# บทที่ 8
# ภาษาคอมพิวเตอร์
# (COMPUTER LANGUAGES)

ภาษาโปรแกรม แบ่งออกเป็น 5 ระดับ ได้แก่ ภาษาเครื่อง ภาษาแอสเซมบลี ภาษาระดับสูง ภาษาระดับสูงมาก และภาษาธรรมชาติ เนื้อหาในบทนี้ จะอภิปราย เกี่ยวกับ ประวัติของภาษาเหล่านี้ กล่าวถึงภาษาโปรแกรมซึ่งนิยมใช้อย่างแพร่หลาย 6 ภาษา ได้แก่ FORTRAN, COBOL, BASIC, Pascal, Ada และภาษา C จะชี้ให้เห็น คุณสมบัติที่สำคัญ ตัวอย่างของภาษา นอกจากนี้แล้ว ภาษาที่สำคัญอื่นๆ จะอธิบายโดยย่อ

**ภาษาโปรแกรม** (Programming Languages)

ภาษาโปรแกรม หมายถึง เครื่องมือที่ใช้สื่อสารกับเครื่องคอมพิวเตอร์ แต่ละภาษามีคุณสมบัติที่ชัดเจน มีคำศัพท์ที่ใช้จำนวนจำกัด แต่ละคำ (word) มีความหมายถูกต้อง ถึงแม้ว่าภาษาโปรแกรมจะมีข้อจำกัดต่างๆ แต่ สามารถ ใช้ ในลักษณะทำทีละขั้นตอน เพื่อแก้ปัญหาที่ซับซ้อน ปัจจุบันนี้มีภาษาโปรแกรม มากกว่า 200 ภาษา และยังคงใช้กันอยู่จนทุกวันนี้ และมีอีกเป็น จำนวนหลายร้อยภาษา ซึ่งไม่ได้นำมาใช้เลยในหลายๆ ปีที่ผ่านมา บางภาษามีชื่อแปลกๆ เช่น INTELLECT, DOCTOR, UFO ภาษาเหล่านี้มาจากที่ไหน?

เริ่มแรก ภาษาโปรแกรม สร้างโดยบุคคล ในมหาวิทยาลัยต่างๆ หรือรัฐบาล เพื่อให้ทำหน้าที่ เฉพาะด้าน บางภาษามีอายุยืนนาน เพราะว่า สามารถสนองตอบความต้องการ ทางด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ และงานอื่นที่มีลักษณะ คล้ายกัน อย่างไรก็ตาม ในเวลาต่อมา เห็นได้ชัดเจนว่า สิ่งที่จำเป็นคือ ความเป็นมาตรฐาน ในแง่ที่ว่า สำหรับ การทำงาน ของงานที่มีลักษณะคล้ายกัน จะใช้ภาษาเดียวกัน

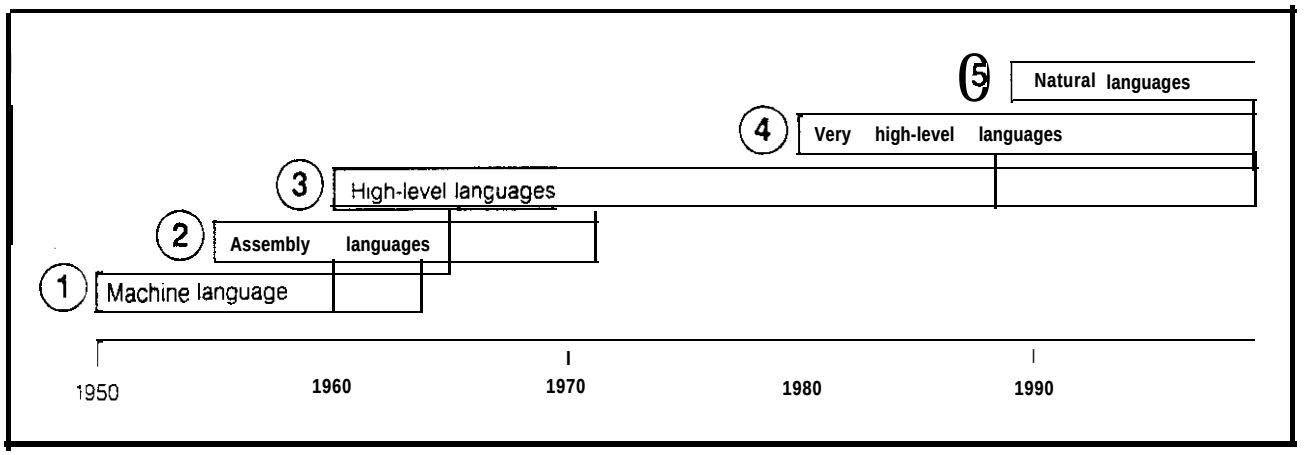
ทุกวันนี้ มีหลายภาษา ที่ใช้ร่วมกัน ซึ่งจะได้อภิปราย ภาษาซึ่งนิยมใช้กันมากที่สุด อย่างไรก็ตาม ก่อนที่จะกล่าวถึง ภาษาต่างๆ จำเป็นจะต้องอภิปรายถึง ระดับของภาษา ก่อน

**ระดับของภาษา** (Levels of Language)

ภาษาโปรแกรม จะเรียกว่า มีระดับ "ต่ำกว่า" หรือ "สูงกว่า" ขึ้นอยู่กับว่า ภาษานั้นๆ เข้าใกล้กับ ภาษาเครื่องคอมพิวเตอร์ แค่ไหน? ($0_S$ และ $1_S$ หมายถึงต่ำ) หรือ เข้าใกล้กับ ภาษาซึ่งมนุษย์ใช้สื่อสาร (คล้ายภาษาอังกฤษมากกว่า หมายถึง สูง) ภาษาโปรแกรมแบ่งออกเป็น 5 ระดับ (levels) หรือ 5 ยุค (generation) เรากำหนดเลข ให้เป็น 1 ถึง 5 ในเทอมของ ง่ายต่อการนำไปใช้ และ มีความสามารถสูงขึ้น แต่ละยุค หมายถึง การปรับปรุง ให้ดีขึ้น เหนือกว่าภาษายุคก่อนหน้า ทั้งห้ายุค ของ ภาษา ได้แก่

1. ภาษาเครื่อง (Machine language)
2. ภาษาแอสเซมบลี (Assembly languages)
3. ภาษาระดับสูง (High-level languages)
4. ภาษาระดับสูงมาก (Very high-level languages)
5. ภาษาธรรมชาติ (Natural languages)

โปรดสังเกต เส้นแสดงเวลาของยุคของภาษา ในรูป 8-1




รูป 8-1 ยุคของภาษาบนเส้นเวลา (Language generations on a time line)

ส่วนที่แรเงา แสดงถึง ช่วงเวลาของการใช้ที่มากกว่า โดยโปรแกรมเมอร์ และส่วนที่ไม่แรเงา แสดงถึง ช่วงเวลา ซึ่งเป็นยุคที่การใช้ลดลง หรือ คาดว่า จะลดลง (fade)

**ภาษาเครื่อง** (Machine Language)

มนุษย์ มักจะไม่ชอบทำงานกับสิ่งที่เป็นตัวเลขทั้งหมด เขาชอบตัวอักษร หรือคำ มากกว่า แต่ ภาษาเครื่อง เป็นตัวเลข ทั้งหมด ภาษาเครื่อง หรือ ภาษาโปรแกรมมีระดับต่ำที่สุด แทน สารสนเทศด้วย เลข 1 และ เลข 0 เท่านั้น เรียกว่า เลขฐานสอง ซึ่งสมนัยกับ สถานะทางไฟฟ้า "on" และ "off" ใน เครื่องคอมพิวเตอร์มากกว่า

ตัวอย่าง ภาษาเครื่อง ที่แสดงให้เห็นในรูป 8-2 เป็นภาษาเครื่อง ของ คอมพิวเตอร์ชนิด เมนเฟรม (main frame) ในช่วงยุคต้นๆ ของการคำนวณ คอมพิวเตอร์ แต่ละเครื่อง มีภาษาเครื่องของมันเอง และโปรแกรมเมอร์มี ระบบความรู้เบื้องต้น สำหรับจัดกลุ่มเลข เพื่อแทนคำสั่ง เช่น บวก หรือ เปรียบเทียบ เมื่อเปรียบเทียบกับปัจจุบันนี้ จะเห็นว่า ตัวโปรแกรมไม่สะดวก สำหรับให้คนอ่าน และใช้ อุตสาหกรรมคอมพิวเตอร์ จึงมีการพัฒนา ภาษาแอสเซมบลี ขึ้นมาใช้

```
FD 71 431F 4153
F3 63 4267 4321
96 FO 426D
F9 10 41F3 438A
47 40 40DA
47 FO 4050
```

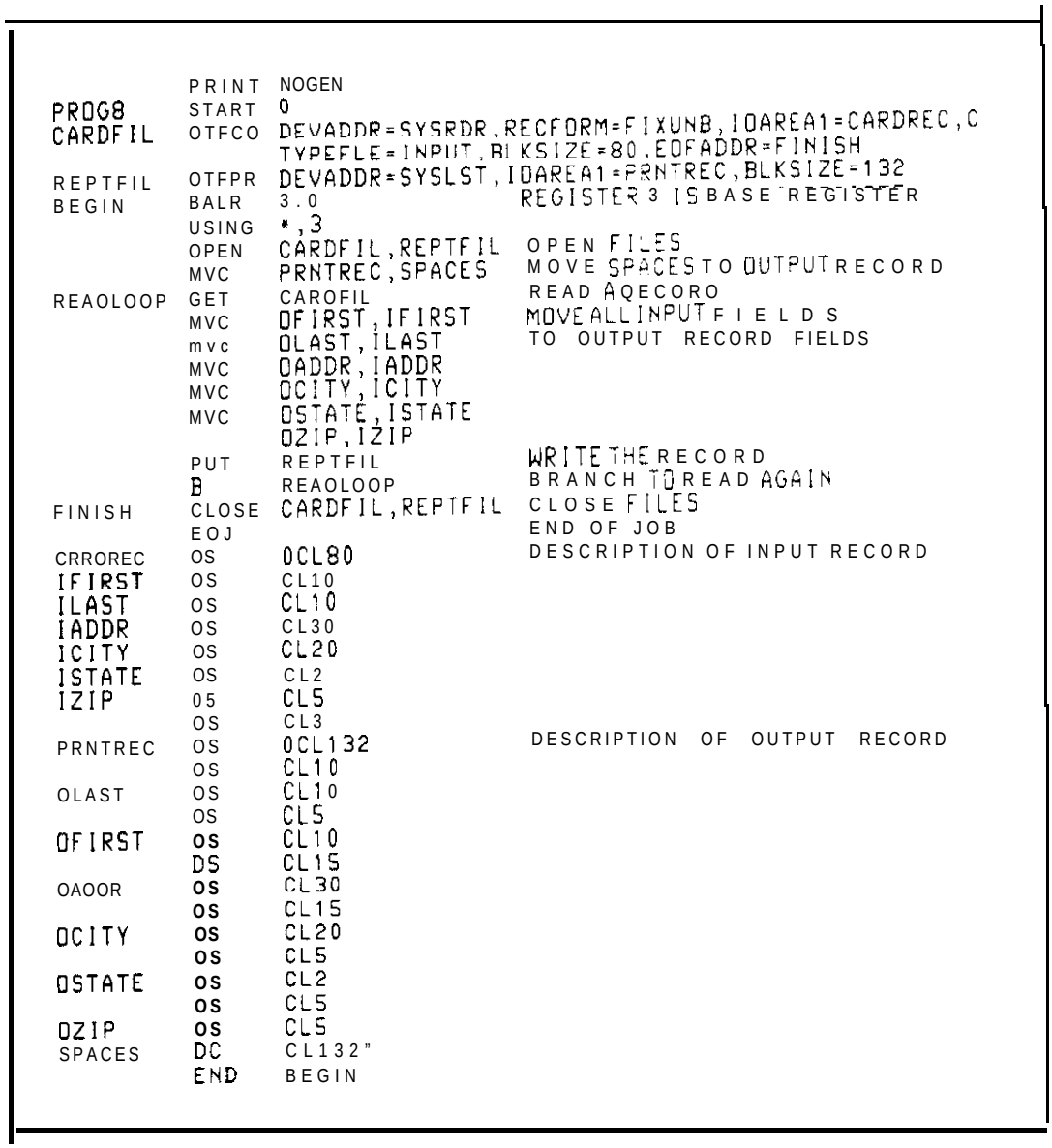
รูป 8-2 ภาษาเครื่อง (Machine Language)

ภาษาเครื่องจริงๆ แล้ว จะเป็นเลข ฐานสองทั้งหมด คือมีเฉพาะ $0_s$ และ $1_s$ เท่านั้น แต่ที่แสดงให้เห็นในตัวอย่างข้างต้น เป็นภาษาเครื่อง ในระบบเลขฐานสิบหก (อักษร A ถึง F ในเลขฐานสิบหก แทน เลข 10 ถึงเลข 15 ในระบบฐานสิบ) คำสั่งงานข้างต้น เป็นภาษาเครื่องของคอมพิวเตอร์ IBM360/370

**ภาษาแอสเซมบลี** (Assembly Languages)

ทุกวันนี้ ภาษาแอสเซมบลี ยังถูกจัดให้เป็นภาษาระดับต่ำมาก (very low-level) เพราะว่าภาษานี้ ไม่สะดวก สำหรับ การใช้ เหมือนเช่น ภาษาที่ใหม่กว่า อย่างไรก็ตาม ณ เวลาที่ ภาษานี้ถูกพัฒนานั้น ถือว่าเป็นการก้าวกระโดดอย่างมาก ไม่ใช่การใช้เลข $1_s$ และ $0_s$ อีกต่อไป

ภาษาแอสเซมบลี ใช้ตัวย่อ หรือรหัสช่วยจำ (mnemonic codes) เพื่อแทน เลข ตัวอย่างเช่น A สำหรับ Add, C สำหรับ Compare, MP สำหรับ Multiple เช่นนี้เป็นต้น ถึงแม้ว่า รหัสเหล่านี้ ไม่ใช้คำภาษาอังกฤษ และยังคงไม่สะดวกในการใช้ แต่ดีกว่าที่จะเป็นเลขทั้งหมด



```
         PRINT   NOGEN
PROG8    START   0
CARDFIL  OTFCO   DEVADDR=SYSRDR,RECFORM=FIXUNB,IOAREA1=CARDREC,C
                 TYPEFLE=INPUT,BLKSIZE=80,EOFADDR=FINISH
REPTFIL  OTFPR   DEVADDR=SYSLST,IOAREA1=PRNTREC,BLKSIZE=132
BEGIN    BALR    3,0                 REGISTER 3 IS BASE REGISTER
         USING   *,3
         OPEN    CARDFIL,REPTFIL     OPEN FILES
         MVC     PRNTREC,SPACES      MOVE SPACES TO OUTPUT RECORD
READLOOP GET     CARDFIL             READ A RECORD
         MVC     OFIRST,IFIRST       MOVE ALL INPUT FIELDS
         MVC     OLAST,ILAST         TO OUTPUT RECORD FIELDS
         MVC     OADDR,IADDR
         MVC     OCITY,ICITY
         MVC     OSTATE,ISTATE
                 OZIP,IZIP
         PUT     REPTFIL             WRITE THE RECORD
         B       READLOOP            BRANCH TO READ AGAIN
FINISH   CLOSE   CARDFIL,REPTFIL     CLOSE FILES
         EOJ                         END OF JOB
CRROREC  DS      0CL80               DESCRIPTION OF INPUT RECORD
IFIRST   DS      CL10
ILAST    DS      CL10
IADDR    DS      CL30
ICITY    DS      CL20
ISTATE   DS      CL2
IZIP     DS      CL5
         DS      CL3
PRNTREC  DS      0CL132              DESCRIPTION OF OUTPUT RECORD
         DS      CL10
OLAST    DS      CL10
         DS      CL5
OFIRST   DS      CL10
         DS      CL15
OADDR    DS      CL30
         DS      CL15
OCITY    DS      CL20
         DS      CL5
OSTATE   DS      CL2
         DS      CL5
OZIP     DS      CL5
SPACES   DC      CL132' '
         END     BEGIN
```

**รูป 8-3 ภาษาแอสเซมบลี ของ IBM**

โปรแกรมนี้ อ่าน ข้อมูลหนึ่งระเบียน แล้วพิมพ์ คอลัมน์ ซ้ายมือสุด เป็น เลขที่อยู่สัญลักษณ์ (symbolic addresses) ของ คำสั่ง หรือ ข้อมูล คอลัมน์ ที่สองเป็น รหัสดำเนินการจริง (actual operation codes) บอก ชนิดของกิจกรรมที่ต้องการทำ ตัวอย่างเช่น MVC หมายถึง "move characters" คอลัมน์ที่สาม อธิบาย ข้อมูล ซึ่ง คำสั่ง ให้กระทำ คอลัมน์ ขวามือสุด เป็น คำอธิบาย (comments) สัมพันธ์ กับ บรรทัด ตรงกันข้าม ตัวอย่างข้างต้นนี้ เมื่อเขียนด้วยภาษาระดับสูง จะลดลงเหลือเพียงสอง หรือสามบรรทัด เท่านั้น

โปรแกรมเมอร์ ซึ่งใช้ ภาษาแอสเซมบลี จะต้อง **ใช้ตัวแปลภาษา** (translator) เพื่อ แปลงผัน (convert) โปรแกรมภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง ตัวแปลภาษา เป็นสิ่งจำเป็นต้องใช้ เพราะว่าคอมพิวเตอร์ สามารถ กระทำการ (execute) ได้จริงเฉพาะ ภาษาเครื่องเท่านั้น ตัวแปลภาษา ในที่นี้คือ โปรแกรม แอสเซมเบลอ (assembler program) หรือ เรียกว่า แอสเซมเบลอ โปรแกรมนี้ จะแปล โปรแกรม ที่เขียนด้วยภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง โปรแกรมเมอร์ไม่จำเป็นต้องวิตกกังวล เกี่ยวกับ การแปลแต่อย่างใด เพียงแต่ ให้เขียนเฉพาะโปรแกรม ด้วยภาษาแอสเซมบลี จากนั้น การแปล จะถูกกระทำด้วย แอสเซมเบลอ

ถึงแม้ว่า ภาษาแอสเซมบลี เป็น ขั้นตอนของการก้าวไปข้างหน้า แต่ ภาษานี้ ยังคงมี ข้อไม่ดี อยู่มาก เช่น ภาษาแอสเซมบลี มี รายละเอียดมาก มีการเขียนซ้ำมาก น่าเบื่อ และข้อผิดพลาด เกิดขึ้นง่าย ให้ดูรูป 8-3 จะเห็นว่า ภาษาแอสเมบลี อ่านง่ายกว่าภาษาเครื่อง แต่ไม่ได้หมายความว่าอย่างชัดเจน

**ภาษาระดับสูง** (High-Level Languages)

ในช่วงต้นของ ค.ศ. $1960_s$ เป็นครั้งแรก ที่มีการใช้ภาษาระดับสูงอย่างกว้างขวาง เปลี่ยนแปลง การเขียนโปรแกรม ไปเป็น บางสิ่ง ซึ่งแตกต่างอย่างมาก จากสิ่งที่เคยเห็น โปรแกรมเมอร์ ซึ่งเคยทำงานกับ รายละเอียด ของ การลงรหัส และเครื่องคอมพิวเตอร์ กลายมาเป็น โปรแกรมเมอร์ ซึ่ง ให้ความสนใจกับ การแก้ปัญหาของลูกค้า มากกว่า โปรแกรม สามารถ แก้ปัญหาซึ่งซับซ้อน มากกว่า ได้มากขึ้น ในเวลาเดียวกัน โปรแกรมเขียน ในลักษณะคล้ายภาษาอังกฤษ ซึ่งทำให้ การใช้สะดวกมากขึ้น ผลลัพธ์จากการเปลี่ยนแปลงเหล่านี้ ทำให้ โปรแกรมเมอร์ สามารถทำงานประสบผลสำเร็จมากขึ้น ด้วยความพยายามที่น้อยกว่าเดิม

ภาษายุคที่สาม เกิดเพิ่มขึ้นอย่างมาก ในการประมวลผลข้อมูล ช่วง ค.ศ. $1960_s$ และ $1970_s$ ระหว่างเวลานั้น มีการใช้เครื่องคอมพิวเตอร์ ชนิด mainframe เพิ่มขึ้น จาก จำนวนร้อยๆ เครื่อง เป็น จำนวน หมื่นๆ เครื่อง ผลกระทบของภาษายุคที่สาม ($3GL_s$) เกิดขึ้นบน สังคมของ

เรา มากมาย

แน่นอน ตัวแปลภาษา จำเป็นต้องใช้ แปล ข้อความสั่งเชิงสัญลักษณ์ (symbolic statements) ของภาษาระดับสูง ให้เป็น ภาษาเครื่อง ซึ่ง คอมพิวเตอร์ กระทำการได้ ตัวแปลภาษานี้ ปกติคือ **คอมไพเลอร์** (compiler) หรือเรียกว่า ตัวแปลโปรแกรม

ภาษาโปรแกรมแต่ละภาษาจะมีคอมไพเลอร์ หลายตัว แต่ละตัว สำหรับเครื่องคอมพิวเตอร์ แต่ละชนิด (There are many compilers for each language and one for each type of computer.) **ตัวอย่างเช่น** ภาษาเครื่อง ซึ่งก่อกำเนิดจาก คอมไพเลอร์ COBOL ของคอมพิวเตอร์เครื่องหนึ่ง จะไม่ใช่ ภาษาเครื่อง ของ คอมพิวเตอร์เครื่องอื่น ดังนั้น จึงจำเป็นต้องมี คอมไพเลอร์ COBOL สำหรับ คอมพิวเตอร์ แต่ละชนิด ซึ่ง จะนำโปรแกรม COBOL ไปวิ่ง (run)

หลายภาษา สร้างขึ้นมาเพื่อวัตถุประสงค์ ในการใช้งานเฉพาะด้าน (specific purposes) เช่น การควบคุม หุ่นยนต์ในโรงงานอุตสาหกรรม หรือ การสร้างงานกราฟฟิก (graphics) อย่างไรก็ตาม มีภาษาโปรแกรมจำนวนมาก ซึ่ง ยืดหยุ่น เป็นพิเศษ และใช้งานได้ทั่วไป (general-purpose) ในอดีตนั้น งานเขียนโปรแกรมที่สำคัญ เขียนด้วยภาษา BASIC, FORTRAN หรือ COBOL ซึ่งทั้งหมดนี้ เป็น ภาษา ใช้งานทั่วไป นอกจากสามภาษานี้แล้ว ภาษาระดับสูงอื่นๆ ซึ่งเป็นที่นิยมใช้กันในทุกวันนี้ ได้แก่ Pascal, Ada และ C

โปรดสังเกตว่า ภาษาระดับสูง ทำให้ โปรแกรมเมอร์ หมดภาระ จากรายละเอียดของฮาร์ดแวร์ อย่างไรก็ตาม ข้อดีเหล่านี้ ทำให้ สิ่งที่เป็นความยืดหยุ่นขาดหายไป ภาษาระดับสูงจำนวนน้อย เช่น C และ FORTH มีทั้งความยืดหยุ่น ของ ภาษาแอสเซมบลี รวมทั้ง ข้อดี (power) ของภาษาระดับสูง แต่ภาษาเหล่านี้ อาจจะไม่เหมาะสม กับ โปรแกรมเมอร์หัดใหม่ (beginning programmer)

**ภาษาระดับสูงมาก** (Very High-Level Languages)

ภาษาระดับสูงมาก หรือ เรียกว่า ภาษายุคที่สี่ (fourth generation languages)

**บทนิยาม** (definition)

ยังไม่มีการ รวบรวมข้อคิดเห็น เกี่ยวกับอะไรทำให้ เกิดภาษายุคที่สี่ ภาษายุคนี้ สิ่งที่สำคัญคือ เป็นภาษาเขียนโปรแกรมแบบลัด (shorthand programming languages) การดำเนินการ ซึ่งต้องเขียน เป็น หลายร้อยบรรทัด ใน ภาษายุคที่สาม เช่น COBOL ปกติแล้ว เมื่อเขียนด้วย ภาษายุคที่สี่ เขียนเพียง ห้าถึงสิบ บรรทัดเท่านั้น อย่างไรก็ตาม ภายใต้ กฎเกณฑ์พื้นฐาน ของ ความรวบรัด (conciseness) ภาษายุคที่สี่ ยากในการอธิบาย

**คุณสมบัติ** (Characteristics)

ภาษายุคที่สี่ มี คุณสมบัติ ร่วมบางอย่าง สิ่งแรกคือ มักแยกตัวออกจากยุคก่อนหน้านั้น โดยพื้นฐานแล้ว ภาษายุคที่สี่ เป็น **ภาษาไร้กระบวนคำสั่ง** (nonprocedural language)

**ภาษาเชิงกระบวนคำสั่ง** (Procedural language) ต้องบอกคอมพิวเตอร์ว่า จะทำงานนั้นอย่างไร (how a task is done) ตัวอย่างเช่น ให้บวก ให้เปรียบเทียบสิ่งนั้น ทำสิ่งนี้ ถ้าเงื่อนไขเป็นจริง และอื่นๆ กรรมวิธีทำทีละขั้นตอน ชัดเจนมาก ภาษาโปรแกรม สามยุคแรก ทั้งหมดเป็นภาษาเชิงกระบวนคำสั่ง (The first three generations of languages are all procedural.)

**ภาษาไร้กระบวนคำสั่ง** (Nonprocedural language) มีแนวคิดเปลี่ยนแปลงไป ในที่นี้ ผู้ใช้นิยามเพียงแค่ ต้องการให้คอมพิวเตอร์ ทำอะไร (What they want the computer to do?) ผู้ใช้ไม่ต้องให้รายละเอียดว่า สิ่งนั้นจะทำได้อย่างไร สิ่งนี้นำไปสู่ หัวข้อ ของ ผลิตผล (productivity) ซึ่งเป็น คุณสมบัติที่สำคัญ ของ ภาษายุคที่สี่

**ผลิตผล** (Productivity)

ภาษายุคที่สี่ สามารถปรับปรุงผลิตผล ได้มาก 5 ถึง 50 เท่า คำกล่าวนี้เป็นจริง ผู้เชี่ยวชาญส่วนใหญ่ พูดว่า ปัจจัยการปรับปรุงโดยเฉลี่ย ประมาณ 10 เท่า นั่นคือ ภาษายุคที่สี่ ให้ผลิตผลเป็น สิบเท่า มากกว่า ผลิตผลที่เขียนด้วยภาษายุคที่สาม

**ตัวอย่าง** จงพิจารณา request ต่อไปนี้ "ให้สร้าง รายงาน แสดง หน่วยทั้งหมดของ การขายสินค้าแต่ละชิ้น จำแนกตามลูกค้า ในแต่ละเดือน และปี และให้มียอดรวมย่อย ของ ลูกค้าแต่ละคน นอกจากนี้แล้ว ลูกค้าใหม่แต่ละคน ให้ขึ้นต้น กระดาษแผ่นใหม่" ภาษายุคที่สี่เขียนดังนี้

```
TABLE FILE SALES
SUM UNITS BY MONTH BY CUSTOMER BY PRODUCT
ON CUSTOMER SUBTOTAL PAGE BREAK
END
```

ถึงแม้ว่า การฝึกอบรม จะต้องใช้เวลา แต่จะเห็นว่าเขียนง่ายมาก อย่างไรก็ตาม ถ้าเป็นภาษายุคที่สาม เช่น COBOL งานอย่างเดียวกันนี้ จะต้องเขียนมากกว่า 500 บรรทัด

ถ้าเรานิยาม ผลิตผล คือ การให้ผลลัพธ์ในเวลาที่น้อยกว่า ภาษายุคที่สี่ จะให้ ผลิตผลมากกว่า

**ข้อไม่ดีของ** $4GL_S$ (The Downside of $4GL_S$)

ภาษายุคที่สี่ ไม่ใช่ จะดีทั้งหมด ภาษายุคที่สี่ยังคงมีวิวัฒนาการ และการวิวัฒนาการนั้นยัง ไม่เป็น มาตรฐานหรือนิยาม อย่างบริบูรณ์ สิ่งที่เหนือกว่าคือ $4GL_S$ ส่วนใหญ่ ใช้ง่าย ภาษา

เหล่านี้จึง ดึงดูด ผู้สนใจใหม่ได้ จำนวนมากมาย ผู้ซึ่ง หลังจากนั้น บุคคลเหล่านี้ จะเข้ามาในระบบคอมพิวเตอร์ มากเกินไป การยอมรับร่วมของ $4GL_S$ คือ ภาษาเหล่านี้ ไม่ได้ใช้ทรัพยากรคอมพิวเตอร์ อย่างมีประสิทธิภาพ อย่างไรก็ตาม ประโยชน์ของการที่งานสำเร็จ เร็วขึ้น สามารถทำให้ลดค่าใช้จ่ายของการวิ่งโปรแกรม

**ข้อดีของ $4GL_S$** ($4GL_S$ Benefits)

ภาษายุคที่ 4 มี ข้อดี ดังนี้

- เป็นภาษาเชิงผลลัพธ์ คือ เน้นอะไร ไม่ใช่ ทำอย่างไร

  (They are results - oriented; they emphasize **what** instead of **how**.)
- เป็นภาษา ซึ่งทำให้ ผลิตผลดีขึ้น เพราะว่า เขียนโปรแกรมง่าย และเปลี่ยนแปลงง่าย

  (They improve productivity because program are easy to write and change.)
- ใช้เวลา ในการอบรมน้อยที่สุด ทั้ง โปรแกรมเมอร์ และ ไม่ใช่โปรแกรมเมอร์

  (They can be used with minimum of training by both programmers and nonprogrammers.)
- เป็นโล่กำบังของผู้ใช้ จากความจำเป็นต้อง รับรู้ เกี่ยวกับ ฮาร์ดแวร์ และ ตรรกะของโปรแกรม

  (They shield users from needing and awareneess of hardware and program logic.)

ในไม่ช้า ราวช่วงท้าย ค.ศ. $1970_S$ มีผู้คนจำนวนหนึ่ง เชื่อว่า ในช่วง ค.ศ. $1980_S$ นั้น $4GL_S$ จะมาแทนที่ ภาษายุคที่สาม ขณะนี้การเปลี่ยนแปลง เห็นชัดขึ้น รูป 8-4 แสดงตัวอย่าง $4GL_S$ ชื่อ Focus

ตัวอย่าง รหัสภาษา Focus

```
TABLE FILE ENROLL
HEADING CENTER
"RAMKHAMHAENG UNIVERSITY"
"SEMEESTER> <YR>          <CNAME> <CNUM>"
"</2>"
PRINT LNAMEE AND FNAME
BY CNUM NOPRINT PAGE-BREAK
BY STDN
IF SEMESTER EQ "SECOND"
IF YR EEQ 96
END
```

เอ้าพุท เป็นดังนี้

RAMKHAMHAENG UNIVERSITY

SECOND 96 INTRODUCTION TO COMPUTER SYSTEM IT 10

| STDN | LNAME | FNAME |
|---|---|---|
| 39310073 | AAAAA | MALEE |
| 39310105 | BBBCC | APISIT |
| 39400672 | CCDDE | SOMSRI |
| 39400825 | DDDDD | NARONGSAK |
| 39607463 | FFDDD | RATTANA |

รูป 8-4 ตัวอย่างภาษายุคที่สี่ ชื่อ FOCUS และเอ้าพุท

**ภาษาธรรมชาติ** (Natural Languages)

ภาษาธรรมชาติ หรือ ภาษายุคที่ห้า คล้ายกับ ภาษาพูดตามธรรมชาติ คำว่า ธรรมชาติ หมายถึง เหมือนมนุษย์ (natural means human-like) คือแทนที่จะเป็น การบังคับ ให้ใส่คำสั่งงานถูกต้อง และ ชื่อข้อมูล เรียงลำดับถูกต้อง ผู้ใช้ เพียงบอกคอมพิวเตอร์ว่า ให้ทำอะไร โดย พิมพ์คำพูดของตนเอง

ภาษาธรรมชาติ บางครั้ง อ้างถึง ภาษาฐานความรู้ (knowledge-based languages) เพราะว่าภาษาธรรมชาติ ใช้ โต้ตอบกับ ฐานของความรู้บนหัวเรื่องบางอย่าง การใช้ภาษาธรรมชาติเพื่อเข้าถึงฐานความรู้ เรียกว่า **ระบบฐานความรู้** (knowledge-based systems) **ส่วนระบบผู้เชี่ยวชาญ** (expert systems) เป็นระบบฐานความรู้ชนิดหนึ่ง ระบบผู้เชี่ยวชาญ ทำให้คอมพิวเตอร์เป็นเช่น ผู้เชี่ยวชาญ บน สาขาใดสาขาหนึ่ง ผู้เชี่ยวชาญของระบบ จะมีความหมายเหมือนกับผู้เชี่ยวชาญซึ่งเป็นมนุษย์ สามารถสอบถาม ตอบคำถาม ในลักษณะคล้ายกัน

การใช้ภาษาธรรมชาติ เพื่อเข้าถึง ฐานความรู้ เป็นพื้นฐานของ งานทางด้านปัญญาประดิษฐ์ (artificial intelligence)

**ตัวอย่าง** คำร้องขอ (request) ซึ่งกำหนด ใน ภาษาชุดที่สี่ ชื่อ FOCUS :

"SUM ORDERS BY DATE BY REGION."

ถ้าเราเปลี่ยน คำร้องขอ แต่ ยังคงเป็น Focus เช่น

"Give me the dates and the regions after you've added up the orders."

เครื่องคอมพิวเตอร์ จะตอบกลับ ในลักษณะเป็นเพื่อน ดังนี้

"You've got to be kidding."

และยกเลิก แต่ภาษาธรรมชาติ บางชุด สามารถจัดกระทำ (handle) คำร้องขอเช่นนี้ได้ ผู้ใช้สามารถ ผ่อนคลาย โครงสร้างของคำร้องขอ และเพิ่มอิสระ ของการโต้ตอบ กับ ข้อมูล

**ตัวอย่าง** ภาษาธรรมชาติ โปรแกรมสำเร็จชุดนี้ เรียกว่า ระบบจัดการเงินสด (Cash Management System)

Hello

How may I help you?

Who are my customers in Chicago?

Just a sec. I'll see.

The customers in that city are :

| I. D. | Name |
|---|---|
| Ballard | Ballard and Sons, Inc. |
| Fremont | Henry Fremont Associatives |
| Greenlake | Greenlake Consortium |
| Wallingford | Wallingford, Inc. |

What can I do for you now?

What is Fremont's balance?

Hang on. I'll see.

| | |
|---|---|
| Accounts Receivable | 563.47 |
| Unapplied Credit | 79.16 |
| Balance | 484.31 |

What else can I do for you?

Give me Fremont's phone number!

Please wait while I check the files.

(312) 789-5562

What can I do for you now?

รูป 8-5 ภาษาธรรมชาติ

ตัวอย่าง ภาษาธรรมชาติ ที่แสดงในรูป 8-5 นี้ เข้าถึงข้อมูลได้ง่าย งานประยุกต์ร่วม (common application) ส่วนใหญ่ สำหรับ ภาษาธรรมชาติ คือ การโต้ตอบกับฐานข้อมูล ดังนั้น ถ้าเราจำกัดความต้องการ เพียง ข้อสอบถามข้อมูล คอมพิวเตอร์ เข้าถึง สื่อต่างๆ คล้ายกับทำด้วยมือ อย่างไรก็ตาม ภาษายุคที่ห้า ยังไม่พร้อมที่จะจัดกระทำ กับ ตรรกะซึ่งซับซ้อน ดังนั้น

ภาษานี้ จึงดูไม่เหมือนกับ เป็นเครื่องมือ สำหรับ โปรแกรมเมอร์อาชีพในอนาคตอันใกล้

**การเลือกใช้ภาษา** (Choosing a Language)

เราเลือกภาษา สำหรับ เขียน โปรแกรมอย่างไร? บางที เราใช้ ภาษานั้น เพราะว่า มัน เป็นภาษาเดียวเท่านั้นที่มีให้ใช้ ณ เวลาติดตั้ง บางครั้ง ผู้จัดการ สั่งให้ พนักงาน ในโครงการของ เขา ใช้ภาษาโปรแกรม ซึ่งกำหนดให้แล้ว บางครั้ง อาจเป็นเพราะ เรามีความรู้ เพียงภาษาเดียว

การเข้าถึงซึ่ง รับรู้ไว คือ เลือกภาษา ซึ่ง เหมาะสมมากที่สุด สำหรับ งานเขียนโปรแกรม เฉพาะของเรา หัวข้อต่อไปนี้ จะกล่าวถึงภาษาต่างๆ ซึ่ง จะให้ภาพ ของภาษา ที่ใช้ร่วมกัน ได้แก่ ภาษา FORTRAN, COBOL, BASIC, Pascal, Ada และภาษา C ทั้งหมดนี้ เป็น ภาษายุคที่สาม ซึ่งใช้ร่วมกันในทุกวันนี้ แต่ละภาษา มีลักษณะสำคัญ อย่างไร จะให้ข้อสังเกตไว้ รวมทั้ง ชนิด ของงาน ซึ่งใช้ภาษานั้น ตาราง 8-1 ทำข้อสรุปงานต่างๆ ซึ่งปกติ ใช้ ภาษาเหล่านี้

ตาราง 8-1 Applications of some important programming languages

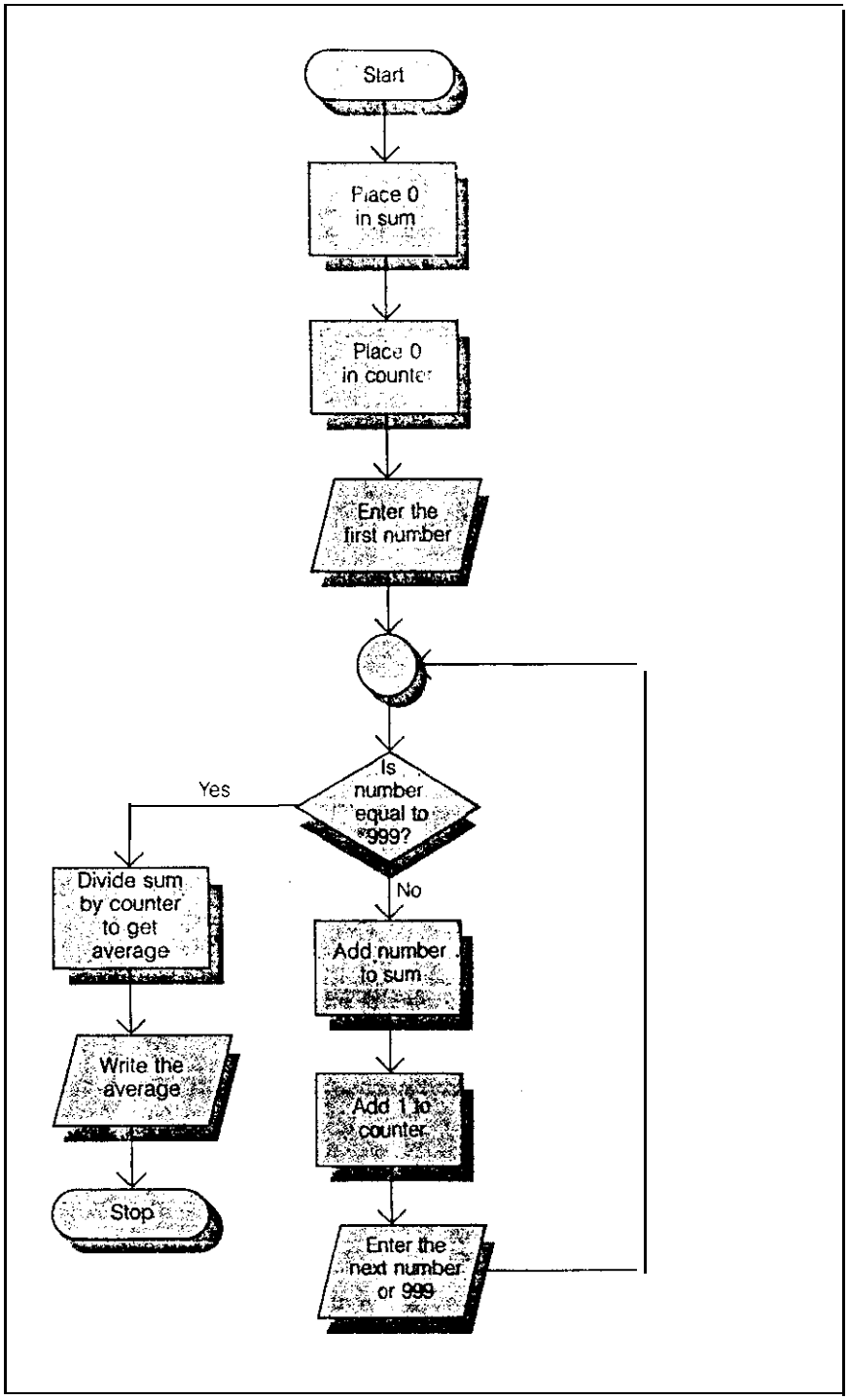
| Language | Application |
|---|---|
| FORTRAN - FORmula TRANslator (1954) | Scientific |
| COBOL - Common Business - Oriented Language (1959) | Business |
| BASIC Beginner's All-purpose Symbolic Instruction Code (1965) | Education, Busines |
| Pascal - named after French inventor Blaise Pascal (1971) | Education, System programming, scientific |
| Ada - named after Ada, the Countess of Lovelace (1980) | Military, general |
| C - evolved from the language B at Bell Labs (1972) | system programming, general |

ในการอภิปราย ภาษาโปรแกรมเหล่านี้ จะแสดง โปรแกรม และเอ้าพุท เพื่อให้เห็น ลักษณะของแต่ละภาษา โปรแกรมทั้งหมด ออกแบบมาเพื่อให้ คำนวณค่าเฉลี่ย ใน เอ้าพุท ตัวอย่าง คำนวณหาค่าเฉลี่ย ของเลข 3 จำนวน ทั้ง หก โปรแกรม ทำงานอย่างเดียวกัน เราจะ เห็นความแตกต่างกัน และความเหมือนกัน ระหว่างภาษาต่างๆ ผู้แต่งตำราเล่มนี้ ไม่คาดหวังว่า

นักศึกษาจะเข้าใจทุกบรรทัด ของโปรแกรม เพียงแต่ให้นักศึกษาเห็น หน้าตาของ โปรแกรม ว่าแต่ละภาษาเป็นอย่างไร

รูป 8-6 เป็น ผังงาน และ รหัสเทียม สำหรับ งานหาค่าเฉลี่ยของเลข ซึ่งจะได้อภิปราย ทีละภาษา พร้อมทั้งตัวโปรแกรม

```
Pseudocode:
start
Place  zero  in  sum
Place  zero  in  counter
Enter  the  first  number
DOWHILE  the  number  is  not  equal  to  333
    Add  number  to  sum
    Add  1  to  counter
    Enter  the  next  number  or  333
ENDDO
Divide  sum  by  counter  to  get  average
Write  the  average
End'
```




รูป 8-6 ผังงานและรหัสเทียม สำหรับการหาค่าเฉลี่ยของเลข

ผังงานและรหัสเทียม ที่จับคู่กันข้างต้นนี้ แสดงให้เห็น ตรรกะ (logic) ของโปรแกรม ทั้งนี้ ผู้ใช้ (user) ใส่เลขผ่านทางแป้นพิมพ์ (keyboard) จากนั้น โปรแกรมจะหาค่าเฉลี่ยของเลข ผู้ใช้สามารถใส่ข้อมูล จำนวนเท่าใดก็ได้ ใส่ทีละจำนวน ในการจบข้อมูล ให้ใส่ เลข 999 ตรรกะของ การใส่เลข ทำให้เกิดการวนซ้ำ : การใส่เลข, บวกเลข กับ ผลรวมสะสม และบวก 1 กับตัวนับ เมื่อใส่เลข 999 จะเป็นการออกจากลูป หลังจากนั้น คำนวณค่าเฉลี่ย แสดงผลบนจอภาพ

## ฟอร์แทรน : ภาษาระดับสูงภาษาแรก

(FORTRAN : The First High-Level Language)

ภาษานี้ พัฒนาโดย บริษัท IBM ประกาศใช้ ในปี ค.ศ. 1954 FORTRAN ย่อมาจาก FORmula TRANslator เป็นภาษาโปรแกรมระดับสูงภาษาแรก สร้างขึ้นมาโดยมีวัตถุประสงค์เพื่อทำงานทางด้านวิทยาศาสตร์ ในช่วงแรกนั้น เครื่องคอมพิวเตอร์ ใช้ สำหรับงานด้านวิศวกรรมศาสตร์ คณิตศาสตร์ และงานวิจัยทางวิทยาศาสตร์ ปัจจุบันนี้ ฟอร์แทรนยังคงเป็นภาษาโปรแกรมซึ่งเป็นที่นิยมใช้กันมากที่สุด ใน งานทางด้านวิทยาศาสตร์

ข้อสังเกต ของ ฟอร์แทรน สำหรับ ความรวบรัดของมัน (its brevity) เป็นส่วนหนึ่งของเหตุผลที่ว่า ทำไมภาษานี้ จึงยังคงเป็นที่นิยม ภาษานี้ดีมาก ในการสนองตอบต่อวัตถุประสงค์แรกของมัน ซึ่ง กระทำการกับสูตรซับซ้อน เช่น ใช้ในการวิเคราะห์ทางเศรษฐศาสตร์ และวิศวกรรมศาสตร์ อย่างไรก็ตาม จะใช้ได้ไม่ดี กับ การประมวลผลแฟ้มข้อมูล หรือ งานประมวลผลข้อมูล เนื่องจาก โครงสร้างควบคุมมีข้อจำกัดมาก เช่น การอธิบายรายละเอียดของข้อมูล นอกจากนี้แล้ว ภาษาฟอร์แทรน ยัง ไม่เหมาะสมอย่างมาก กับงานทางด้านธุรกิจ นอกจากนั้น ภาษาฟอร์แทรน ไม่จำเป็นต้อง ให้นิยาม สมาชิกข้อมูล ก่อนนำมาใช้ สิ่งนี้ ทำให้ภาษานี้ง่าย และมีขนาดสั้น แต่ทำให้เกิดข้อผิดพลาดได้ง่ายเช่นกัน

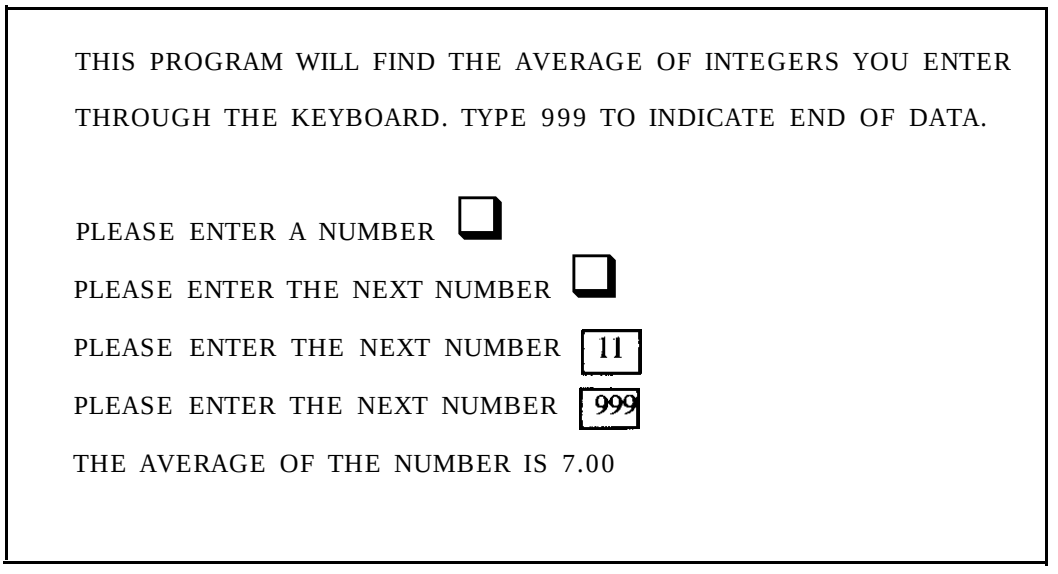
โปรแกรมทั้งหมด ไม่ใช่จัดระเบียบ ในวิธีเดียวกัน ทั้งนี้ หลากหลาย ขึ้นอยู่กับ ภาษาที่ใช้ ภาษาจำนวนมาก (เช่น COBOL) ตัวโปรแกรม ถูกแบ่งออกเป็นส่วนๆ โปรแกรมภาษาฟอร์แทรน ไม่ได้แบ่งเป็นส่วนๆ (ถึงแม้ว่า เป็นไปได้ที่จะเชื่อม (link) โปรแกรมต่างๆ เข้าด้วยกัน)

โปรแกรมภาษาฟอร์แทรน ประกอบด้วย ข้อความสั่ง ทีละบรรทัด ชนิดต่างๆ กันของข้อมูล จำแนกตามลักษณะข้อมูลที่ใช้ รายละเอียดของระเบียนข้อมูล ปรากฏ ใน ข้อความสั่ง format ซึ่งใช้ คู่กันกับ ข้อความสั่ง READ และข้อความสั่ง WRITE รูป 8-7 แสดงให้เห็นตัวโปรแกรมภาษาฟอร์แทรน และตัวอย่างเอ้าพุท ของโปรแกรม

```
C       FORTRAN  PROGRAM
C       AVERAGE INTEGER ENTERED THROUGH THE KEYBOARD
        WRITE (6, 10)
        SUM = 0
        COUNTER = 0
        WRITE (6, 60)
        READ (5, 40) NUMBER
    I IF (NUMBER .EQ. 999) GOT0 2
        SUM = SUM + NUMBER
        COUNTER = COUNTER + I
        WRITE (6, 70)
        READ (5, 40) NUMBER
        GOT0 1
    2 AVERAGE = SUM / COUNTER
        WRITE (6, 80) AVERAGE
   10 FORMAT (IX, 'THIS PROGRAM WILL FIND THE AVERAGE OF',
     * 'INTEGERS YOU ENTER',/1X, 'THROUGH THE ',
     * 'KEYBOARD. TYPE 999 TO INDICATE END OF DATA.',/)
   40 FORMAT (13)
   60 FORMAT (IX, 'PLEASE ENTER A NUMBER j
   70 FORMAT (IX, 'PLEASE ENTER THE NEXT NUMBER j
   80 FORMAT (IX, 'THE AVERAGE OF THE NUMBER IS ', F6.2)
        STOP
        END
```

(a)

เอ้าพุท เป็นดังนี้




(b)

รูป 8-7 โปรแกรมภาษา FORTRAN และตัวอย่างเอ้าพุท

โปรแกรมนี้ ทำงานทันที มี ข้อความบอกผู้ใช้ ให้ใส่ข้อมูล

(a) สองบรรทัดแรก เป็น คอมเมนต์ (comment) อยู่ในส่วนที่เหลือ ของ โปรแกรม ในบทนี้ ข้อความสั่ง WRITE ส่งเอ้าพุท ไปบน จอภาพ ใน รูปแบบ ซึ่งกำหนดโดย เลขตัวที่สองใน วงเล็บ ข้อความสั่ง READ รับข้อมูล จากผู้ใช้ และใส่ข้อมูล ในตำแหน่ง NUMBER ซึ่ง จะบวกกับ ผลรวมสะสม SUM ข้อความสั่ง IF ตรวจสอบว่า เป็น 999 หรือไม่ ถ้าใช่ มี การ เปลี่ยน ตรรกะของโปรแกรม ไปยังข้อความสั่ง 2 ซึ่ง คำนวณหาค่าเฉลี่ย หลังจากนั้น แสดงผลค่าเฉลี่ย

(b) จอภาพ แสดงให้เห็น การโต้ตอบระหว่างโปรแกรม กับ ผู้ใช้

## โคบอล : ภาษาโปรแกรมสำหรับงานธุรกิจ

(COBOL : The Language of Business)

ในช่วง ค.ศ. $1950_s$ ภาษาฟอร์แทรน ได้พัฒนาขึ้นมาแล้ว แต่ ยังไม่เป็นที่ยอมรับว่าเป็นภาษาโปรแกรมภาษาระดับสูงที่เหมาะสม กับ งานทางธุรกิจ กระทรวงกลาโหมของสหรัฐอเมริกา มีความสนใจ โดยเฉพาะ ที่จะสร้างภาษา ซึ่งเป็นมาตรฐาน จึงเรียก ตัวแทนจากรัฐบาล และโรงงานอุตสาหกรรมต่างๆ รวมทั้งอุตสาหกรรมทางด้านคอมพิวเตอร์ มาประชุมกัน ตัวแทนเหล่านี้รวมกันเป็น CODASYS - Conference of Data SYStem Language ในปี ค.ศ. 1959 CODASYS แนะนำ ภาษา COBOL ย่อมาจาก Common Business - Oriented Language รัฐบาลสหรัฐอเมริกา สนับสนุน โดยให้ทุกคน ที่ชนะข้อตกลง กับ โครงการเกี่ยวกับคอมพิวเตอร์ต้องใช้ ภาษาโคบอล สถาบันมาตรฐานแห่งชาติของประเทศสหรัฐอเมริกา (The American National Standard Institute (ANSI) ได้ให้ COBOL มาตรฐาน ในปี 1968 และในปี 1974 มีมาตรฐาน อีกเวอร์ชัน หนึ่งเรียกว่า ANSCOBOL และ หลังจาก อีกมากกว่า 7 ปี ของ การโต้เถียงทางอุตสาหกรรม ภาษามาตรฐาน ซึ่งเรียกว่า COBOL 85 ได้ถูกรับรอง ทำให้ COBOL เป็นเครื่องมือซอฟต์แวร์ ที่ ทันสมัยมากขึ้นในขณะนั้น ข้อดีของการเป็นภาษามาตรฐาน คือ โคบอล เป็นอิสระจากเครื่อง หมายถึง โปรแกรม ซึ่งเขียนเพื่อทำงานบน คอมพิวเตอร์ ชนิดหนึ่ง เพียงปรับปรุงเล็กน้อย เท่านั้น สามารถ วิ่ง (run) ได้กับคอมพิวเตอร์อีก ชนิดหนึ่ง ซึ่งมีคอมไพเลอร์ภาษาโคบอล

คุณสมบัติที่สำคัญของภาษา COBOL คือ เป็นภาษาโปรแกรมที่คล้ายกับภาษาอังกฤษมากกว่า ความคล้ายของ FORTRAN หรือ BASIC ชื่อตัวแปร กำหนดได้ในวิธีซึ่งแม้เราจะไม่รู้เกี่ยวกับการเขียนโปรแกรม ยังสามารถที่จะเข้าใจ วัตถุประสงค์ทั่วไปของโปรแกรมได้

**ตัวอย่างเช่น**

```
IF SALES-AMOUNT IS GREATER THAN SALES_QUOTA
        COMPUTE COMMISSION = MAX-RATE * SALES-AMOUNT
ELSE
        COMPUTE COMMISSION = MIN-RATE * SALES-AMOUNT.
```

ถ้าเราเข้าใจ หลักของการเขียนโปรแกรม ไม่ยากเลยสำหรับการเรียนรู้ COBOL เพิ่มขึ้น ภาษา COBOL ใช้สำหรับงานใดๆ ซึ่งสัมพันธ์กับ การเขียนโปรแกรมทางด้านธุรกิจ โดยเฉพาะ เหมาะสมกับการประมวลผล ข้อมูลชนิด ตัวอักษรเลข (alphanumeric data) เช่น ที่อยู่

สินค้าที่ซื้อ และ ปริมาณ มีหน่วยเป็น ดอลล่าร์ หรือ ข้อมูลเชิงธุรกิจ อย่างไรก็ตาม คุณสมบัติซึ่งทำให้ COBOL มีประโยชน์มาก คือ COBOL เป็นภาษาที่คล้ายกับภาษาอังกฤษ อ่านง่าย แต่เป็น จุดอ่อน เพราะว่า โปรแกรมภาษา COBOL มีคำศัพท์ เพิ่มเติมมาก ตัวโปรแกรมเขียนเป็นจำนวนบรรทัดมาก แม้จะทำงานง่ายๆ ดังนั้น สำหรับเรื่องความเร็ว และความง่าย ภาษา BASIC, FORTRAN และ Pascal น่าจะดีกว่า

ในรูป 8-8 จะเห็นว่า โปรแกรมภาษา COBOL ถูกแบ่งออกเป็นสี่ส่วน แต่ละส่วน *เรียกว่า ดิวิชั่น (divisions)*

- Identification division มีการบอกชื่อ ของโปรแกรม และบ่อยครั้ง ประกอบด้วย คอมเมนต์ ซึ่งจะช่วยเหลือ ผู้ใช้ได้
- Environment division อธิบายเกี่ยวกับตัวเครื่อง ซึ่ง จะนำโปรแกรม ไปคอมไพล์ และกระทำการ และยังเกี่ยวข้องกับ แต่ละแฟ้ม (file) ของ โปรแกรม เพื่อ กำหนด อุปกรณ์ทางกายภาพ โดยเฉพาะ เช่น tape drive หรือ เครื่องพิมพ์ (printer) ซึ่งจะ อ่านหรือเขียน ไฟล์
- Data division ประกอบด้วย สารสนเทศอย่างละเอียด เกี่ยวกับ การประมวลผลข้อมูล โดย โปรแกรม เช่น ชนิดของตัวอักขระ (ว่าเป็นเลข หรือ ตัวอักษรเลข) จำนวนตัวอักขระ และ การใส่จุดทศนิยม
- Procedure division ประกอบด้วย ข้อความสั่ง ซึ่งให้ คอมพิวเตอร์ กำหนด คำสั่งเพื่อทำงาน ตรรกะ ของโปรแกรม ออกมาจนเป็นผลสำเร็จ

ข้อสังเกตบางประการเกี่ยวกับ COBOL : COBOL เป็นภาษาเก่า ขนาดใหญ่ เทอะทะ และไม่สวยงาม แต่ที่มีชีวิตยาวนาน และใช้กันมาจนทุกวันนี้ และจากการอภิปรายทั้งหมดนี้ ไม่ได้เปลี่ยนแปลงความจริงที่ว่า ถ้าท่านสนใจที่จะทำให้มีเงินมากๆ โดยการเป็น โปรแกรมเมอร์ ทางธุรกิจ COBOL ยังเป็นภาษาที่ดีที่สุด

```
**************************************************************************
 IDENTIFICATION  DIVISION.
**************************************************************************
 PROGRAM-ID.    AVERAGE.
* COBOL  PROGRAM
* AVERAGING  INTEGERS  ENTERED  THROUGH  THE  KEYBOARD.
```

```
**************************************************************************
ENVIRONMENT DIVISION.
**************************************************************************
CONFIGURATION SECTION.
SOURCE-COMPUTER.  VAX-VMS
OBJECT-COMPUTER.  VAX-AMS.
**************************************************************************
DATA DIVISION.
**************************************************************************
FILE SECTION.
WORKING-STORAGE SECTION.
01  AVERAGE        PICTURE     ---9.99.
01  COUNTER        PICTURE     9(02)          VALUE  ZERO.
01  NUMBER-ITEM    PICTURE     S9(03).
01  SUM-ITEM       PICTURE     S9(06).        VALUE ZERO.
01  BLANK-LINE     PICTURE     X(80)          VALUE  SPACES.
**************************************************************************
PROCEDURE DIVISION
**************************************************************************
100-CONTROL-ROUTNE
     PERFORM 200-DISPLAY-INSTRUCTIONS.
     PERFORM 300-INITIALIZATION-ROUTINE.
     PERFORM 400-ENTER-AND-ADD
               UNTIL NUMBER-ITEM = 999
     PERFORM 500-CALCULATE-AVERAGE.
     PERFORM 600-DISPLAY-RESULTS.
     STOP RUN
200-DISPLAY-INSTRUCTIONS.
     DISPLAY
       "THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER"
```

```
        DISPLAY
          "THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA".
        DISPLAY  BLANK-LINE.
    300-INITIALIZATION-ROUTINE.
        DISPLAY "PLEASE ENTER A NUMBER".
        ACCEPT NUMBER-ITEM.
    400-ENTER-AND-ADD.
        ADD NUMBER-ITEM TO SUM-ITEM.
        ADD 1 TO COUNTER.
        DISPLAY "PLEASE ENTER THE NEXT NUMBER".
        ACCEPT NUMBER-ITEM.
    500-CALCULATE-AVERAGE.
        DIVIDE SUM-ITEM BY COUNTER GIVING AVERAGE.
    600-DISPLAY-RESULTS.
        DISPLAY "THE AVERAGE OF THE NUMBER IS", AVERAGE.
```

เอ้าพุท เป็นดังนี้

```
THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTEF
THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA.
PLEASE ENTER A NUMBER
6
PLEASE ENTER THE NEXT NUMBER
4
PLEASE ENTER THE NEXT NUMBER
11
PLEASE ENTER THE NEXT NUMBER
999
THE AVERGAE OF THE NUMBER IS  7.00
```

รูป 8-8 โปรแกรมภาษา COBOL และเอ้าพุท

## เบสิก สำหรับผู้เริ่มเรียนภาษาโปรแกรม และคนอื่นๆ

(BASIC : For Beginners and Others)

เราได้สัมผัส ภาษา BASIC ย่อมาจาก Beginners' All purpose Symbolic Instruction Code ซึ่งได้กล่าวมาแล้วในบทที่ 7 จะเห็นว่า BASIC เป็นภาษาร่วม ซึ่งเรียนรู้ได้ง่าย พัฒนาที่ Dartmouth College, ผู้ออกแบบคือ John Kemeny และ Thomas Kurtz ในปี ค.ศ. 1965 โดยมีความตั้งใจตั้งแต่แรกว่า จะให้เป็นภาษาที่ใช้โดย นักศึกษา ในสิ่งแวดล้อมทางวิชาการ ในช่วงท้าย ปี ค.ศ. $1960_s$ ภาษานี้กลายเป็นที่นิยมใช้ กันอย่างแพร่หลายมาก ใน สิ่งแวดล้อม ของการใช้การใช้เวลาร่วมกันแบบโต้ตอบ (interactive time-sharing environments) ในมหาวิทยาลัย และวิทยาลัยต่างๆ การใช้ภาษา BASIC ได้ขยายไป ในธุรกิจ และระบบคอมพิวเตอร์ส่วนตัว ระดับเล็กมาก และคอมพิวเตอร์ระดับเล็ก

คุณสมบัติข้อแรกของภาษา BASIC ซึ่งเป็นที่สนใจของ ผู้อ่านจำนวนมาก ของ หนังสือเล่มนี้ คือ ภาษา BASIC เรียนรู้ง่าย แม้ผู้นั้นจะ ไม่เคยเขียนโปรแกรมมาก่อนก็ตาม ดังนั้น บ่อยครั้งที่ ภาษานี้ ใช้สำหรับ การฝึกอบรม นักศึกษา ในชั้นเรียน ภาษา COBOL ยังถูกใช้โดยบุคคลซึ่งไม่ใช่โปรแกรมเมอร์ เช่น วิศวกร ซึ่งพบว่าภาษานี้มีประโยชน์ ในการแก้ปัญหา อย่างไรก็ตาม BASIC มีข้อจำกัด กล่าวคือ เป็นภาษาซึ่งไม่เหมาะสม สำหรับโปรแกรมซับซ้อน ตัวอย่างของโปรแกรมและเอ้าพุท แสดงให้เห็นในรูป 8-9

```
10 REM  BASIC PROGRAM
20 REM AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD.
30 PRINT "THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS
   YOU ENTER."
40 PRINT "THROUGH THE KEYBOARD.  TYPE 999 TO INDICATE END OF
   DATA."
50 PRINT
60  SUM = 0
70  COUNTER = 0
80.  PRINT "PLEASE ENTER A NUMBER"
90  INPUT  NUMBER.
100  IF NUMBER = 999 THEN 160
110  SUM = SUM + NUMBER
```

```
120. COUNTER = COUNTER + I
130. PRINT "PLEASE ENTER THE NEXT NUMBER"
140. INPUT NUMBER
150. GOT0 100
160. AVERAGE = SUM/COUNTER
170. PRINT "THE AVERAGE OF THE NUMBER IS"; AVERAGE
180. END
```

(a)

```
THIS PRGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER
THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA.
PLEASE ENTER A NUMBER
? 6
PLEASE ENTER THE NEXT NUMBER
?
PLEASEENTERTHENEXTNUMBER
? 11
PLEASE ENTER THE NEXT NUMBER
? 999
THE AVERAGE OF THE NUMBER IS 7
```

(b)

รูป 8-9 โปรแกรมภาษา BASIC

(a) โปรแกรมภาษา BASIC มองดูคล้ายกับ โปรแกรมภาษา FORTRAN มาก ข้อแตกต่างหลัก คือ ใน ข้อความสั่ง อินพุท และข้อความสั่ง เอ้าพุท ในที่นี้ PRINT แสดงผลของข้อมูล ซึ่งอยู่ทางขวามือ ใน ข้อความสั่ง บนจอภาพ และ INPUT รับข้อมูลจากผู้ใช้

(b) จอภาพนี้แสดงผลให้แห็นการโต้ตอบระหว่างโปรแกรม กับ ผู้ใช้

## ปาสคาล : ภาษาโปรแกรมอย่างง่าย

(Pascal : The Language of Simplicity)

Pascal เป็นชื่อที่ตั้งให้เป็นเกียรติแก่ Blaise Pascal นักคณิตศาสตร์ ชาวฝรั่งเศส ในศตวรรษ ที่ 17 ภาษา Pascal พัฒนาขึ้นมา เพื่อให้เป็นภาษาสำหรับการเรียนการสอนโดยเฉพาะ ผู้ออกแบบ คือ นักวิทยาศาสตร์คอมพิวเตอร์ ชาวสวิส ชื่อ Niklaus Wirth ประกาศใช้ครั้งแรก ในปี ค.ศ. 1971 ในช่วงเวลานั้น Pascal กลายเป็นภาษาที่นิยมใช้กันมาก ครั้งแรกในยุโรป ขณะนี้ ในประเทศสหรัฐอเมริกา โดยเฉพาะในมหาวิทยาลัย และวิทยาลัยต่างๆ ซึ่งมีการเรียนการสอน วิชาวิทยาศาสตร์คอมพิวเตอร์

คุณสมบัติที่สำคัญ ของ Pascal คือ เป็นภาษาที่ง่ายกว่า ภาษาโปรแกรมอื่นๆ กล่าวคือ มีคุณสมบัติ จำนวนน้อยกว่า และ ใช้ถ้อยคำ น้อยกว่า Pascal กลายเป็นภาษา ซึ่งนิยมกันมาก ในแผนกวิชาวิทยาศาสตร์คอมพิวเตอร์ ในวิทยาลัย เนื่องจาก ข้อจำกัด ของ ความสามารถทางอินพุท และเอ้าพุท ซึ่งไม่เหมือนกับ ในรูปแบบปัจจุบัน ทำให้มี ผลกระทบรุนแรง บน งานด้านธุรกิจ แต่ Pascal ทำให้เกิด ก้าวกระโดดอย่างมาก บนตลาดเครื่องคอมพิวเตอร์ระดับเล็กลงมา เนื่องจากเป็นภาษาที่ง่าย และทันสมัย เมื่อเทียบกับ BASIC ตัวอย่างของโปรแกรม และ เอ้าพุท ตัวอย่าง แสดงให้เห็นในรูป 8-10

```
program average (input, output);
(*  pascal program      *)
(*  averaging integers entered through the keyboard  *)

var
    counter, number, sum : integer;
    average : real;
begin
    writeln ('This program will find the average of integers you enter');
    writeln ('through the keyboard. Type 999 to indicate end of data.');
    writeln;
    sum := 0;
    counter := 0;
```

```
        writel" ('Please enter a number ');
        read (number);
        while numbeer <> 999 do
            begin
                sum := sum + number;
                counter := counter + I;
                writeln ('Please enter the next "umber');
                read (number);
            end;
        average := sum / counter;
        writel" ('The average of thee numbers is', average : 6 : 2);
    end.
```

เอ้าพุท เป็นดังนี้

```
This program will fmd the average of integers you enter
through the keyboard. Type 999 to indicate end of data.

Please enter a "umber
6
Please enter the next number
4
Please enter the next number
11
Please enter the next numbeı
999
The average of the number is 7.00
```

รูป 8-10 โปรแกรมภาษา Pascal และเอ้าพุทตัวอย่าง

(a) คอมเมนต์อยู่ใน เครื่องหมาย (* ถึง *) ตัวแปรทุกตัวต้องประกาศ สัญลักษณ์ := กำหนดค่าหนึ่งค่า ให้กับ ตัวแปรทางซ้ายมือ สัญลักษณ์ <> หมายถึง ไม่เท่ากับ ข้อความสั่ง writeln หมายถึง ใส่บรรทัดว่าง บนจอภาพ

(b) จอภาพนี้ แสดงผล ให้เห็นการโต้ตอบระหว่าง โปรแกรม กับ ผู้ใช้

## เอดา : ภาษาที่เป็นมาตรฐาน

(Ada : The Language of Standardization?)

ซอฟต์แวร์ ที่มีมูลค่ามากกว่า 25 พันล้านเหรียญดอลลาร์ มีหรือไม่? ไม่มีอีกแล้ว ตามที่ผู้เชี่ยวชาญ ของ กระทรวงกลาโหม กล่าว ในปี ค.ศ. 1974 กระทรวงกลาโหม ของ ประเทศสหรัฐอเมริกา ได้ใช้จ่ายเงิน จำนวนที่กล่าวข้างต้นนี้ บน ซอฟต์แวร์ ทุกชนิด สำหรับ การได้มาของภาษา สำหรับ ความจำเป็นของมัน คำตอบ ให้กับปัญหานี้ ทำให้ได้ภาษาใหม่ที่ชื่อ Ada ตั้งชื่อให้เป็นเกียรติแก่ เคาเตส Ada Lovelace "โปรแกรมเมอร์คนแรก" ผู้ให้การสนับสนุนทางการเงิน คือ เพนตากอน (Pentagon) เมื่อเริ่มต้นนั้น ตั้งใจให้ Ada เป็นภาษามาตรฐาน สำหรับระบบอาวุธ แต่ภาษานี้ ประสบผลสำเร็จ สำหรับ งานทางด้านธุรกิจ เช่นกัน ประกาศ เป็นทางการใน ปี ค.ศ. 1980 ภาษา Ada ไม่เพียงแต่ได้รับการสนับสนุนจากกองทัพเท่านั้น แต่ยังมีอุตสาหกรรมขนาดใหญ่ เช่น บริษัท IBM และ Intel ให้การสนับสนุนด้วย Ada ยังใช้ได้กับไมโครคอมพิวเตอร์ บางชนิด ถึงแม้ว่า ผู้เชี่ยวชาญ โรงงานอุตสาหกรรมบางคน กล่าวว่า ภาษา Ada ซับซ้อนมาก แต่มีบางคน กล่าวว่า ภาษานี้ เรียนรู้ได้ง่าย (easy to learn) และ จะสามารถเพิ่มผลิตผล จริงๆ แล้ว ผู้เชี่ยวชาญบางคน เชื่อว่า ในอนาคต Ada จะเป็นภาษาเชิงพาณิชย์ที่สำคัญเช่นเดียวกับ COBOL และ FORTRAN

ตัวอย่างโปรแกรม และ เอ้าพุทแสดงให้เห็นในรูป 8-11

```
-- ADA PROGRAM
-- AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD.
with TEXT_IO; use TEXT_IO;
procedure AVERAGE is
        package INT_IO is new INTEGER-IO (INTEGER);
        AVERAGE :           FLOAT;
        COUNTER :           INTEGER := 0;
        NUMBER :            INTEGER;
        SUM :               INTEGER := 0;
begin
        PUT_LINE ("THIS PROGRAM WILL FIND THE AVERAGE OF
                    INTEGERS YOU ENTER");
        PUT_LINE ("THROUGH THE KEYBOARD.  TYPE 999 TO INDICATE
```

```
                END OF DATA.");
        NEW-LINE;
        PUT ("PLEASE ENTER A NUMBER");
        INT-IO. GET(NUMBER);
        while NUMBER /= 999 loop
                SUM := SUM + NUMBER,
                COUNTER := COUNTER + I;
                PUT ("PLEASE ENTER THE NEXT NUMBER");
                INT_IO.GET (NUMBER);
        end loop;
        AVEERAGE := SUM / COUNTER;
        PUT (THE AVERAGEE OF THE NUMBER Is");
        FLO_IO.PUT (AVERAGE);
end AVERAGE;
```

(a)

**เอ้าพุท เป็นดังนี้**

```
THIS PRGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER
THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA.

PLEASE ENTER A NUMBER ❑
PLEASE ENTER THE NEXT NUMBER ❑
PLEASE ENTER THE NEXT NUMBER [11]
PLEASE ENTER THE NEXT NUMBER [999]
THE AVERAGE OF THE NUMBER IS 7.00
```

(b)

**รูป 8-11 โปรแกรมภาษา Ada และเอ้าพุทตัวอย่าง**

(a) คอมเมนต์ ให้เริ่มต้นด้วย hyphen สองตัว ภาษา Ada มีคุณสมบัติว่า ตัวแปรทุกตัว ต้องประกาศ ก่อน เริ่ม logic ข้อความสั่ง NEW-LINE แสดงผลบรรทัดว่าง, ข้อความสั่ง PUT-LINE แสดงผลข้อมูลบนจอภาพ สัญลักษณ์ /= หมายถึง ไม่เท่ากับ

(b) จอภาพนี้ แสดงให้แห็นการโต้ตอบระหว่าง โปรแกรม กับ ผู้ใช้

## ซี : ภาษาสมัยใหม่

(C : A Sophisticated Language)

ภาษาซึ่ง โดยตัวมันเอง ใช้งานด้านการเขียนโปรแกรมระบบ (ระบบปฏิบัติการ และงานอื่นๆ ที่คล้ายกัน) ภาษา C พัฒนาโดย Dennis Ritchie ที่บริษัท Bell Labs ในปี ค.ศ. 1972 ภาษานี้ พัฒนา ต่อมาจาก เวอร์ชัน ก่อนหน้านั้น ซึ่งมีชื่อเรียกว่า ภาษา B ภาษา C ให้รหัส ซึ่งมีประสิทธิภาพ ใกล้เคียงกับภาษาแอสเซมบลี ในขณะที่ ยังคงมี คุณสมบัติ ของ ภาษาระดับสูง เช่น การเขียนโปรแกรมแบบโครงสร้าง ภาษา C ประกอบด้วยคุณสมบัติที่ดีที่สุดจากภาษาอื่นๆ รวมทั้ง PL/1 และ Pascal ตัวคอมไพเลอร์ของภาษา C ง่าย และรัดกุม (simple and compact) เนื่องจาก ภาษา C เป็นอิสระจากสถาปัตยกรรมของตัวเครื่องคอมพิวเตอร์ จึงเป็นภาษาที่เหมาะสมสำหรับการเขียนโปรแกรม ซึ่ง เคลื่อนย้ายง่าย [1] นั่นคือ โปรแกรม ซึ่งสามารถวิ่ง บนคอมพิวเตอร์ ได้มากกว่า หนึ่งชนิด

ถึงแม้ว่า ภาษา C จะง่ายและสวยงาม แต่ไม่ง่าย สำหรับการเรียนรู้ ภาษา C พัฒนาขึ้นมาสำหรับ โปรแกรมเมอร์ ที่มีความสามารถสูง (gifted programmers) และ เส้นโค้งของการเรียนรู้มาก จริงๆ แล้ว งานชนิดตรงไปตรงมา อาจแก้ปัญหา ได้ง่าย ในภาษา C แต่ ปัญหาซับซ้อน จำเป็นต้องใช้คนเก่ง (mastery) ของภาษานั้น

ข้อสังเกตที่น่าสนใจ คือ ภาษา C ใช้ได้ บน ไมโครคอมพิวเตอร์ มีส่วนส่งเสริมอย่างมาก กับ ค่าของ ไมโครคอมพิวเตอร์ สำหรับ budding entrepreneurs นั่นคือ อุตสาหกรรมซอฟต์แวร์ สามารถใช้ เครื่องมือพื้นฐานอย่างเดียวกัน - ภาษา C ซึ่งใช้ได้ ประสบผลสำเร็จโดยบริษัทซอฟต์แวร์ เช่น ไมโครซอฟต์ (Microsoft)

ตัวอย่างของโปรแกรมและเอ้าพุท แสดงให้เห็นในรูป 8-12

```
/* c program */
/* Averaging integers entered through the keyboard */
```

[1] "portable" prgrams - that is, programs that can be run on more than one type of computer.

```
main ( )
{float average;
 int counter = 0 ; number ; sum = 0;
printf ("THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER\n");
printf ("THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA.\n\n");
printf ("PLEASE ENTER A NUMBER");
scanf ("%d", &number);
while (number != 999)
    {
       sum = sum + number;
    counter ++;
       printf ("PLEASE ENTER THE NEXT NUMBER");
       printf ("%d", &number):
    }
    average = sum / counter;
    printf ("THE AVERAGE OF THE NUMBERS IS %F", AVERAGE);
}
```

(a)

```
THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER
THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA.

PLEASE ENTER A NUMBER 6
PLEASE ENTER THE NEXT NUMBER 4
PLEASEENTERTHENEXTNUMBER 11
PLEASE ENTER THE NEXT NUMBER 999
THE AVERAGE OF THE NUMBERS IS 7.00
```

(b)

รูป 8-12 โปรแกรมภาษา c และเอ้าพุท

(a) คอมเมนต์ อยู่ภายในเครื่องหมาย /* และ */ ชื่อตัวแปรทั้งหมด เช่น number ต้องมีการประกาศ คำสั่งงาน printf ส่ง เอ้าพุท ไปยังจอภาพ และ scanf นำ ข้อมูลมาจาก ผู้ใช้

(b) จอภาพนี้ แสดงให้เห็น การโต้ตอบระหว่าง โปรแกรม กับ ผู้ใช้

## ภาษาอื่นๆ

(Some Other Languages)

ภาษาโปรแกรมต่างๆ ซึ่งอธิบายมาแล้วข้างต้น น่าจะเป็นภาษาหลักๆ ซึ่งใช้กันอยู่ทุกวันนี้ มีหลายภาษาในกลุ่มนี้ ซึ่งมีอภิสิทธิ์ สำหรับ การไม่มีเหตุผลว่า ทำไมจึงมาอยู่ใน อันดับแรก หรือ อยู่ อันดับหลัง โดยองค์กร ที่มีกำลัง แต่ภาษาอื่นๆ แม้จะไม่นิยมแพร่หลายเท่ากับภาษาเหล่านี้ แต่ก็ยังคงเป็นที่นิยมใช้อยู่ และยังมีความสำคัญที่จะต้องทำความรู้จัก โปรดสังเกตว่า ภาษาเหล่านี้ ส่วนใหญ่เป็น ภาษาใช้งานแฉพาะด้าน (special-purpose languages) จึงมีข้อจำกัด ของการใช้มากกว่า

**อัลกอล** (ALGOL)

ย่อมาจาก ALGOrithmic Language ภาษานี้ ใช้ในปี 1960 เป็นที่นิยมมากในยุโรป และนิยมใช้ในประเทศสหรัฐอเมริกา ALGOL พัฒนาขึ้นมาโดยมีวัตถุประสงค์ตั้งแต่แรกว่า จะใช้งานด้านเขียนโปรแกรมวิทยาศาสตร์ และเป็นรากฐานของภาษา PL/I และ Pascal

**ลิสป์** (LISP)

ย่อมาจาก **LIS**t **P**rocessing พัฒนาในปี 1958 ที่สถาบันเทคโนโลยีแมสสาชูเสต (Massachuseetts Institute of Technology) ผู้ออกแบบคือ John McCarthy ภาษานี้ออกแบบมาเพื่อ ประมวลผลข้อมูลที่ไม่ใช่เลข ได้แก่ สัญลักษณ์ ตัวอักขระ หรือ คำ LISP สามารถใช้ เชิงโต้ตอบ ที่ เทอร์มินัล ได้ เป็นภาษาที่นิยมใช้ สำหรับ เขียนโปรแกรม เกี่ยวกับ งานด้านปัญญาประดิษฐ์ (artificial intelligence)

**โปรล็อก** (PROLOG)

เป็นภาษาโปรแกรม สำหรับ งานด้านปัญญาประดิษฐ์ ชื่อ PROLOG ย่อมาจาก **PRO**gramming in **LOG**ic ภาษานี้ กำลังได้รับความสนใจเพิ่มขึ้น ในฐานะที่เป็น เครื่องมือสำหรับ การเขียนโปรแกรม ภาษาธรรมชาติ ประดิษฐ์ขึ้นในปี ค.ศ. 1972 โดย Alan Colmerauer ณ มหาวิทยาลัยมาแซล แต่ได้รับความสนใจอย่างกว้างขวาง ในปี 1979 เฉพาะเวอร์ชันที่มีประสิทธิภาพมากกว่า ได้ถูกนำมาใช้ ภาษา PROLOG ถูกเลือก โดย ประเทศญี่ปุ่น ให้เป็น ภาษาทางการ ของ โครงการ คอมพิวเตอร์ยุคที่ 5 และภาษานี้ กลายเป็นที่นิยม ในผู้มีอาชีพทางด้านปัญญาประดิษฐ์ จำนวนมาก

**พีแอล/วัน** (PL/1)

นำมาใช้ในปี 1964 ชื่อ PL/1 ย่อมาจาก Programming Language One ผู้ให้การสนับ

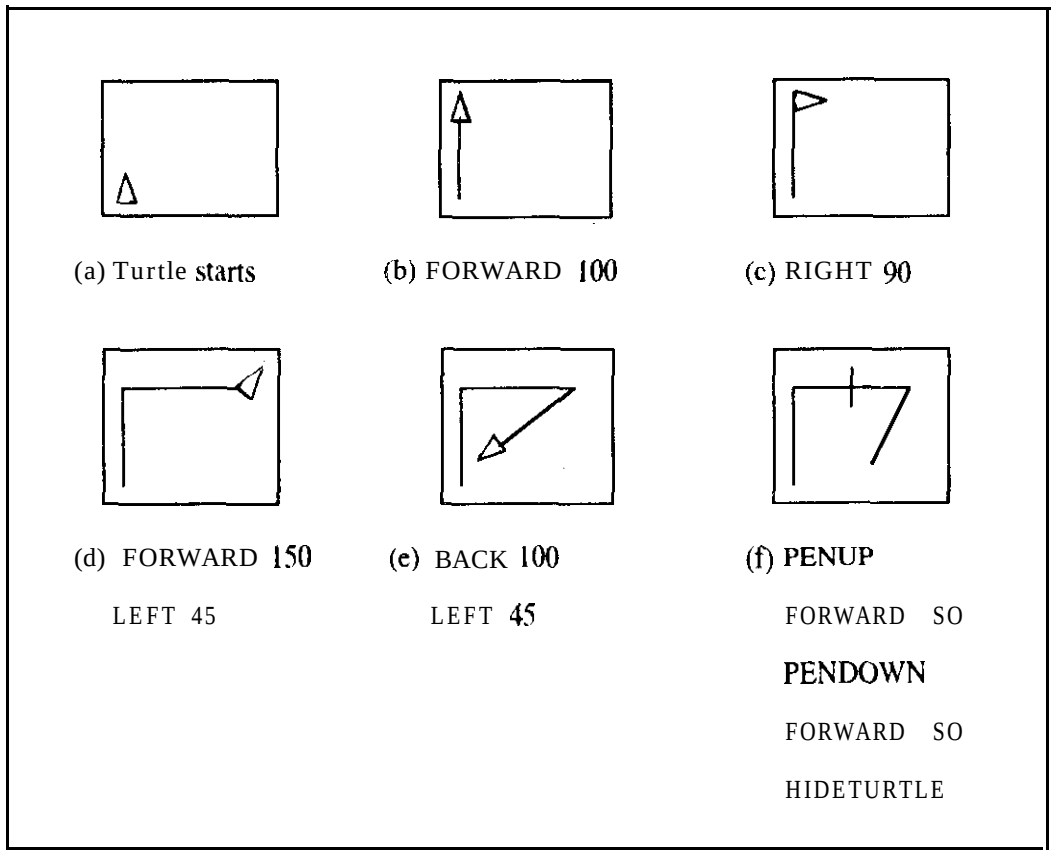
สนุนคือบริษัท IBM ภาษานี้ ออกแบบมา เพื่อให้ใช้ได้ทั้ง งานวิทยาศาสตร์ และงานธุรกิจ PL/1 จึงค่อนข้างยืดหยุ่นมาก จริงๆ แล้วเป็นภาษา ซึ่งเรียนรู้ง่าย โดย ศึกษาขั้นต้นจากตัวอย่างต่างๆ อย่างไรก็ตาม เนื่องจากเป็นภาษานำทุกสิ่ง มารวมกันเพื่อให้ใช้งานได้ทุกด้าน จึงทำให้ภาษานี้ มีขนาดใหญ่ ด้วยมี ทางเลือก (options) มาก ทำให้การใช้ประโยชน์ ลดลง

**เอพีแอล** (APL)

ย่อมาจาก A Programming Language ผู้ออกแบบคือ Kenneth Iverson นำมาใช้โดย IBM ในปี 1968 ภาษา APL มีกำลัง (powerful) การโต้ตอบ และเหมาะสมกับการจัดทำตาราง นั่นคือ การประมวลผล กลุ่มของเลขที่สัมพันธ์กับ ในตาราง ภาษา APL มี สัญลักษณ์พิเศษจำนวนมาก ซึ่งเป็นเหตุผลหนึ่ง ซึ่ง อาจทำให้เกิดปัญหา ใน การวิ่งโปรแกรมบนคอมพิวเตอร์ เนื่องจากสัญลักษณ์ จำนวนมาก นี้ไม่ใช่ส่วน ซึ่งเราคุ้นแคย ใน ชุดอักขระ ASCII สัญลักษณ์บางตัว แทน การปฏิบัติการ ที่มีกำลังมาก เนื่องจาก ภาษา APL มี ตัวปฏิบัติการ จำนวนมาก นั้นหมายความว่า คอมไพเลอร์ ต้องมีขนาดใหญ่ด้วย ดังนั้นภาษานี้ จึงใช้ได้เฉพาะบนระบบคอมพิวเตอร์ ที่มีหน่วยความจำมาก

**โลโก** (LOGO)

ถ้าเราเคยได้ยิน โปรแกรมเมอร์ สองคน (หรือครูในโรงเรียน) ใช้คำ เต่า (turtle) ในการสนทนากัน เราเดาได้ว่า เขากำลังพูดถึง ภาษาโลโก ภาษาย่อย ของ LISP ซึ่งพัฒนาที่ สถาบันแมซซาชูเสต (Massachusetts Institute of Technology) โดย Seymour Papert ขณะนี้เป็นที่รู้จักกันว่า เป็นภาษาซึ่งเด็กๆ สามารถใช้ได้ คำว่า "เต่า" จริงๆ แล้วเป็นตัวชี้รูปสามเหลี่ยม บนจอภาพ ซึ่งโต้ตอบ กับ คำสั่งงาน ง่ายๆ เช่น FORWARD และ LEFT เป็นภาษาเชิงโต้ตอบ หมายความว่า ผู้คนสามารถเรียนรู้ โดยใช้ LOGO ผ่านบทสนทนา กับ คอมพิวเตอร์ รูป 8-13 เป็นตัวอย่าง ของ การออกแบบโปรแกรม LOGO




รูป 8-13 LOGO Logic

"เต่า" หรือ ตัวชี้รูปสามเหลี่ยม เคลื่อนย้ายด้วย ลำดับ ของ คำสั่งงาน LOGO แบบง่ายๆ เช่น FORWARD ย้าย เต่า ในทิศทางด้านหน้า BACK ย้าย เต่า กลับข้างหลัง ตัวเลข ซึ่งตามหลัง คำสั่งงาน หมายถึง ความยาวของเส้นที่ให้วาด RIGHT และ LEFT ตามด้วยเลข แสดงถึงทิศทาง ของ เต่า ซึ่งจะหมุนเวียน และจำนวน ดีกรี ของ การหมุนเวียน PENUP และ PENDOWN หมายถึง การยกปากกาขึ้น และ การจรดปากกาลง เมื่อเต่าเคลื่อนย้าย จะทิ้งรอยทางเดินไว้ คำสั่งงาน HIDETURTLE ทำให้เต่า หายไปจากจอภาพ

**ไพลอต (PILOT)**

ภาษานี้ประดิษฐ์ขึ้นในปี ค.ศ. 1973 แต่แรก ออกแบบมาเพื่อให้เด็กๆ คุ้นเคยกับเครื่องคอมพิวเตอร์ ปัจจุบันนี้ ภาษา PILOT ถูกนำมาใช้มากที่สุด เพื่อเขียนการสอนใช้คอมพิวตอร์ช่วย (computer-aided instruction หรือ CAI) ในโครงการทั้งหมด โดยเฉพาะ เหมาะสมกับ งานที่เกี่ยวกับ คำสั่งฝึกหัด และการทดสอบ ภาษา PILOT ไม่ใช่ทางเลือกที่ดี สำหรับ ปัญหาการคำนวณที่ซับซ้อน

**สมอลทอค** (Smalltalk)

การโต้ตอบมากที่สุด กับ เครื่องคอมพิวเตอร์ ประกอบด้วย "การจำ และ การพิมพ์" อย่างไรก็ตาม ภาษา smalltalk เรา "เห็น" และ "พิมพ์" ในที่นี้ คือวิธีทำงานของภาษานี้ คีย์บอร์ด ใช้ ใส่ ข้อความ (text) เข้าไปยังคอมพิวเตอร์ แต่ งานอื่นๆ ทั้งหมด ทำให้ ประสบผลสำเร็จ ได้โดยการใช้ เมาส์ (mouse) เราเคลื่อนเมาส์ไปรอบรอบ เพื่อกำหนดทิศทางการเคลื่อนที่ของตัวชี้บนจอภาพ กดปุ่มบนเมาส์ เพื่อเลือกคำสั่งงาน ภาษานี้ ประดิษฐ์ โดย Alan Kay ที่ ศูนย์วิจัย Palo Alto รัฐ California และพัฒนาโดย Xerox Corporation ภาษา Smalltalk แตกต่างไปจาก วิทยาศาสตร์คอมพิวเตอร์ ในอดีต เพราะว่า ภาษานี้ สนับสนุนระบบคอมพิวเตอร์ภาพ (visual computer system) โดยเฉพาะ หลักของภาษา Smalltalk คือ มันเป็น **ภาษาเชิงวัตถุ** (object-oriented language) ไม่ใช่ ภาษาเชิงกระบวนงาน (procedure-oriented language) : เป็นการโต้ตอบ ระหว่าง คน และ สิ่งของต่างๆ หรือ ชนิดต่างๆ ของวัตถุ

**FORTH**

ภาษานี้ ออกแบบโดย Charles Moores ในปี ค.ศ. 1975 สำหรับ งานควบคุม แบบทันที (real-time control task) เช่น การแนะนำกล้องดาราศาสตร์ เช่นเดียวกับ โปรแกรมกราฟฟิก และงานธุรกิจหลากหลาย ภาษานี้ออกแบบมา เพื่อทำให้การใช้ หน่วยความจำของคอมพิวเตอร์ และความเร็วเป็นไปได้ดีที่สุด ดังนั้น FORTH จึงเป็นภาษาโปรแกรมที่ดีเยี่ยม สำหรับ ไมโครคอมพิวเตอร์ โดยปกติแล้ว โปรแกรม FORTH วิ่งได้เร็วมาก และ ใช้หน่วยความจำ น้อยมากกว่าโปรแกรม BASIC ซึ่งทำงานอย่างเดียวกัน ทุกวันนี้ FORTH มีให้ใช้ได้ บน คอมพิวเตอร์เกือบทุกชนิด จากคอมพิวเตอร์ ระดับเล็กมาก จนถึง คอมพิวเตอร์ ระดับใหญ่ ภาษา FORTH ใช้หน่วยความจำของคอมพิวเตอร์ เพียงเล็กน้อย แต่ โปรแกรมเมอร์ ที่มีประสบการณ์เท่านั้น จึงจะสามารถใช้ภาษานี้ได้อย่างถูกต้อง

**Modula-2**

โปรแกรมเมอร์ ภาษา Pascal จะไม่มีปัญหาเลย เมื่อทำความรู้จักกับ Modula-2 เพราะว่าทั้งสองภาษานี้ เกือบจะเหมือนกัน สิ่งนี้ ไม่น่าประหลาดใจ เพราะว่า ผู้ออกแบบทั้งสองภาษานี้ คือ Niklaus Wirth ภาษา Pascal ตั้งใจ ให้เป็นภาษา สำหรับ การเรียนการสอน งานด้านนี้ จึงกระทำ ได้ดีมาก แต่ Modula-2 เห็นได้ชัดเจน ว่าออกแบบมา เพื่อเขียน ซอฟต์แวร์ระบบ โดยเฉพาะ

**RPG**

เป็นภาษาเชิงปัญหา (problem-oriented language) ชื่อนี้ ย่อมาจาก Report Program

Generator เป็นภาษาที่ออกแบบมาเพื่อแก้ปัญหา ของ การผลิตรายงานเชิงธุรกิจ โดยเฉพาะ สามารถทำการปรับปรุงแฟ้ม ให้ทันสมัย บริษัท IBM เป็นผู้พัฒนาขึ้นมา นำมาใช้ในปี 1964 แต่แรกนั้น ตั้งใจ ให้ใช้กับ ระบบคอมพิวเตอร์ ขนาดเล็ก เวอร์ชัน ปรับปรุงใหม่ ชื่อ RPG II นำมาใช้ในปี 1970 และ ขยาย ความสามารถดั้งเดิม ของ ภาษา ส่วนเวอร์ชัน ใหม่กว่า คือ RPG III เป็นภาษาเชิงโต้ตอบ ซึ่งใช้ เมนู เพื่อให้โปรแกรมเมอร์ เลือกได้ง่าย เพื่อวางแผนเขียน โปรแกรม RPG เรียนรู้ได้ง่ายมาก ซึ่ง นักธุรกิจเชื่อว่า เขา สามารถ ได้ ผลกลับคืน มากที่สุด จากการลงทุน

## คำถามวิจารณ์ (Review Questions)

1. โดยทั่วไป ภาษาโปรแกรม แตกต่างจาก ภาษามนุษย์ อย่างไร?
2. คำว่า "สูง" หรือ "ต่ำ" ในความหมายของภาษาโปรแกรม คืออะไร?
3. จงอธิบาย ความแตกต่าง ของ ภาษา ต่อไปนี้
   - ภาษาเครื่อง
   - ภาษาแอสเซมบลี
   - ภาษาระดับสูง
4. จงอธิบายว่า ทำไม ภาษายุคที่สี่ จึงแทน การเพิ่มผลิตผล เมื่อเปรียบเทียบ กับ ภาษายุคที่สาม
5. จงอภิปราย ข้อดี และ ข้อจำกัด ของ ภาษาธรรมชาติ
6. ภาษา COBOL แตกต่างจาก ภาษา FORTRAN และ BASIC อย่างไร?
7. ภาษา BASIC และ Pascal เหมือนกัน อย่างไร และสองภาษานี้ แตกต่างกันอย่างไร
8. จงอภิปราย ความเหมาะสม และความไม่เหมาะสม ของ ภาษาต่อไปนี้
   - FORTRAN
   - COBOL
   - Pascal
9. จงอภิปราย ข้อดี ของ ภาษา C
10. จงอภิปราย ข้อดี ของ ความเป็นมาตรฐาน ของ ภาษาโปรแกรม
11. จงอภิปรายว่า ทำไม บางภาษา จึงยังคงมี ใช้อยู่จนทุกวันนี้ และบางภาษา ไม่ได้นำมาใช้เลย
12. ท่านคิดว่า ความต้องการของโปรแกรมเมอร์ ที่ ให้มี ภาษาใช้ได้ง่ายขึ้น มี จำนวนน้อยลง หรือไม่? อธิบาย

13. จากตัวอย่าง โปรแกรม หาค่าเฉลี่ย ของเลข 3 จำนวน จงแปลี่ยนแปลง ให้เป็น โปรแกรม หาผลรวม ของ เลขจำนวนเต็ม 2 ตัว