแก้ (edit) ข้อมูลในบัฟเฟอร์อินพุต โดยใช้ปุ่ม backarrow เพื่อลบอักขระและพิมพ์ใหม่ การกดปุ่ม Enter คือ ใส่อักขระ <eoln> ในบัฟเฟอร์อินพุตและทำให้โปรแกรมของเรา เริ่มต้นประมวลผล บรรทัดข้อมูล

ในทำนองเดียวกัน การแสดงผลอักขระบนจอภาพสมมูลกับการเขียนอักขระไว้ที่แฟ้มระบบ Output กระบวนงาน WriteLn ใส่อักขระ <eoln> ในแฟ้มนี้ ด้วยเหตุนี้ คือ การย้ายตัวชี้ตำแหน่ง (cursor) ไปยังตอนเริ่มต้นของบรรทัดถัดไปของจอภาพ

Input และ Output ทั้งคู่เป็นแฟ้มข้อความ เพราะว่าแต่ละส่วนประกอบของมันเป็นตัวอักขระ

บัฟเฟอร์อินพุต หมายถึงพื้นที่หน่วยเก็บชั่วคราวสำหรับข้อมูลอินพุต (Input buffer is a temporary storage area for input data.)

ตัวอย่าง 8.2

ข้อความสั่ง

```
Write (Output, 'Enter a salary value> ') ;
ReadLn (Input, Salary)
```

แสดงข้อความพร้อมรับบนจอภาพ (แฟ้ม Output) และอ่านหนึ่งค่า พิมพ์ที่คีย์บอร์ด (แฟ้ม Input) ไว้ใน Salary ทั้งสองคำสั่งนี้มีความหมายเหมือนกับ

```
Write ('Enter a salary value > ') ;
ReadLn (Salary)
```

แบบฝึกหัด 8.1

1. a) สำหรับโปรแกรมคอมพิวเตอร์ ซึ่งจัดกระทำ (handles) การจองการสำรองที่นั่งของสายการบิน (booking of airline reservations) เราควรใช้แฟ้มข้อมูล หรือใส่ข้อมูลเชิงโต้ตอบแบบไหนดีกว่ากัน ให้อธิบาย

b) สำหรับโปรแกรมพิมพ์ใบแสดงผลการศึกษารายวิชา (transcripts) ของนักศึกษาในมหาวิทยาลัย ควรใช้แบบแฟ้มข้อมูล หรือข้อมูลเชิงโต้ตอบ ให้อธิบาย

2. จงอธิบายว่าเมื่อใดใช้ Read ดีกว่า ReadLn ในความหมายของการรับข้อมูลตัวเลขจากผู้ใช้ของโปรแกรมเชิงโต้ตอบ

3. ถ้าแฟ้มข้อความเก็บเชิงกายภาพบนดิสก์ เป็นกระแสขนาดยาวของอักขระ มันจะปรากฏเป็นหลายบรรทัดบนจอภาพของโปรแกรม text editor อย่างไร

4. ให้ x เป็นตัวแปรชนิด Real, N เป็นชนิด Integer, C ชนิด Char และ S เป็นชนิด String จงบอก contents ของตัวแปรแต่ละตัว หลังจากการดำเนินการ Read หรือ ReadLn แต่ละชุด สมมติว่าแฟ้มประกอบด้วย บรรทัดต่อไปนี้

123 3.145 XYZ <eoln>

35 Z <eoln>

a) ReadLn (Indata, N, X) ;
   Read (Indata, C)

b) Read (Indata, N, X, C)

c) Read (Indata, N, X, C, C)

d) ReadLn (Indata, N) ;
   Read (Indata, C)

e) ReadLn (Indata, X) ;
   Read (Indata, C, N)

f) Read (Indata, C, C, C, X) ;
   Read (Indata, N)

g) Read (In ata, N, X, C, C, C, C, N)

h) Read (InData, N, X, C, C, C, C) ;
   Read (InData, N, X) ; Read (InData, C)

i) Read (InData, N, S) ;
   Read (InData, C)

j) Read (InData, N, S) ;
   ReadLn (InData) ;
   Read (InData, C)

5. ทำซ้ำแบบฝึกหัดข้อ 4 โดยการแทน WriteLn สำหรับ ReadLn Write สำหรับ Read และ OutData สำหรับ InData จงแสดงแฟ้ม OutData หลังจากแต่ละส่วน สมมติว่า N เท่ากับ 512, X เท่ากับ 0.123 และ C เท่ากับ '*'

8.2 กระบวนงานและฟังก์ชันสำหรับแฟ้มข้อความ (Procedures and Functions for Text Files)

เราได้แสดงให้เห็นแล้วว่าประกาศแฟ้มข้อความอย่างไรอ่านจากแฟ้มข้อความใช้เป็นแฟ้มข้อมูลอย่างไร และเขียนผลลัพธ์เอาต์พุต ไปไว้ในแฟ้มข้อความใช้เป็นแฟ้มเอาต์พุตอย่างไรในหัวข้อนี้ จะอธิบายกระบวนงานและฟังก์ชันของ Turbo Pascal เพิ่มเติมซึ่งจำเป็นต้องใช้แฟ้มข้อความในโปรแกรม Pascal

การใช้แฟ้มข้อความ เราต้องกระทำการดำเนินการดังต่อไปนี้

1. ประกาศตัวแปรซึ่งจะใช้เป็นแฟ้มข้อความให้เป็นชนิด Text

2. จัดหาชื่อสารบบ (directory name) ของแฟ้มดิสก์จริง ซึ่งสมนัยกับตัวแปรแฟ้มแต่ละตัว

3. สำหรับแฟ้มข้อความซึ่งกำลังจะใช้เป็นแฟ้มข้อมูล เปิดแฟ้มและ reset เครื่องหมายตำแหน่งแฟ้มของมัน (its file location marker) ไปที่อักขระตัวแรกในแฟ้ม

4. สำหรับแฟ้มข้อความซึ่งกำลังใช้เป็นแฟ้มเอาต์พุต เปิดแฟ้มโดยการสร้างแฟ้มว่างเริ่มต้นบนดิสก์

5. ใช้ตัวแปรแฟ้มเป็นพารามิเตอร์ตัวแรกในการดำเนินการอ่าน หรือเขียนทั้งหมดที่เกี่ยวข้องกับแฟ้มข้อความ

6. ปิดแฟ้มข้อความแต่ละชุดที่ตอนจบโปรแกรม สิ่งนี้จะเขียนอักขระ <eof> ที่ตอนจบของแฟ้มเอาต์พุต

เราได้อภิปรายการดำเนินการข้อ 1 และข้อ 5 มาเรียบร้อยแล้ว การดำเนินการส่วนที่เหลือจะอธิบายต่อไป

ชื่อภายนอกและชื่อภายในสำหรับแฟ้ม (External and Internal Names for Files)

เพื่อให้ Turbo Pascal สามารถเข้าถึงแฟ้มซึ่ง save บนดิสก์ได้เราต้องจัดชื่อสารบบ (directory name) ของมัน ซึ่งหมายถึงชื่อที่ใช้ระบุถึงมันในสารบบของดิสก์

สารบบดิสก์จะประกอบด้วย ชื่อสารบบของแฟ้มทั้งหมดที่เก็บบนดิสก์ รวมทั้งสารสนเทศที่เกี่ยวข้องอื่นๆ เช่น แฟ้มมีการดัดแปรครั้งสุดท้ายเมื่อใด ขนาดของแฟ้มมีหน่วยเป็นไบต์, และเลขที่อยู่ดิสก์ของมัน

สารบบดิสก์ หมายถึง รายการของสารสนเทศเชิงพรรณาที่เกี่ยวกับแฟ้มต่างๆ ซึ่งเก็บบนดิสก์ รวมทั้งชื่อสารบบของแต่ละแฟ้ม (Disk directory is a list of descriptive information about the files stored on a disk, including each file's directory name.)

ตัวอย่าง ข้อความสั่งเรียกกระบวนงาน

```
Assign (MyData, 'A : INDATA.DAT') ;
```

เกี่ยวข้องตัวแปรแฟ้ม MyData กับแฟ้มชื่อ INDATA.DAT ในสารบบสำหรับหน่วยขับดิสก์ (disk drive) A ถ้าไม่กำหนดหน่วยขับดิสก์, Turbo Pascal ใช้ default disk

เราสามารถใช้ตัวแปร string เป็นพารามิเตอร์ตัวที่สองสำหรับข้อความสั่ง Assign ในกรณีเช่นนี้ content ของตัวแปร string ถูกใช้เป็นชื่อสารบบของแฟ้ม ซึ่งทำให้เราเข้าถึงชื่อสารบบของแฟ้มเป็นหน่วยข้อมูลระหว่างการกระทำการโปรแกรม

ข้อความสั่ง Assign เป็นเพียงข้อความสั่งที่อ้างถึง (references) ชื่อสารบบของแฟ้มเท่านั้น จากนั้นเราใช้ชื่อตัวแปรแฟ้มในข้อความสั่งโปรแกรมอื่นๆ ทั้งหมด ด้วยเหตุผลนี้โปรแกรมเมอร์ จึงเรียก MyData เป็นชื่อภายในของแฟ้ม และเรียก A : INDATA.DAT เป็นชื่อภายนอกของแฟ้ม

Syntax Display

กระบวนงาน Assign

Form : | Assign (filevar, dirname) |

ตัวอย่าง : Assign (MyData, 'A : INDATA.DAT')

มีความหมายดังนี้ : แฟ้มซึ่งชื่อสารบบของมันกำหนดโดยสายอักขระ dirname ที่เกี่ยวข้องกับตัวแปรแฟ้ม filevar ข้อสังเกต : ไม่ให้ใช้ Assign กับแฟ้มเปิด ถ้า dirname เป็นสายอักขระว่าง (' '), filevar เกี่ยวข้องกับแฟ้ม Input มาตรฐาน หรือ แฟ้ม Output มาตรฐาน ขึ้นอยู่กับว่า ภายหลัง filevar คือ อินพุตที่เปิดหรือเอาต์พุตที่เปิดตัวแปรแฟ้ม แต่ละตัวต้องมีข้อความสั่ง Assign ของมันเอง Assign ไม่ใช่ส่วนของ Standard Pascal

การเปิดแฟ้มข้อความ (Opening a Text File)

ก่อนที่โปรแกรมจะสามารถใช้แฟ้มข้อความได้แฟ้มจะต้องถูกจัดเตรียมสำหรับอินพุต หรือเอาต์พุต แฟ้มข้อความหนึ่งชุดไม่สามารถเปิดสำหรับอินพุต และเอาต์พุตทั้งคู่ ณ เวลาเดียวกัน นั่นคือ ถ้าเราอยู่ในการประมวลผลของการอ่านข้อมูลจากแฟ้มข้อความ เราไม่สามารถเริ่มต้นเขียนผลลัพธ์ไปยังแฟ้มเดียวกัน

การเปิดแฟ้ม หมายถึง การจัดเตรียมแฟ้มสำหรับอินพุต หรือสำหรับเอาต์พุต (Opening a file is the preparing a file for input or for output.)

กระบวนงาน Reset (Reset Procedure)

ข้อความสั่งเรียกกระบวนงาน

Reset (InData)

จัดเตรียมแฟ้ม InData สำหรับอินพุตให้โปรแกรม โดยการย้ายเครื่องหมายชี้ตำแหน่งแฟ้มของมัน ไปที่ตอนเริ่มต้นของแฟ้ม **เครื่องหมายชี้ตำแหน่งแฟ้ม** (file location marker) ชี้ที่อักขระถัดไปที่จะถูกประมวลผลในแฟ้ม หลังจากการดำเนินการ Reset ถูกกระทำแล้วอักขระถัดไปที่จะถูกอ่าน จะเป็นอักขระตัวแรกในแฟ้มเสมอการดำเนินการ Reset ต้องทำให้เสร็จก่อนอักขระใดๆ จะถูกอ่านจากแฟ้ม InData แต่การดำเนินการจะล้มเหลว ถ้าแฟ้ม InData ไม่ได้ถูก save บนดิสก์การอ่านและประมวลผลแฟ้มครั้งที่สอง ในการวิ่งโปรแกรมเดียวกัน ต้องกระทำการดำเนินการ Reset อีกครั้งหนึ่ง

เครื่องหมายชี้ตำแหน่งแฟ้ม หมายถึง เครื่องหมายในแต่ละแฟ้มซึ่งชี้อักขระตัวถัดไปที่จะถูกประมวลผลในแฟ้ม (File location marker is a marker in each file that points to the nest character to be processed in the file)

รูป 8.4 แสดงแฟ้ม MyData หลังจากการดำเนินการ Reset เครื่องหมายชี้ตำแหน่งแฟ้ม ชี้ที่อักขระตัวแรก

Syntax Display

กระบวนงาน Reset

Form : | Reset (infile) |

ตัวอย่าง : Reset (MyData)

มีความหมายดังนี้ : แฟ้ม infile ถูกจัดเตรียมสำหรับอินพุต และเครื่องหมายชี้ตำแหน่งแฟ้มสำหรับ infile ย้ายไปยังอักขระตัวแรกในแฟ้ม การดำเนินการ Reset กระทำอัตโนมัติบนแฟ้มระบบ Input ดังนั้น จำไม่ต้องมี Reset (Input)

15900.00 Johnny Jones<eoln> 25000.00 Sally Smythe <eoln> <eof>

↑

File location marker

รูป 8.4 แฟ้ม MyData หลังกระทำ Reset (MyData)

กระบวนงาน Rewrite (Rewrite Procedure)

ข้อความสั่งเรียกกระบวนงาน

Rewrite (OutData)

จัดเตรียมแฟ้ม OutData สำหรับการรับเอาต์พุต ถ้าดิสก์ไม่มีแฟ้ม ซึ่งสมนัยกับแฟ้ม OutData แฟ้มว่างใหม่เริ่มต้นถูกสร้างขึ้นถ้ามีแฟ้มซึ่งสมนัยกับ OutData ถูก save เรียบร้อยแล้วบนดิสก์

เครื่องหมายชี้ตำแหน่งแฟ้มของมัน ย้ายไปตอนต้นของแฟ้ม ผลคือการลบข้อมูลแฟ้มเก่า

ให้แน่ใจว่ามีการเรียกกระบวนการ Assign ที่เกี่ยวข้องกับตัวแปรแฟ้มกับชื่อสารบบของมัน ก่อนการเรียกเริ่มต้นของ Reset หรือ Rewrite กรณีอื่นๆ จะเกิดข้อผิดพลาดเวลาดำเนินงาน file not assigned

Syntax Display

กระบวนงาน Rewrite

Form : Rewrite (outfile)

ตัวอย่าง : Rewrite (MyData)

มีความหมายดังนี้ : แฟ้ม outfile ถูกจัดเตรียมสำหรับเอาต์พุตและ outfile เริ่มต้นเป็นแฟ้มว่าง ข้อมูลใดๆ ก็ตามที่เคยเก็บอยู่ในแฟ้ม outfile จะหายไป (lost) การดำเนินการ Rewrite กระทำอัตโนมัติบนแฟ้มระบบ output ดังนั้นจึงไม่ต้องมี Rewrite (output)

การปิดแฟ้ม (Closing a File)

หลังจากการประมวลผลของแฟ้มอินพุต หรือแฟ้มเอาต์พุตเสร็จสิ้นแล้วใช้กระบวนงาน Close ของ Turbo Pascal (ไม่ใช้ส่วนของ standard Pascal) เพื่อตัดการเชื่อมต่อแฟ้มจากโปรแกรมของเรา ถึงแม้ว่าเราควรปิดแฟ้มทั้งหมดให้ดูแลพิเศษกับแฟ้มเอาต์พุต เพราะว่าการเขียนอักขระแต่ละตัวไปยังแฟ้มเอาต์พุต ไม่มีประสิทธิผลอย่างมาก Pascal เก็บอักขระเอาต์พุตในบัฟเฟอร์เอาต์พุต (เนื้อที่หน่วยเก็บ) เมื่อบัฟเฟอร์เอาต์พุตเต็ม, contents ของมันถูกเขียนเป็นหนึ่งบล็อกของตัวอักขระให้สมนัยกับแฟ้มเอาต์พุต เมื่อเราปิดแฟ้มเอาต์พุต อักขระใดๆ ที่เหลืออยู่ในบัฟเฟอร์เอาต์พุตของมันจะถูกเขียนบนแฟ้มนั้น ตามด้วยอักขระ <eof> ถ้าเราไม่ปิดแฟ้มเอาต์พุต อักขระในบัฟเฟอร์เอาต์พุตของมันจะไม่ถูกทำสำเนา ดังนั้นผลลัพธ์โปรแกรมบางส่วนอาจหายไป (lost)

บัฟเฟอร์เอาต์พุต หมายถึง เนื้อที่หน่วยเก็บชั่วคราว สำหรับผลลัพธ์ที่จะถูกเขียนไปยังแฟ้มเอาต์พุต

(Output buffer is a temporary storage area for results to be written to an output file.)

Syntax Display

กระบวนงาน Close

Form : Close (filevar)

ตัวอย่าง Close (InData)

มีความหมายดังนี้ : แฟ้มชื่อ filevar ถูกปิดถ้า filevar เป็นแฟ้มเอาต์พุต ผลลัพธ์ใดๆ ซึ่งเหลืออยู่ในหน่วยความจำถูกเขียนไปยังแฟ้มเกี่ยวข้องกับ filevar ก่อนอักขระ <eof>

ข้อสังเกต : ถ้า filevar ถูกปิดไปเรียบร้อยแล้ว เมื่อเรียก Close เกิดข้อผิดพลาด เวลาดำเนินงาน Close ไม่ใช่ส่วนของ standard Pascal

การทดสอบการจบแฟ้ม : ฟังก์ชัน EOF (Testing for the End of a File : Function EOF)

เมื่ออ่านแฟ้มข้อมูล เราจำเป็นต้องทราบว่าเมื่อใดเรามาถึงตอนจบของแฟ้ม วิธีหนึ่งคือนับจำนวนอักขระซึ่งถูกประมวลผล และเปรียบเทียบเลขตัวนี้กับจำนวนอักขระในแฟ้ม อีกวิธีหนึ่ง คือ เพิ่ม **อักขระตัวเฝ้ายาม (sentinel)** ที่ตอนจบของแต่ละแฟ้ม ทั้งสองวิธีไม่ใช่สิ่งจำเป็น

ทุกแฟ้มจบด้วยอักขระ <eolf> อัตโนมัติ และฟังก์ชันในตัว EOF ของ Pascal ทดสอบว่าอักขระตัวถัดไป ซึ่งจะถูกอ่านในแฟ้มเป็นอักขระ <eof> หรือไม่

ฟังก์ชัน designator

EOF (InFile)

กลับคืน True ถ้าอักขระข้อมูลทั้งหมดในแฟ้ม InFile ถูกประมวลผลแล้ว (นั้นคือ อักขระถัดไปที่จะถูกอ่าน คืออักขระ <eof>) ฟังก์ชันนี้กลับคืน False ถ้ายังมีอักขระที่จะถูกอ่านเหลืออยู่

Syntax Display

ฟังก์ชัน EOF

Form : | EOF (filename) |

ตัวอย่าง : EOF (InFile)

มีความหมายดังนี้ : EOF กลับคืน True ถ้าอักขระถัดไปในแฟ้ม filename คือ <eof> กรณีอื่นๆ EOF กลับคืน False

ข้อสังเกต : ถ้าไม่มี filename, แฟ้มอินพุต ถือว่าเป็นแฟ้มระบบ INPUT (คีย์บอร์ด) เราสามารถใส่อักขระ <eof> ที่คีย์บอร์ดโดยกดปุ่ม Ctrl-Z ถ้าการดำเนินการอ่าน (read) มีความพยายามใน standard Pascal เมื่อ EOF (filename) เป็น True เกิดข้อผิดพลาด an Attempt to read past the end of the input file และโปรแกรมหยุดการทำงาน

รวมทั้งหมดเข้าด้วยกัน (Putting it All Together)

ตัวอย่าง 8.3 และ 8.4 ใช้ตัวดำเนินการแฟ้มทั้งหมดที่ได้อภิปรายมาแล้ว

ตัวอย่าง 8.3 เขียนตารางไว้ที่แฟ้มเอาต์พุต

โปรแกรมในรูป 8.5 อ่านแฟ้มซึ่งประกอบด้วยรายการเลข (หนึ่งตัวต่อหนึ่งบรรทัด) และเขียนเลขแต่ละตัวตามด้วยรากที่สอง และกำลังสองของมันไว้ที่แฟ้มเอาต์พุต โปรแกรมอ่านชื่อสารบบของแฟ้มข้อมูลไว้ในตัวแปรแฟ้ม InFileName หลังจากออกจากลูป แฟ้มถูกปิด และชื่อสารบบของแฟ้มเอาต์พุต (SQUARES.TXT) เขียนบนจอภาพ ดังนั้นผู้ใช้โปรแกรมจึงทราบว่าจะดูแฟ้มที่ใด ปกติโปรแกรมเมอร์ใช้ extension .TXT (สำหรับ TEXT) หรือ .DAT (สำหรับ DATA) กับแฟ้มข้อความ

Edit Window

```
Program SquareAndRoot ;
{  Writes a table of squares and square roots to a file.}
   var
      InFile,                  {input file}
      OutFile : Text ;         {output file}
      InFileName : String ;    {directory name of input file}
      NextNum : Real ;         {next number}

begin    {SquareAndRoot}
   {Prepare file for input and output}
   Write ('Input file name > ') ;
   ReadLn (InFileName) ;        {Get name of InFile}
   Assign (InFile, InFileName) ;
   Reset (InFile) ;             {Open InFiLe for input}

   Assign (OutFile, 'SQUARES.TXT') ;
   Rewrite (OutFile) ;          {Open OutFile for output}

   {Write table heading to output file}
    WriteLn (OutFile, 'N' : 10 : 2,
         'Square root' : 15 : 2, ' Square ' : 12: 2) ;

   {Read each number and write each line of the output file}
   while not EOF (InFile) do
      begin
         ReadLn (InFile, NextNum) ; {Read next number}
```

```
        WriteLn (OutFile, NextNum : 10 :2
            Sqrt (NextNum) : 15 : 2, Sqr (NexTNum) : 12 : 2)
    end ; {while}

  {Close the files}
  Close (InFile) ;
  Close (OutFile) ;
  WriteLn ('Table of roots and squares is in file SQUARES. TXT')
end.  {SquareAndRoot}
```

Output Window

```
Input file name > NUMBERS.TXT
Table of roots and squares is in file SQUARES.TXT
```

รูป 8.5 โปรแกรม SauareAndRoot

การวิ่งตัวอย่างในรูป 8.5 แสดงให้เห็นว่ามีสองบรรทัดแสดงผลบนจอภาพ บรรทัดเอาต์พุตบรรทัดแรก ประกอบด้วยข้อความพร้อมรับตามด้วย ชื่อภายนอกของแฟ้มอินพุต ผู้ใช้โปรแกรมพิมพ์ชื่อแฟ้มอินพุต (NUMBERS.TXT) แต่ข้อความสั่ง Assign ครั้งที่สองกำหนดชื่อแฟ้มเอาต์พุต (SQUARES.TXT)

หลังจากเปิดแฟ้ม โปรแกรมเขียนหัวเรื่องตารางไปที่แฟ้มเอาต์พุต ภายในลูปข้อความสั่ง ReadLn อ่านเลขแต่ละตัวจากแฟ้มอินพุต และข้อความสั่ง WriteLn เขียนผลลัพธ์เอาต์พุตสามตัวไปที่บรรทัดถัดไปของแฟ้มเอาต์พุต ถึงแม้ว่าจำนวนอักขระในบรรทัดข้อมูลอาจแปรเปลี่ยนได้ โปรแกรมเขียนอักขระ 37 ตัวต่อหนึ่งบรรทัดเอาต์พุตเสมอ รวมทั้งอักขระ <eoln> เลขซึ่งอ่านจากแฟ้มข้อมูล และเขียนที่แฟ้มเอาต์พุตจะไม่ปรากฏบนจอภาพ โปรดสังเกตว่าไม่มีความจำเป็นใดๆ สำหรับข้อความพร้อมรับภายในลูป เพราะว่าหน่วยข้อมูลไม่ได้ใส่แบบเชิงโต้ตอบ

โปรดสังเกตว่าไม่มีความจำเป็นต้องอ่าน priming เมื่อใช้ฟังก์ชัน EOF เพื่อควบคุมการทำซ้ำลูป ข้อความสั่ง ReadLn ภายในลูป อ่านบรรทัดข้อมูลทั้งหมดรวมทั้งบรรทัดแรก

ถ้าแฟ้มข้อมูลว่าง while ลูปจะไม่ถูกกระทำการ และแฟ้มจะถูกปิด ในกรณีนี้แฟ้มเอาต์พุตจะมีแต่หัวเรื่องของตารางเท่านั้น ซึ่งเขียนไว้เรียบร้อยแล้ว

ตัวอย่างแฟ้มข้อมูล และแฟ้มเอาต์พุต ซึ่งสมนัยกัน แสดงให้เห็นดังนี้

แฟ้ม NUMBERS.TXT

100 < eoln>

4.00 <eoln> <eof>

แฟ้ม SQUARES.TXT

N	Square root	Square <eoln>
100.00	10.00	10000.00 <eoln>
4.00	2.00	16.00 <eoln> <eof>

ตัวอย่าง 8.4 ทำสำเนาแฟ้ม (Copying a File)

โปรแกรมทำสำเนาแฟ้มในรูป 8.6 ทำสำเนาครั้งละหนึ่งบรรทัด จากแฟ้มอินพุตของมัน (INFILE) ไปยังแฟ้มเอาต์พุตของมัน (OUTFILE) ก่อนการทำซ้ำแต่ละครั้งของ while ลูป ฟังก์ชัน EOF ทดสอบว่าจบแฟ้ม INFILE หรือไม่ ถ้าอักขระ <eof> คือตัวถัดไป EOF (InFile) จะเป็น Ture และ not EOF (InFile) จะเป็น False ดังนั้นเกิดการออกจากลูป ถ้าแฟ้มยังมีอักขระข้อมูลอีก EOF (InFile) จะเป็น False และ not EOF (InFile) จะเป็น True ดังนั้น body ของลูป จะกระทำการต่อเนื่อง ข้อความสั่ง ReadLn อ่านบรรทัดถัดไปของแฟ้ม InFile ไว้ที่ตัวแปรสายอักขระ Line และข้อความสั่ง WriteLn เขียนบรรทัดนั้นบนแฟ้มเอาต์พุต

Edit Window

```
Program CopyFile ;
{Copies a data file by writing each line to an output file}
  var
    InFile,                  {input file}
    OutFile : Text ;         {output file}
    InFileName,              {directory name of input file}
    Line    : String ;       {each file line}
```

```
begin {CopyFile}
   {Prepare files for input and output}
   Write ('File to be copied > ') ;
   ReadLn (InFileName) ;     {Get name of InFile}
   Assign (InFile, InFileName) ;
   Reset (InFile) ;          {Open InFile for input}

   Assign (OutFile, 'CopyFILE.TXT') ;
   Rewrite (OutFile) ;       {Open OutFile for output}

   {Copy the data file line by line}
   while not EOF (InFile) do
      begin
         ReadLn (InFile, Line) ; {Read next line}
         WriteLn (OutFile, Line) ; {Write it to output file}
      end ; {while}

   {Close the files}
   Close (InFile) ;
   Close (OutFile) ;
   WriteLn ('File', InFileName, 'is copied to COPYFILE. TXT')
end. {CopyFile}
```

Output Window

```
File to be copied > A : MYDATA.TXT
File A : MYDATA. TXT is copied to COPYFILE.TXT
```

รูป 8.8 โปรแกรม Copy File

Syntax Display

ฟังก์ชัน EOLN

Form : | EOLN (filename) |

ตัวอย่าง : EOLN (InFile)

มีความหมายดังนี้ : ผลลัพธ์ของฟังก์ชันเป็น True ถ้าอักขระถัดไปในแฟ้ม filename คือ <eoln> กรณีอื่นๆ ผลลัพธ์เป็น False

ข้อสังเกต : ถ้าไม่มี filename แฟ้มอินพุต ถูกสมมติว่าเป็นแฟ้มระบบ INPUT (คีย์บอร์ด) ใน standard Pascal การเรียกฟังก์ชัน EOLN จะเกิดข้อผิดพลาดถ้า EOF (filename) เป็นจริง

ตัวอย่าง 8.5 การทำสำเนาหนึ่งบรรทัดของแฟ้ม (Copying a Line of a File)

กระบวนงาน CopyLine ในรูป 8.7 ทำสำเนาหนึ่งบรรทัดของพารามิเตอร์แฟ้มตัวแรกไว้ที่พารามิเตอร์แฟ้มตัวที่สองของมัน while ลูป สำเนาอักขระแต่ละตัวจนถึง <eoln> จากแฟ้ม InFile ไปยัง OutFile หลังจากออกจากลูป ข้อความสั่ง

```
ReadLn (InFile) ;        {skip the <eoln>}
WriteLn (OutFile);       {write it to OutFile}
```

ข้ามอักขระ <eoln> ใน InFile และเขียนอักขระ <eoln> ที่ OutFile

เนื่องจาก standard Pascal ไม่มีแบบชนิดข้อมูล String เราจึงไม่สามารถใช้ ReadLn อ่านหนึ่งบรรทัดข้อมูล ไว้ในตัวแปร string เช่นที่เราทำในโปรแกรม CopyFile (รูป 8.6) แต่เราสามารถใช้ CopyLine เพื่อเขียนหนึ่งบรรทัดข้อมูลไปที่แฟ้มเอาต์พุตใน standard Pascal ได้ ตัวอย่างเช่น เราเรียกกระบวนงาน CopyLine ในโปรแกรม CopyFile เพื่อสำเนาแต่ละบรรทัดของแฟ้มอินพุตไปยังแฟ้มเอาต์พุต ในกรณีนี้ while ลูปต้องเขียนใหม่ดังนี้

```
{Copy each line from InFile to OutFile}
while not EOF (InFile) do
   CopyLine (InFile, OutFile) ;
```

```
procedure CopyLine (var InFile  {input file},
                    OutFile {output file} : Text) ;
{
   Copies one line of file InFile to file OutFile
   Pre  :  InFile is opened for input and OutFile for output
   Post :  Each character on the current line of file InFile is copied to
           OutFile. The <eoln> character is the last character written to
           OutFile and the file location marker for InFile is at the start of the
           next line.
{
   var
      Next : Char ;  {each character}

begin {CopyLine}
   {Copy all characters up to the <eoln>}
   while not EOLN (InFile) do
      begin
         Read (InFile, Next) ;
         Write (OutFile, Next) ;
      end ; {while}

   {Process the <eoln> character}
   ReadLn (InFile) ;    {skip the <eoln>}
   WriteLn (OutFile)    {write it to OutFile}
.end ; {CopyLine}
```

รูป 8.7 กระบวนงาน CopyLine

ถึงแม้ว่ากระบวนงาน CopyLine จะไม่อ่านอักขระ <eoln> จริงไว้ที่ Next โปรแกรม Pascal สามารถอ่านอักขระ <eoln> เมื่อเกิดเหตุการณ์นี้ Turbo Pascal เก็บอักขระที่สมนัยกับ Ctrl-M ในตัวแปรรับ <eoln> เป็นหน่วยข้อมูลของมัน แต่ standard Pascal เก็บอักขระว่างแทน

สไตล์โปรแกรม (Program Style)

การใช้ ReadLn หลังลูป (Using ReadLn After a Loop) หลังจากออกจากลูป ในรูป 8.7 ข้อความสั่ง ReadLn (InFile) ; {Skip the <eoln>}

กระทำการ โดยข้ามอักขระ <eoln> การใส่ ReadLn (InFile) ถัดจากลูป ซึ่งใช้ not EOLN (InFile) เป็นตัวทดสอบการทำซ้ำลูป ทำให้ ReadLn ประมวลผลอักขระ <eoln> ซึ่งเป็นเหตุให้การทำซ้ำจบ (terminate) ถ้าอักขระ <eoln> ไม่ถูกประมวลผล Read ถัดไป หรือ ReadLn ถัดไปจะอ่านอักขระ <eoln> แทนที่จะเป็นอักขระตัวแรกในบรรทัดข้อมูลถัดไป น่าจะทำให้เกิดผลลัพธ์โปรแกรมที่ไม่คาดคิดขึ้น และข้อผิดพลาดเวลาดำเนินการ ที่จริงถ้าข้อความสั่งนี้ เอาออกไปจาก CopyLine ลูป

```
while not EOF (InFile) do
    CopyLine (InFile, OutFile) ;
```

จะกระทำการไม่มีการหยุด อักขระตัวถัดไปที่จะประมวลผลหลังจากกลับคืน (return) จาก CopyLine จะต้องเป็นอักขระ <eoln> เสมอ ดังนั้นในการเรียกถัดไปของ CopyLine, while ลูปของมันจะถูกข้าม และข้อความสั่ง

```
WriteLn (OutFile) {Write it to OutFile}
```

จะเขียนอักขระ <eoln> อีกตัวหนึ่ง ไปที่ OutFile แทนที่จะเขียนบรรทัดข้อมูลถัดไป กระบวนการนี้จะทำต่อเนื่องจนกระทั่งผู้ใช้โปรแกรมจบ (terminated) โปรแกรม

พารามิเตอร์แฟ้มในกระบวนงานนิยาม โดยผู้ใช้ (File Parameters in User – Defined Procedures)

พารามิเตอร์แฟ้ม InFile เป็นพารามิเตอร์ตัวแปรใน parameter list ของกระบวนงาน CopyLine เพราะว่า InFile แทนแฟ้มอินพุต มันจึงดูแปลกที่ประกาศเป็นพารามิเตอร์ตัวแปร อย่างไรก็ตาม Pascal ต้องให้พารามิเตอร์แฟ้มทั้งหมดเป็นพารามิเตอร์ตัวแปร เพราะว่าการสำเนาเฉพาะที่ (local copy) ของแฟ้มขนาดใหญ่อาจจะใช้หน่วยความจำมากเกินไป

การใช้ EOLN กับข้อมูลคีย์บอร์ด (Using EOLN with Keyboard Data)

เราสามารถใช้ฟังก์ชัน EOLN กับพารามิเตอร์ INPUT เพื่อทดสอบอักขระ <eoln> เมื่ออ่านข้อมูลจากคีย์บอร์ดได้โปรดจำไว้ว่าอักขระ <eoln> ถูกใส่แฟ้มระบบ Input เมื่อกดปุ่ม Enter

ข้อความสั่ง

```
CopyLine (Input, OutFile)
```

ส่ง Input เป็นพารามิเตอร์แฟ้มตัวแรกไปยังกระบวนงาน CopyLine (ดูรูป 8.7) ดังนั้น บรรทัดข้อมูลถัดไป ซึ่งพิมพ์ที่คีย์บอร์ด จะเขียนไปที่แฟ้ม OutFile

เป็นการง่ายที่ใช้ EOLN หรือ EOF ผิด เพื่อทดสอบอักขระ <eoln> หรือ <eof> ในข้อมูลคีย์บอร์ด ถ้าเราลืมระบุพารามิเตอร์แฟ้ม และเขียน function designator เป็นแค่ EOLN (เช่นใน while not EOLN do), Pascal จะใช้แฟ้มระบบ Input เป็น default file parameter ในทำนองเดียวกัน ถ้าเราลืมใช้พารามิเตอร์แฟ้มกับ Read หรือ ReadLn Pascal จะใช้ Input โดย default ดังนั้นโปรแกรมจะหยุดจนกว่าเราพิมพ์บรรทัดข้อมูล

แบบฝึกหัด 8.2 Self – Check

1. จงหาข้อผิดพลาด ในส่วนของโปรแกรมข้างล่างนี้ สมมติว่า InFile เป็นชนิด text, Next เป็นชนิด Char แฟ้มชุดใดเป็นแฟ้มอินพุต ชุดใดเป็นแฟ้มเอาต์พุต และแฟ้มชุดใดถูกทดสอบโดยฟังก์ชัน EOLN

```
Assign  ('MYDATA.TXT', InFile) ;
Rewrite (InFile) ;
Reset (Input) ;
While not EOLN do
   begin
      Read (InFile, Next) ;
      Write (Next)
   end; {while}
```

2. สมมติว่าเรากำหนดให้แฟ้มอินพุต NUMBERS.TXT ด้วยจำนวนเต็มหนึ่งตัวต่อหนึ่งบรรทัด สมมติว่าตัวแปร NumData เป็นชนิด Text มีข้อความสั่งอะไรบ้างที่ต้องใช้

เพื่อจัดเตรียมสำหรับการอ่านแฟ้ม และมีข้อความสั่งอะไรที่ควรถูกกระทำการ หลังจากการอ่านตลอดแฟ้ม

เขียนโปรแกรม

1. จงเขียนกระบวนงานอ่านแฟ้มซึ่งมีเลขจำนวนเต็ม, หนึ่งตัวต่อหนึ่งบรรทัด และคำนวณผลบวกของจำนวนเต็ม อินพุตของกระบวนงานเป็นตัวแปร FileName ชนิด String ประกอบด้วยชื่อของแฟ้มที่จะอ่าน เอาต์พุตของกระบวนงาน คือ ผลบวก ถ้าเป็นแฟ้มว่างผลบวกเป็นศูนย์

2. จงเขียนโปรแกรมบีบอัด (compresses) แฟ้มโดยการเขียนอักขระเว้นอักขระ (every other character) ในแฟ้มไปยังแฟ้มเอาต์พุต แฟ้มใหม่จะประกอบด้วย ตัวอักขระประมาณครึ่งหนึ่งของแฟ้มเดิม

3. จงเขียนกระบวนงาน ซึ่งกลับคืน เลขนับจำนวนการเกิดของตัวอักขระเฉพาะหนึ่งตัวบนบรรทัดปัจจุบันของแฟ้มข้อมูล ชื่อแฟ้มภายในและอักขระซึ่งกำลังจะนับเป็นอินพุตของกระบวนงาน จำนวนนับของอักขระ เป็นเอาต์พุตของกระบวนงาน

8.3 การใช้แฟ้มข้อความ (Using Text Files)

ถ้าโปรแกรมที่หนึ่ง เขียนเอาต์พุตของมันไว้ที่แฟ้มดิสก์ ไม่ใช่ไว้ที่จอภาพ โปรแกรมที่สองใช้แฟ้มเอาต์พุตนั้น เป็นแฟ้มข้อมูลของมัน สองโปรแกรมนี้ สื่อสารซึ่งกันและกันผ่านแฟ้มดิสก์ กรณีศึกษาต่อไปนี้ พัฒนาโปรแกรมซึ่งแฟ้มเอาต์พุตของมันเป็นแฟ้มข้อมูลของอีกโปรแกรมหนึ่ง

กรณีศึกษา การเตรียมแฟ้มบัญชีเงินเดือน (Case Study : Preparing a Payroll File)

ปัญหา (Problem)

นักบัญชีของบริษัทต้องการให้เราเขียนโปรแกรมบัญชีเงินเดือนสองโปรแกรม โปรแกรมแรกอ่านแฟ้มข้อมูล ซึ่งประกอบด้วยข้อมูลเงินเดือนของพนักงาน ข้อมูลของพนักงานแต่ละคนเก็บในสองบรรทัดติดต่อกัน บรรทัดแรกจะมีชื่อพนักงาน บรรทัดที่สองเริ่มต้นด้วยตัวชี้บอก (indicator) สถานภาพของพนักงาน (status) ตัว H หมายถึง พนักงานจ้างรายชั่วโมง หรือ ตัว S หมายถึง พนักงานเงินเดือน ไม่ว่าจะเป็นอัตรารายชั่วโมงของพนักงาน (สำหรับพนักงานจ้างรายชั่วโมง) หรือเงินเดือนรายปีของพนักงาน (สำหรับพนักงานเงินเดือน) จะเป็นดังนี้

สำหรับพนักงานจ้างรายชั่วโมงเท่านั้นซึ่งบรรทัดที่สองจะจบด้วยค่าข้อมูลเพิ่มเติม ได้แก่จำนวนชั่วโมงทำงานของสัปดาห์นั้น แฟ้มข้อมูลตัวอย่างที่มีพนักงานเงินเดือนหนึ่งคน และพนักงานจ้างรายชั่วโมงหนึ่งคน เป็นดังนี้

แฟ้มข้อมูล (Data File)

Peter Liacouras <eoln>

S 260000.00 <eoln>

Caryn Koffman <eoln>

H 10.00 40.0 <eoln> <eof>

โปรแกรมแรกจะเขียนชื่อของพนักงานแต่ละคนไปที่แฟ้มเอาต์พุต ตามด้วยบรรทัดซึ่งประกอบด้วยเงินสุทธิที่คำนวณได้ของพนักงานต่อสัปดาห์ คำนวณและแสดงผลจำนวนเงินเดือนทั้งหมดสำหรับสัปดาห์ เช่นที่แสดงในแฟ้มเอาต์พุต ข้างล่างนี้ บรรทัดที่สองของแฟ้มเอาต์พุต เป็นรายการ เงินเดือนทั้งปีของพนักงาน (เช่น $ 260,000) หารด้วย 52 บรรทัดที่สีของแฟ้มเอาต์พุตควรจะเป็นรายการผลคูณของอัตราของพนักงานจ้างรายชั่วโมง ($ 10.00) และชั่วโมงทำงาน (40.0)

แฟ้มเอาต์พุต (Output File)

Peter Liacouras <eoln>

5000.0 <eoln>

Caryn Koffman <eoln>

400.00 <eoln> <eof>

โปรแกรมที่สองจะอ่านแฟ้มเอาต์พุต และเขียนเช็คเงินเดือน (payroll checks) ขึ้นอยู่กับ contents ของแฟ้มนั้น ตัวอย่างเช่น เช็คใบแรกควรเป็น $5000.00 จ่ายให้ Peter Liacouras

วิเคราะห์ (Analysis)

ขณะนี้เราจะเขียนโปรแกรมแรก และทิ้งโปรแกรมที่สองเป็นแบบฝึกหัดท้ายบท โปรแกรมต้องทำสำเนา ชื่อของพนักงานแต่ละคนไปที่แฟ้มเอาต์พุต พร้อมทั้งคำนวณเงินเดือนรายสัปดาห์ของพนักงานแต่ละคนแล้วเขียนไปที่แฟ้มเอาต์พุต และบวกเลขตัวนี้กับเงินเดือนรวมทั้งหมด (payroll total)

ข้อมูลที่ต้องการ (Data Requirements)

อินพุตปัญหา (จากแฟ้มอินพุต InEmp)

ชื่อของพนักงานแต่ละคน

ประเภท (รายชั่วโมงหรือรายเดือน) ของพนักงานแต่ละคน

อัตราค่าจ้างของพนักงานแต่ละคน

จำนวนชั่วโมงทำงานของพนักงานรายชั่วโมง

เอาต์พุตปัญหา (ไปยังแฟ้มเอาต์พุต OutEmp)

ชื่อพนักงานแต่ละคน

เงินเดือนรายสัปดาห์ของพนักงานแต่ละคน

เอาต์พุตปัญหา (ไปยังจอภาพ)

·Payroll : Real {the payroll total}

ออกแบบ (Design)

โปรแกรมหลักเตรียมแฟ้มต่างๆ สำหรับอินพุต และเอาต์พุตและเรียกกระบวนงาน ProcessEmp เพื่อประมวลผล พนักงานทั้งหมดและสะสมเงินเดือนรวม (total payroll) (ขั้นที่ 2 ในอัลกอริทึมข้างล่างนี้) หลังจาก ProcessEmp เสร็จแล้ว โปรแกรมหลักแสดงผลเงินเดือนรวมสุดท้าย

อัลกอริทึมสำหรับโปรแกรมหลัก

1. เตรียมแฟ้ม InEmp และ OutEmp
2. ประมวลผลพนักงานทั้งหมด และคำนวณเงินเดือนรวมทั้งหมด
3. แสดงผลเงินเดือนรวมทั้งหมด

วิเคราะห์และออกแบบ ProcessEmp (Analysis and Design of ProcessEmp)

ProcessEmp จัดสรรหน่วยเก็บสำหรับตัวแปร ซึ่งเก็บข้อมูลเงินเดือนของพนักงานและเงินเดือนสุดท้าย ข้อมูลที่ต้องการ สำหรับ ProcessEmp ได้แก่

ข้อมูลที่ต้องการสำหรับ ProcessEmp (Data Requirement for ProcessEmp)

พารามิเตอร์อินพุต

InEmp : Text {file of employee data}

พารามิเตอร์เอาต์พุต

OutEmp : Text {payroll file}

พารามิเตอร์ อินพุต/ เอาต์พุต

Payroll : Real {the payroll amount so far}

ตัวแปรเฉพาะที่ (Local Variables)

HourOrSal : Char {input – hourly or salaried indicator}

Hours : Real {input – hours worked}

Rate : Real {input – hourly rate}

Salary : Real {output – gross salary}

อัลกอริทึม สำหรับ ProcessEmp อยู่ข้างล่างนี้ ขั้นที่ 3 ของ ProcessEmp กระทำโดยกระบวนงาน CopyLine (ดูรูป 8.7) ขั้นที่ 4 กระทำโดยกระบวนงาน ReadData และขั้นที่ 5 กระทำโดยฟังก์ชัน ComputeSal รูป 8.8 แสดงผังโครงสร้าง

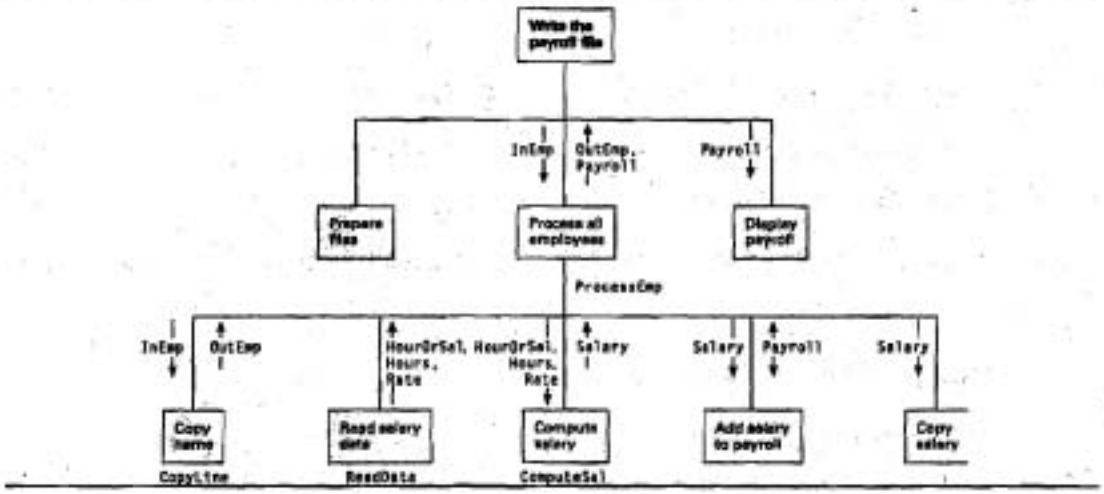
อัลกอริทึม สำหรับ ProcessEmp

1. Initialize payroll total to 0.0
2. While there are more employee do

   begin

   3. Read next employee's name from InEmp and write it to OutEmp
   4. Read next employee's salary data
   5. Compute next employee's weekly salary
   6. Write next employee's salary to OutEmp and add it to payroll total

   end



รูป 8.8 Structure Chart สำหรับการเขียนแฟ้ม Payroll

การทำให้เกิดผล (Implementation)

การลงรหัสโปรแกรมหลัก

รูป 8.9 แสดงให้เห็นโปรแกรมหลัก นอกจากแฟ้มความ InEmp และ OutEmp แล้ว มีเฉพาะ Payroll เท่านั้น ซึ่งเป็นตัวแปรที่ประกาศในโปรแกรมหลัก เราประกาศตัวแปรที่จำเป็นเพื่อเก็บและประมวลผลข้อมูลของพนักงานแต่ละคนใน ProcessEmp ขั้นตอนนี้ต้องกันกับ (consistent with) นโยบายการประกาศตัวแปรในส่วนจำเพาะ (module) ระดับสูงสุดซึ่งเข้าถึงมัน

```
Edit Window
program WritePayroll ;
{
   Write each employee's name and weekly salary to an output file and
   computes total payroll amount.
}
   var
      InEmp,              {data file}
      Out Emp : Text ;    {output file}
      Rayroll : Real ;    {output – total payroll}

{Insert ReadData, ComputeSal, and ProcessEmp here}

begin {Write Payroll}

   {Prepare InEmp and OutEmp}
   Assign (InEmp, 'A : INEMP.DAT') ;
   Reset (InEmp) ;

   Assign (OutEmp, 'A : OUTEMP.DAT') ;
   Rewrite (OutEmp) ;
```

```
    {Process all employee and compute payroll total}
    ProcessEmp (InEmp, OutEmp, Payroll) ;

    {Display result and close files}
    WriteLn ('Total payroll is $ ', Payroll : 4 :2) ;
    Close (InEmp) ;
    Close (OutEmp)
end. {WritePayroll}
```

Output Window

```
Total payroll is $5400.00
```

รูป 8.9 การเขียน Payroll File

การลงรหัสส่วนจำเพาะ (Coding the Modules)

รูป 8.10 แสดงให้เห็นกระบวนงาน ProcessEmp และส่วนจำเพาะย่อยของมัน (subordinate modules) กระบวนงาน ReadData อ่านอักขระข้อมูลตัวแรกและหลังจากนั้นอ่านค่าตัวเลขหนึ่งค่าหรือสองค่าขึ้นอยู่กับว่าอักขระตัวแรกเป็น S หรือ H ฟังก์ชัน Compute-Sal คำนวณเงินเดือนรายสัปดาห์สุทธิ (gross weekly salary) สำหรับพนักงาน และเลขตัวนี้กำหนดให้กับ Salary หลังจากกลับคืนฟังก์ชัน

```
{Insert procedure CopyLine (ดูรูป 8.7) here.}

procedure ReadData   (var  InEmp {input file} : Text ;
                      var  HourOrSal {output} : Char ;
                      var  Rate {output},
                           Hours {output} : Real) ;
```

```
{
Reads the employee salary data
Pre  :  File InEmp is opened for input
Post :  HourOrSal contains the first data character on the data line.
        If HourOrSal is H, Rate contains the hourly rate and Hours contains
        the hours warked.
        If HourOrSal is S, Rate contains the annual Salary and Rate is
        undefined. Otherwise, Hours and Rate are undefined.
}
begin {ReadData}
Read (InEmp, HourOrSal) ;
HourOrSal := UpCase (HourOrSal) ;
If  HourOrSal = 'H' then
   ReadLn (InEmp, Rate, Hours) {Read hourly rate and hours}
else if HourOrSal = 'S' then
   ReadLn (InEmp, Rate) {Read annual salary}
else
   begin
      ReadLn (InEmp) ; {Skip past <eoln>}
      WriteLn ('Invalid character', HourOrSal)
   end {if}
end, {ReadData}

function ComputeSal   (HourOrSal : Char ; Rate, Hours : Real)
                      : Real ;
{
Computes the weekly gross salary.
Pre  :  HourOrSal and Rate are defined.
```

```
Post  :  If HourOrSal is H, returns Rate * Hours ;
         If HourOrSal is S, returns Rate/52 ;
         Otherwise, result is undefined.
}
begin
   if HourOrSal = 'H' then
      ComputeSal := Rate * Hours
   else if HourOrSal = 'S' then
      ComputeSal := Rate/52.0
end ; {ComputeSal}

procedure  ProcessEmp  (var  InEmp   {input file},
                             OutEmp {output file} : Text ;
                       var   Payroll  {output} : Real) ;
{
Processes all employees and computes payroll total.
Pre   :  InEmp is prepared for input and OutEmp for output.
Post  :  All employee names and weekly salaries are written to OutEmp and
         the sum of their salaries is returned through Payroll.
}
   var
      HourOrSal : Char ;  {input-hourly or salaries indicator}
      Hours,              {input – hours worked}
      Rate,               {input- hourly rate}
      Salary : Real ;     {output – gross salary}

begin {ProcessEmp}
   Payroll := 0.0 ;
```

```
    while not EOF (InEmp) do
      begin
        CopyLine (InEmp, OutEmp) ; {Copy employee name.}
        {Read salary data}
        ReadData (InEmp, HourOrSal, Rate, Hours) ;
        Salary := ComputeSal (HourOrSal, Rate, Hours) ;
        If (HourOrSal = 'H') or (HourOrSal = 'S') then
          begin {good data}
            Payroll := Payroll + Salary ;
            WriteLn (OutEmp, Salary :4 :2)
          end {good data}
        else
          WriteLn (OutEmp, 'Salary is undefined')
      end {while}
  end ; {ProcessEmp}
```

รูป 8.10 กระบวนงาน ReadData, ฟังก์ชัน ComputeSal และกระบวนงาน ProcessEmp

-while ลูปในกระบวนงาน ProcessEmp ทดสอบว่าอักขระตัวถัดไปคือ <eof> หรือไม่ ถ้าไม่ใช่ ProcessEmp เรียก CopyLine เพื่อทำสำเนาชื่อของพนักงานถัดไปจากแฟ้มอินพุต (InEmp) ไปยังแฟ้มเอาต์พุต (OutEmp) CopyLine ประมวลผล บรรทัดเว้นบรรทัดของแฟ้ม InEmp เริ่มต้นด้วยบรรทัดแรกต่อไป ข้อความสั่ง

ReadData (InEmp, HourOrSal, Rate, Hours) ;

ใน ProcessEmp เรียก ReadData เพื่ออ่านข้อมูลเงินเดือนของพนักงานจาก InEmp และเลื่อนเครื่องหมายชี้ตำแหน่งแฟ้มไปยังอักษรตัวแรกของชื่อพนักงานถัดไป ถ้าอักขระตัวที่ถูกอ่านไปไว้ HourOrSal ไม่ใช่ H หรือ S, กระบวนงาน ReadData จะเขียน ข้อความระบุความผิดพลาดไปที่จอภาพ และข้ามไปเริ่มต้นยังบรรทัดข้อมูลถัดไป ข้อความสั่งถัดไปใน ProcessEmp

Salary := ComputeSal (HourOrSal, Rate, Hours) ;

เรียกฟังก์ชัน ComputeSal เพื่อคำนวณ เงินเดือนสุทธิรายสัปดาห์ และกำหนดค่าไปที่ Salary สุดท้ายข้อความสั่ง

```
Payroll := Payroll + Salary ;
WriteLn (OutEmp, Salary :4 :2)
```

บวกเงินเดือนสุทธิของพนักงานปัจจุบันกับ Payroll และเขียนที่แฟ้ม OutEmp ซึ่งนิยามเป็น Salary

การทดสอบ (TestInd)

วิ่งโปรแกรมบัญชีเงินเดือนด้วยข้อมูลบัญชีเงินเดือนตัวอย่างดูว่ามันคำนวณเงินเดือนรายสัปดาห์ และ payroll ถูกต้องหรือไม่ ทั้งพนักงานจ้างรายชั่วโมง และพนักงานเงินเดือน ทั้งคู่ดูว่าเกิดอะไรขึ้น เมื่ออักขระตัวแรกของข้อมูลเงินเดือนของพนักงานไม่ใช้ H หรือ S ในกรณีนี้โปรแกรมควรจะแสดงข้อความระบุความผิดพลาด และประมวลผลต่อไปกับข้อมูลของพนักงานถัดไปตรวจสอบว่าโปรแกรมประมวลผลถูกต้องหรือไม่กับพนักงานคนต่อมา หรือว่ามีข้อผิดพลาดขยายต่อเป็นเหตุให้ข้อผิดพลาดในข้อมูลของพนักงานหนึ่งคนทำให้ผลลัพธ์อื่นๆ ทั้งหมดไม่ถูกต้องหรือไม่

ความสำคัญของการเลื่อนผ่านอักขระ <eoln> (The Important of Advancing Past the <eoln> character)

เป็นเรื่องง่ายที่จะเกิดข้อผิดพลาดเมื่ออ่านข้อมูลอักขระผสมกับข้อมูลตัวเลข ปัญหาจำนวนมากเกิดขึ้นจากการไม่เลื่อนผ่านอักขระ <eoln> ตัวอย่างเช่น จงพิจารณาว่าจะเกิดอะไรขึ้นถ้า ReadData ใช้ข้อความสั่ง

```
Read (InEmp, Rate) ;
```

เมื่ออ่านข้อมูลของพนักงานรายเดือน แทนที่จะเรียกกระบวนงาน ReadLn ความยากคือกระบวนงาน Read เลื่อนเครื่องหมายชี้ตำแหน่งแฟ้มสำหรับแฟ้ม InEmp ไปจนถึงอักขระ <eoln> แต่ไม่ผ่านอักขระตัวนี้

```
S 260000.00 <eoln>
            ↑
        File location marker
```

เมื่อ CopyLine ถูกเรียกเพื่อทำสำเนาชื่อของพนักงานคนที่สอง while ลูปออกทันทีโดยไม่อ่านชื่อ เพราะว่าอักขระตัวถัดไป คือ อักขระ <eoln> และอักขระ <eoln> ถูก

ประมวลผลก่อน CopyLine กลับคืนไปกระบวนงาน ProcessEmp และอักขระถัดไปขณะนี้คือ อักษร ตัวแรกของชื่อพนักงาน เมื่อ ReadData ถูกเรียกอีกครั้งหนึ่ง ข้อความสั่ง

Read (InEmp, HourOrSal)

จะอ่านตัวอักษรตัวแรก คือ C ของชื่อของพนักงานคนที่สอง ดังนั้น ข้อความระบุความผิดพลาดจะแสดงผลบนจอภาพ

แบบฝึกหัด 8.3

1. a) จะเกิดผลอะไรขึ้น ถ้ามีอักขระว่างที่ตอนท้ายของบรรทัดข้อมูลในแฟ้มข้อมูลสำหรับโปรแกรมในรูป 8.9

b) จะเกิดผลอะไรขึ้นกับบรรทัดว่าง

2. ทำไมจึงใส่ชื่อพนักงานและเงินเดือนแยกต่างหากคนละบรรทัดของแฟ้ม OutEmp ทำให้การเขียนโปรแกรมเพื่อพิมพ์ เช็คบัญชีเงินเดือนง่ายขึ้น

เขียนโปรแกรม

1. จงเขียนโปรแกรมอ่านแฟ้ม OutEmp ซึ่งได้จากโปรแกรม WritePayroll (รูป 8.9) และแสดงผลนับจำนวนพนักงานที่ประมวลผลโดย WritePayroll และเงินเดือนเฉลี่ยของพนักงานกลุ่มนี้

8.4 การแก้จุดบกพร่องด้วยแฟ้ม (Debugging with Files)

การเตรียมแฟ้มข้อมูล ซึ่งประกอบด้วยเซตตัวแทนของข้อมูล ทดสอบสำหรับโปรแกรมใหม่ทำให้การแก้จุดบกพร่องโปรแกรมง่ายขึ้นทุกครั้งที่วิ่งโปรแกรม โปรแกรมสามารถอ่านข้อมูลทดสอบจากแฟ้มนี้ ขจัดความจำเป็นที่ต้องพิมพ์หน่วยข้อมูลใหม่ หลังจากโปรแกรมเชิงโต้ตอบมีการแก้ไขจุดบกพร่องแล้ว ให้แทนที่ชื่อสารบบแฟ้ม (directory name) ในข้อความสั่ง Assign ด้วยสายอักขระว่าง (' ') เพื่อว่า Turbo Pascal จะอ่านข้อมูลจากคีย์บอร์ดในการวิ่งภายหลัง

เราสามารถใช้ไอเดนติไฟเออร์แฟ้มเป็นตัวแปร Watch ค่าซึ่งแสดงผลใน Watch window สำหรับไอเดนติไฟเออร์ แฟ้มคือ รายการภายในวงเล็บ ซึ่งประกอบด้วย ชื่อสารบบสำหรับแฟ้ม เหมือนเช่นกำหนดในข้อความสั่ง Assign รายการนี้ จะประกอบด้วย สายอักขระเข้าซึ่งบอกว่าแฟ้มเปิดสำหรับอินพุต, เปิดสำหรับเอาต์พุต หรือปิด แล้วหรือยัง

8.5 ข้อผิดพลาดร่วมของการเขียนโปรแกรม

อย่าลืมใช้กระบวนงาน Assign ที่เกี่ยวข้องไอเดนติไฟเออร์ แฟ้มแต่ละตัวกับชื่อสารบบของมันก่อนการเปิดแฟ้ม ชื่อแฟ้มที่ใช้ในโปรแกรมของเราอาจแตกต่างจากชื่อสารบบของแฟ้มดิสก์ที่เกี่ยวข้อง ชื่อแฟ้มทั้งหมด ต้องประกาศเป็นตัวแปร (ชนิด Text) ยกเว้นแฟ้มระบบ Input และ Output

ให้ใช้กระบวนงาน Reset หรือ Rewrite เสมอเพื่อเตรียมแฟ้มสำหรับอินพุต หรือเอาต์พุต (ยกเว้นแฟ้มระบบ Input และ Output) ถ้าเราเขียนใหม่บนแฟ้มที่มีอยู่แล้ว ข้อมูลเดิมบนแฟ้มนั้นจะหายไปต้องมั่นใจว่าเราไม่เผลอใส่ข้อความสั่ง Reset หรือ Rewrite ในลูป ถ้าเราทำการดำเนินการ Read ในลูปจะอ่านส่วนประกอบแฟ้มแรกซ้ำๆ กัน การดำเนินการ Write ในลูปจะเขียนส่วนประกอบของแฟ้มแรกซ้ำๆ กัน

กระบวนงาน Read หรือ ReadLn สามารถใช้ได้เฉพาะหลังจากแฟ้มมีการเตรียมสำหรับอินพุตแล้วเท่านั้น กระบวนงาน Write หรือ WriteLn สามารถใช้ได้เฉพาะหลังจากแฟ้มมีการเตรียมสำหรับเอาต์พุตแล้วเท่านั้น เมื่อกระทำแฟ้มอินพุต หรือ แฟ้มเอาต์พุต ให้แน่ใจว่าได้กำหนดชื่อแฟ้มเป็นพารามิเตอร์ กระบวนงาน ตัวแรก กรณีอื่นๆ แฟ้มระบบ Input หรือ output จะถูกสมมติขึ้น ลูปข้างล่างนี้จะกระทำการไม่หยุด (forever) เพราะว่าข้อมูลถูกอ่านจากแฟ้มระบบ Input (default file) แทนที่จะเป็นแฟ้ม MyData ซึ่งทดสอบอักขระ <eoln>

```
While not EOLN (MyData) do
  begin
    Read (Next) ; {incorrect read from keyboard}
    Write (Next)
  end ; {while}
```

เมื่อเราใช้ฟังก์ชัน EOLN หรือ EOF เพื่อควบคุมการใส่ข้อมูลอย่าลืมใส่ชื่อแฟ้มข้อมูลเป็นอาร์กิวเมนต์ของฟังก์ชัน ให้จำว่าต้องข้าม (skip over) อักขระ <eoln> โดยใช้ ReadLn หลังจากถึงตอนจบของบรรทัดข้อมูล จงรอบคอบไม่พยายามที่จะอ่านข้อมูลเมื่อเครื่องหมายชี้ตำแหน่งแฟ้มอยู่ที่อักขระ <eof>

ถ้าเรากดปุ่ม Enter นานเกินไป เมื่อจบการสร้างแฟ้มข้อมูล เราอาจใส่บรรทัดว่างเพิ่มที่ตอนท้ายของแฟ้มข้อมูล ตัวอย่างแฟ้มข้างล่างนี้ ประกอบด้วย เลขหนึ่งตัวต่อหนึ่งบรรทัด และมีบรรทัดว่างตอนท้าย

500 <eoln>

37 <eoln>

<eoln> <eof>

ถึงแม้ว่าบรรทัดว่างดูเหมือนไม่มีอันตราย ถ้าเราใช้หัวเรื่อง while

while not EOF (InData) do

เพื่อควบคุมลูป ซึ่งอ่านและประมวลผลเลขหนึ่งตัวต่อหนึ่งบรรทัด บรรทัดว่างจะทำให้ลูป กระทำการโดยใช้เวลาเพิ่มอีกหนึ่งรอบ

ถ้าแฟ้มเอาต์พุตหายไป หรือเขียนไม่สมบูรณ์ เช่นเราอาจจะลืมปิดแฟ้มเอาต์พุต Turbo Pascal เก็บ (save) ข้อมูลที่กำลังเขียนไปยังแฟ้มในบัฟเฟอร์เอาต์พุตชั่วคราว ในหน่วยความจำ

ข้อความสั่ง Close ทำสำเนาข้อมูลส่วนที่เหลือใดๆ ในบัฟเฟอร์เอาต์พุตไปยังแฟ้มดิสก์

ข้อสรุปของ New Pascal Constructs

Construct	Effect
การประกาศแฟ้ม var MoreChars, MoreDigits : Text ; I : Integer ; NextChar : Char ;	MoreChars และ MoreDigits เป็นแฟ้มข้อความ
กระบวนงาน Assign Assign (MoreDigits, 'A : DICITS.DAT') ;	ตัวแปรแฟ้ม MoreDigits เกี่ยวข้องกับแฟ้มดิสก์ DIGITS.DAT ซึ่งอยู่ในหน่วยขับ A
กระบวนงาน Reset และ Rewrite Reset (MoreDigits) ; Rewrite (MoreChars) ;	MoreDigits ถูกเตรียมเป็นอินพุต และ MoreChars ถูกเตรียมเป็นเอาต์พุต

Construct	Effect
กระบวนงาน Read และ Write	
Read (MoreDigits, I) ;	จำนวนเต็มถัดไปถูกอ่านจากแฟ้ม MoreDigits ไว้ในตัวแปร I (ชนิด Integer) สายอักขระ number : ถูกเขียนที่ MoreChars และตามด้วยค่าของ I
WriteLn (MoreChars, 'number : ', I) ;	
ฟังก์ชัน EOF	
Reset (MoreDigits) ;	แฟ้ม MoreDigits ถูกเตรียมสำหรับอินพุต ของโปรแกรม และแฟ้ม MoreChars สำหรับเอาต์พุต จากโปรแกรม ค่าจำนวนเต็มตัวแรกบนแต่ละบรรทัดของแฟ้ม MoreDigits ถูกเขียนบนบรรทัดแยกจากกันของแฟ้ม MoreChars
Rewrite (MoreChars) ;	
while not EOF (MoreDigits) do	
begin	
ReadLn (MoreDigits, I) ;	
WriteLn (MoreChars, I)	
end ; {while}	
ฟังก์ชัน EOLN	
Reset (MoreDigits) ;	แฟ้ม MoreDigits ถูกเตรียมสำหรับอินพุต อักขระแต่ละตัวบนบรรทัดแรกถูกอ่านไปยัง NextCh และแสดงผลบนจอภาพ ReadLn เลื่อนเครื่องหมายชี้ตำแหน่งแฟ้มสำหรับ MoreDigitsไปยังอักขระตัวแรกของบรรทัดที่สอง
while not EOLN (MoreDigits) do	
begin	
Read (MoreDigits, NextCh) ;	
Write (Output, NextCh)	
end ; {while}	
ReadLn (MoreDigits)	
กระบวนงาน Close	
Close (MoreChars) ;	แฟ้ม MoreChars ถูกปิด

แบบฝึกหัด Quick – Cheek

1. การดำเนินการ_ _ _ _ _ _ _ _ _ _ _ _ _ _ เกี่ยวข้องกับแฟ้มดิสก์ด้วยตัวแปรแฟ้ม แต่_ _ _ _ _ _ _ _ _ _ _ _ หรือ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ต้องกระทำก่อนอ่านหรือเขียนแฟ้ม

2. ตัวอักขระ_ _ _ _ _ _ _ _ _ _ _ ใช้ดัน _ _ _ _ _ _ _ _ _ _ _ แฟ้มให้เป็นบรรทัด และอักขระ_ _ _ _ _ _ _ _ _ _ _ _ _ปรากฏตอนท้ายของแฟ้ม

3. แบบชนิดข้อมูลประเภทใดซึ่งสามารถอ่าน หรือเขียนจากแฟ้มข้อความ

4. แฟ้มสามารถส่งเป็น พารามิเตอร์ค่าไปยังกระบวนงานได้เสมอหรือไม่

5. จริงหรือเท็จ : ชื่อของตัวแปรชนิด Text ต้องเหมือนกับชื่อของแฟ้มชนิดสมนัยกัน

6. แฟ้มข้อความสามารถใช้เป็นทั้งอินพุต และเอาต์พุตในโปรแกรมเดียวกันได้หรือไม่

7. จงแก้ไขที่ผิดในโปรแกรมข้างล่างนี้

```
Reset (Number) ;
While not EOF do
   Read (InFile, Number) ;
```

คำถามทบทวน (Review Questions)

1. จงบอกข้อดีสามข้อของการใช้แฟ้มสำหรับอินพุต และเอาต์พุตแทนที่อินพุตและเอาต์พุตเชิงโต้ตอบ

2. a) จงอธิบายความแตกต่างระหว่าง ชื่อภายในและชื่อภายนอกของแฟ้ม
   b) ในการเลือกชื่อแต่ละชื่อมีข้อตกลงอะไรบ้าง
   c) ชื่อใดที่ปรากฏ ในการประกาศตัวแปรแฟ้ม
   d) ชื่อใดที่ปรากฏในคำสั่งงานของระบบปฏิบัติการ

3. จงอธิบายว่า Read และ ReadLn แตกต่างกันอย่างไร ในการอ่านหน่วยข้อมูลจากแฟ้มข้อความ

4. ให้ X เป็นชนิด Real, N เป็นชนิด Integer และ Ch เป็นชนิด Char จงบอก contents ของตัวแปรแต่ละตัว หลังจากการดำเนินการอินพุตแต่ละชุดถูกกระทำ สมมติว่าแฟ้มประกอบด้วยบรรทัดข้างล่างนี้ และการดำเนินการ Reset (InData) เกิดขึ้นก่อน สำดับแต่ละชุดของข้อความสั่ง

23 53.2 ABC <eoln>

145 Z <eoln> <eof>

a) Read (InData, N, X) ;
   ReadLn (InData, Ch)

b) ReadLn (InData, Ch, N)

c) ReadLn (InData) ;
   Read (InData, X, Ch)

5. จงเขียนลูป ซึ่งอ่านค่าจำนวนเต็ม 10 ค่าจากแฟ้มข้อมูลและแสดงผลบนจอภาพ ถ้ามีจำนวนเต็มไม่ถึง 10 ค่าในแฟ้มให้แสดงข้อความ That's all folks หลังจากเลขตัวสุดท้าย

6. จงเขียนกระบวนงาน ซึ่งทำสำเนาบรรทัดข้อมูลหลายบรรทัดที่คีย์บอร์ดไปยังแฟ้มข้อความ กระบวนการทำสำเนา จะหยุดเมื่อผู้ใช้ใส่บรรทัดว่าง

7. จงเขียนกระบวนงาน ซึ่งจะเปิดแฟ้มชื่อ MANY.TXT, อ่านอักขระตัวแรกจากแฟ้มและเขียนที่แฟ้มใหม่ชื่อ FIRST.TXT กระบวนงาน ไม่ควรเขียนแฟ้มที่สอง ถ้าแฟ้มแรกเป็นแฟ้มว่าง

เขียนโปรแกรม

1. จงเขียนกระบวนงานอ่านข้อมูล ของพนักงานหนึ่งคนจากแฟ้ม OutEmp ซึ่งได้จากโปรแกรม WritePayroll (ดูหัวข้อ 8.3) และเขียนเช็คเงินเดือน (payroll check) ไปที่แฟ้มเอาต์พุต

รูปแบบของเช็คควรเป็นดังนี้

Temple University Check No. 12372
Philadelphia. PA Date : 03-17-94
Pay to the
Order of : Peter Liacouras $ 5000.00
Jane Smith
Bursar

2. จงเขียนโปรแกรมอ่านหมายเลขเช็คเริ่มต้นและข้อมูลจากคีย์บอร์ด จากนั้นเขียนเช็คโดยใช้กระบวนงานจากแบบฝึกหัดเขียนโปรแกรมข้อ 1 และแฟ้มข้อมูล ซึ่งเกิดจากการวิ่งโปรแกรม WritePayroll และเขียนอีกบรรทัดหนึ่งประกอบด้วยอักขระขีดเส้นได้ 80 ตัวระหว่างเช็คแต่ละใบ

3. จงคำนวณ การชำระเงินกู้รายเดือน และเงินชำระคืนทั้งหมดสำหรับเงินให้กู้ของธนาคารแห่งหนึ่ง โดยกำหนดสิ่งต่อไปนี้

a) จำนวนเงินให้กู้ (the amount of the loan)

b) ระยะเวลาของการกู้เงินเป็นเดือน (the duration of loan in months)

c) อัตราดอกเบี้ยสำหรับเงินกู้ยืม

โปรแกรมที่เขียนควรอ่านเงินให้กู้ครั้งละหนึ่งราย กระทำการคำนวณที่ต้องการ และพิมพ์เงินกู้ยืมที่ต้องชำระคืนรายเดือน และจำนวนเงินชำระคืนทั้งหมด

ทดสอบโปรแกรมด้วยข้อมูลข้างล่างนี้ เป็นอย่างน้อย (และมากกว่านี้ได้ถ้าต้องการ)

Loan	Months	Rate
16000	300	12.50
24000	360	13.50
30000	300	15.50
42000	360	14.50
22000	300	15.50
300000	240	15.25

ข้อแนะนำ (Hints)

a) สูตรสำหรับคำนวณเงินกู้ชำระคืนรายเดือน

$$\text{monthpay} = \frac{\text{ratem} \times \text{expm}^{\text{months}} \times \text{loan}}{\text{expm}^{\text{months}} - 1.0}$$

เมื่อ

$$\text{ratem} = \frac{\text{rate}}{1200.00}$$

$$\text{expm} = (1.0 + \text{ratem})$$

เราจำเป็นต้องใช้ลูป เพื่อคูณ expm ด้วยตัวมันเอง months ครั้ง

b) สูตรสำหรับคำนวณ เงินกู้ชำระคืนทั้งหมด

total + monthpay x months

4. จงใช้ผลเฉลยของโปรแกรม ข้อ 3 เป็นฐานสำหรับเขียนโปรแกรมซึ่งจะเขียนแฟ้มข้อมูล เป็นตารางของรูปแบบข้างล่างนี้ :

Loan Amount : $ 1000

Interest Rate	Duration (years)	Monthly Payment	Total Payment
6.00	20	–	–
6.00	25	–	–
6.00	30	–	–
6.25	20	–	–

แฟ้มเอาต์พุตของโปรแกรม ประกอบด้วย สารสนเทศการชำระเงินกู้จำนวน $1000 อัตรดอกเบี้ยจาก 6% จนถึง 10% ซึ่งการเพิ่มเป็น 0.25% ระยะเวลากู้ยืมเงินเป็น 20, 25 และ 30 ปี

5. Whatsamata U มีบริการให้กับนักศึกษา ในการคำนวณเกรดที่ตอนจบแต่ละภาคการศึกษา (semester) โปรแกรมนี้ประมวลผลคะแนนสอบซึ่งมีการถ่วงน้ำหนัก (weighted test scores) สามชุด และคำนวณคะแนนเฉลี่ยของนักศึกษาและเกรดเป็นตัวอักษร (A คือ คะแนน 90-100, B คือคะแนน 80-89 เป็นต้น) จงอ่านข้อมูลของนักศึกษาจากแฟ้มและเขียนชื่อของนักศึกษาแต่ละคน คะแนนสอบ และเกรดไว้ที่แฟ้มเอาต์พุต

จงเขียนโปรแกรม เพื่อจัดบริการข้างต้นนี้ ข้อมูลจะประกอบด้วยการถ่วงน้ำหนักการทดสอบสามชุด ตามด้วย คะแนนทดสอบและ เลขประจำตัวของนักศึกษา (เลขสี่หลัก) ของนักศึกษาแต่ละคน คำนวณคะแนนถ่วงน้ำหนักแต่ละคนและตัวอักษรเกรดที่สมนัยกัน สารสนเทศควรจะพิมพ์พร้อมกับคะแนนทดสอบสามครั้ง ค่าเฉลี่ยถ่วงน้ำหนักของนักศึกษาแต่ละคนเท่ากับ

(weight1, x score1) + (weight2 x score2) + (weight3 x score3)

สำหรับข้อสรุป สถิติ ให้พิมพ์ คะแนนเฉลี่ยสูงสุด คะแนนเฉลี่ยต่ำสุด คะแนนเฉลี่ยของคะแนนเฉลี่ยและจำนวนนักศึกษาทั้งหมด ที่ถูกประมวลผล

ข้อมูลตัวอย่าง :

0.35 0.25 0.4 testweights

00 76 88 1014 testscores and ID

6. จงเขียนโปรแกรมอ่าน สายของตัวอักขระ ซึ่งแทนเลขโรมัน (Roman numeral) จากนั้นเปลี่ยนให้เป็นรูปแบบ Arabic (จำนวนเต็ม)

ค่าของตัวอักขระสำหรับเลขโรมัน เป็นดังนี้

M	1000
D	500
C	100
L	50
X	10
V	5
I	1

จงทดสอบโปรแกรมด้วยข้อมูล ต่อไปนี้ :

IX (9), LXXXVII (87), CCXIX (219)

M CCCLIV (1354), MMDCLXXIII (2673)

MCDLXXVI (1476)