บทที่ 2
การแก้ปัญหาและ Pascal
(Problem Solving and Pascal)

2.1 วิธีการพัฒนาซอฟต์แวร์
2.2 การประยุกต์ใช้วิธีการพัฒนาซอฟต์แวร์
กรณีศึกษา : การเปลี่ยนมาตรหน่วยวัด
2.3 ภาพรวมของ Pascal, คำสงวนและไอเดนต์ไฟเออร์
2.4 แบบชนิดข้อมูลและการประกาศ
2.5 ข้อความสั่งกระทำการ
2.6 รูปแบบทั่วไปของโปรแกรม Pascal
2.7 นิพจน์คำนวณ
กรณีศึกษา : การประเมินค่าเหรียญต่างๆ
2.8 การจัดรูปแบบและการดูเอาต์พุตของโปรแกรม
2.9 การแก้จุดบกพร่องและข้อผิดพลาดของการเขียนโปรแกรม

การเขียนโปรแกรม คือ กิจกรรมของการแก้ปัญหาอย่างหนึ่ง (Programming is a problem-solving activity)

ถ้าเราเป็นนักแก้ปัญหาที่ดี เรามีศักยภาพที่จะเป็นโปรแกรมเมอร์ที่ดี เป้าหมายอย่างหนึ่งของหนังสือเล่มนี้คือ ช่วยให้เราปรับปรุงความสามารถในการแก้ปัญหา เพื่อให้ถึงจุดหมายนี้ จึงแนะนำวิธีเชิงระบบในหัวข้อ 2.1 เพื่อใช้แก้ปัญหาการเขียนโปรแกรม เรียกว่า วิธีการพัฒนาซอฟต์แวร์ และจะแสดงให้เห็นว่าจะประยุกต์ใช้อย่างไรในหัวข้อ 2.2

เนื้อหาในบทนี้แนะนำ Pascal ซึ่งเป็นภาษาเขียนโปรแกรมระดับสูง พัฒนาในปี ค.ศ. 1971 โดย Nicklaus Wirth แห่งมหาวิทยาลัย Zurich ประเทศ Switzerland Pascal เป็นภาษาเขียนโปรแกรมซึ่งเป็นที่นิยมแพร่หลายมากที่สุด สำหรับการสอนแนวคิดของการเขียนโปรแกรม เพราะว่า วากยสัมพันธ์ของมันค่อนข้างง่ายต่อการเรียนรู้ อีกเหตุผลหนึ่ง

ซึ่งทำให้ Pascal เป็นที่นิยมทั่วไป คือ คอมไพเลอร์ Pascal ซึ่งมีประสิทธิภาพมีให้ใช้สำหรับคอมพิวเตอร์ส่วนใหญ่

Pascal สะดวกในการเขียนโปรแกรมเชิงโครงสร้าง (Structured programs) โปรแกรมซึ่งง่ายต่อการอ่าน ทำความเข้าใจ และมีการทำงานเป็นอันดับที่ดี Pascal เป็นที่นิยมในอุตสาหกรรม เพราะว่า การเขียนโปรแกรมเชิงโครงสร้าง คือ การปฏิบัติการเขียนโปรแกรมที่เป็นมาตรฐาน ข้อดีอีกประการหนึ่ง สำหรับการศึกษา Pascal คือ มันเป็นพื้นฐานสำหรับการออกแบบภาษาโปรกรม Ada ซึ่งเป็นภาษาทางการรับรองโดยกระทรวงกลาโหมของประเทศสหรัฐอเมริกาสำหรับการพัฒนาซอฟต์แวร์

จงเชื่อมั่นว่า โปรแกรม Pascal ซึ่งเขียนบนคอมพิวเตอร์เครื่องหนึ่ง กระทำการได้บนคอมพิวเตอร์อีกเครื่องหนึ่ง รหัสคำสั่งต้องตรงกันกับภาษา Pascal มาตรฐาน ซึ่งอธิบายตัวสร้างภาษา Pascal ทั้งหมดและวากยสัมพันธ์ของมัน โปรแกรมในตำราเล่มนี้เขียนด้วย Turbo Pascal ซึ่งพัฒนาโดย Borland International สำหรับใช้บนคอมพิวเตอร์ใช้แทนกันได้กับเครื่อง IBM (IBM-compatible computers) Turbo Pascal ขยายต่อ (extands) เป็น Standard Pascal และมีคุณสมบัติซึ่งไม่ได้อธิบายในมาตรฐานของภาษา เมื่อใดก็ตามที่แนะนำส่วนขยายของ Turbo Pascal จะเปรียบเทียบให้เห็นระหว่าง Turbo Pascal และ Standard Pascal

ในบทนี้จะอธิบายส่วนของโปรแกรม Pascal และชนิดของข้อมูล ซึ่งสามารถถูกประมวลผลโดย Turbo Pascal จากนั้นอธิบายข้อความสั่งของ Pascal สำหรับกระทำการคำนวณ สำหรับการใส่ข้อมูล และสำหรับการแสดงผลของผลลัพธ์

2.1 วิธีการพัฒนาซอฟต์แวร์ (The Software Development Method)

วิธีการพัฒนาซอฟต์แวร์แบ่งออกเป็น 6 ขั้นตอนดังนี้

1. การระบุความต้องการของปัญหา (Specify the problem requirements)
2. การวิเคราะห์ปัญหา (Analyze the problem)
3. การออกแบบอัลกอริทึมเพื่อแก้ปัญหา (Design the algorithm to solve the problem)
4. การทำให้เกิดผลของอัลกอริทึม (Implement the algorithm)
5. การทดสอบและการทวนสอบโปรแกรมซึ่งสร้างเสร็จแล้ว (Test and verify the completed program)

6. การบำรุงรักษาโปรแกรมและปรับให้เป็นปัจจุบัน (Maintain and update the program)

(1) ปัญหา (Problem)

การระบุความต้องการของปัญหา บังคับให้เรากำหนด (state) ปัญหาอย่างชัดเจน และไม่กำกวม และเพื่อให้ได้รับความเข้าใจชัดเจนว่าต้องการให้แก้ปัญหาอะไร วัตถุประสงค์ของเราคือ ขจัดสิ่งที่ไม่มีความสำคัญในปัญหารากซึ่งไม่ใช่เรื่องง่าย เราอาจจำเป็นต้องได้รับสารสนเทศมากขึ้นจากบุคคลซึ่งเป็นเจ้าของปัญหา

(2) วิเคราะห์ (Analysis)

การวิเคราะห์ปัญหาเกี่ยวข้องกับการระบุปัญหา

(a) อินพุต (inputs) ได้แก่ข้อมูลซึ่งเราจะต้องนำมาใช้ทำงาน

(b) เอาต์พุต (outputs) ได้แก่ผลลัพธ์ที่ต้องการ และ (c) ความต้องการเพิ่มเติมใดๆ หรือข้อบังคับของการแก้ปัญหา ณ ขั้นตอนนี้ เราควรกำหนดรูปแบบที่ต้องการของผลลัพธ์ ซึ่งจะให้แสดงผล (ตัวอย่างเช่น เป็นตาราง มีหัวเรื่องของแต่ละสดมภ์) และพัฒนารายชื่อของตัวแปรปัญหาและความสัมพันธ์ของมัน ความสัมพันธ์เหล่านี้อาจเป็นสูตร

ถ้าขั้นตอนที่ 1 และ 2 กระทำไม่ถูกต้อง เราจะแก้ปัญหาผิด ให้อ่านประโยคปัญหาอย่างรอบคอบ ขั้นแรกให้ได้ความคิดที่ชัดเจนของปัญหา และขั้นที่สองกำหนดอินพุตและเอาต์พุต สิ่งที่จะช่วยเหลือได้คือการขีดเส้นใต้วลีในประโยคปัญหา เพื่อระบุถึงอินพุตและเอาต์พุต ดังแสดงในประโยคปัญหาข้างล่างนี้

Determine the total cost of apples given the number of pounds of apples and the cost per pound of apples.

อินพุต

ปริมาณของผลแอปเปิลที่ซื้อ (มีหน่วยเป็น pounds)

ราคาผลแอปเปิลต่อ pound (ดอลลาร์ต่อปอนด์)

เอาต์พุต

จำนวนเงินทั้งหมดของผลแอปเปิล (มีหน่วยเป็นดอลลาร์)

เมื่อรู้อินพุตและเอาต์พุตแล้ว ต่อไปพัฒนารายการสูตร ซึ่งระบุความสัมพันธ์ระหว่างสิ่งเหล่านี้ สูตรคือ

total cost = unit cost x number of units

คำนวณจำนวนเงินทั้งหมดของสิ่งของที่ซื้อ แล้วแทนตัวแปรเข้าไป ด้านนี้สูตรของปัญหา คือ

total cost of apples = cost per pound x pond of apples

ในบางสถานการณ์ เราอาจต้องทำข้อสมมติ หรือทำให้เกิดความง่ายขึ้น เพื่อให้ได้มาซึ่งความสัมพันธ์เหล่านี้

กระบวนการของการตัดทอนตัวแปรที่สำคัญ และความสัมพันธ์ของปัญหาออกมา เรียกว่า การนิยามนามธรรม (The process of extracting the essential variables and relationships of a problem is called abstraction.)

(3) ออกแบบ (Design)

การออกแบบอัลกอริทึมเพื่อแก้ปัญหา เราต้องเขียนกระบวนงานทำที่ละขั้นตอน จากนั้นทวนสอบว่าอัลกอริทึมแก้ปัญหาได้ตามต้องการ

อัลกอริทึม หรือขั้นตอนวิธี หมายถึง วิธี หรือรายการของขั้นตอนต่างๆ สำหรับการแก้ปัญหา (**Algorithm** is a recipe, or list of steps, for solving a problem.)

การเขียนอัลกอริทึม เป็นงานส่วนยากที่สุดของกระบวนการแก้ปัญหา เพราะฉะนั้นในตอนเริ่มต้นไม่ควรแก้ปัญหาในรายละเอียดของทุกส่วนของปัญหา แต่ควรฝึกวินัยตัวเราเองให้ใช้การออกแบบจากบนลงล่าง (top down design)

การออกแบบจากบนลงล่าง อาจเรียกว่าการแบ่งแยกและเอาชนะ (divide and conquer) ขั้นแรกเขียนขั้นตอนที่สำคัญ (major steps) หรือปัญหาย่อย (subproblems) ซึ่งจำเป็นต้องแก้ปัญหา จากนั้นแก้ปัญหาเดิม (original problem) โดยการแก้ปัญหาย่อยทุกชุดของมัน อัลกอริทึมคอมพิวเตอร์ส่วนใหญ่อย่างน้อยที่สุดจะประกอบด้วยปัญหาย่อยต่อไปนี้

อัลกอริทึมสำหรับปัญหาการเขียนโปรแกรม

1. อ่านข้อมูล (Read the data)
2. กระทำการคำนวณ (Perform the compentations)
3. แสดงผลลัพธ์ (Desplay the results)

เมื่อเรารู้จักปัญหาย่อยแล้ว เราสามารถทำงานกับปัญหาย่อยแต่ละชุดครั้งละหนึ่งชุด ตัวอย่างเช่น ขั้นตอนกระทำการคำนวณ อาจจำเป็นต้องแบ่งให้เป็นรายการของขั้นตอนต่างๆ ซึ่งมีรายละเอียดมากขึ้น เรียกว่า การแบ่งละเอียดของอัลกอริทึม (algorithm refinements)

การแบ่งละเอียดของอัลกอริทึม หมายถึง รายการอย่างละเอียดของขั้นตอนต่างๆ ซึ่งจำเป็นเพื่อแก้ปัญหาของหนึ่งขั้นตอน ในอัลกอริทึมเดิม (Algorithm refinement is a detailed list of steps needed to solve a particular step in the original algorithm.)

การตรวจสอบโดยผู้เขียนอัลกอริทึมเองมีความสำคัญ แต่บ่อยครั้งมักถูกมองข้าม ในส่วนของการออกแบบอัลกอริทึม

การตรวจสอบอัลกอริทึม กระทำอย่างรอบคอบทุกขั้นตอนของอัลกอริทึม (หรือการแบ่งละเอียดของมัน) เสมือนกับว่าเป็นคอมพิวเตอร์ทวนสอบว่าอัลกอริทึมทำงานได้เช่นที่เราตั้งใจให้เป็นเช่นนั้น

เราจะประหยัดเวลาและแรงงาน ถ้าพบข้อผิดพลาดของอัลกอริทึมเสียแต่เนิ่นๆ ในกระบวนการแก้ปัญหา

การตรวจสอบอัลกอริทึมโดยผู้เขียนอัลกอริทึม หมายถึง การจำลองการทำงานทีละขั้นตอนของคอมพิวเตอร์กระทำการกับอัลกอริทึม (**Desk checking** is the step-by-step simulation of the computer execution of an algorithm.)

(4) การทำให้เกิดผล (Implementation)

การนำอัลกอริทึมมาปฏิบัติให้เกิดผล อยู่ในขั้นตอนที่ 4 ของวิธีพัฒนาซอฟต์แวร์ เกี่ยวข้องกับการเขียนอัลกอริทึมให้เป็นโปรแกรม เราต้องแปลงผัน (convert) ขั้นตอนแต่ละชุดของอัลกอริทึมให้เป็นหนึ่งข้อความสั่ง หรือมากกว่าหนึ่งข้อความสั่ง ในภาษาเขียนโปรแกรม

การเขียนโปรแกรมเชิงโครงสร้าง (Structured programming)

หมายถึงวิธีเชิงวินัยของการเขียนโปรแกรมให้ได้โปรแกรมผลลัพธ์ที่อ่านและทำความเข้าใจได้ง่าย และมีข้อผิดพลาดน้อยลง องค์การของรัฐบาลและอุตสาหกรรมให้การสนับสนุนอย่างแข็งขันกับการเขียนโปรแกรมเชิงโครงสร้างเพราะว่า โปรแกรมเชิงโครงสร้างออกแบบในตอนเริ่มต้นง่ายมาก และง่ายต่อการบำรุงรักษาในระยะเวลายาวนาน

โปรแกรมเชิงโครงสร้าง หมายถึง โปรแกรมซึ่งเขียนง่าย ทำความเข้าใจง่าย และบำรุงรักษาง่าย (Structured program is a program that is easy to read, understand, and maintain.)

(5) การทดสอบ (Testing)

การทดสอบและการทวนสอบโปรแกรม หมายถึง การทดสอบโปรแกรมที่เสร็จสมบูรณ์แล้ว และทวนสอบว่ามันทำงานได้ตามต้องการ และไม่ควรเขียนเพียงแค่ทดสอบ

ครั้งเดียว แต่ต้องวิ่งโปรแกรมหลายๆ ครั้ง ใช้ชุดข้อมูลแตกต่างกัน ทำให้มั่นใจว่าโปรแกรมทำงานถูกต้องทุกสถานการณ์ ซึ่งจัดให้ในอัลกอริทึม

(6) การบำรุงรักษา (Maintenance)

การบำรุงรักษาและการปรับโปรแกรมให้เป็นปัจจุบัน หมายถึง การดัดแปรโปรแกรม เพื่อลบข้อผิดพลาดซึ่งตรวจพบก่อนหน้านั้น และรักษาโปรแกรมให้เป็นปัจจุบันเมื่อข้อบังคับของรัฐบาลหรือนโยบายของบริษัทมีการเปลี่ยนแปลง องค์การจำนวนมากบำรุงรักษาโปรแกรมให้ใช้งานได้นาน 5 ปีหรือมากกว่านั้น บ่อยครั้งหลังจากโปรแกรมเมอร์คนที่เขียนรหัสเดิมลาออกหรือย้ายไปสู่ตำแหน่งอื่น

2.2 การประยุกต์ใช้วิธีพัฒนาซอฟต์แวร์ (Applying the Software Development Method)

ตลอดหนังสือเล่มนี้ เราใช้ห้าขั้นตอนแรกของวิธีการพัฒนาซอฟต์แวร์ เพื่อแก้ปัญหาการเขียนโปรแกรม ปัญหาเหล่านี้นำเสนอเป็นกรณีศึกษา เริ่มต้นด้วย ประโยคปัญหา วิเคราะห์ปัญหา ออกแบบและแบ่งละเอียดอัลกอริทึมเริ่มต้น สุดท้ายปฏิบัติให้เกิดผลเป็นโปรแกรม Pascal วิ่งโปรแกรมตัวอย่างและอภิปรายว่าจะทดสอบโปรแกรมอย่างไร

กรณีศึกษา : การเปลี่ยนมาตรหน่วยวัด (Converting Units of Measurement)

ปัญหา เราทำงานในร้านขายผ้านำเข้าจากต่างประเทศ ผ้าส่วนใหญ่ที่เราซื้อมามีหน่วยวัดเป็นเมตร (square meters) แต่ลูกค้าของเรา ต้องการทราบปริมาณผ้าซึ่งจำนวนเท่ากันให้มีหน่วยวัดเป็นหลา (square yards)

จงเขียนโปรแกรมทำการเปลี่ยนหน่วยวัดนี้

วิเคราะห์

ขั้นแรก ในการแก้ปัญหานี้คือ กำหนดว่าเราต้องทำอะไร เราต้องเปลี่ยนมาตรหน่วยวัดจากระบบหนึ่งไปยังอีกระบบหนึ่ง เช่น จากหน่วยวัดเมตรเป็นหลา หรือในทางกลับกันจากหน่วยวัดหลาเป็นเมตร ปัญหาคือ เราซื้อผ้ามีหน่วยวัดเป็นเมตร ดังนั้น อินพุตคือ ปริมาณผ้ามีหน่วยวัดเป็นเมตร ลูกค้าต้องการทราบปริมาณผ้าซึ่งมีจำนวนเท่ากันในหน่วยวัดหลา ดังนั้น เอาต์พุตคือปริมาณผ้ามีหน่วยวัดเป็นหลา การเขียนโปรแกรม เราจำเป็นต้องทราบความสัมพันธ์ระหว่างเมตรกับหลา จากตารางการวัดแสดงไว้ว่า 1 เมตรเท่ากับ 1.196 หลา

ต่อไปคือรายการข้อมูลและสูตรที่เกี่ยวข้อง

SqMeters คือ หน่วยความจำซึ่งเก็บอินพุต

SqYards คือ หน่วยความจำเก็บผลลัพธ์โปรแกรมหรือเอาต์พุต

สูตร 1 เมตร = 1.196 หลา

ออกแบบ

ต่อไป เขียนอัลกอริทึมเพื่อแก้ปัญหา เริ่มต้นจากเขียนรายการหลักสามขั้นตอนหรือปัญหาย่อยของอัลกอริทึม

อัลกอริทึม

1. อ่านปริมาณผ้ามีหน่วยเป็นเมตร
2. เปลี่ยนหน่วยวัดปริมาณผ้าจากเมตรให้มีหน่วยเป็นหลา
3. แสดงผลปริมาณผ้ามีหน่วยเป็นหลา

ต่อไปตรวจดูว่ามีขั้นตอนใดหรือไม่ในอัลกอริทึม ซึ่งต้องทำการแบ่งให้ละเอียด หรือขั้นตอนนั้นชัดเจนอย่างบริบูรณ์หรือไม่ ขั้นที่ 1 (อ่านข้อมูล) และขั้นที่ 3 (แสดงผลค่า) เป็นขั้นตอนหลัก ไม่ต้องการแบ่งละเอียด ขั้นที่ 2 ตรงไปตรงมา แต่เพิ่มรายละเอียดได้

ขั้นที่ 2 การแบ่งละเอียด

2.1 ปริมาณผ้ามีหน่วยวัดเป็นหลา เท่ากับ 1.196 คูณด้วยปริมาณผ้าซึ่งมีหน่วยวัดเป็นเมตร อัลกอริทึมที่สมบูรณ์ มีการแบ่งละเอียด เป็นดังนี้

1. อ่านปริมาณผ้ามีหน่วยวัดเป็นเมตร
2. เปลี่ยนปริมาณผ้าให้มีหน่วยเป็นหลา
   2.1 ปริมาณผ้ามีหน่วยวัดเป็นหลา เท่ากับ 1.196 คูณด้วยปริมาณผ้า ซึ่งมีหน่วยวัดเป็นเมตร
3. แสดงผลปริมาณผ้ามีหน่วยวัดเป็นหลา

ตรวจสอบอัลกอริทึมแบบ desk check ก่อนจะทำการในขั้นตอนต่อไป ขั้นที่ 1 ถ้าปริมาณผ้าเท่ากับ 2 เมตร ขั้นที่ 2.1 เปลี่ยนเป็น 1.196 x 2.00 เท่ากับ 2.392 หลา ในขั้นที่ 3 ผลลัพธ์ถูกต้องควรแสดงผลเป็น 2.392 หลา

การปฏิบัติให้เกิดผล

จากอัลกอริทึมเขียนเป็นโปรแกรม Pascal ดังแสดงในรูป 2.1 Metric Conversion Program

```
Program Metric;
{Converts square metres to square yards.}

   const
      MetersTYards = 1.196;

   var
      SqMeters,          {input - fabric size in meters}
      SqYards : Real;    {output - fabric size in yards}

begin
   WriteLn ('Enter the fabric size in square meters>');
   ReadLn (Sqmeters);

   {Convert the fabric size to square yards.}
   Sqyards := MetersToYard * SqMeters;

   {Display the fabric size in square yards.}
   WriteLn ('The fabric size in square yards is', SqYards)
end.
```

เอาต์พุต เป็นดังนี้

```
Enter the fabric size in square meterd>
2.00
The fabric size in square yards is  2.3920000000E+00
```

10 ตัว

รูป 2.1 Metric Conversion Program

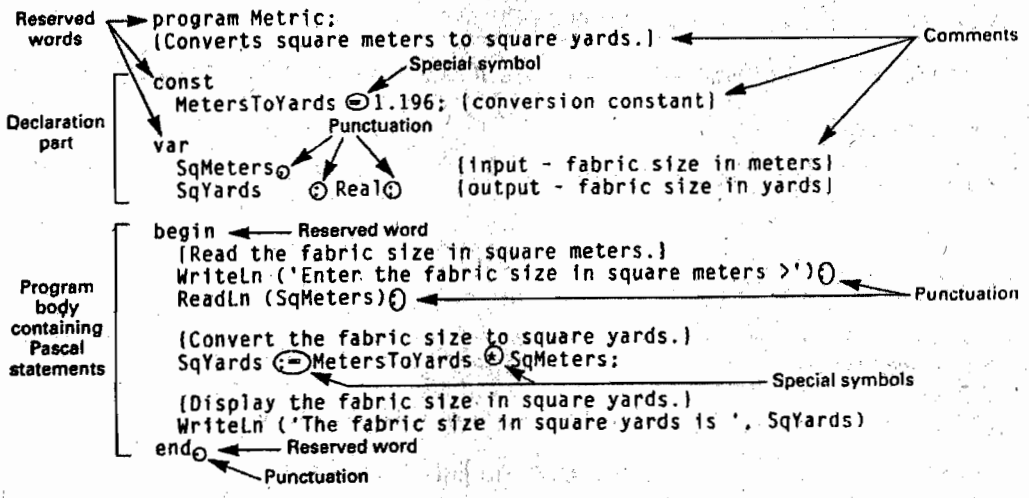
แบบฝึกหัด

1. จงเปลี่ยนอัลกอริทึมสำหรับโปรแกรมเปลี่ยนหน่วยวัด จากหน่วยวัดเป็นหลา ให้เป็นหน่วยวัดเมตร

2. จงเขียนรายการข้อมูล สูตร และอัลกอริทึมสำหรับโปรแกรมเปลี่ยนน้ำหนัก จากปอนด์เป็นกิโลกรัม

2.3 ภาพรวมของ Pascal, คำสงวน และไอเดนต์ไฟเออร์ (Overview of Pascal, Reserved Words, and Identifiers)

ข้อดีของ Pascal คือ โปรแกรมที่เราเขียนคล้ายภาษาอังกฤษ ซึ่งเราใช้ในชีวิตประจำวัน ถึงแม้ว่าเรายังไม่รู้ว่าจะเขียนโปรแกรมด้วยตนเองอย่างไร แต่เราสามารถอ่านและทำความเข้าโปรแกรมได้ ดูรูป 2.2



รูป 2.2

โปรแกรม Pascal แบ่งออกเป็นสองส่วน คือ

- ส่วนประกาศ (declaration part) และ
- ตัวโปรแกรม (program body)

ส่วนประกาศ หมายถึง ส่วนของโปรแกรม ซึ่งบอกตัวแปลชุดคำสั่ง หรือคอมไพเลอร์ (compiler) ถึงชื่อต่างๆ ของเซลล์หน่วยความจำในโปรแกรม (**Declaration part** is the part of a program that tells the compiler the names of memory cells in a program.)

ตัวโปรแกรม หมายถึง ข้อความสั่งกระทำการในโปรแกรม (Program body is the executable statements in a program.) เริ่มจากบรรทัด begin และตัวโปรแกรมประกอบด้วยข้อความสั่ง (ได้มาจากอัลกอริทึม) ซึ่งถูกแปลให้เป็นภาษาเครื่องและถูกกระทำการในเวลาต่อมา

คำในโปรแกรม เช่น begin และ end เรียกว่า คำสงวน ซึ่งมีความหมายเฉพาะให้กับคอมไพเลอร์

บรรทัดซึ่งประกอบด้วยข้อความในเครื่องหมายวงเล็กปีกกา เรียกว่า คอมเมนต์ (comments)

คอมเมนต์ คือ สารสนเทศเพิ่มเติมให้กับบุคคลซึ่งอ่านโปรแกรม ส่วนที่เป็นคอมเมนต์ จะไม่ได้รับการสนใจ (are ignosed) จากคอมไพเลอร์

สุดท้าย โปรแกรมประกอบด้วยเครื่องหมายกำกับวรรคตอนและสัญลักษณ์พิเศษ (*, =, :=)

comma (,) ใช้คั่น items ในรายการ

semicolon (;) ใช้จบบรรทัด (statement terminator) หรือตัวคั่นข้อความสั่ง (statement separator)

period (.) ใช้เมื่อจบบรรทัดสุดท้าย

ตาราง 2.1 อธิบายคำสงวนในรูป 2.2

Reserved word	Meaning
program	The first word of a Pascal program
const	Precedes the list of constants
var	Precedes the list of variables
begin	Start the program body
end	the last word of a Pascal program

ไอเดนติไฟเออร์มาตรฐาน (Standard Identifiers)

คำอื่นๆ ในรูป 2.2 คือ ไอเดนต์ไฟเออร์แบ่งออกเป็นสองชนิดคือ

- ไอเดนติไฟเออร์มาตรฐาน (stardard identifiers) หรือเรียกอีกอย่างหนึ่งว่า predefined identifiers
- ไอเดนติไฟเออร์นิยามโดยผู้ใช้ (naer-defined identifiers)

ไอเดนติไฟเออร์มาตรฐาน หมายถึง คำซึ่งมีความหมายเฉพาะ คล้ายคำสงวน แต่เป็นคำที่โปรแกรมเมอร์อาจให้นิยามใหม่เพื่อวัตถุประสงค์อื่นได้ (Standard identifier is a word having special meaning but one that a programmer may redefine.)

ตัวอย่างเช่น ชื่อของ predefined operations ได้แก่ ReadLn สำหรับ Read a Line WriteLn สำหรับ Write a Line

และชื่ออื่นๆ เช่น predefined data type Real สำหรับ real number Integer, Input, Output, Write, Read Maxint เป็นต้น

ดูภาคผนวก B รายชื่อของคำสงวนและไอเดนติไฟเออร์มาตรฐานของ Pascal

ไอเดนติไฟเออร์นิยามโดยผู้ใช้ (User-Defined Identifiers)

หมายถึง ชื่อซึ่งโปรแกรมเมอร์กำหนดเองให้กับเซลล์หน่วยความจำ เพื่อเก็บข้อมูลและผลลัพธ์โปรแกรม ชื่อการปฏิบัติการ (ดูบทที่ 3) ชื่อโปรแกรม ตัวอย่างเช่น Metric ในรูป 2.2 เป็นชื่อโปรแกรม และต้องอยู่ที่หัวเรื่องโปรแกรม (program heading)

กฎวากยสัมพันธ์สำหรับไอเดนติไฟเออร์ (Syntax Rules for Identifiers)

1. ไอเดนติไฟเออร์ต้องขึ้นต้นด้วยตัวอักษร (A → Z, a → z)

2. ไอเดนติไฟเออร์ประกอบด้วยการรวมกัน (combination) ของตัวอักษร และตัวเลข (0 → 9) ตัวขีดเส้นใต้ (underscore (_)) สำหรับ Turbo Pascal เท่านั้น

3. คำสงวนนำมาใช้เป็นไอเดนติไฟเออร์ไม่ได้

4. ใน Turbo Pascal ถ้าความยาวของไอเดนติไฟเออร์มากกว่า 63 ตัวอักขระ เฉพาะอักขระ 63 ตัวแรกเท่านั้นที่ถูกต้อง (valid)

ตัวอย่าง valid identifiers

Letter1, Letter2, Inches, Cent, CentPerInch, Hello, Variable

ตาราง 2.2 Invalid Identifiers

Invalid Identifier	Reason Invalid
1Letter	ขึ้นต้นด้วยตัวเลข
const	คำสงวน
var	คำสวงน
Two*Four	ตัวอักษร * ใช้ไม่ได้
Joe´s	ตัวอักขระ ใช้ไม่ได้

ข้อสังเกต

ไอเดนติไฟเออร์ Cont_Per_Inch เป็น invalid ใน standard Pascal แต่ Tutbo Pascal สามารถใช้สัญลักษณ์ขีดเส้นใต้ (underscor (_)) ได้ ดังนั้น ชื่อ Cont_Per_Inch จึงเป็น valid identifier ใน Turbo Pascal อย่างไรก็ตาม เพื่อให้โปรแกรมสามารถเคลื่อนย้ายได้ จึงแนะนำว่าควรหลีกเลี่ยงการใช้สัญลักษณ์ขีดเส้นใต้

ตาราง 2.3 คำสงวนและไอเดนติไฟเออร์ในรูป 2.2

คำสงวน	ไอเดนติไฟเออร์มาตรฐาน	ไอเดนติไฟเออร์นิยามโดยผู้ใช้
program, var, const, begin, and	Real, ReadLn WriteLn	Metric MetersToYards Sqmeters, SqYards

วากยสัมพันธ์การแสดงผลสำหรับหัวเรื่องโปรแกรม (Syntax Display for Program Heading)

หัวเรื่องโปรแกรม

รูปแบบ (1)	program pname;
standard Pascal (2)	program pname (Input, Output);

ตัวอย่าง program Hello;

program Hello (Input, Output);

ในที่นี้ pnane เป็นชื่อโปรแกรม หัวเรื่องโปรแกรมมีสองรูปแบบ รูปแบบที่สองแสดงว่าข้อมูลอินพุตอ่านจากระบบ Input (คีย์บอร์ด) ผลลัพธ์เอาต์พุตจะ write ไประบบ Output (จอภาพ) รูปแบบที่สองเท่านั้น จึงจะถูกต้อง (valid) ใน Standard Pascal

สไตล์โปรแกรม (Program Style)

การใช้อักษรตัวเล็กและตัวใหญ่ในโปรแกรม Pascal (The Use of Lowercase and Uppercase in Pascal Programs)

โปรแกรมที่ดูดี (look good) หมายถึง โปรแกรมซึ่งอ่านง่าย และทำความเข้าใจง่าย โปรแกรมส่วนใหญ่ถูกใช้งานหรือศึกษาโดยผู้อื่นซึ่งไม่ใช่ผู้เขียนโปรแกรมดังเดิม ในโลกที่เป็นจริงประมาณ 25% ของเวลาถูกใช้ไปกับการออกแบบและลงรหัส ส่วนเวลาที่เหลืออีก 75% ใช้เพื่อการบำรุงรักษา (ได้แก่ การปรับให้เป็นปัจจุบัน และการดัดแปรโปรแกรม) โปรแกรมที่เขียนอย่างปราณีต (neatly) และมีความหมายชัดเจน ทำให้งานของทุกคนง่ายขึ้น

โปรแกรมในหนังสือเล่มนี้ คำสงวนใช้อักษรตัวเล็ก (lowercase) ไอเดนติไฟเออร์ใช้ผสมกันระหว่างอักษรตัวเล็กและอักษรตัวใหญ่ ทั้งนี้อักษรตัวแรกของไอเดนติไฟเออร์ใช้ตัวใหญ่ ถ้าไอเดนติไฟเออร์ตัวนั้นประกอบด้วยคำสองคำหรือมากกว่าขึ้นไป อักษรตัวแรกของทุกคำให้ใช้ตัวใหญ่ ตัวอย่างเช่น MetersToYards ขอแนะนำว่าถ้าเราทำตามข้อตกลงนี้ (this convention) การแยกความแตกต่างระหว่างคำสงวนและไอเดนติไฟเออร์อื่นๆ จะง่ายขึ้น

คอมไพเลอร์ของ Turbo Pascal ไม่ให้ความแตกต่างระหว่างอักษรตัวเล็กและอักษรใหญ่ หมายความว่า คำสงวน const อาจเขียนเป็น CONST หรือไอเดนติไฟเออร์ ReadLn เขียนเป็น READLN ไม่แตกต่างกัน อย่างไรก็ตาม ตามข้อตกลงของเรา รูปแบบที่ควรเขียน คือ const และ ReadLn แต่ไอเดนติไฟเออร์มาตรฐาน มีความหมายพิเศษคล้ายคำสงวน นักวิทยาศาสตร์คอมพิวเตอร์จำนวนมากจึงเขียนไอเดนติไฟเออร์มาตรฐานในวิธีเดียวกับเขียนคำสงวน (คือเป็นอักษรตัวเล็กทั้งหมด) ถ้าเราชอบเช่นนี้ ให้เขียน ReadLn เป็น readln

สไตล์โปรแกรม (Program Style)

การเลือกชื่อไอเดนติไฟเออร์ (Choosing Identifier names)

เลือกชื่อที่มีความหมายสำหรับไอเดนติไฟเออร์ซึ่งนิยามโดยผู้ใช้ เพื่อให้ง่ายต่อการทำความเข้าใจ ตัวอย่างเช่น ไอเดนติไฟเออร์ ชื่อ Salary เป็นชื่อที่เหมาะสำหรับตัวแปรใช้เก็บเงินเดือนของบุคคล ในขณะที่ ไอเดนติไฟเออร์ ชื่อ S หรือ Bagel เป็นชื่อที่ไม่เหมาะกับตัวแปรเก็บเงินเดือนของบุคคล

การตั้งชื่อโดยใช้ตัวอักษรน้อยกว่าสามตัวและให้มีความหมายนั้นอาจเป็นงานยาก แต่ในทางตรงกันข้ามถ้าชื่อนั้นยาวเกินไป อาจมีข้อผิดพลาดในการพิมพ์ กฎที่มีเหตุมีผลคือใช้ชื่อที่มีความยาวระหว่าง 3 ถึง 10 ตัวอักขระ

ถ้าเราพิมพ์ไอเดนติไฟเออร์ผิด คอมไพเลอร์จะตรวจพบ เป็นข้อผิดพลาดวากย-สัมพันธ์ (syntax error) และแสดง error message เป็น undefined identifier ระหว่างการแปลโปรแกรม

แบบฝึกหัด 2.3

1. โปรแกรม Pascal หนึ่งโปรแกรม แบ่งออกเป็นกี่ส่วน อะไรบ้าง

2. ทำไมจึงไม่ควรใช้ไอเดนติไฟเออร์มาตรฐานเป็นชื่อเซลล์หน่วยความจำในโปรแกรม เราใช้คำสงวนเป็นชื่อเซลล์หน่วยความจำในโปรแกรมได้หรือไม่

3. จงบอกว่าไอเดนติไฟเออร์ต่อไปนี้เป็นกลุ่มใด

   a) Pascal reserved words

   b) standard identifiers

   c) valid identifiers

   d) invalid identifiers

end	ReadLn	Bill	program	Sue's	
Rate	start	begin	const	XYZ123	
123XYZ	ThisIsALongOne		Y=Z	Prog#2	'MaxCores'

2.4 แบบชนิดข้อมูลและการประกาศ (Data Types and Declarations)

ส่วนการประกาศของโปรแกรม Pascal สื่อสารคอมไพเลอร์กับชื่อของไอเดนติไฟเออร์ทั้งหมดซึ่งนิยามโดยผู้ใช้ที่ปรากฏในโปรแกรม และการใช้ไอเดนติไฟเออร์แต่ละตัวมันบอกคอมไพเลอร์ว่าเป็นสารสนเทศชนิดอะไรซึ่งจะเก็บในเซลล์หน่วยความจำแต่ละแห่ง

แบบชนิดข้อมูล (Data Types)

ค่าหนึ่งค่าถูกแทนในหน่วยความจำอย่างไรกำหนดโดยแบบชนิดข้อมูลของค่านั้น

แบบชนิดข้อมูล หมายถึง เซตของค่าต่างๆ และการดำเนินการซึ่งสามารถกระทำได้บนค่าเหล่านั้น (Data type is a set of values and operations that can be performed on those values.)

ใน standard Pascal มี predefined data types 4 ชนิด ได้แก่ Real (สำหรับ real numbers)

Integer (สำหรับ integers)

Char (สำหรับ single character values)

และ Boolean (สำหรับค่า True และ False)

ส่วน Turbo Pascal มี data type อีกหนึ่งชนิดคือ String ซึ่งเป็น nonstandard และง่ายต่อการประมวลผลลำดับของตัวอักขระ (ตัวอย่างเช่น ชื่อของบุคคล) แบบชนิดข้อมูลแต่ละชุดจะมีเซตของค่าต่างๆ และการดำเนินการซึ่งกระทำกับค่าเหล่านั้น ซึ่งแบบชนิดข้อมูลเหล่านี้จะอธิบายโดยละเอียดในบทที่ 7

ค่าหนึ่งค่าซึ่งเขียนภายในหนึ่งบรรทัดของโปรแกรม Pascal หรือพิมพ์เป็นหน่วยข้อมูล (data item) หนึ่งตัว เพื่อให้อ่านโดยโปรแกรมค่าซึ่งปรากฏในหนึ่งบรรทัดของโปรแกรมเรียกว่า **สัญพจน์** (literal)

(1) แบบชนิดข้อมูล Integer

ในวิชาคณิตศาสตร์ เลขจำนวนเต็มอาจเป็นบวกศูนย์หรือเป็นลบ และเลขที่ไม่มีเครื่องหมายกำกับ ถือว่าเป็นบวก แบบชนิดข้อมูล Integer ใช้แทนเลขจำนวนเต็มใน Pascal

เนื่องจากขนาดของหน่วยความจำมีจำกัด เพราะฉะนั้นเลขจำนวนเต็มทั้งหมดจึงไม่สามารถถูกแทนได้ Turbo Pascal สามารถแทนเลขจำนวนเต็มในพิสัยของค่า -32768 (เท่ากับ -2^{15}) ถึง 32767 (เท่ากับ $2^{15}-1$) มี predefined constant ชื่อ MaxInt ซึ่งค่าของมันแทนเลขจำนวนเต็มบวกใหญ่ ที่สุด เลข Integer มีเครื่องหมาย comma ไม่ได้ ตัวอย่าง valid integers·

-10500 0 435 15 -25

เราสามารถอ่านและแสดงผล integers กระทำการดำเนินการคำนวณ (บวก ลบ คูณ และหาร) และเปรียบเทียบ integer สองตัว

(2) แบบชนิดข้อมูล Real

เลขจำนวนจริงมีภาคจำนวนเต็ม (integral part) และภาคเศษส่วน (fractional part) ทั้งสองส่วนนี้คั่นด้วยจุดทศนิยม ใน Pascal แบบชนิดข้อมูล real แทนเลขจำนวนจริง และ Real lileral ต้องขึ้นต้นและจบด้วยเลขโดด ดังนั้นเศษส่วน -.25 ใน Pascal เขียนเป็นสัญพจน์ real -0.25 และเลข 64. ใน Pascal ต้องเขียนเป็น 64.0 ตามลำดับ

เราสามารถใช้สัญกรณ์เชิงวิทยาศาสตร์ (scientific notation) แทนค่าที่ใหญ่มาก และแทนค่าที่เล็กมากได้

ตัวอย่างเช่น เลขจำนวนจริง 1.23×10^5 มีค่าเท่ากับ 123000.0 เมื่อเลขชี้กำลังเท่ากับ 5 หมายความว่า จุดทศนิยมย้ายไปทางขวามือ 5 ตำแหน่ง สัญกรณ์เชิงวิทยาศาสตร์ใน Pascal เขียนดังนี้ 1.23E5 หรือ 1.23E+5 ถ้าเลขชี้กำลังมีเครื่องหมายเป็นลบ จุดทศนิยมจะย้ายไปทางซ้ายมือ ตัวอย่างเช่น 0.34E-4 มีค่าเท่ากับ 0.000034

ตาราง 2.4 Valid และ Invalid real literals

Valid real literal	Invalid real literal
3.14159	150 (ไม่มีจุดทศนิยม)
0.005	.12345 (ไม่มีเลขหน้าจุดทศนิยม)
12345.0	16. (ไม่มีเลขหลังจุดทศนิยม)
15.0E-04 (มีค่าเท่ากับ 0.0015)	-15E-0.3 (เลขชี้กำลังมีจุดทศนิยมไม่ได้)
2.345E2 (มีค่าเท่ากับ 234.5)	12.5E.3 (เลขชี้กำลังมีจุดทศนิยมไม่ได้)
1.15E-3 (มีค่าเท่ากับ 0.00115)	.123E3 (ไม่มีเลขหน้าจุดทศนิยม)
12E + 5 (มีค่าเท่ากับ 1200000.0)	

บรรทัดสุดท้ายแสดงให้เห็นว่าสัญพจน์ Real ในสัญกรณ์เชิงวิทยาศาสตร์ของ Pascal เขียนโดยไม่มีจุดนิยมได้

เราสามารถอ่านและแสดงผลเลขจำนวนจริง กระทำการดำเนินการคำนวณ (บวก, ลบ, คูณ และหาร) และเปรียบเทียบ

(3) แบบชนิดข้อมูล Char

แบบชนิดข้อมูล Char แทนค่าตัวอักขระหนึ่งตัว - ตัวอักษร เลขโดด หรือสัญลักษณ์พิเศษ สัญพจน์ (literal) ชนิด Char แต่ละตัวต้องอยู่ภายในเครื่องหมาย apostrophes (single quotes) ตัวอย่างเช่น

'A', 'Z', '2', '9', 't', ':', '"', ' '

สัญพจน์ตัวซึ่งอยู่ติดกับสัญพจน์ตัวสุดท้ายคืออักขระ "

สัญพจน์ตัวสุดท้ายแทนอักขระว่าง (blank character)

ถึงแม้ว่า literal ชนิด Char ในโปรแกรมต้องอยู่ในเครื่องหมาย apostrophe แต่ค่าของข้อมูล (data value) ชนิด Char ไม่ต้องมีเครื่องหมาย apostrophe ตัวอย่างเช่น การใส่ตัวอักษร Z เป็นหน่วยข้อมูล (data item) ชนิดตัวอักขระ ซึ่งจะถูกอ่านโดยโปรแกรมกดปุ่ม Z ไม่ใช่ลำดับ 'Z'

ข้อมูลชนิด Char นำไปคำนวณไม่ได้ หมายความว่า นิพจน์ '3' + 5 เขียนผิด (invalid) ใน Pascal อย่างไรก็ตาม เราสามารถเปรียบเทียบตัวอักขระ อ่าน และแสดงผลได้

(4) แบบชนิดข้อมูล Boolean

ไม่เหมือนกับแบบชนิดข้อมูลอื่นๆ แบบชนิดข้อมูล Boolean มีเพียงสองค่าที่เป็นไปได้ คือ True และ False

เราสามารถใช้แบบชนิดข้อมูลนี้ แทนค่ามีเงื่อนไข เพื่อให้โปรแกรมทำการตัดสินใจ เราสามารถแสดงผลข้อมูล Boolean แต่ไม่สามารถอ่านหน่วยข้อมูลชนิด Boolean

(5) แบบชนิดข้อมูล String

แบบชนิดข้อมูล Real Interger Char และ Boolean ทั้งหมดนี้เป็น **แบบชนิดข้อมูลมาตรฐาน (Standard data type)** ใน Turbo Pascal มีแบบชนิดข้อมูลประเภทที่ห้าเรียกว่า string (เป็นคำสงวนใน Turbo Pascal) ซึ่งหมายถึงลำดับของตัวอักขระอยู่ภายในเครื่องหมาย apostrophes ตัวอย่างเช่น บรรทัดข้างล่างนี้มีสัญพจน์ชนิด string 4 ชุด

'ABCD', '1234', 'True', 'Enter fabric size in square meters'

โปรดสังเกตว่า สายอักขระ '1234' ไม่ได้ถูกเก็บในวิธีซึ่งเหมือนกับ integer 1234 และ string นี้ใช้กับตัวดำเนินการคำนวณไม่ได้

สายอักขระ 'True' เก็บแตกต่างจากค่า Boolean True เช่นกัน

ใน Turbo Pascal, เราสามารถอ่าน strings, เก็บในหน่วยความจำ, เปรียบเทียบและแสดงผลได้ ค่าของ string ประกอบด้วยตัวอักขระได้มากถึง 255 ตัว เมื่อใส่ค่า string ให้ถูกอ่านโดยโปรแกรมไม่ต้องคีย์เครื่องหมาย apostrophes เพียงแค่พิมพ์ค่าข้อมูลชนิด Char

ถึงแม้ว่า standard Pascal ไม่มีแบบชนิดข้อมูล string แต่เราสามารถใช้ string literals ได้ ซึ่งปกติปรากฏในคำสั่งใช้แสดงผลสารสนเทศ

วัตถุประสงค์ของแบบชนิดข้อมูล (Purpose of Data Types)

การใช้แบบชนิดข้อมูลแตกต่างกัน ทำให้คอมไพเลอร์ทราบว่าการดำเนินการชุดไหนไม่ถูกต้อง (invalid) สำหรับเซลล์หน่วยความจำที่ใช้ในโปรแกรม ถ้าเราพยายามจัดดำเนินการค่าในหน่วยความจำในวิธีไม่ถูกต้อง (ตัวอย่างเช่น บวกค่าชนิดตัวอักขระสองจำนวน) คอมไพเลอร์จะแสดงผล error message บอกว่า สิ่งนี้เป็นการดำเนินการที่ผิด ในทำนองเดียวกัน ถ้าเราพยายามเก็บค่าชนิดที่ผิดในเซลล์หน่วยความจำ (ตัวอย่างเช่น สายอักขระในหน่วยความจำซึ่งมีชนิดเป็น Interger) เราจะได้รับ error message การตรวจพบข้อผิดพลาดเหล่านี้ ทำให้คอมพิวเตอร์ไม่กระทำการดำเนินการในสิ่งที่ไม่มีความหมายในหัวข้อถัดไป เราจะอภิปรายว่า การบอกคอมไพเลอร์ Pascal ถึงแบบชนิดข้อมูลของเซลล์หน่วยความจำแต่ละเซลล์ทำอย่างไร

การประกาศ (Declarations)

เราบอกคอมไพเลอร์ Pascal ถ้าชื่อต่างๆ ของเซลล์หน่วยความจำซึ่งใช้ในโปรแกรมและชนิดของสารสนเทศซึ่งเก็บในเซลล์เหล่านี้ผ่านทางการประกาศค่าคงตัวและการประกาศตัวแปร

การประกาศค่าคงตัว (Constant Declarations)

การประกาศค่าคงตัว

```
const
    MetersToYares = 1.196;
```

ระบุว่าไอเดนติไฟเออร์ MetersToYards เป็นชื่อของเซลล์หน่วยความจำเก็บเลขจำนวนจริง 1.196 ตลอดเวลา

ไอเดนติไฟเออร์ MetersToYards เรียกว่า ค่าคงตัว

ค่าคงตัว หมายถึง เซลล์หน่วยความจำซึ่งค่าของมันเปลี่ยนแปลงไม่ได้ (Constant is a memory cell whose value cannot change.)

Pascal กำหนดแบบชนิดข้อมูลของ MetersToYards (ชนิด Real) จากรูปแบบของสัญพจน์ (1.196) ที่มันแทน เราใช้ค่าคงตัวเฉพาะเมื่อค่าข้อมูลนั้นไม่มีการเปลี่ยนแปลง (ตัวอย่างเช่น 1 หลามีค่าเท่ากับ 1.196 เมตรเสมอ) เราไม่สามารถเขียนคำสั่งให้เปลี่ยนค่าของค่าคงตัว

ตัวอย่าง การประกาศค่าคงตัว

```
const
    MyLargeInteger = 9999;              {value is 9999}
    MySmallInteger = -MyLargeInteger; {value is -9999}
    Star = ' * ';                       {value is symbol *}
    FirstMonth = 'January',             {value is string 'Junuary'}
```

ประกาศค่าคงตัวสี่ตัวที่แตกต่างกัน ค่าของค่าคงตัวตัวที่สอง, MySmallInteger, ขึ้นอยู่กับค่าคงตัวตัวแรก, MyLargeInteger, ค่าคงตัวตัวที่สาม, Star, มีค่าเป็นชนิด Char และค่าคงตัวตัวที่สี่, FirstMonth, มีค่าเป็น string

Syntax Display

Constant Declaration

Form : `Const constant = value;`

ตัวอย่าง `Const MyPi = 3.14159;`

มีความหมายดังนี้ value เป็นค่าที่เกี่ยวข้องกับไอเดนติไฟเออร์ constant และค่านี้เปลี่ยนแปลงไม่ได้ value อาจเป็น literal หรือ constant ซึ่งให้นิยามมาแล้ว สำหรับ numeric value มีเครื่องหมายได้

การประกาศ constant อาจมีมากกว่าหนึ่งตัว ซึ่งอยู่หลังคำสงวน const ให้ใส่เครื่องหมาย ; ที่ตอนจบของการประกาศค่าคงตัวแต่ละตัว (ดูตัวอย่างข้างต้น)

การประกาศตัวแปร (Variable Declarations)

เซลล์หน่วยความจำซึ่งใช้เก็บข้อมูลอินพุตของโปรแกรม และผลลัพธ์จากการคำนวณของมันเรียกว่า ตัวแปร เพราะว่าค่าซึ่งเก็บไว้ในหน่วยความจำเปลี่ยนแปลงได้ขณะที่โปรแกรมกระทำการ

ตัวแปร หมายถึง เซลล์หน่วยความจำซึ่งค่าของมันเปลี่ยนแปลงได้ (Variable is a memory cell whose value can change.)

ตัวอย่าง การประกาศตัวแปร

```
var
    SqMeters,          {input-fabric size in meters}
    SqYards : Real;    {output-fabric size in yards}
```

ในรูป 2.1 มีการกำหนดชื่อตัวแปรสองตัว (SqMeters, SqYards) ใช้เก็บเลขจำนวนจริง โปรดสังเกตว่า Pascal ไม่สนใจคอมเมนต์ในเครื่องหมายวงเล็บปีกกา ซึ่งอธิบายการใช้ของตัวแปรแต่ละตัว ในการประกาศตัวแปร ไอเดนติไฟเออร์มาตรฐาน (ตัวอย่างเช่น Real) หลังเครื่องหมาย : บอกคอมไพเลอร์ว่าเป็นชนิดของข้อมูล (เช่นเป็นเลขจำนวนจริง) เก็บในชื่อตัวแปรนั้น

เราสามารถประกาศตัวแปรสำหรับแบบชนิดข้อมูลได้ทุกชนิด

Syntax Display

การประกาศตัวแปร (Variable Declaration)

Form : `var variable list : data type;`

ตัวอย่าง :
```
var
    x, y     : Real;
    Me, You  : Integer;
```

มีความหมายดังนี้ : เซลล์หน่วยความจำถูกจัดสรรให้กับตัวแปรแต่ละตัวใน variable list แบบชนิดข้อมูล (Real, Integer เป็นต้น) ของตัวแปรแต่ละตัวถูกกำหนดระหว่าง colon (:) และ semicolon (;) เครื่องหมาย comma ใช้คั่นตัวแปรใน variable list คำสงวน var อาจเขียนเพียงครั้งเดียว แล้วตามด้วย variable list หลายชุด และ data types ที่เกี่ยวข้องแต่ละชุดจบด้วยเครื่องหมาย semicolon

แบบฝึกหัด 2.4

1. a) จงเขียนเลขต่อไปนี้ในสัญกรณ์ฐานสิบปกติ (normal decimal notation)

   103E-4 1.2345E+6 123.45E+3

   b) จงเขียนเลขต่อไปนี้ในสัญกรณ์เชิงวิทยาศาสตร์ของ Pascal (Pascal scientific notation)

   1300 123.45 0.00426

2. จงระบุว่า ค่าสัญพจน์ (literal values) ต่อไปนี้ชุดใดถูกต้อง (valid) ใน Pascal และชุดใดไม่ถูกต้อง (Invalid) และให้บอกแบบชนิดข้อมูล (data types) ของค่าสัญพจน์ที่ถูกต้องทุกตัว

15 'XYX' ' * ' $25.123 15. -999 .123

'X' "X" '9' '-5' True 'True'

3. จงบอกเหตุผลที่ค่าของ pi (3.14159) ควรจะเก็บเป็นค่าคงตัวในหน่วยความจำ

4. อะไรควรเป็นแบบชนิดข้อมูลของตัวแปรที่ดีที่สุด สำหรับ

a) พื้นที่ของวงกลมซึ่งมีหน่วยเป็นตารางนิ้ว (square inches)

b) จำนวนรถยนต์ซึ่งวิ่งผ่านสี่แยกแห่งหนึ่งในหนึ่งชั่วโมง

c) ชื่อของเรา

d) ตัวอักษรตัวแรกของนามสกุลของเรา

การเขียนโปรแกรม

1. จงเขียนหัวเรื่องของโปรแกรมและการประกาศสำหรับโปรแกรมชื่อ Mine ซึ่งมีค่าคงตัว MyPi (3.14159)

ตัวแปร Radins, Area และ Circumf นิยามเป็น Real

ตัวแปร NumCirc เป็น Integer และตัวแปร CircName เป็น string

2.5 ข้อความสั่งกระทำการ (Excentable Statements)

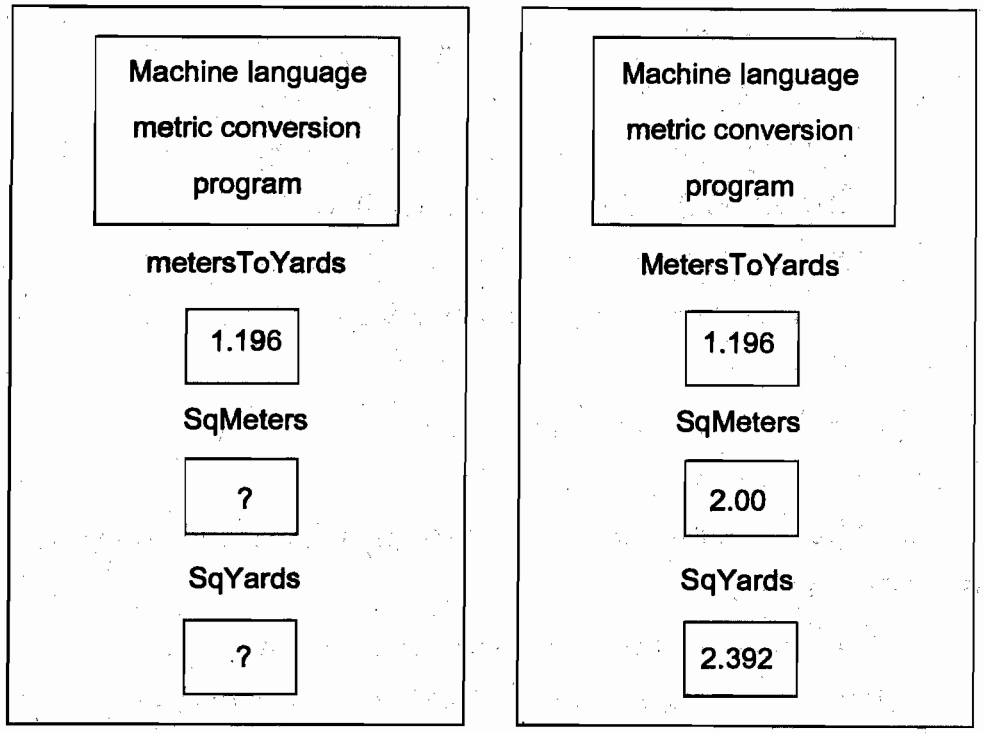
ข้อความสั่งกระทำการอยู่ถัดจากส่วนการประกาศ และอยู่หลังคำสงวน begin

ข้อความสั่งกระทำการ หมายถึง ข้อความสั่ง Pascal ใช้เขียนหรือลงรหัสอัลกอริทึมและการแบ่งละเอียดของอัลกอริทึม

คอมไพเลอร์ Pascal แปลข้อความสั่งกระทำการให้เป็นภาษาเครื่อง (machine language) คอมพิวเตอร์กระทำการ (executes) เวอร์ชันภาษาเครื่องของข้อความสั่งเหล่านี้เมื่อเราวิ่งโปรแกรม

โปรแกรมในหน่วยความจำ (Programs in Memory)

ก่อนตรวจสอบข้อความสั่งกระทำการในโปรแกรมเปลี่ยนหน่วยวัดขนาด (รูป 2.1) ให้ดูหน่วยความจำคอมพิวเตอร์ก่อนและหลังการกระทำการ รูป 2.3 (a) แสดงการบรรจุโปรแกรมในหน่วยความจำและพื้นที่หน่วยความหลังจากกระทำการโปรแกรม เครื่องหมายคำถามในเซลล์หน่วยความจำ SqMeters และ SqYards แสดงว่าค่าของเซลล์เหล่านี้ยังไม่ถูกนิยาม (undefined) ก่อนเริ่มต้นกระทำการโปรแกรม ระหว่างกระทำการโปรแกรมค่าข้อมูล 2.00 ถูกอ่านไว้ในตัวแปร SqMeters หลังจากโปรแกรมถูกกระทำการแล้ว ตัวแปรถูกนิยามดังแสดงในรูป 2.3 (b)



รูป 2.3 หน่วยความจำก่อนและหลังการกระทำการของโปรแกรม

ข้อความสั่งกำหนดค่า (Assignment Statements)

ข้อความสั่งกำหนดค่า หมายถึง คำสั่งซึ่งเก็บค่าหรือผลลัพธ์ของการคำนวณในตัวแปร (Assignment statement is an instruction that stores a value or a computational result in a variable.)

การดำเนินการส่วนใหญ่ในโปรแกรมใช้ข้อความสั่งกำหนดค่า

ตัวอย่าง

```
SqYards := MetersToYards * SqMeters
```

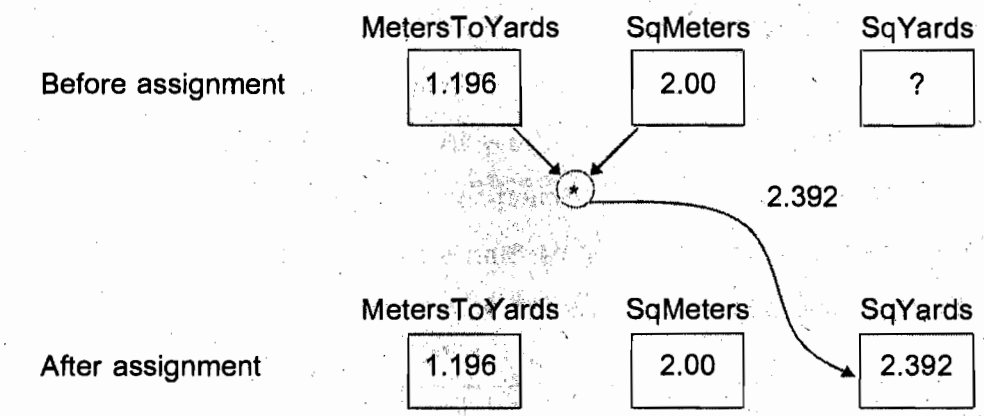
มีความหมายดังนี้ ค่าคงตัว MetersToYard คูณกับตัวแปร SqMeters ได้ผลลัพธ์เก็บไว้ที่ตัวแปร SqYards

เซลล์หน่วยความจำสำหรับ MetersToYards และ SqMeters ทั้งคู่ต้องเก็บสารสนเทศที่ถูกต้อง (ในกรณีนี้คือเลขจำนวนจริง) ก่อนกระทำการข้อความสั่งกำหนดค่า รูป 2.4

แสดงให้เห็นว่า contents ในหน่วยความจำก่อนและหลังกระทำการข้อความสั่งกำหนดค่า มีเฉพาะ contents ใน SqYards เท่านั้นที่เปลี่ยนแปลง ใน Pascal

สัญลักษณ์ := เรียกว่าตัวกำหนดค่า (assignment operator)

อ่านว่า "becomes", "gets" หรือ "takes the value of" ไม่ใช่ "equals" ระหว่างสัญลักษณ์ : และ = มีอักขระว่างไม่ได้



รูป 2.4 ผลของ SqYards := MetersToYards * SqMeters

Syntax Display

ข้อความสั่งกำหนดค่า (Assignment Statement)

Form : variable := expression

ตัวอย่าง X := Y + Z + 2.0

มีความหมายดังนี้ ตัวแปร หน้าเครื่องหมายตัวกำหนดค่า ถูกกำหนดให้มีค่าเป็นค่าของนิพจน์ขวามือ และในการประมวลผลค่าก่อนหน้า (previous value) ของตัวแปรจะถูกทำลาย

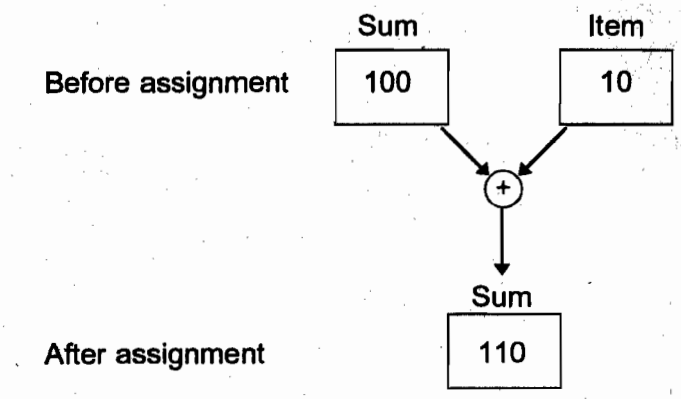
นิพจน์ในที่นี้อาจเป็น ตัวแปร ค่าคงตัว สัญพจน์ หรือ combination ของสิ่งเหล่านี้เชื่อมต่อด้วยตัวดำเนินการที่เหมาะสม (ตัวอย่างเช่น +, -, และ /) แบบชนิดข้อมูลของนิพจน์และตัวแปรทางซ้ายมือของเครื่องหมาย := ต้องเหมือนกัน ยกเว้นเมื่อนิพจน์มีชนิดข้อมูลเป็น Integer ส่วนตัวแปรมีชนิดข้อมูลเป็น Real หรือนิพจน์มีชนิดข้อมูลเป็น Char ส่วนตัวแปรมีชนิดข้อมูลเป็น string

ตัวอย่าง 2.2

ใน Pascal เราสามารถเขียนข้อความสั่งกำหนดค่าอยู่ในรูปแบบ

Sum := Sum + Item

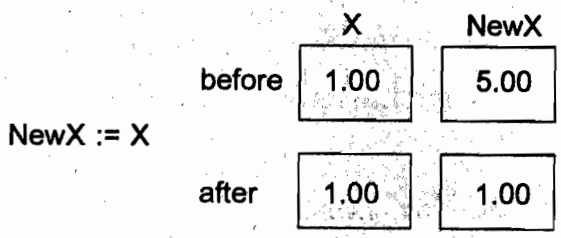
เมื่อตัวแปร Sum ปรากฏทั้งสองด้านของตัวกำหนดค่า สิ่งนี้ไม่ใช่สมการพีชคณิต แต่เป็นการฝึกเขียนโปรแกรมร่วมชนิดหนึ่ง คำสั่งนี้มีความหมายว่า ให้บวกค่าปัจจุบันของ Sum กับค่าของ Item ได้ผลลัพธ์แล้วเก็บไว้ใน Sum ค่าก่อนหน้าของ Sum จะถูกทำลาย ดังแสดงในรูป 2.5 ส่วน ค่าของ Item ไม่เปลี่ยนแปลง



รูป 2.5 ผลของ Sum := Sum + Item

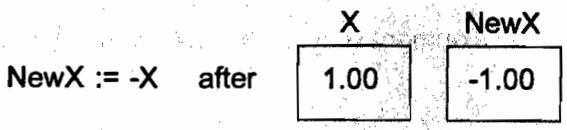
ตัวอย่าง 2.3

เราสามารถเขียนข้อความสั่งกำหนดค่าเพื่อกำหนดค่าของตัวแปรหนึ่งตัวหรือค่าคงตัวให้กับตัวแปร ถ้า X และ NewX เป็นตัวแปรชนิด Real ข้อความสั่งข้างล่างนี้



หมายถึง ทำสำเนาค่าของตัวแปร X ไว้ที่ตัวแปร NewX

ข้อความสั่ง



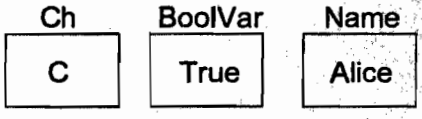
หมายถึง สั่งคอมพิวเตอร์ให้เอาค่าของ X เปลี่ยนค่านี้เป็นลบ และเอาผลลัพธ์ไปเก็บใน NewX ตัวอย่างเช่น ถ้า X มีค่าเท่ากับ 3.5 ค่าของ NewX เท่ากับ -3.5 ทั้งสองคำสั่งข้างต้นนี้ ไม่เปลี่ยนค่าของ X

ตัวอย่าง 2.4

สมมติว่า Ch เป็นข้อมูลชนิด Char, BoolVar เป็นข้อมูลชนิด Boolean และ Name เป็นข้อมูลชนิด string ข้อความข้างล่างนี้ถูกต้องทั้งหมดใน Pascal โปรดสังเกตว่า ระหว่างข้อความสั่งกระทำการ เราต้องใส่เครื่องหมาย semicolon คั่นเสมอ

```
Var       Ch   :  Char;   Name  :  string;
        Boolvar  :  Boolean;
           Ch    := 'C';
        BoolVar  := True;
           Name  := 'Alice';
```

ต่อไปเราจะแสดงหน่วยความจำ หลังจากข้อความสั่งเหล่านี้ถูกกระทำการ จะเห็นว่าเครื่องหมาย apostrophes จะไม่ถูกเก็บเมื่อข้อมูลเป็นตัวอักขระ หรือข้อมูลสายอักขระ และค่า Boolean True เก็บในตัวแปร BoolVar



เนื่องจาก Pascal แทนค่าของแบบชนิดข้อมูลแต่ละชนิดแตกต่างกัน ดังนั้นค่าที่กำหนดให้และตัวแปรซึ่งรับค่านั้นจึงจำเป็นต้องเข้ากันได้ (assignment compatible) หมายความว่าตัวแปรและค่าต้องเป็นชนิดเดียวกัน ยกเว้นค่าชนิด Integer เท่านั้นที่ตัวแปรซึ่งรับค่าของมันเป็นชนิด Real ได้

ข้อความสั่งกำหนดค่าต่อไปนี้ ไม่ถูกต้อง (invalid)

```
Ch      := 5;       {Invalid assignment of integer to type Char variable}
Ch      := Name,    {invalid assignment of type string variable to type Char
                     variable}
Name    := True;    {invalid assignment of type Boolean value to type string
                     variable}
BoolVar := 'False'  {invalid assignment of type string literal to type Boolean
                     variable}
```

การดำเนินการและกระบวนงานอินพุต/เอาต์พุต (Input/Output Operations and Procedures)

ข้อมูลเก็บในหน่วยความจำได้สามวิธี : เป็นค่าคงตัว กำหนดค่าให้กับตัวแปรหรืออ่านข้อมูลเก็บไว้ในตัวแปร

สองวิธีแรกอภิปรายไปแล้ว เราอาจจะใช้วิธีที่สามอ่านข้อมูลเก็บไว้ในตัวแปร ถ้าเราต้องการให้โปรแกรมจัดดำเนินการข้อมูลต่างๆ ทุกครั้งที่มันกระทำการ การอ่านข้อมูลเก็บไว้ในหน่วยความจำ เรียกว่า การดำเนินการอินพุต

การดำเนินการอินพุต หมายถึง คำสั่งซึ่งอ่านข้อมูลเก็บไว้ในหน่วยความจำ (Input operation is an instruction that reads data into memory.)

ขณะกระทำการ โปรแกรมกระทำการคำนวณและเก็บผลลัพธ์ในหน่วยความจำ ผลลัพธ์เหล่านี้สามารถถูกนำออกแสดงผลให้กับผู้ใช้โปรแกรมด้วยการดำเนินการเอาต์พุต

การดำเนินการเอาต์พุต หมายถึง คำสั่งซึ่งแสดงผลสารสนเทศที่เก็บในหน่วยความจำ (Output operation is an instruction that displays information stored in memory.)

การดำเนินการอินพุต/เอาต์พุต ทั้งหมดใน Pascal ถูกกระทำโดยหน่วยโปรแกรมพิเศษ ซึ่งเรียกว่า กระบวนงานอินพุต/เอาต์พุต

กระบวนงานอินพุต/เอาต์พุต หมายถึง กระบวนงาน Pascal ซึ่งกระทำการดำเนินการอินพุตหรือเอาต์พุต (Input/Output procedure is a Pascal procedure that performs an input or output operatins.)

กระบวนงานอินพุต/เอาต์พุต เป็นส่วนของคอมไพเลอร์ Pascal และชื่อของมันเป็นไอเดนติไฟเออร์มาตรฐาน (ตัวอย่างเช่น ReadLn และ WriteLn) ใน Pascal ข้อความสั่งเรียกกระบวนงาน ใช้เพื่อเรียกหรือใช้งานกระบวนงาน

ข้อความสั่งเรียกกระบวนงาน หมายถึง คำสั่งซึ่งเรียกหรือใช้งานกระบวนงาน (A procedure call statement is an instruction that calls or activates a procedure.)

การเรียกกระบวนงานคล้ายกับการขอร้องเพื่อนให้ทำงานด่วน เราบอกเพื่อนว่าให้ทำอะไร (แต่ไม่ได้บอกว่าสิ่งนั้นจะทำอย่างไร) และคอยเพื่อนให้รายงานกลับว่างานนั้นทำเสร็จแล้ว หลังจากได้ยินแล้ว เราจึงสามารถดำเนินการต่อไป และทำสิ่งอื่นๆ

กระบวนงาน WriteLn (The WriteLn Procedure)

การดูผลลัพธ์ของการกระทำการโปรแกรม เราต้องมีวิธีการกำหนดว่าค่าของตัวแปรอะไร ที่ต้องการแสดงผล ในรูป 2.1 ข้อความสั่งเรียกกระบวนงาน

WriteLn ('The fabric size in square yards is', SqYards) เรียกกระบวนงาน WriteLn ให้แสดงผลหนึ่งบรรทัดของโปรแกรมเอาต์พุต ซึ่งมี items สองตัว คือ string literal 'The baric size ... is' และค่าของ SqYards สำหรับ string literal ตัวอักขระภายในเครื่องหมาย apostrophes จะถูกพิมพ์ แต่ไม่พิมพ์เครื่องหมาย apostrophes กระบวนงาน WriteLn แสดงผลดังนี้

The fabric size in square yards is 2.3920000000E+00

ถ้าไม่จัดรูปแบบข้อมูลเอาต์พุต ค่าที่เป็น real จะแสดงผลในรูปสัญกรณ์เชิงวิทยาศาสตร์ (2.3920000000E+00 คือ 2.392)

WriteLn โดยไม่มีรายการเอาต์พุต (WriteLn Without an Output List)

ข้อความสั่ง

```
WriteLn ('The fabric size in square yards is', SqYards);
WriteLn;
WriteLn ('Metric conversion completed')
```

จะแสดงผลดังนี้

The fabric size in square yards is 2.3920000000E+00

Metric conversion completed

เนื่องจาก WriteLn บรรทัดที่สองไม่มีรายการเอาต์พุต จึงมีบรรทัดว่าง (blank line) อยู่ตรงกลางของเอาต์พุต การกระทำของ WriteLn ทำให้ตัวชี้ตำแหน่ง (cursor) ข้ามไปหนึ่งบรรทัดแล้วขึ้นบรรทัดถัดไป

Syntax Display

WriteLn Procedure

รูปแบบ :

```
WiteLn (output list)
WriteLn
```

ตัวอย่าง

```
WriteLn ('My height in inches is.' ; Heigh)
```

มีความหมายดังนี้ กระบวนงาน WriteLn แสดงผลค่าของตัวแปรแต่ละตัวหรือค่าคงตัวตามลำดับที่ปรากฏในรายการเอาต์พุต จากนั้นขึ้นบรรทัดใหม่ ตำแหน่งตัวชี้อยู่ที่บรรทัดถัดไป สำหรับ string literal พิมพ์โดยไม่มีเครื่องหมาย apostrophes ถ้ากระบวนงาน WriteLn ไม่มีรายการเอาต์พุต ตำแหน่งตัวชี้จะขึ้นบรรทัดใหม่และอยู่ที่สดมภ์แรก

กระบวนงาน Write (The Write Procedure)

Pascal มีกระบวนงานเอาต์พุต ชนิดที่สองคือ Write ซึ่งคล้ายกับ WriteLn ทุกอย่างยกเว้นตำแหน่งตัวชี้ไม่เลื่อนขึ้นบรรทัดใหม่ หลังจากแสดงผลรายการเอาต์พุตแล้ว

```
Write ('The fabric size in square yards is ');
WriteLn (Sqyards)
```

ซึ่งแสดงผลบรรทัดเอาต์พุตเหมือนกับข้อความสั่งหนึ่งบรรทัดข้างล่างนี้

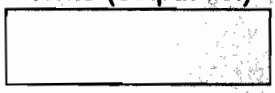
```
WriteLn ('The fabric size in square yards is ', SqYards)
```

ปกติจะเหมาะสมมากกว่าคือให้ใช้รูปแบบหลัง

Syntax Display

Write Procedure

รูปแบบ : Write (output list)



ตัวอย่าง

Write ('My height in inches is ', Height, 'and my ')

มีความหมายดังนี้ แสดงผลค่าของตัวแปรแต่ละตัวหรือค่าคงตัวในรายการเอาต์พุต string literal ถูกแสดงผลโดยไม่มีเครื่องหมาย apostrophes ตัวชี้ตำแหน่งไม่เลื่อนขึ้นบรรทัดถัดไป หลังจากแสดงผลเอาต์พุต

สไตล์ของโปรแกรม (Program Style)

ข้อชี้แนะสำหรับ Write และ WriteLn

โดยทั่วไป กระบวนงาน WriteLn ใช้แสดงผลลัพธ์โปรแกรม แต่ถ้ารายการเอาต์พุตยาวมาก เราสามารถแบ่งเป็นส่วนๆ และแสดงผลทีละขั้น ในกรณีนี้ ใช้กระบวนการ Write เพื่อแสดงผลทั้งหมด ยกเว้นส่วนสุดท้ายของรายการเอาต์พุต ใช้กระบวนงาน WriteLn เพื่อแสดงผลเฉพาะส่วนสุดท้ายของรายการเอาต์พุต

ตัวอย่างเช่น

```
Write ('This line displays ');
Write ('the value of X (', X, ') and ');
WriteLn ('the value of Y (', Y, '). ')
```

X	Y
55	7

รายการเอาต์พุตบรรทัดที่สองและบรรทัดที่สามประกอบด้วยสายอักขระ, ตัวแปร และสายอักขระ ข้อความสั่งสามบรรทัดนี้จะสร้างเอาต์พุตหนึ่งบรรทัด เมื่อให้ X มีค่าเท่ากับ 55 Y มีค่าเท่ากับ 7 จะแสดงผลดังนี้

This line displays the value of X (55) and the value of Y (7).

กระบวนงาน ReadLn (The ReadLn Procedure)

เมื่อต้องการข้อมูลอินพุต ใช้กระบวนงาน Write เพื่อแสดงข้อความพร้อมรับ (prompt message) ซึ่งบอกผู้ใช้โปรแกรมให้ใส่ข้อมูลอะไร ข้อความสั่งข้างล่างนี้

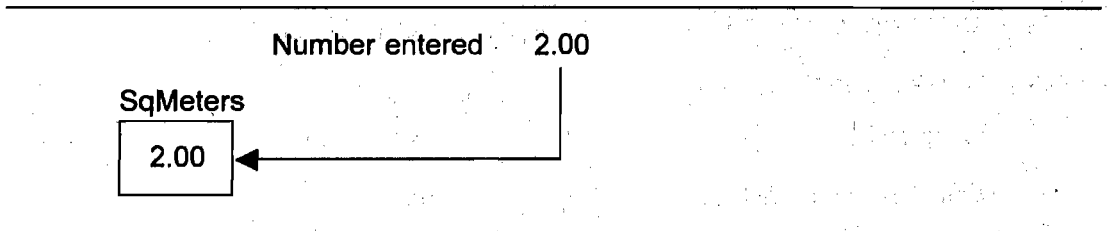
```
Write ('Enter the fabric size in square meters > ');
ReadLn (SqMeters)
```

แสดงข้อความพร้อมรับสำหรับค่าตัวเลขมีหน่วยเป็นเมตร ในรูปแบบของ strine literal และเลื่อนตำแหน่งตัวชี้ไปยังตำแหน่งจอภาพหลังสัญลักษณ์ > หลังจากนั้นผู้ใช้โปรแกรมคีย์ค่าข้อมูลที่ต้องการ และกระบวนงาน ReadLn จะประมวลผลอินพุต

ข้อความสั่ง

ReadLn (SqMeters)

เรียกกระบวนงาน ReadLn เพื่ออ่านข้อมูลไว้ที่ตัวแปร SqMeters กระบวนงาน ReadLn เอาข้อมูลซึ่งเก็บในตัวแปร SqMeters จากที่ไหน มันอ่านข้อมูลจากอุปกรณ์อินพุตมาตรฐาน (ใน Pascal เรียกว่า Input) กรณีส่วนใหญ่อุปกรณ์อินพุตมาตรฐานคือคีย์บอร์ด (keyboard) เมื่อใดก็ตามที่ผู้ใช้โปรแกรมคีย์ข้อมูลที่คีย์บอร์ด คอมพิวเตอร์จะเก็บข้อมูลนั้นใน SqMeters เนื่องจากประกาศ SqMeters เป็นชนิด Read การดำเนินการอินพุตจะประมวลผลโดยไม่มีข้อผิดพลาด ถ้าผู้ใช้โปรแกรมพิมพ์ตัวเลข ผลของการดำเนินการ ReadLn แสดงในรูป 2.6



รูป 2.6 ผลของ ReadLn (SqMeters)

เมื่อใดก็ตามที่กระบวนงาน ReadLn ถูกกระทำการ โปรแกรมจะหยุดชั่วคราวจนกว่ามีการใส่ข้อมูล และกดปุ่ม Enter ที่คีย์บอร์ด ถ้าคีย์ตัวอักขระข้อมูลไม่ถูกต้อง ผู้ใช้โปรแกรมสามารถกดปุ่ม backspace (←) เพื่อแก้ไขข้อมูลได้ แต่ถ้ากดปุ่ม Enter ไปแล้ว ข้อมูลถูกอ่านขณะพิมพ์ และสายเกินไปที่จะแก้ข้อผิดพลาดจากการใส่ข้อมูล

กระบวนงาน ReadLn สามารถอ่านค่าข้อมูล สำหรับแบบชนิดข้อมูลซึ่งให้นิยามแล้วชนิดอื่นๆ ได้ด้วย ยกเว้นชนิด Boolean จำนวนตัวอักขระ ซึ่งอ่านด้วยกระบวนงาน ReadLn ขึ้นอยู่กับชนิดของตัวแปรซึ่งรับข้อมูล ตาราง 2.5 แสดงตัวอย่างของ ReadLn กับตัวแปรอินพุตหนึ่งตัว สมมติว่า ปุ่ม Enter ถูกกด หลังจากตัวอักขระตัวสุดท้ายที่แสดงใน

สดมภ์ Data Line ข้อควรจำคือ เมื่อพิมพ์ข้อมูลตัวอักขระหนึ่งสายอักขระในโปรแกรม ไม่ต้องใส่เครื่องหมาย apostrophes

ตาราง 2.5 กฎสำหรับการอ่านข้อมูล

Type of var	Effect of ReadLn (var)
Char	อ่านตัวอักขระข้อมูลตัวถัดไป (Read next data character)
ตัวอย่าง	
Data Line	**Effect**
XYX	เก็บ X ในตัวแปร var
Type of var	Effect of ReadLn (var)
Integer	ข้ามตัวอักขระว่างข้างหน้า อ่านตัวอักขระทั้งหมดจนถึงอักขระว่างตัวถัดไป ตัวอักขระควบคุม หรือ Enter ตัวอักขระที่อ่าน ประกอบกันเป็นค่า integer
ตัวอย่าง	
Data Line	**Effect**
35 55	เก็บ 35 ในตัวแปร var
Type of var	Effect of ReadLn (var)
Real	ข้ามอักขระว่างข้างหน้า อ่านตัวอักขระทั้งหมดจนถึงอักขระว่างตัวถัดไป ตัวอักขระควบคุมหรือ Enter ตัวอักขระที่อ่านประกอบกันเป็นค่า real หรือ integer
ตัวอย่าง	
Data Line	**Effect**
1.54	เก็บ 1.54 ในตัวแปร var

Type of var	Effect of ReadLn (var)
string	อ่านตัวอักขระทั้งหมดจนถึง Enter
ตัวอย่าง	
Data Line	**Effect**
Sam SBD 55	เก็บ Sam SBD 55 ในตัวแปร var

จะเกิดอะไรขึ้น เมื่อตัวอักขระข้อมูลมีมากเกินในบรรทัดข้อมูล (ตัวอย่างเช่น ใน ตาราง 2.5 YZ ในบรรทัดข้อมูลบรรทัดแรก และ 55 ในบรรทัดข้อมูลบรรทัดที่สอง) ตัวอักขระเหล่านี้ถูกประมวลผลโดยกระบวนการ ReadLn แต่ไม่เก็บในหน่วยความจำ โปรดสังเกตตัวอักขระทั้งหมดบนบรรทัดข้อมูลถูกเก็บ เมื่อ var เป็นชนิด string

Syntax Display

กระบวนงาน ReadLn

รูปแบบ :

```
ReadLn (input list)
ReadLn
```

ตัวอย่าง ReadLn (Age, NumDepend)

มีความหมายดังนี้ กระบวนงาน ReadLn อ่านข้อมูลซึ่งผู้ใช้โปรแกรมพิมพ์ที่คีย์-บอร์ดเข้าไปไว้ในหน่วยความจำระหว่างกระทำการโปรแกรม ผู้ใช้โปรแกรมต้องใส่หน่วยข้อมูล (data item) หนึ่งตัวสำหรับตัวแปรแต่ละตัวที่กำหนดในรายการอินพุต (input list) จากนั้นกดปุ่ม Enter เครื่องหมาย comma ใช้คั่นชื่อตัวแปรในรายการอินพุต

อันดับของข้อมูลต้องสมนัยกับอันดับของตัวแปรในรายการอินพุต ให้ใส่อักขระว่างหนึ่งตัวหรือมากกว่าหนึ่งตัวระหว่างหน่วยข้อมูลที่เป็นตัวเลข แต่ภายในค่าตัวเลข (numeric value) หรือระหว่างค่าตัวเลข ใส่เครื่องหมาย comma ไม่ได้ ห้ามใส่อักขระว่างใดๆ ระหว่างหน่วยข้อมูลตัวอักขระสืบเนื่อง (consecutive character data items)

ยกเว้น อักขระว่างนั้นเป็นอักขระหนึ่งตัวของหน่วยข้อมูลซึ่งถูกอ่านและนำไปเก็บ ตัวอักขระข้อมูลส่วนเกินใดๆ ก็ตามซึ่งอยู่ตอนจบของบรรทัดถูกประมวลผล ถูก ignored (คือไม่เก็บในหน่วยความจำ)

ถ้าไม่มี input list (บรรทัดรูปแบบที่สอง) ตัวอักขระข้อมูลทั้งหมด ตลอดจนปุ่ม Enter จะถูกประมวลผล แต่ถูก ignored

กระบวนงาน Read

ในบทที่ 8 เราจะอภิปรายอีกวิธีหนึ่ง ในการใส่ข้อมูล กระบวนงาน Read ข้อแตกต่างที่สำคัญระหว่าง Read และ ReadLn คือ ตัวอักขระส่วนเกินใดๆ (any extra characters) บนบรรทัดข้อมูล หลังจากการดำเนินการ Read จะไม่ถูกอ่าน จนกระทั่งการดำเนินการ Read หรือ ReadLn ถัดไป

ในทางตรงกันข้าม การดำเนินการ ReadLn จะประมวลผลตัวอักขระทั้งหมดบนบรรทัดข้อมูล โดยไม่สนใจอักขระส่วนเกินใดๆ ที่ตอนท้ายของบรรทัด

การอ่านหน่วยข้อมูลหลายตัว

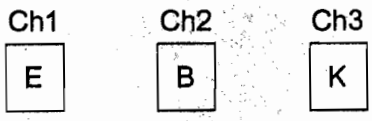
ส่วนใหญ่แล้ว ReadLn จะอ่านหน่วยข้อมูลครั้งละหนึ่งตัว แต่เป็นไปได้ที่ ReadLn หนึ่งบรรทัดอ่านค่าข้อมูลหลายตัว บ่อยครั้งที่กระทำกับตัวแปรชนิด Char

ตัวอย่าง 2.5

สมมติว่า Ch1, Ch2 และ Ch3 เป็นตัวแปรชนิด Char ข้อความสั่งข้างล่างนี้

```
ReadLn (Ch1, Ch2, Ch3)
```

อ่านและเก็บอักขระข้อมูลสามตัว ข้อมูลตัวแรกเก็บใน Ch1 ตัวที่สองเก็บใน Ch2 และตัวที่สามเก็บใน Ch3 ใส่อักขระข้อมูล EBK ผลลัพธ์เป็นดังนี้



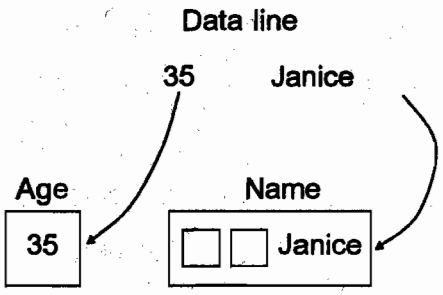
ถ้าเราไม่ตั้งใจเก็บอักขระว่างในหน่วยความจำ ไม่ต้องกด space bar ระหว่างอักขระข้อมูล มิฉะนั้นมันจะนับเป็นหนึ่งตัวอักขระของอักขระสามตัวซึ่งอ่านและเก็บ ตัวอย่างเช่น ถ้าเราใส่ E B K อักขระสามตัวที่เก็บจะเป็น E อักขระว่าง และ B ส่วนอักขระว่าง และ K จะถูกประมวลผลแต่ไม่เก็บไว้

ตัวอย่าง 2.6

สมมติว่า Age เป็นข้อมูลชนิด Integer และ Name เป็นข้อมูลชนิด string ข้อความสั่งสองบรรทัดข้างล่างนี้

```
WriteLn ('Enter your age and your name > ');
ReadLn (Age, Name)
```

อ่านและเก็บหน่วยข้อมูลตัวแรก (เป็นเลขจำนวนเต็ม) ใน Age และหน่วยข้อมูลตัวที่สอง ใน Name ดังแสดงในรูป 2.7 อย่างน้อยที่สุดจะต้องมีอักขระว่างหนึ่งตัวหลังเลขจำนวนเต็ม และ Turbo Pascal เก็บอักขระว่างเป็นสัญลักษณ์ ☐ ใน string



รูป 2.7 การอ่าน Integer และ String

สไตล์โปรแกรม (Program Style)

หลีกเลี่ยงข้อผิดพลาดการใส่ข้อมูลด้วยข้อมูลสายอักขระ (Avoiding Data Entry Error with String Data)

ผู้ใช้โปรแกรมต้องใส่หน่วยข้อมูลในลำดับที่ถูกต้อง เมื่อข้อความสั่ง ReadLn อ่านหน่วยข้อมูลหลายตัว

ตัวอย่างเช่น จงพิจารณาว่าจะเกิดอะไรขึ้น ถ้าผู้ใช้โปรแกรมพิมพ์บรรทัดข้อมูลดังนี้

Janice 35

เมื่อ ReadLn (Age, Name) กระทำการ เนื่องจาก Age เป็นตัวแปรตัวแรก ในรายการอินพุต ReadLn จะพยายามอ่านและเก็บชื่อผู้ใช้ (Janice) ในตัวแปร Age ชนิด Integer สิ่งนี้ทำให้เกิดข้อความระบุความผิดพลาด invalid numeric format เพราะว่า ตัวอักขระ J ไม่ใช่ตัวเลข

เราต้องพยายามอ่านและเก็บบรรทัดข้อมูลนี้โดยใช้ข้อความสั่ง

ReadLn (Name, Age); {invalid attempt to read string first}

อย่างไรก็ตาม ข้อความสั่งข้างต้นนี้จะอ่านและเก็บตัวอักขระทั้งหมดบนบรรทัดข้อมูล (Janice 35) ใน Name หลังจากนั้นโปรแกรมจะหยุดจนกว่าผู้ใช้โปรแกรมพิมพ์ค่าของ Age ในบรรทัดข้อมูลที่สอง เพื่อหลีกเลี่ยงปัญหาการใส่ข้อมูลเหล่านี้ ขอแนะนำว่า ให้

อ่านหนึ่งหน่วยข้อมูลด้วยข้อความสั่ง ReadLn แต่ละชุด ยกเว้นเมื่ออ่านข้อมูลอักขระตัวแปรชนิด Char

แบบฝึกหัด 2.5 Self-Check

1. จงแสดงผลของเอาต์พุตจากบรรทัดโปรแกรมข้างล่างนี้ เมื่อใส่ข้อมูล 5 และ 7

```
WriteLn ('Enter two integers > ');
ReadLn (M, n) {data line is : 5 7}
M := M + 5 ;
N := 3 + N ;
WriteLn ('M = ', M);
WriteLn ('N = ', N)
```

2. จงแสดงผลของเอาต์พุตจากบรรทัดต่อไปนี้

```
Write ('My name is : ' );
WriteLn ('Doe, Jane');
WriteLn;
Write ('I live in');
Write ('Ann arbor, MI');
Write ('and my zip code is ', 48109)
```

3. จงบอกว่าการกำหนดค่าข้างล่างนี้แต่ละชุดมีชุดใดถูกต้อง (valid) หรือไม่ถูกต้อง (invalid) ถ้าการกำหนดค่านั้นถูกต้อง ให้บอกผลลัพธ์ สมมติว่าตัวแปร R เป็นชนิด Real, I เป็นชนิด Integer, B เป็นชนิด Boolean, C เป็นชนิด Char และ S เป็นชนิด string

```
a) R := 3.5 + 5.0
b) I := 2 * 5
c) C := 'My name'
d) S := Your name
e) B := Boolean
f)  S := C
g) C := S
```

h) R := I

i) I := R

j) R := 10 + I

โปรแกรมมิง (Programming)

1. จงเขียนข้อความสั่งซึ่งแจ้งผู้ใช้ให้ใส่เลขสามจำนวน จากนั้นอ่านเลขสามจำนวนไว้ใน First, Second และ Third

2. a) จงเขียนข้อความสั่งแสดงผลบรรทัดข้างล่างนี้โดยให้ค่าของ X อยู่ดอนท้าย

The value of X is _____.

b) สมมติว่า Radius และ Area เป็นตัวแปรชนิด Real หมายถึง รัศมีและพื้นที่ของวงกลม จงเขียนข้อความสั่งซึ่งจะแสดงผลสารสนเทศในรูปแบบข้างล่างนี้

The area of a circle with radius _____ is _____.

3. จงเขียนโปรแกรมแจ้งผู้ใช้ให้ใส่รัศมีของวงกลม จากนั้นคำนวณพื้นที่ของวงกลมและแสดงผล โดยใช้สูตร

Area = MyPi X Radius X Radius

เมื่อ MyPi คือค่าคงตัว 3.14159

2.6 รูปแบบทั่วไปของโปรแกรม Pascal (General Form of a Pascal Program)

ถึงขณะนี้เราได้อภิปรายข้อความสั่งแต่ละชุดที่ปรากฏในโปรแกรม Pascal ต่อไปเราจะทบทวนกฎสำหรับการรวม (combining) ข้อความสั่งเหล่านี้เป็นโปรแกรมรวมทั้งการใช้เครื่องหมายกำกับวรรคตอน การเว้นวรรค และคอมเมนต์ในโปรแกรม

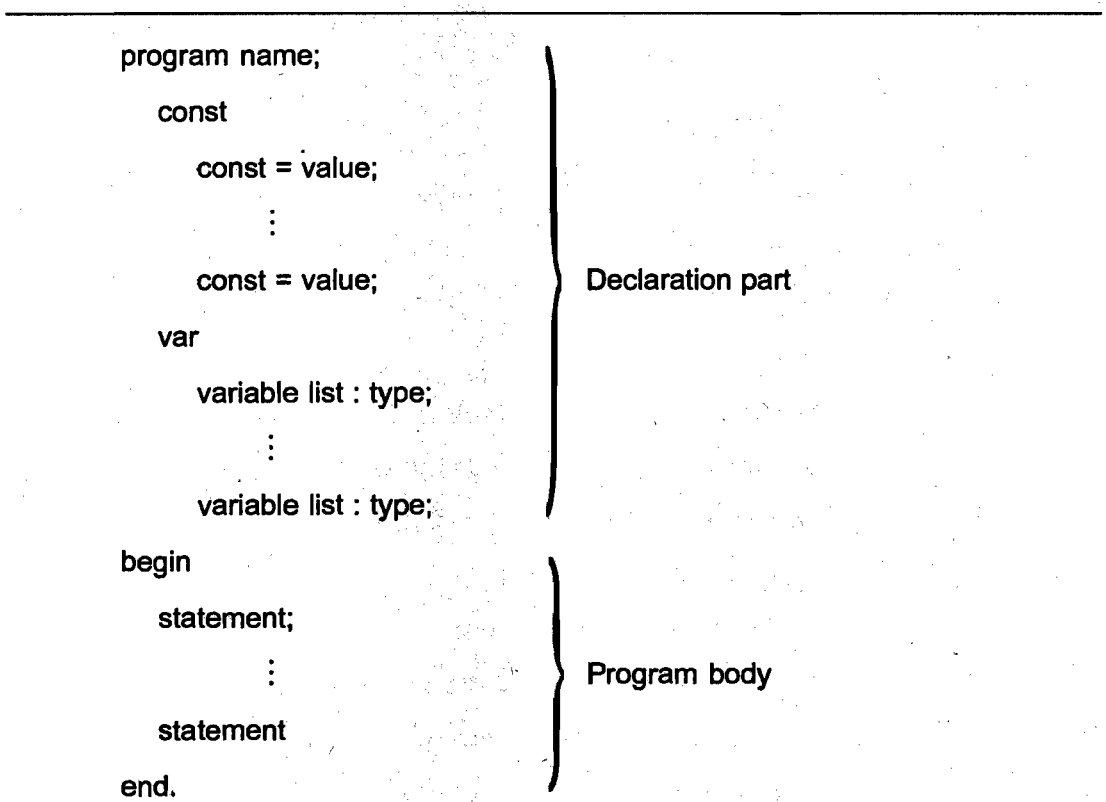
รูป 2.8 แสดงรูปแบบทั่วไปของโปรแกรม ซึ่งแต่ละโปรแกรมต้องขึ้นต้นด้วยหัวเรื่องระบุชื่อของโปรแกรม ต่อไปคือส่วนการประกาศ ซึ่งเราประกาศไอเดนติไฟเออร์ทุกตัวที่ใช้ในโปรแกรม ยกเว้นไอเดนติไฟเออร์มาตรฐาน การประกาศค่าคงตัวทั้งหมด ให้ตามหลังคำสงวน const การประกาศตัวแปรทั้งหมด ให้ตามหลังคำสงวน var

อาจมีการประกาศค่าคงตัวมากกว่าหนึ่งตัว และอาจมีรายการตัวแปรมากกว่าหนึ่งชุด เครื่องหมาย comma (,) ใช้คั่นไอเดนติไฟเออร์แต่ละตัว ในรายการตัวแปร

เครื่องหมาย semicolon (;) ใช้จบการประกาศแต่ละชุด

ใน standard Pascal คำสงวน const และ var ปรากฏมากกว่าหนึ่งครั้งไม่ได้ และต้องอยู่ในอันดับเหมือนที่แสดงในรูป 2.8

ในทางตรงกันข้าม Turbo Pascal คำสงวน const และ var มีมากกว่าหนึ่งครั้งได้ และจะเรียงอันดับอย่างไรก็ได้ อย่างไรก็ตาม การฝึกปฏิบัติเขียนโปรแกรมที่ดีควรทำตามกฎของ standard Pascal เพื่อให้สามารถเคลื่อนย้ายได้



รูป 2.8 รูปแบบทั่วไปของโปรแกรม Pascal

คำสงวน begin คือ สัญญาณให้เริ่มต้นตัวโปรแกรม (program body) ตัวโปรแกรมประกอบด้วยข้อความสั่งต่างๆ ซึ่งจะถูกแปลให้เป็นภาษาเครื่องและกระทำการในที่สุด ข้อความสั่งซึ่งเราได้เรียนมาบ้างแล้ว กระทำการดำเนินการคำนวณและการดำเนินการอินพุต/เอาต์พุต บรรทัดสุดท้าย คือ end.

เครื่องหมาย semicolon ใช้คั่นข้อความสั่ง เนื่องจาก semicolon ไม่จำเป็นต้องมีก่อนข้อความสั่งแรกในลำดับ หรือหลังข้อความสั่งสุดท้าย ถึงแม้ว่าจะไม่แนะนำให้ทำ แต่เราสามารถพิมพ์เครื่องหมาย semicolon หนึ่งตัวหลังข้อความสั่งสุดท้ายในโปรแกรมได้ ถ้าเป็นดังนี้ เครื่องหมาย semicolon มีผลคือ ใส่ข้อความสั่งว่าง (empty statement) ระหว่างข้อความสั่งจริงสุดท้ายกับตัวจบโปรแกรม (end.)

ภาษา Pascal ไม่สนใจการแยกบรรทัด (line breaks) ดังนั้น ข้อความสั่ง Pascal หนึ่งคำสั่งอาจขยายเกินเป็นหลายบรรทัด ตัวอย่างเช่น การประกาศตัวแปรและค่าคงตัว

ในรูป 2.1 ซึ่งเริ่มต้นบนหนึ่งบรรทัดและจบในบรรทัดถัดไป

ข้อความสั่งซึ่งขยายมากกว่าหนึ่งบรรทัดไม่สามารถแบ่งแยกในตอนกลางของไอเดนติไฟเออร์ คำสงวน หรือสัญพจน์

เราสามารถเขียนข้อความสั่งมากกว่าหนึ่งคำสั่งบนหนึ่งบรรทัดได้ ตัวอย่างเช่น

```
WriteLn ('Enter two letters >'); ReadLn (Letter1, Letter2)
```

ประกอบด้วย หนึ่งข้อความสั่ง แสดงผลข้อความพร้อมรับ และอีกหนึ่งข้อความสั่งซึ่งอ่านข้อมูลที่ต้องการ

เครื่องหมาย semicolon ใช้คั่นข้อความสั่งสองชุด

เครื่องหมาย semicolon อีกตัวหนึ่งอาจอยู่ตอนจบของบรรทัด ถ้ามีข้อความสั่งตามมาอีก แนะนำคือ ให้เขียนหนึ่งข้อความสั่งเท่านั้นบนหนึ่งบรรทัด เพราะว่าจะทำให้อ่านโปรแกรมง่าย และการบำรุงรักษาโปรแกรมทำได้ง่าย

ที่ว่างในโปรแกรม (Spaces in Programs)

ความต้องกันและการใช้อักขระว่างอย่างระมัดระวังทำให้ปรับปรุงสไตล์ของโปรแกรม ตัวอักขระว่างจะใช้ระหว่างคำ (words) ในบรรทัดโปรแกรม

คอมไพเลอร์ไม่สนใจตัวอักขระว่างที่เกินมาระหว่างคำและสัญลักษณ์ แต่เราอาจใส่อักขระว่าง เพื่อทำให้โปรแกรมอ่านง่ายและเป็นสไตล์ของโปรแกรม เราควรใส่อักขระว่างหนึ่งตัวเสมอหลัง comma และใส่ก่อน และใส่หลังตัวดำเนินการ เช่น *, -, และ := จำไว้เสมอว่า ให้ย่อหน้าแต่ละบรรทัดของโปรแกรม ยกเว้นบรรทัดแรกและบรรทัดสุดท้าย และบรรทัด begin และให้เขียนคำสงวน const, var และ begin บนบรรทัด ซึ่งจะแสดงคำให้เด่นขึ้น บรรทัดทั้งหมดยกเว้นบรรทัดแรกและบรรทัดสุดท้ายของโปรแกรมและบรรทัด begin ให้ย่อหน้าสองหรือมากกว่าสองที่ สุดท้ายให้ใส่บรรทัดว่าง (blank lines) ระหว่างแต่ละหัวข้อของโปรแกรม

ถึงแม้ว่าประเด็นสไตล์การเขียน ไม่มีผลกระทบแต่อย่างใดกับความหมายของโปรแกรม แต่ทำให้ผู้คนอ่านและทำความเข้าใจโปรแกรมง่ายขึ้น อย่างไรก็ตาม จะต้องไม่ใส่อักขระว่างในที่ห้ามใส่ ตัวอย่างเช่น ระหว่างตัวอักขระ : และตัวอักขระ = เมื่อมันเป็นตัวกำหนดค่า (assignment operator)

คอมเมนต์ในโปรแกรม (Comments in Programs)

โปรแกรมเมอร์ทำให้โปรแกรมน่าอ่านมากขึ้นโดยการใช้คอมเมนต์เพื่ออธิบายวัตถุประสงค์ของโปรแกรม การใช้ไอเดนติไฟเออร์ และวัตถุประสงค์ของแต่ละขั้นตอนของโปรแกรม คอมเมนต์เป็นส่วนของการทำเอกสารโปรแกรม เพราะว่ามันช่วยผู้อื่นให้อ่านและทำความเข้าใจโปรแกรมได้ อย่างไรก็ตาม คอมไพเลอร์ไม่สนใจ (ignored) คอมเมนต์ และไม่แปลคอมเมนต์ให้เป็นภาษาเครื่อง

คอมเมนต์ปรากฏโดยตัวมันเองบนบรรทัดโปรแกรมที่ตอนจบของบรรทัดตามหลังข้อความสั่ง หรือ**ฝังตัว** (embedded) ในข้อความสั่ง การประกาศตัวแปรข้างล่างนี้ คอมเมนต์ชุดแรกฝังตัวในการประกาศ ในขณะที่ชุดที่สองตามหลังการประกาศ ส่วนใหญ่เราทำเอกสารตัวแปรในวิธีนี้

```
var
  SqMeters,        {input-fabric size in meters}
  SqYards : Real;  {output-fabric size in yards}
```

Syntax Display

คอมเมนต์ (Comments)

Form : | {comment} |

ตัวอย่าง
```
{This is a comment}
(* and so is this *)
{one comment (* inside another *) comment}
```

มีความหมายดังนี้ วงเล็บปีกกาเปิด แสดงการเริ่มต้นของคอมเมนต์ วงเล็บปีกกาปิดระบุการจบคอมเมนต์ เช่นเดียวกับ (* และ *) เป็นเครื่องหมายการเริ่มต้นและการจบตามลำดับของคอมเมนต์ คอมเมนต์ปรากฏเป็นรายการในโปรแกรม แต่คอมไพเลอร์ไม่สนใจ

ข้อสังเกต Turbo Pascal (แต่ไม่ใช่ standard Pascal) ยอมให้คอมเมนต์ ซ้อนใน (nested) ภายในคอมเมนต์อีกชุดหนึ่งได้ ถ้าคอมเมนต์ชุดแรกเริ่มต้นด้วย { คอมเมนต์ชุดที่สองต้องเริ่มต้นด้วย (* และในทางกลับกัน ถ้าคอมเมนต์ชุดแรกเริ่มต้นด้วย (* คอมเมนต์ชุดที่สองต้องเริ่มต้นด้วย {

สไตล์โปรแกรม (Program style)

การใช้คอมเมนต์ (Using Comments)

ทุกโปรแกรมควรเริ่มต้นด้วยส่วนหัวเรื่อง ซึ่งประกอบด้วยชุดของคอมเมนต์ระบุสิ่งต่อไปนี้

(1) ชื่อโปรแกรมเมอร์

(2) วันเดือนปีที่เขียนโปรแกรมเวอร์ชันปัจจุบัน

(3) ถ้าอธิบายโดยสรุปว่าโปรแกรมนี้ทำอะไร

เมื่อโปรแกรมที่กำลังเขียนเป็นงานซึ่งกำหนดให้ทำในชั้นเรียน นักศึกษาควรใส่รายละเอียดของชั้นเรียน และชื่อของอาจารย์ผู้สอน ตัวอย่างเช่น

```
program Metric ;
{
   Programmer: William Bell    Date completed : May 9, 2008
   Instructor    : Somphit Kosallawat        Class : IT257
   This program reads a value in square meters
   and convert it to square yards.
}
```

ก่อนการ implement แต่ละขั้นตอนในอัลกอริทึมเริ่มต้น เราควรเขียนคอมเมนต์ซึ่งสรุปวัตถุประสงค์ของขั้นอัลกอริทึม

คอมเมนต์นี้ควรอธิบายว่าขั้นตอนนั้นทำอะไร ตัวอย่างเช่น

คอมเมนต์

```
   { Convert the fabric size to square yards }
     SqYares := MetersToYards * SqMeters ;
```

แบบฝึกหัด 2.6 Self-Check

1. จงเปลี่ยนคอมเมนต์เหล่านี้ให้ถูกต้องเชิงวากยสัมพันธ์

   { This is a comment? *)

   { How about this one { it seems like a comment } doesn't it }

2. เครื่องหมาย semicolon ในตัวโปรแกรมมีไว้เพื่อวัตถุประสงค์อะไร

3. จงแก้ไขข้อผิดพลาดวากยสัมพันธ์ในโปรแกรมข้างล่างนี้ และเขียนโปรแกรมใหม่ให้เป็นไปตามข้อตกลงสไตล์โปรแกรมของเรา จากนั้นอธิบายว่า ข้อความสั่งแต่ละชุดของโปรแกรมที่ถูกต้องทำอะไร พิมพ์ค่าอะไรบ้าง

```
program SMALL VAR X, Y, X, real;
BEGIN Y = 15.0
Z := -Y + 3.5; Y + z =: x;
writeln (n; y; z); end;
```

เขียนโปรแกรม (Programming)

1. จงเขียนโปรแกรมซึ่งเก็บค่า 'X', 76.1 และ 'MyDog' ในเซลล์หน่วยความจำแยกต่างหากจากกัน โปรแกรมของเราควรอ่านค่าต่างๆ เป็นหน่วยข้อมูล และแสดงผลค่าเหล่านี้อีกครั้งหนึ่งสำหรับผู้ใช้เมื่อทำงานเสร็จแล้ว

2.7 นิพจน์คำนวณ (Arithmetic Expressions)

การแก้ปัญหาของการเขียนโปรแกรมส่วนใหญ่เราจำเป็นต้องเขียนนิพจน์คำนวณซึ่งจัดดำเนินการ (manipulate) ข้อมูลชนิด Integer และชนิด Real หัวข้อนี้จะอธิบายตัวดำเนินการ (operators) ซึ่งใช้ในนิพจน์คำนวณ และกฎสำหรับการเขียนและการประเมินผลนิพจน์เหล่านี้

ตาราง 2.6 แสดงให้เห็นตัวดำเนินการคำนวณทั้งหมด ตัวดำเนินการแต่ละตัว จัดดำเนินการกับตัวถูกดำเนินการ (operands) สองตัว

ตัวถูกดำเนินการอาจเป็นค่าคงตัว (constants) ตัวแปรหรือนิพจน์คำนวณอื่นๆ

ตัวดำเนินการ +, -, * และ / ใช้กับตัวถูกดำเนินการชนิด Interger หรือชนิด Real ดังแสดงไว้ในสตมภ์ท้ายสุด

สำหรับตัวดำเนินการ +, -, และ * แบบชนิดข้อมูลของผลลัพธ์ จะเหมือนกับแบบชนิดข้อมูลของตัวถูกดำเนินการของมัน

ตัวดำเนินการหารข้อมูลชนิด real คือ / จะให้ผลลัพธ์เป็นเลขจำนวนจริงเสมอ ดังนั้นนิพจน์ X/2 จะให้ผลลัพธ์เป็นชนิด Real เสมอ แม้แต่เมื่อ X เป็นชนิด Integer ก็ตาม ตัวอย่างเช่น ถ้า X เท่ากับ 4 ค่าของ X/2 จะเท่ากับ 2.0

ตัวดำเนินการสองตัวสุดท้าย div และ mod ใช้เฉพาะกับข้อมูลชนิด Integer เท่านั้น

ตาราง 2.6 ตัวดำเนินการคำนวณ

Arithmetic	Meaning	Example
+	Addition	5 + 2 is 7
		5.0 + 2.00 is 7.0
-	Subtraction	5 - 2 is 3
		5.0 - 2.0 is 3.0
*	Multiplication	5 * 2 is 10
		5.0 * 2.0 is 10.0
/	Real division	5.0 / 2.0 is 2.5
		5 / 2 is 2.5
div	Integer division	5 div 2 is 2
mod	Modulus operator	5 mod 2 is 1
		↓ ↓ ↓
		ตัวตั้ง ตัวหาร เศษเหลือ

ตัวดำเนินการ div และ mod

div คือ ตัวดำเนินการหารเลขจำนวนเต็ม เพื่อคำนวณส่วนที่เป็นเลขจำนวนเต็มของผลลัพธ์ของการหารโดยที่ตัวถูกดำเนินการตัวแรกเป็นตัวตั้ง และตัวถูกดำเนินการตัวที่สองเป็นตัวหาร (The integer division operator, div, computes the integral part of the result of dividing its first operand by its second.)

ตัวอย่างเช่น 7/2 คำตอบคือ 3.5

แต่ 7 div 2 คือส่วนที่เป็นเลขจำนวนเต็มของผลลัพธ์นี้ คำตอบคือ 3 ในทำนองเดียวกัน ค่าของ 299/100 คำตอบคือ 2.99 แต่ค่าของ 299 div 100 คำตอบคือ 2 (คิดเฉพาะส่วนที่เป็นเลขจำนวนเต็มของผลลัพธ์)

ตัวถูกดำเนินการทั้งสองตัวของ div ต้องเป็น integers ทั้งคู่ และการดำเนินการ div จะไม่ถูกนิยาม (undefined) เมื่อตัวหาร (ตัวถูกดำเนินการตัวที่สอง) มีค่าเป็นศูนย์

ตาราง 2.7 ตัวอย่างของตัวดำเนินการ div

3 div 15 = 0	3 div - 15 = 0
15 div 3 = 5	15 div - 3 = - 5
16 div 3 = 5	16 div - 3 = - 5
17 div 3 = 5	- 17 div 3 = - 5
18 div 3 = 6	- 18 iv 3 = - 6

ใน Pascal ผลลัพธ์ของนิพจน์ 6/2 และ 6 div 2 จะไม่เหมือนกัน เพราะว่าค่าของ 6/2 เป็นเลขจำนวนจริง 3.0 แต่ค่าของ 6 div 2 เป็นเลขจำนวนเต็ม 3 ถึงแม้ว่าในเชิงคณิตศาสตร์จะสมมูล (equivalent) กันก็ตาม

mod คือตัวดำเนินการ modulus ส่งกลับเศษเหลือซึ่งเป็นเลขจำนวนเต็มของผลลัพธ์จากการหารของตัวถูกดำเนินการตัวแรกด้วยตัวถูกดำเนินการตัวที่สอง (The modulus operator, mod, returns the integer remainder of the result of dividing its first operand by its second.)

ตัวอย่างเช่น ค่าของ 7 mod 2 คือ 1 เพราะว่าเศษเหลือที่เป็นเลขจำนวนเต็มคือ 1

เราสามารถใช้การหารยาว (long division) เพื่อคำนวณหาผลลัพธ์ของตัวดำเนินการ div หรือ mod

ตัวอย่างเช่น การคำนวณทางซ้ายมือ แสดงให้เห็นผลของการหาร 7 ด้วย 2 ด้วยวิธีหารยาว ผลหารคือ 3 (7 div 2 คือ 3) และเศษเหลือคือ 1 (7 mod 2 คือ 1)

ส่วนการคำนวณทางขวามือ แสดงให้เห็นว่า 299 div 100 คำตอบคือ 2 (ผลหาร) และ 299 mod 100 คือ 99 (เศษเหลือ)

7 div 2	299 div 100
↓	↓
3	2
$2\overline{)7}$	$100\overline{)299}$
6	200
1 ← 7 mod 2	99 ← 299 mod 100

ขนาดของ M mod N ต้องน้อยกว่าตัวหาร N เสมอ

ถ้า M เป็นเลขจำนวนเต็มบวก 100 ค่าของ M mod N ต้องอยู่ระหว่าง 0 และ 99 การดำเนินการ mod จะไม่ถูกนิยาม (undefined) เมื่อ n มีค่าเป็นศูนย์ ตาราง 2.8 แสดงให้เห็นตัวอย่างของตัวดำเนินการ mod เมื่อเปรียบเทียบสตมภ์ที่สองและสตมภ์ที่สามในตารางนี้จะเห็นว่า - M mod N สมมูลกับ - (M mod N)

ตาราง 2.8 ตัวอย่างของดัวดำเนินการ mod

3 mod 5 = 3	5 mod 3 = 2	- 5 mod 3 = - 2
4 mod 5 = 4	5 mod 4 = 1	- 5 mod 4 = - 1
5 mod 5 = 0	15 mod 5 = 0	- 15 mod 5 = 0
6 mod 5 = 1	15 mod 6 = 3	- 15 mod 6 = - 3
7 mod 5 = 2	15 mod 7 = 1	- 15 mod - 7 = - 1
8 mod 5 = 3	15 mod 8 = 7	- 15 mod 0 is undefined

แบบชนิดข้อมูลของนิพจน์คำนวณ (Data Type of an Arithmetic Expression)

จากที่ได้กล่าวมาแล้ว นิพจน์ชนิด Integer อาจกำหนดให้กับตัวแปรชนิด Real ได้ แต่ในทางกลับกันทำไม่ได้ (but not vice versa) Pascal กำหนดแบบชนิดข้อมูลของนิพจน์คำนวณ โดยใช้กฎต่อไปนี้

นิพจน์จะมีชนิดเป็น Integer ถ้าตัวถูกดำเนินการทุกตัวเป็นชนิด Integer และไม่มีตัวดำเนินการใดๆ เป็น / กรณีอื่นๆ นิพจน์จะมีชนิดเป็น Real

กฎข้างต้นนี้ขึ้นอยู่กับคุณสมบัติสองข้อของตัวดำเนินการคำนวณ

(1) ตัวดำเนินการ / จะให้ผลลัพธ์เป็นชนิด Real เสมอ

(2) ถ้าตัวดำเนินการมีตัวถูกดำเนินการหนึ่งตัวเป็นชนิด Real และตัวถูกดำเนินการอีกหนึ่งตัวมีชนิดเป็น Integer จะให้ผลลัพธ์เป็นชนิด Real (ตัวอย่างเช่น 5 + 2.0 ผลลัพธ์คือ 7.0 ไม่ใช่ 7)

นิพจน์ชนิดผสม (Mixed-type expression)

หมายถึงนิพจน์ซึ่งมีตัวถูกดำเนินการเป็นชนิด Real และชนิด Integer ทั้งคู่ จากกฎข้างต้น นิพจน์ชนิดผสมต้องมีชนิดเป็น Real การฝึกเขียนโปรแกรมที่ดีควรหลีกเลี่ยงการเขียนนิพจน์ชนิดผสม

ข้อความสั่งกำหนดค่าชนิดผสม (Mixed-Type Assignment Statement)

เมื่อข้อความสั่งกำหนดค่าถูกกระทำการ อันดับแรกนิพจน์ถูกประเมินผลจากนั้นผลลัพธ์จะถูกกำหนดให้ตัวแปรซึ่งอยู่ก่อนหน้าตัวกำหนดค่า (:=) ไม่ว่านิพจน์จะมีชนิดเป็น Real หรือเป็นชนิด Integer ก็ตาม อาจถูกกำหนดให้กับตัวแปรชนิด Real ข้อความสั่งกำหนดค่าทั้งหมดต่อไปนี้ถูกต้อง (valid) ถ้า M และ N เป็นชนิด Integer X และ Y เป็นชนิด Real

```
M := 3;
N := 2;
X := M/N;          {assigns 1.5 to X}
Y := M + N;        {assigns 5.0 to Y}
```

ข้อความสั่งท้ายสุด นิพจน์ M + N ประเมินผลเป็นเลขจำนวนเต็ม 5 และค่านี้จะถูกเปลี่ยนให้เป็นชนิด Real ซึ่งสมมูลกัน คือ 5.0 ก่อนเก็บในตัวแปร Y

โปรดสังเกตว่า การเปลี่ยนชนิดนั้น กระทำขึ้นหลังจากค่าจำนวนเต็มสองค่าบวกกัน นั่นคือ 3 บวก 2 ได้ผลลัพธ์เป็น 5 จากนั้นจึงเปลี่ยนเป็น 5.0 นี่คือตัวอย่างของข้อความสั่งกำหนดค่าชนิดผสม เพราะว่าตัวแปรที่กำลังจะถูกกำหนดค่า และนิพจน์มีแบบชนิดข้อมูลแตกต่างกัน

Pascal ไม่ยอมให้นิพจน์ชนิด Real กำหนดค่าให้กับตัวแปรชนิด Integer เพราะว่าภาคเศษส่วน (fractional part) ของนิพจน์ไม่ถูกแทนที่ค่านี้จะหายไป

ตัวอย่างข้อความสั่งกำหนดค่าชนิดผสมต่อไปนี้ไม่ถูกต้อง (invalid) ถ้า Count มีชนิดเป็น Integer

```
Count := 3.5;              {assign real value to integer variable}
Count := Count + 1.0;      {1.0 is Real, so resutl is Real}
Count := Count / 2;        {result of division is Real}
```

นิพจน์ที่มีตัวดำเนินการหลายตัว (Expression with Multiple Operators)

นิพจน์ที่มีตัวดำเนินการหลายตัวเป็นสิ่งปกติ การทำความเข้าใจและเขียนนิพจน์ซึ่งมีตัวดำเนินการหลายตัว เราต้องทราบกฎของ Pascal สำหรับการประเมินผลนิพจน์ ตัวอย่างเช่น

10 + 5 / 2 ทำ + ก่อน / หรือทำ / ก่อน +

10 / 5 * 2 ทำ / ก่อน * หรือทำ * ก่อน /

การทดสอบจะเห็นว่า อันดับของการประเมินผลจะให้ผลลัพธ์แตกต่างกัน นิพจน์ทั้งสองชุดข้างต้น ตัวดำเนินการ / ถูกประเมินผลเป็นอันดับแรก เหตุผลอธิบายไว้ในกฎของ Pascal สำหรับการประเมินผลนิพจน์ ซึ่งมีหลักการคล้ายกับกฎพีชคณิต (algebraic rults)

กฎสำหรับการประเมินผลนิพจน์ (Rules for Evaluating Expressions)

a) กฎของวงเล็บ (Parentheses rules) : นิพจน์ทั้งหมดซึ่งอยู่ในวงเล็บต้องประเมินผลแยกจากกัน นิพจน์ที่มีวงเล็บซ้อนในต้องประเมินผลจากข้างในไปข้างนอก นิพจน์อยู่ในสุดถูกประเมินผลเป็นอันดับแรก

b) กฎการทำก่อนของตัวดำเนินการ (Operator precedence rule) : ตัวดำเนินการในนิพจน์เดียวกันถูกประเมินผลในอันดับต่อไปนี้

*, /, div, mod	อันดับแรก (High)
+, -	สุดท้าย (Low)

c) กฎการเปลี่ยนหมู่จากซ้าย (Left associative rule) : ตัวดำเนินการในนิพจน์เดียวกัน ที่มีระดับการทำก่อนเหมือนกัน (เช่น + และ -) ถูกประเมินผลจากซ้ายไปขวา

กฎเหล่านี้ช่วยให้เราเข้าใจว่า Pascal ประเมินผลนิพจน์อย่างไร ให้ใช้วงเล็บเท่าที่จำเป็นเพื่อกำหนดอันดับของการประเมินผล สำหรับนิพจน์ที่มีความซับซ้อนการใช้วงเล็บเพิ่มเป็นสิ่งที่ดี ทำให้เห็นอันดับของการประเมินผลชัดเจน ตัวอย่างเช่น นิพจน์

X * Y * Z + A / B - C * D

เขียนในรูปแบบที่อ่านง่ายขึ้นโดยการใช้วงเล็บ

(X * Y * Z) + (A / B) - (C * D)

ตัวอย่าง 2.8

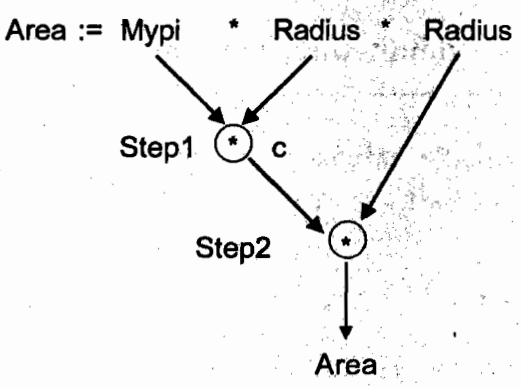
สูตร คำนวณหาพื้นที่ของวงกลม

$a = \pi r^2$

ใน Pascal เขียนดังนี้

Area := MyPi * Radius * Radius

เมื่อ MyPi คือ ค่าคงตัว 3.14159 รูป 2.9 แสดงให้เห็นต้นไม้การประเมินผลของสูตรนี้ ในต้นไม้นี้ อ่านจากบนลงล่าง ลูกศรเชื่อมต่อตัวถูกดำเนินการแต่ละตัวกับตัวดำเนินการของมัน อันดับของการประเมินผลของตัวดำเนินการ แสดงด้วยเลขทางซ้ายมือของตัวดำเนินการแต่ละตัว ตัวอักษรทางขวามือของตัวดำเนินการหรือกฎที่เราใช้



เหตุผล Rule c : lelf associative rule

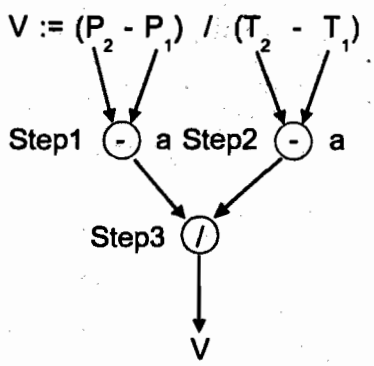
รูป 2.9 ต้นไม้การประเมินผล (Evaluation tree) สำหรับข้อความสั่งกำหนดค่า
Area := MyPi * Radius * Radius

ตัวอย่าง 2.9

สูตรคำนวณความเร็วเฉลี่ย v, เมื่อวัตถุเดินทางบนเส้นระหว่างจุด P_1 และ P_2 ในเวลา T_1 ถึง T_2 คือ

$$V = \frac{P_2 - P_1}{T_2 - T_1}$$

ใน Pascal สูตรนี้ เขียนและประเมินผลดังแสดงไว้ในรูป 2.10



เหตุผล Rule a : นิพจน์ที่มีวงเล็บกำกับประเมินผลแยกจากกัน

รูป 2.10 ต้นไม้การประเมินผลสำหรับข้อความสั่งกำหนด
$V := (P_2 - P_1) / (T_2 - T_2 - T_1)$

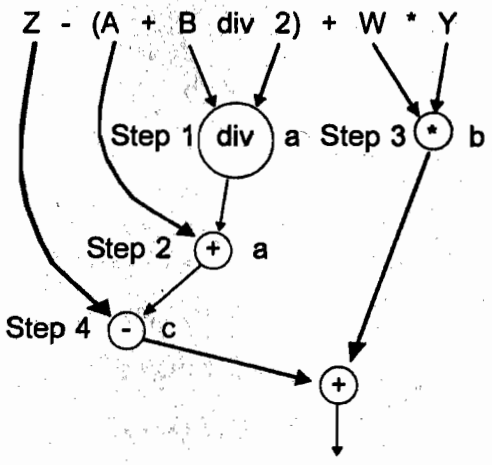
ตัวอย่าง 2.10 จงพิจารณานิพจน์

$$Z - (A + B \text{ div } 2) + W * Y$$

สมมติว่าตัวแปรทุกตัวเป็นชนิด Integer นิพจน์ (A + B div 2) ถูกประเมินผลเป็นอันดับแรก (กฎข้อ a)

เมื่อค่าของ B div 2 ถูกคำนวณ จากนั้นเอาไปบวกกับ A จึงได้ค่าของ (A + B div 2) ต่อไป กระทำการดำเนินการคูณ (กฎข้อ b) คำนวณค่าของ W * Y จากนั้น เอาค่าของ (A + B div 2) ลบออกจาก Z (กฎข้อ c)

สุดท้าย ผลลัพธ์ที่ได้บวกกับ W * Y สำหรับต้นไม้การประเมินผล สำหรับนิพจน์นี้แสดงไว้ในรูป 2.11



เหตุผล

Step 1 : ทำ div ก่อน +

Step 2 : นิพจน์ที่อยู่ภายในวงเล็บประเมินผลแยกจากกัน

Step 3 : ทำ * ก่อน + และ -

Step 4 : กฎการเปลี่ยนหมู่ทำจากซ้าย

รูป 2.11 ต้นไม้การประเมินผลสำหรับนิพจน์
Z - (A + B div 2) + W * Y

การเขียนสูตรคณิตศาสตร์ใน Pascal (Writing Mathematical Formulas in Pascal)

เราอาจพบปัญหาสองอย่างในการเขียนสูตรคณิตศาสตร์ใน Pascal เรื่องแรกเกี่ยวกับการคูณ บ่อยครั้งสูตรเขียนตัวตั้งคูณ (multiplicands) สองตัวติดกัน ตัวอย่างเช่น a = bc

ใน Pascal เราต้องใช้ตัวดำเนินการ * เพื่อแสดงถึงการคูณเสมอ เช่น

A := B * C

สิ่งที่ยากในสูตร คือ การหาร ซึ่งปกติเขียนตัวเศษ (numerator) และตัวส่วน (denominator) บนบรรทัดแยกจากกัน เช่น

$$m = \frac{y - b}{x - a}$$

ใน Pascal, ตัวเศษและตัวส่วนต้องเขียนบนบรรทัดเดียวกัน ดังนั้น เครื่องหมายวงเล็บจึงจำเป็นต้องนำมาใช้แยกตัวเศษออกจากตัวส่วน และเพื่อแสดงอันดับอย่างชัดเจนของการประมวลผลของตัวดำเนินการในนิพจน์ จากสูตรข้างต้นใน Pascal เขียนดังนี้

```
M := (Y - B) / (X - A)
```

ตาราง 2.9 เป็นตัวอย่างของสูตรคณิตศาสตร์ ซึ่งเขียนใหม่ด้วยภาษา Pascal

ตาราง 2.9 สูตรคณิตศาสตร์เป็นนิพจน์ Pascal

Mathematical Formula	Pascal Expression
1. $b^2 - 4ac$	`B * B - 4 * A * C`
2. $a + b - c$	`A + B - C`
3. $\frac{a + b}{c + d}$	`(A + B) / (C + D)`
4. $\frac{1}{1 + x^2}$	`1 / (1 + X * X)`
5. $a X - (b + c)$	`A * X (- (B + C))`

สูตรที่ 1 และสูตรที่ 4 ระบุการคูณอย่างชัดแจ้งโดยใช้ตัวดำเนินการ *

สูตรที่ 3 และสูตรที่ 4 ใช้เครื่องหมายวงเล็บกำกับ เพื่อควบคุมอันดับของการประเมินผลตัวดำเนินการ

ตัวดำเนินการคำนวณสองตัวเขียนติดกันไม่ได้ ดังนั้นจึงต้องแยกจากกันด้วยตัวถูกดำเนินการ หรือวงเล็บเปิด (สูตรที่ 5)

ตัวดำเนินการลบชนิดเอกภาค (Unary Minus)

ในตาราง 2.9 นิพจน์ Pascal ชุดที่ห้า ใช้ตัวดำเนินการลบชนิดเอกภาค เพื่อทำให้ค่าของ (B + C) เป็นลบ ก่อนกระทำคูณ ตัวดำเนินการลบชนิดเอกภาค จะมีตัวถูกดำเนินการเพียงหนึ่งตัวเท่านั้น และมีการทำก่อน เหมือนกับตัวดำเนินการลบ

นิพจน์ชนิดผสมซึ่งมีทั้ง div และ mod (Mixed-Type Expressions with div and mod)

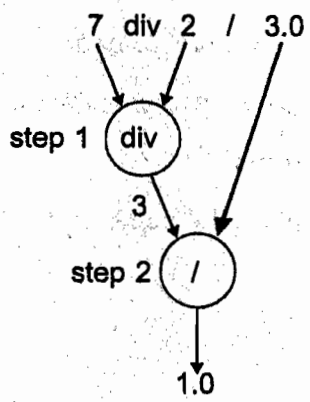
เราต้องระมัดระวังอย่างมากเมื่อใช้ div และ mod ในนิพจน์ชนิดผสม หรือใช้กับตัวดำเนินการหาร / ชนิด Real นิพจน์

7 div 2 / 3.0 เขียนถูกต้อง (valid) ประเมินผลเป็น 1.0

ดังแสดงในรูป 2.12 แต่อย่างไรก็ตาม นิพจน์

7 / 2 div 3. เขียนไม่ถูกต้อง (invalid)

เพราะว่า 7 / 2 คือ 3.5 (/ ให้ผลลัพธ์เป็นชนิด Real) และ 3.5 เป็นตัวถูกดำเนินการของ div ไม่ได้



รูป 2.12 ต้นไม้การประเมินผลสำหรับนิพจน์ 7 div 2 / 3.0

กรณีศึกษา การประเมินผลเหรียญ (Evaluating Coins)

ปัญหา

ลูกสาวของเราเก็บเหรียญ nickels และ pennies ไว้ในกล่องสะสม สิ้นปีเธอต้องการแลกเปลี่ยนเป็นธนบัตรดอลลาร์เพื่อนำไปซื้อของขวัญปีใหม่ เธอต้องการทราบว่าเหรียญที่เธอสะสมไว้มีมูลค่าเป็นกี่ดอลลาร์ กี่เซ็นต์

วิเคราะห์

วิธีการแก้ปัญหานี้ เราต้องนับจำนวนเหรียญที่สะสมไว้ แยกเป็นชนิด nickels มีกี่เหรียญ และชนิด pennies มีจำนวนกี่เหรียญ จากนั้นคำนวณมูลค่าทั้งหมดของเหรียญให้มีหน่วยเป็นเซ็นต์ เมื่อได้เลขตัวนี้แล้ว ใช้ 100 เป็นตัวหารแบบ integer จะได้ผลหารเป็นจำนวนดอลลาร์ ส่วนเศษเหลือจะมีหน่วยเป็นเซ็นต์ ให้คืนเธอไป

ใน data requirement รายการมูลค่าทั้งหมดหน่วยเป็นเซ็นต์ (Total Cents) เป็นตัวแปรในโปรแกรม เพราะว่าจำเป็นต้องนำไปประมวลผล แต่ไม่ใช่เป็นส่วนของเอาต์พุต

Data Requirements

Problem Inputs

Name	: string	{your danghter's name}
Nickels	: Integer	{the count of nickels}
Pennies	: Integer	{the count of pennies}

Problem Outputs

Dollas	: Integer	{The number of dollars she should receive}
Change	: Integer	(The loose change she should receive}

Additional program Variables

TotalCents	: Integer	{the total number of cents}

Relevent Formulas

1 dollar = 100 pennies (cents)

1 nickel = 5 pennies

ตัวแปรอินพุต Name เก็บชื่อของผู้ใช้และช่วยในการโต้ตอบเป็นส่วนตัวระหว่างลูกสาวของเรากับโปรแกรม

ออกแบบ

อัลกอริทึมเริ่มต้น

1. อ่านและแสดงผลชื่อผู้ใช้โปรแกรม
2. อ่านจำนวนเหรียญชนิด nickels และชนิด pennies
3. คำนวณมูลค่าทั้งหมดของเหรียญให้มีหน่วยเป็นเซ็นต์
4. คำนวณค่ามีหน่วยเป็นดอลลาร์และเงินที่เป็นเศษเหลือ
5. แสดงผลค่ามีหน่วยเป็นดอลลาร์และเงินที่เป็นเศษเหลือ

ขั้นที่ 3 และ 4 จำเป็นต้องแบ่งละเอียดดังนี้

ขั้นที่ 3 การแบ่งละเอียด

3.1 TotalCents หมายถึง ค่าของ Nickels มีหน่วยเป็นเซ็นต์ บวกกับ Pennies

ขั้นที่ 4 การแบ่งละเอียด

4.1 Dollas หมายถึง ผลหารเป็นเลขจำนวนเต็ม มี TotalCents เป็นตัวตั้ง 100 เป็นตัวหาร

4.2 Change หมายถึง เศษเหลือซึ่งเป็นเลขจำนวนเต็ม มี TotalCents เป็นตัวตั้ง 100 เป็นตัวหาร

การปฏิบัติให้เกิดผล

โปรแกรมแสดงในรูป 2.13

อินพุตวินโดว์ (Input Window)

```
Program Coins;
{Determines the value of a coin collection}

var
   Name : string;          {input - name of program user}
   Pennies,                {input - count of pennies}
   Nickels,                {input - count of nickels}
   Dollars,                {output - number of dollars}
   Change,                 {output - loose change}
   TotalCents : Integer;   {total cents}
```

```
begin
   {Read and display the program user's name.}
   Write (' Type in your name and press Enter > ');
   ReadLn (Name);
   Write (' Hello ' , Name);
   WriteLn (' . Let ' ' s find the value of your coins . ');
   {Read in the count of nickels and pennies .}
   Write (' Number of nickels > ');
   ReadLn (Nickels);
   Write (' Number of pennies > ');
   ReadLn (Pennies);

   {Compute the total value in cents.}
   TotalCents := 5 * Nickels + Pennies;

   {Find the value in dollars and change.}
   Dollars := TotalCents div 100;
   Change := TotalCents mod 100;

   {Display the value in dollars and change.}
   WriteLn;
   Write (' Your coins are worth ' , Dollars : 5 ' dollars ');
   WriteLn (' and ', Change : 5 ' cents.')
end.
```

เอาต์พุตวินโดว์ (Output Window)

```
Type in your name and press Enter > Sally
Hello Sally. Let's find the value of your coins.
Number of nickels > 30
Number of pennies > 77
Your coins are worth 2 dollars and 27 cents.
```

รูป 2.13 การหาค่าของเหรียญ

ข้อความสั่ง

Wrint (' Hello ' , Name);

WriteLn (' . Let ' ' s find the value of your coins . ');

แสดงผลบรรทัดที่สอง ในเอาต์พุตวินโดว์

ในบรรทัดที่สอง คำ Let ' ' s มี apostrophe สองตัวติดกัน ใช้แทน apostrophe หนึ่งตัวภายในสายอักขระ

ถ้านักศึกษาใช้ apostrophe หนึ่งตัวจะเกิด syntax error

ข้อความสั่ง

TotalCents := 5 * Nickels + Pennies;

ใช้ implement อัลกอริทึมขั้น 3.1

ข้อความสั่ง

Dollars := TotalCents div 100;

Change := TotalCents mod 100;

ใช้ตัวดำเนินการคำนวน div และ mod เพื่อ implement อัลกอริทึมขั้น 4.1 และขั้น 4.2 ตามลำดับ

การทดสอบ

การทดสอบโปรแกรมนี้ พยายาม run โปรแกรมโดยให้มีจำนวนเหรียญชนิด nickels และชนิด pennies ซึ่งจะให้จำนวน dollars พอดีไม่มีเศษเหลือ ตัวอย่างเช่น มี 35 nickels และ 25 pennies ซึ่งผลลัพธ์ที่ถูกต้องคือ 2 ดอลลาร์ไม่มีเซ็นต์ หลังจากนั้นให้เพิ่มจำนวนหรือลดจำนวนเหรียญชนิด pennies ครั้งละ 1 เหรียญ (เช่น เป็น 26 nickels และ 24 pennies) เพื่อตรวจสอบให้มั่นใจว่ากรณีเหล่านี้โปรแกรมจัดกระทำได้ถูกต้อง

แบบฝึกหัด 2.7

1. a) จงประเมินผลนิพจน์ต่อไปนี้ ซึ่งมี 7 และ 22 เป็นตัวถูกดำเนินการ (operands)

   22 div 7 22 mod 7

   7 div 22 7 mod 22

   b) ทำซ้ำ (repeat) ข้อ a) โดยใช้คู่ของเลขจำนวนเต็มต่อไปนี้

   15, 16 2, 23 -14, 16

2. กำหนดการประกาศดังนี้

```
const
   MyPi = 3.14159;
   MaxI = 1000;
var
   X, Y   : Real;
   A, B, I : Integer;
```

จงแสดงให้เห็นว่ามีข้อความสั่งชุดใดบ้างที่ถูกต้อง (valid) จงหาค่าของข้อความสั่งที่ถูกต้องทุกข้อ และจงแสดงให้เห็นว่ามีข้อความสั่งชุดใดบ้างไม่ถูกต้อง (invalid) ให้อธิบายเหตุผลประกอบด้วย สมมติว่า A มีค่าเท่ากับ 3 B มีค่าเท่ากับ 4 และ Y มีค่าเท่ากับ -1.0

```
a) I  := A mod B
b) I  := (990 - MaxI) dIv A
c) I  := A mod Y
d) X  := MyPi * Y
e) I  := A / B
f) X  := A / B
g) X  := A mod (A / B)
h) I  := B div O
I) I  := A mod (990 - MaxI)
j) I  := (Max I - 990) div A
k) X  := A / Y
l) I  := MyPi * A
m)X  := MyPi div Y
n) X  := A div B
o) I  := (MaxI - 990) mod A
p) I  := A mod O
q) I  := A mod (MaxI - 990)
```

3. จงวาดรูปต้นไม้การประเมินผล (evaluation trees) ของนิจน์ต่อไปนี้ และค่าสุดท้ายคืออะไร

1.8 * Celsius + 32.0

(Salary - 5000.00) * 0.20 + 1425.00

10 mod 4 + 1 div 2

4. จงเขียนข้อความสั่งกำหนดค่า (assignment statement) เพื่อให้สมการข้างล่างนี้ปฏิบัติได้ (implement) ในภาษา Pascal

$$q = \frac{kA(T1 - T2)}{L}$$

5. สมมติว่าเราได้ประกาศตัวแปรดังนี้

```
var
   Color, Lime, Straw, Red, Orange : Integer;
   White, Green, Blue, Purple, Crayon : Real;
```

จงประเมินผลข้อความสั่งต่อไปนี้ โดยให้ Color = 2, Crayon = -1.3, Strew = 1, Red = 3 และ Purple = 0.2E + 1

a) White := Color * 2.5 / Purple

b) Green := Colow / Purple

c) Orange := Color div Red

d) Blue := (Color + Straw) / (Crayon + 0.3)

e) Line := Red div Color + Red mod Color

f) Purple := Straw / Red * Color

6. ให้ A, B, C และ X เป็นชื่อสี่ชื่อของตัวแปรชนิด Real และให้ I, J และ K เป็นชื่อสามชื่อของตัวแปรชนิด Integer ข้อความสั่งข้างล่างนี้ทุกบรรทัดฝ่าฝืนกฎของการเขียนนิพจน์คำนวณ จงเขียนใหม่เพื่อให้ข้อความสั่งทุกบรรทัดสอดคล้อง (consistent) กับกฎ

a) X := 4.0A * C

b) A := AC

c) I := 2 * -J

d) K := 3(I + J)

e) X := 5A / BC

f) I := 5J3

การเขียนโปรแกรม

จงศึกษาโปรแกรมในรูป 2.13 แล้วเขียนโปรแกรมใหม่เพื่อแลกเหรียญ 1, 2, 5 และ 10 บาท ให้เป็นธนบัตรฉบับราคา 20, 50, 100 และ 500 บาท ตามลำดับ และคืนส่วนที่เป็นเศษเหลือ

ตัวอย่าง อินพุต

สะสม 1 บาท มี 35 เหรียญ = 35
2 บาท มี 16 เหรียญ = 32
5 บาท มี 34 เหรียญ = 170
10 บาท มี 62 เหรียญ = 620
รวม 857 บาท

เอาต์พุต เป็นดังนี้

ธนบัตร 20 บาท ไม่มี
50 บาท มี 1 ใบ
100 บาท มี 3 ใบ
500 บาท มี 1 ใบ และคืนเศษเหรียญ 7 บาท

2.8 การจัดรูปแบบและการดูโปรแกรมเอาต์พุต (Formatting and Viewing Program Output)

Pascal แสดงผลเลขจำนวนจริงทั้งหมดในสัญกรณ์เชิงวิทยาศาสตร์ (scientific notation) ถ้าเราไม่สั่งให้ทำกรณีอื่น ในหัวข้อนี้จะอธิบายว่า ถ้าต้องการกำหนดรูปแบบหรือแสดงผลเอาต์พุตของเราเอง จะทำได้อย่างไร

การจัดรูปแบบค่าจำนวนเต็ม (Formatting Integer Values)

การระบุรูปแบบของตัวแปรชนิด Integer หรือค่าชนิด Integer ให้แสดงผลโดยโปรแกรมทำได้ง่าย เราเพียงใส่สัญลักษณ์ : fw หลังตัวแปรหรือค่า เมื่อ fw หมายถึง ความต้องการเขตข้อมูล หรือจำนวนเลขโดดซึ่งต้องการให้แสดง

ความกว้างของเขตข้อมูล หมายถึง จำนวนตัวอักขระซึ่งใช้เพื่อแสดงผลหนึ่งค่า (Field width is the number of characters used to display a value.)

จงพิจารณาข้อความสั่งข้างล่างนี้

```
Write (' Your coins are worth ' , Dollars : 1, ' dollars ');
WriteIn (' and ' , Change : 2 , ' cents . ')
```

หมายถึง เลขโดดหนึ่งหลักจะใช้แสดงผลค่าของ Dollars และเลขโดดสองหลักใช้แสดงผลค่าของ Change (เลขระหว่าง 0 ถึง 99) ถ้า Dollars มีค่าเท่ากับ 7 และ Change มีค่าเท่ากับ 8 โปรแกรมเอาต์พุตจะเป็นดังนี้

```
Your coins are worth 7 dollars and 8 cents.
```

ในบรรทัดนี้ โปรดสังเกตว่า มีเนื้อที่เพิ่ม 1 ที่ก่อนค่าของ Change (ค่าเท่ากับ 8) เหตุผลคือ รูปแบบข้อกำหนด format specification : 2 ให้เนื้อที่สำหรับเลขโดดสองตัวสำหรับการแสดงผล ถ้าค่าของ Change เป็นเลขหนึ่งตัว (0 ถึง 9) เลขหนึ่งตัวแสดงชิดขวา (right - justified) และมีอักขระว่างหนึ่งตัวนำหน้าตัวเลข เราสามารถใช้ format symbol : 2 เพื่อแสดงผลค่าเอาต์พุตระหว่าง -9 ถึง 99 สำหรับเลขที่มีค่าลบ เราต้องนับรวมเครื่องหมายลบไว้ในเลขที่แสดงผลด้วย

ตาราง 2.10 แสดงให้เห็นว่าค่าของจำนวนเต็มสองค่าแสดงผลโดยใช้ข้อกำหนดรูปแบบแตกต่างกัน ทำได้อย่างไร ในตารางนี้ สัญลักษณ์ ␣ แทนอักขระว่าง ตัวอย่างบรรทัดที่ 4 แสดงให้เห็นว่า Pascal ขยายความกว้างของเขตข้อมูลอัตโนมัติ ถ้าเรากำหนดความกว้างไว้น้อยกว่าค่า integer ที่ต้องการแสดงผล

ตาราง 2.10 แสดงผล 234 และ -234 โดยใช้รูปแบบแตกต่างกัน

Value	Format	Displayed Output
234	: 4	␣234
234	: 5	␣␣234
234	: 6	␣␣␣234
234	: 1	234
234	: Len	␣␣234 (ถ้า Len มีค่าเท่ากับ 5)
-234	: 4	-234
-234	: 5	␣-234

-234	: 6	□□-234
-234	: 1	-234
-234	: Len	□-234 (ถ้า Len มีค่าเท่ากับ 5)

การจัดรูปแบบค่าจำนวนจริง (Formatting Real Values)

การอธิบายข้อกำหนดรูปแบบสำหรับตัวแปรหรือค่าชนิด Real เราต้องระบุทั้งความกว้างของเขตข้อมูลทั้งหมดที่ต้องการและจำนวนจุดทศนิยมไว้ข้างหลังตัวแปรหรือค่า (: fw : dp) ความกว้างของเขตข้อมูลทั้งหมดต้องมีขนาดใหญ่พอที่จะใส่เลขโดดทั้งหมดก่อนและหลังจุดทศนิยม เราควรจะรวมสดมภ์แสดงผลจุดทศนิยมและสดมภ์สำหรับเครื่องหมายลบ ถ้าเลขนั้นสามารถเป็นลบได้

ถ้า X เป็นตัวแปรชนิด Real มีค่าอยู่ระหว่าง -99.9 และ 999.9 เราควรใช้ output list item เป็นดังนี้

X : 5 : 1

เพื่อแสดงผลค่าของ X โดยมีทศนิยมหนึ่งตำแหน่ง

ตาราง 2.11 แสดงให้เห็นค่าแตกต่างกันของ X แสดงผลโดยใช้ข้อกำหนดรูปแบบข้างต้น ค่าซึ่งแสดงผลถูกปัดเศษให้มีจุดทศนิยมหนึ่งหลัก และค่าทั้งหมดแสดงผลแบบชิดขวาในห้าสดมภ์

ตาราง 2.11 แสดงผลค่าของ X โดยใช้ข้อกำหนดรูปแบบ : 5 : 1

Value of X	Display Output
99.42	□99.4
0.123	□□0.1
-9.53	□-9.5
-25.55	-25.6
99.999	100.0
999.43	999.4

ตาราง 2.12 แสดงให้เห็นค่าจำนวนจริง ซึ่งแสดงผลโดยใช้ข้อกำหนดรูปแบบอื่นๆ ตัวอย่างบรรทัดที่ 6 และบรรทัดที่ 13 แสดงให้เห็นว่า Pascal ขยายความกว้างของเขตข้อมูลอัตโนมัติ ถ้าเรากำหนดความกว้างไว้น้อยกว่าค่าจำนวนซึ่งต้องการแสดงผล เราสามารถใช้ข้อกำหนดรูปแบบ 3 : 1 และ 4 : 2 เพื่อแสดงผลค่าจำนวนจริงใดๆ โดยไม่ต้องมีอักขระว่างนำหน้า ตัวอย่างบรรทัดที่ 7 และบรรทัดที่ 14 แสดงให้เห็นว่าเราสามารถใช้ข้อกำหนดรูปแบบ : n สำหรับค่า real ซึ่งจะแสดงผลเป็นสัญกรณ์วิทยาศาสตร์ โดยใช้ความกว้างของเขตข้อมูลเท่ากับ n

ตาราง 2.12 การจัดรูปแบบค่าจำนวนจริง

Value	Format	Displayed Output
3.14159	: 5 : 2	□3.14
3.14159	: 4 : 2	3.14
3.14159	: 5 : 3	3.142
0.1234	: 4 : 2	0.12
-0.006	: 8 : 3	□□-0.006
25.876	: 3: 1	25.9
-0.006	: 9	-6.00E-03
3.14159	: 4 : 2	3.14
3.14159	: 5.1	□□3.1
3.14159	: 8 : 5	□3.14159
-0.006	: 4 : 2	-0.01
-0.006	: 8 : 5	-0.00600
-124.3123	: 4 : 2	-124.31
-3.14159	: 9	-3.14E+00

สไตล์โปรแกรม (Program Style)

การไม่ให้ใส่อักขระว่างนำหน้า (Eliminating Leading Blanks)

จากที่แสดงให้เห็นในตาราง 2.10 จนถึงตาราง 2.12 เลขที่ต้องการแสดงผลใช้สดมภ์น้อยกว่าที่กำหนดในความกว้างของเขตข้อมูล จึงมีอักขระว่างนำหน้า การขจัดอักขระว่างนำหน้าให้เลือกรูปแบบซึ่งจะแสดงผลค่าน้อยที่สุดซึ่งคาดเอาไว้ (To eliminate leading blanks, choose a format that will display the smallest value expected.)

ถ้าค่าจริงต้องการสดมภ์แสดงผลมากกว่าความกว้างของเขตข้อมูลจะขยายเพื่อใส่เองอัตโนมัติ ด้านนี้รูปแบบ : 1 หมายถึง แสดงผลค่าจำนวนเต็มใดๆ โดยไม่มีอักขระว่างนำหน้า (ตัวอย่างเช่น 29397 : 1 แสดงผลเป็น 29397) รูปแบบ : 3 : 1 แสดงผลเลขจำนวนจริงใดๆ โดยต้องการจุดทศนิยมหนึ่งตำแหน่ง และไม่ให้มีอักขระว่างนำหน้า (ตัวอย่างเช่น 99397.567 : 3 : 1 แสดงผลเป็น 99397.6) ในทำนองเดียวกัน รูปแบบ : 4 : 2 แสดงผลเลขจำนวนจริงใดๆ โดยให้มีจุดทศนิยมสองตำแหน่ง และไม่มีอักขระว่างนำหน้า

การจัดรูปแบบสายอักขระ (Formatting Strings)

ค่าของสายอักขระ (string value) แสดงผลในเขตข้อมูล แบบจัดขวา (right-justified) เสมอ เพราะฉะนั้นอักขระว่างจะอยู่หน้าสายอักขระ ถ้าเรากำหนดความกว้างของเขตข้อมูลมากกว่าค่าของสายอักขระ ในกรณีที่ถ้าความกว้างของเขตข้อมูลน้อยกว่าค่าของสายอักขระ Turbo Pascal จะขยายความกว้างของเขตข้อมูล เพื่อให้สายอักขระทั้งหมดสามารถแสดงผลได้ (ส่วนใน standard Pascal จำนวนตัวอักขระที่แสดงผลจะเท่ากับความกว้างของเขตข้อมูล ทำให้ไม่สามารถแสดงผลตัวอักขระได้ทั้งหมด)

ตาราง 2.13 แสดงผลค่าของสายอักขระโดยใช้การจัดรูปแบบ

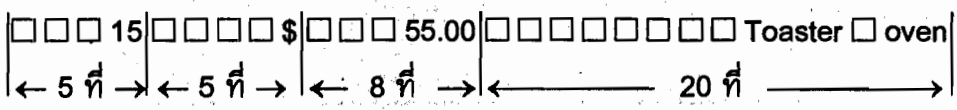
string	Format	Display Output
' * '	: 1	*
' * '	: 2	□*
' * '	: 3	□□*
' ACES '	: 1	ACES (A in standard Pascal)
' ACES '	: 2	ACES (AC in standard Pascal)

' ACES '	: 3	ACES (AC in standard Pascal)
' ACES '	: 4	ACES (AC in standard Pascal)
' ACES '	: 5	ACES (AC in standard Pascal)

ตัวอย่าง 2.11 ข้อความสั่ง

```
Quantity := 15;
Cost := 55.0;
Description := 'Toaster oven";
WriteLn (Quantity : 5, '$' : 5, cost : 8 : 2 Description : 20)
```

แสดงผล จำนวนเต็ม (ค่าของ Quantity) ตัวอักขระ ($) เลขจำนวนจริง (ค่าของ Cost) และสายอักขระ (ค่าของ Description) บรรทัดเอาต์พุต เป็นดังนี้



เมื่อสัญลักษณ์ □ หมายถึง อักขระว่าง มีอักขระว่าง 3 ตัวอยู่หน้าเลขจำนวนเต็ม 15 มีอักขระว่าง 4 อยู่หน้าตัวอักขระ $ มีอักขระว่าง 3 ตัวอยู่หน้าเลขจำนวนจริง 55.00 และมีอักขระว่าง 8 ตัวอยู่หน้าสายอักขระ หากเราต้องการให้ $ อยู่ติดกับเลขจำนวนจริง 55.00 สามารถทำได้โดยการเปลี่ยน output list item รายการที่สามเป็น Cost : 5 : 2

การใช้หน้าต่างเอาต์พุต (Using the Output Window)

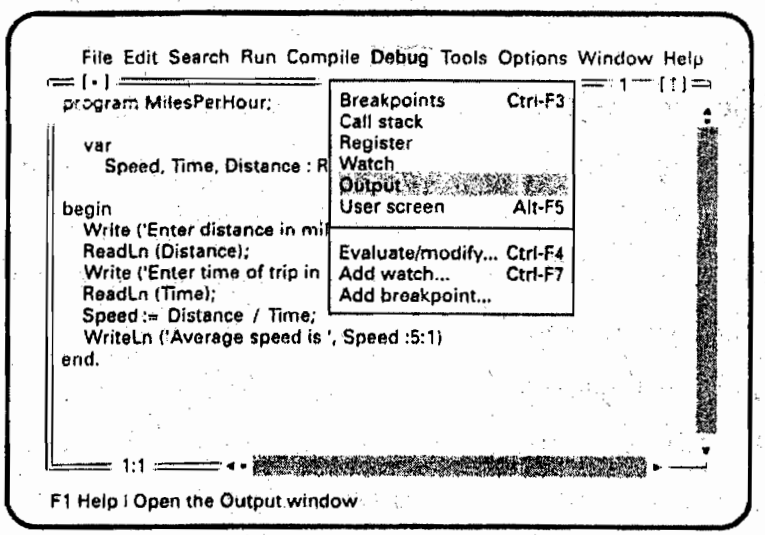
โดยปกติ Turbo Pascal แสดงผลหน้าต่างตรวจแก้ (Edit window) ขณะวิ่งโปรแกรม Turbo Pascal จะสลับทันทีกลับไปยังจอผู้ใช้ (user screen) เมื่อใดก็ตามที่โปรแกรมหยุดชั่วคราว เพื่อใส่ข้อมูล การกดปุ่ม Alt และปุ่ม F5 พร้อมกัน หมายถึงแสดงผลจอผู้ใช้ ทำให้เราเห็นเอาต์พุตโปรแกรม เมื่อโปรแกรมกระทำการเสร็จสิ้นแล้ว อย่างไรก็ตาม โปรแกรมเมอร์บางคนชอบมองดูเอาต์พุตของโปรแกรมอย่างต่อเนื่องขณะที่วิ่งโปรแกรม

Turbo Pascal ยอมให้เราเปิดหน้าต่างเอาต์พุต (Output window) และจะแสดงผลบนจอ ณ เวลาเดียวกับหน้าต่างตรวจแก้ ในการทำสิ่งนี้ ให้เลือก Output option จาก Debug menu ดังแสดงในรูป 2.14 ซึ่งทำให้หน้าต่างเอาต์พุตปรากฏที่ตอนล่างของจอ และเลือกมัน

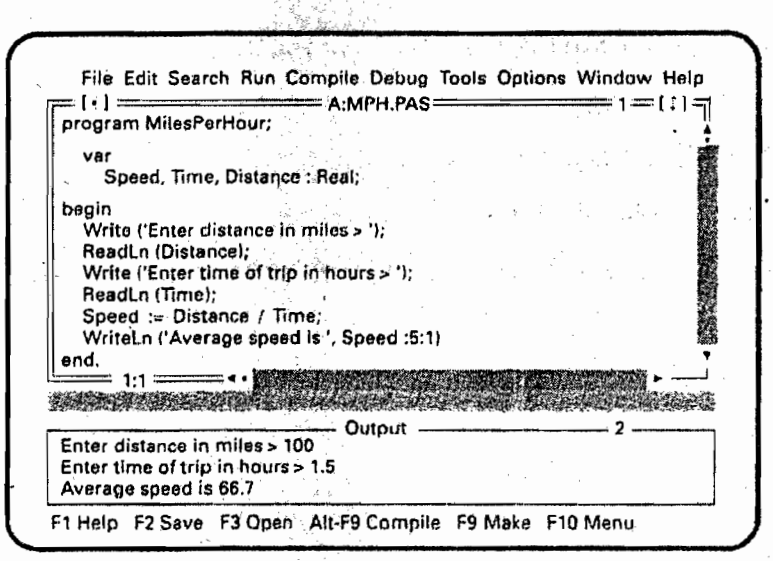
เป็น active window เราสามารถใช้ mouse หรือ arrow keys เพื่อเลื่อนขึ้นหรือลงตลอดหน้าต่างเอาต์พุต และแสดงผลเอาต์พุตซึ่งไม่ปรากฏในหน้าต่างเอาต์พุต การกลับคืนมายัง Edit window และทำให้มันเป็น active window ให้วาง mouse แล้วคลิก (click) บน Edit window หรือกดปุ่ม F6 (ปุ่ม F6 ทำให้เราสลับกลับไปกลับมาระหว่าง Edit window กับ Output window)

เมื่อเรากลับคืนมายัง Edit window จะสังเกตเห็นว่าขณะนี้มันครอบคลุม Output window ไว้ด้วย เพื่อแสดงผลทั้งสองหน้าต่าง ในเวลาเดียวกัน เราสามารถลดขนาดของ Edit window และไม่ครอบคลุม Output window โดยการใช้ mouse (คลิกและ drag มุมขวาล่างสุดของ Edit window) หรือจากการเลือก Size/More option จาก Window menu และใช้ arrow keys ขณะที่กดปุ่ม shift

การดำเนินการเหล่านี้ อธิบายในรายละเอียดมากขึ้น ใน Help screen สำหรับ Size/More รูป 2.15 แสดงให้เห็นจอ ซึ่งลดขนาด Edit window และแสดงทั้ง Edit และ Output windows



รูป 2.14



รูป 2.15

การพิมพ์โปรแกรมและเอาต์พุต (Printing Programs and Output)

มีหลายวิธีในการพิมพ์โปรแกรมและเอาต์พุตของโปรแกรม วิธีที่ง่ายที่สุดคือใช้ปุ่ม PrintScreen บนคีย์บอร์ด

ขั้นแรก แสดง Edit window หรือ user screen บนจอ จากนั้นกดปุ่ม PrintScreen เพื่อให้ได้สำเนาของจอปัจจุบันไปสู่ Printer

อีกวิธีหนึ่งคือพิมพ์สำเนา program file จาก MS-DOS โดยใช้คำสั่งงาน ดังนี้

TYPE FileName.PAS > PRN

หมายถึง พิมพ์ FileName.PAs ถ้าเราไม่มีตัวอักขระ >PRN หมายถึง file จะแสดงออกบนจอภาพ

ถ้าเครื่องคอมพิวเตอร์ของเรามี printer เอง เราสามารถเรียก MS-DOS Logging feature ให้ทำงาน ซึ่งจะทำให้อักขระทุกตัว ส่งไป user screen พร้อมกับส่งไป printer ในเวลาเดียวกัน การเรียก logging feature ให้ทำงาน เลือก DOS shell จาก File submenu และกดปุ่ม Ctrl และปุ่ม PrintScreen เมื่อต้องการกลับมายัง Turbo Pascal ให้พิมพ์ EXIT ถ้าต้องการออกจาก logging feature ให้ทำกระบวนการนี้ซ้ำ

แบบฝึกหัด 2.8

1. จงแสดงบรรทัดเอาต์พุตของข้อความสั่งต่อไปนี้

```
Write (-99 : 4);
WriteLn ('Bottles' : 8);
WriteLn (-99 : 4, -99 : 8)
```

2. จงแสดงค่าของ -15.564 (เก็บใน X) จะแสดงผลอย่างไร เมื่อใช้การจัดรูปแบบต่อไปนี้

```
X : 8 : 4
X : 8 : 3
X : 8 : 2
X : 8 : 1
X : 8 : 0
X : 8
```

3. จงบอกเอาต์พุตของลำดับต่อไปนี้ของข้อความสั่งโปรแกรม สมมติว่า I เป็น Integer ให้ใช้สัญลักษณ์ □ แทนอักขระว่าง

```
I := 1;
WriteLn (I : 5, '□', I : 1, I : 5);
I := I * 10;          {I = 1 * 10 = 10}
WriteLn (I : 5, '□', I : 1, I : 5);
I := I * 10;          {I = 10 * 10 = 100}
WriteLn (I : 5, '□', I : 1, I : 5);
I := I * 10;          {I = 100 * 10 = 1000}
WriteLn (I : 5, '□', I : 1, I : 5)
```

4. สมมติว่า X (มีชนิดเป็น Real) มีค่าเท่ากับ 12.335 และ I (มีชนิดเป็น Integer) มีค่าเท่ากับ 100 จงแสดงบรรทัดเอาต์พุตของข้อความสั่งต่อไปนี้

```
Write (' X is ' : 10, X : 6 : 2, ' I is ' : 4, I : 5);
Write (' I is ' : 10, I : 1);
Write (' X is ' : 10, X : 2 : 1)
```

การเขียนโปรแกรม (Programming)

จงเขียนชุดของข้อความสั่งเพื่อแสดงผลหัวเรื่อง (heading) สามบรรทัดประกอบด้วย ชื่อนักศึกษา โรงเรียน จังหวัด และรหัสไปรษณีย์ ทุกบรรทัดให้อยู่ตรงกลางบนจอ คอมพิวเตอร์

2.9 การแก้จุดบกพร่องและข้อผิดพลาดในโปรแกรม (Debugging and Programming Errors)

ไม่นานนัก โปรแกรมเมอร์มือใหม่จะค้นพบว่าโปรแกรมน้อยมากซึ่งวิ่ง (run) ได้อย่างถูกต้องในครั้งแรกที่กระทำการ

ข้อผิดพลาดในโปรแกรมมีชื่อเฉพาะ เรียกว่า จุดบกพร่อง (bugs)

กระบวนของการแก้ไขจุดบกพร่อง เรียกว่า การแก้จุดบกพร่องของโปรแกรม (Degugging is the process of removing errors from a program.)

เมื่อ Turbo Pascal ตรวจพบข้อผิดพลาด คอมพิวเตอร์จะแสดงข้อความระบุความผิดพลาด (error message) ซึ่งเราได้ทำผิดไว้และอาจเป็นเหตุให้เกิดข้อผิดพลาดขึ้น เราจึงต้องแก้ไขข้อความสั่งที่ผิดตรงตำแหน่ง cursor หลังจากข้อความระบุความผิดพลาดถูกแสดงขึ้น ถ้าเรากดปุ่ม F1 ระบบ Help ของ Turbo Pascal จะให้สารสนเทศเพิ่มเพื่ออธิบายข้อผิดพลาด เมื่อเรามีประสบการณ์ เราจะมีความสามารถมากขึ้นในการหาตำแหน่งที่ผิดพลาดและแก้ไขที่ผิดพลาดให้ถูกต้อง

ข้อผิดพลาดแบ่งออกเป็นสามชนิด ได้แก่

ข้อผิดพลาดวากยสัมพันธ์ (syntax errors)

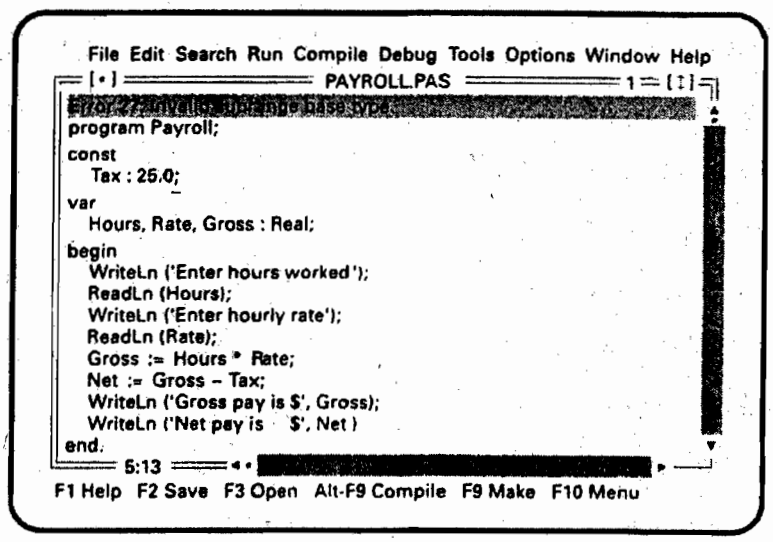
ข้อผิดพลาดเวลาวิ่งโปรแกรม (runtime errors) และข้อผิดพลาดตรรกะ (logic errors)

ข้อผิดพลาดวากยสัมพันธ์ (Syntax Errors)

ข้อผิดพลาดวากยสัมพันธ์เกิดขึ้นเมื่อรหัสของเราฝ่าฝืนกฎไวยากรณ์ของ Pascal หนึ่งข้อหรือมากกว่าขึ้นไป ซึ่งคอมไพเลอร์ (compiler) ตรวจพบ และแสดงผลขณะที่พยายามแปลโปรแกรม ถ้าข้อความสั่งนั้นมีข้อผิดพลาดวากยสัมพันธ์ มันจะแปลไม่ได้ และโปรแกรมจะไม่ถูกกระทำการ (not be executed)

โปรแกรม payroll ซึ่งแสดงให้เห็นใน Edit window รูป 2.16 มี syntax error สองแห่ง หลังจากความพยายามครั้งแรกที่จะคอมไพล์โปรแกรมนี้ เราได้ error message ซึ่งเขียนว่า Invalid subrange base type

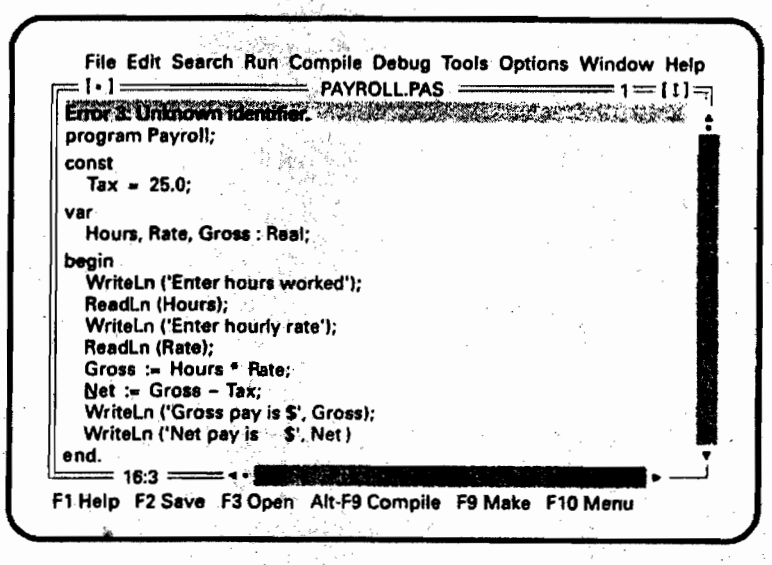
ปัญหาจริง คือ สัญลักษณ์ : ใช้ผิดในตำแหน่ง สัญลักษณ์ = ในการประกาศตัวคงตัว



รูป 2.16

รูป 2.17 แสดง Edit window หลังจากแก้ไขข้อผิดพลาดตำแหน่งแรกแล้ว และคอมไพล์โปรแกรมอีกครั้งหนึ่ง กรณีนี้ error message แสดงว่าคอมไพเลอร์ ไม่พบการประกาศของไอเดนติไฟเออร์ Net หลังจากเราแก้ไขข้อผิดพลาดการประกาศตัวแปรนี้แล้ว โปรแกรมควรจะคอมไพล์ได้สำเร็จ

การแก้ไขข้อผิดพลาดวากยสัมพันธ์ ครั้งละหนึ่งแห่ง จะใช้เวลามากเราควรตรวจสอบโปรแกรมแบบ desk check อย่างระมัดระวัง ก่อนคอมไพล์โปรแกรมในครั้งแรก ข้อผิดพลาดร่วมสองอย่างที่พบจากโปรแกรมเมอร์มือใหม่คือ ลืมประกาศชื่อไอเดนติไฟเออร์ (หรือสะกดผิด) และไม่ใส่ semicolon ข้อผิดพลาดทั้งสองอย่างนี้ตรวจพบง่าย โดยการ deck chedking อย่างระมัดระวัง



รูป 2.17

การใช้แบบชนิดข้อมูล Pascal ไม่ถูกต้องบ่อยครั้งเป็นเหตุให้เกิดข้อผิดพลาดวากยสัมพันธ์ ตัวอย่างเช่น

- ใช้ตัวดำเนินการคำนวณกับข้อมูลชนิด Char หรือ string
- การกำหนดค่าของตัวแปรของชนิดหนึ่งให้กับตัวแปรที่มีแบบชนิดข้อมูลอีกชนิดหนึ่ง (ยกเว้นค่า Integer สามารถกำหนดให้กับตัวแปรชนิด Real ได้)

การใช้ apostrophes ไม่ถูกต้องกับข้อมูลชนิด string ทำให้เกิดข้อผิดพลาดวากย-สัมพันธ์ ข้อมูลชนิด string ต้องอยู่ภายในเครื่องหมาย single quote หรือ apostrophe เสมอ ไม่สามารถใช้ double quotes เมื่อเริ่มต้นหรือจบ string

ข้อผิดพลาดวากยสัมพันธ์ร่วมอีกอย่างหนึ่งคือ ไม่ใส่หรือใส่เกิน apostrophe ใน string ถ้าไม่มี apostrophe ที่ตอนจบของ string คอมไพเลอร์ของ Turbo Pascal จะแสดงข้อความระบุความผิดพลาดดังนี้

.String constant exceeds line

โปรดจำไว้ว่าต้องใช้ apostrophe สองตัวติดกัน เพื่อแทน apostrophe หนึ่งตัวภายใน string ตัวอย่างเช่น

WriteLn ('Enter Joe' 's nickname>');

ข้อผิดพลาดเวลาดำเนินงาน (Run-Time Error)

ข้อผิดพลาดเวลาดำเนินงานตรวจพบและแสดงโดยคอมพิวเตอร์ระหว่างกระทำการโปรแกรม ข้อผิดพลาดชนิดนี้เกิดขึ้นเมื่อผู้ใช้สั่งคอมพิวเตอร์ให้กระทำการดำเนินการที่ไม่ถูกต้อง เช่น การหารเลขด้วยศูนย์ หรือการจัดดำเนินการกับข้อมูลซึ่งไม่ได้ให้นิยามหรือข้อมูลไม่ถูกต้อง

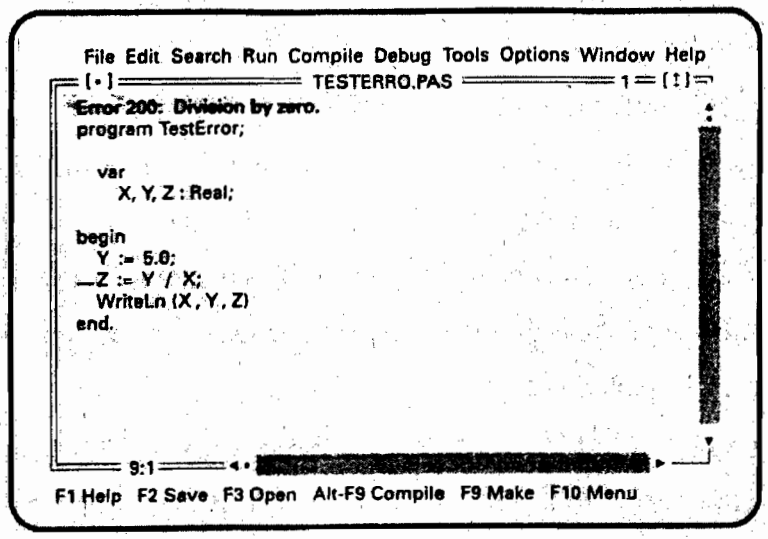
โปรแกรมในรูป 2.18 คอมไพล์เป็นผลสำเร็จ แต่ไม่มีข้อความสั่งกำหนดค่าให้กับตัวแปร X ก่อน ข้อความสั่งกำหนดค่า

Z := Y / X;

การกระทำการเป็นเหตุให้เกิดข้อผิดพลาดเวลาดำเนินการคอมไพเลอร์ Turbo Pascal จะกำหนดค่า default 0.0 ให้ X เมื่อกระทำการข้อความสั่งกำหนดค่าข้างต้น จะเกิดข้อความระบุข้อผิดพลาดดังนี้

Error 200 : Division by Zero

แสดงว่ามีความหมายที่จะหารเลขด้วยเลขศูนย์ ซึ่งเป็นการดำเนินการไม่ถูกต้อง



รูป 2.18

เราจะต้องมาที่ Turbo Pascal editor วาง cursor ตรงตำแหน่งเริ่มต้นของข้อความสั่งกำหนดค่า เลือก Help เพื่อให้ pop up หน้าต่าง แสดงสิ่งที่อาจเป็นเหตุให้มีข้อผิดพลาด

เวลาดำเนินงาน เราควรใส่ข้อความสั่งกำหนดค่า ไม่เท่ากับศูนย์ ให้ X การให้คอมไพเลอร์กำหนดค่า default กับตัวแปร ถือว่าเป็นการฝึกเขียนโปรแกรมที่แย่

ข้อผิดพลาดจากการใส่ข้อมูล (Data entry errors) เป็นข้อผิดพลาดเวลาดำเนินงานร่วมอีกชนิดหนึ่ง เกิดขึ้นจากการอ่านข้อมูลที่เป็นชนิดไม่ถูกต้องให้กับตัวแปร ตัวอย่างเช่น อ่านเลขจำนวนจริงหรือตัวอักขระให้กับตัวแปรชนิด Integer

ข้อผิดพลาดเวลาดำเนินงานร่วมอีกชนิดหนึ่งคือ ส่วนล้นคำนวณ (arithmetic overflow) ข้อผิดพลาดชนิดนี้เกิดขึ้นเมื่อโปรแกรมพยายามจะเก็บค่าซึ่งมีขนาดใหญ่มากกว่าพื้นที่ของตัวแปร

ข้อผิดพลาดตรรกะ (Logic Errors)

ข้อผิดพลาดตรรกะเกิดขึ้นเมื่อโปรแกรมทำตามอัลกอริทึมที่ผิด (faulty algorithm) เนื่องจากข้อผิดพลาดตรรกะปกติไม่ทำให้เกิดข้อผิดพลาดเวลาดำเนินงานและไม่แสดงข้อความระบุข้อผิดพลาด ซึ่งทำให้ยากมากที่จะตรวจพบ สิ่งที่เป็นเพียงสัญญาณ (sign) ของข้อผิดพลาดตรรกะคือเอาต์พุตโปรแกรมอาจจะไม่ถูกต้อง เราสามารถตรวจพบข้อผิดพลาดตรรกะได้ โดยการทดสอบโปรแกรมอย่างละเอียดเปรียบเทียบเอาต์พุตของมันกับผลลัพธ์ที่คำนวณได้ การป้องกันข้อผิดพลาดตรรกะ เราต้อง desk checking อัลกอริทึม และโปรแกรมอย่างระมัดระวังก่อนพิมพ์ (type) โปรแกรม

เนื่องจากการแก้ไขจุดบกพร่องทำให้เสียเวลา การวางแผนแก้ปัญหาโปรแกรมอย่างรอบคอบและ desk checking เพื่อขจัดจุดบกพร่อง (bug) จึงควรทำแต่เนิ่นๆ ถ้าเราไม่มั่นใจในวากยสัมพันธ์ของข้อความสั่งใดก็ตามให้เปิดดูตำรา แล้วทำตามวิธีนี้จะประหยัดเวลาและหลีกเลี่ยงปัญหา

ข้อสรุปของตัวสร้าง Pascal (Summary of New Pascal Constructs)

Construct	Effect
Program Heading program Payroll;	ไอเดนติไฟเออร์ Payroll เป็นชื่อโปรแกรม
Constant Declaration Const Tax = 25.00; Star * '*';	ตัวคงตัว Tax มีค่าเป็นเลขจำนวนจริง 25.00 ตัวคงตัว star เป็นข้อมูลชนิด char มีค่าเป็น '*'

Variable Declaration var X, Y, z : Real, Me, It : Integer	จัดสรร cell หน่วยความจำ ชื่อ X, Y และ Z สำหรับเก็บเลขจำนวนจริง และจัดสรร cell หน่วยความจำ ชื่อ Me และ It สำหรับเก็บจำนวนเต็ม
Assignment Statement Distance := Speed * Time	กำหนดผลคูณของ Speed และ Time ให้เป็นค่าของ Distance
ReadLn Procedure ReadLn (Hours, Rate)	ใส่ข้อมูลไว้ในตัวแปร Hours และ Rate
Write Procedure Write (' Net = ', Net : 4 : 2)	แสดงผลสายอักขระ 'Net =' ตามด้วยค่าของ Net ในเขตข้อมูลความกว้าง 4 สดมภ์ ปัดเศษให้เป็นจุดทศนิยมสองตำแหน่ง
Write Procedure WriteLn (X, Y)	พิมพ์ค่าของ X และ Y จากนั้นเลื่อน cursor ไปยังบรรทัดใหม่

คำถามทบทวน (Review Question)

1. สารสนเทศชนิดใดควรกำหนดไว้ในคอมเมนต์ ซึ่งปรากฏที่ตอนต้นของโปรแกรม
2. จงตรวจสอบว่ามีตัวแปรใดบ้างซึ่งเขียนถูกต้องเชิงวากยสัมพันธ์

Income	Two fold
Itime	C3PO
const	Income#1
Tom's	item
Hour*Rate	MyProgram
ReadLn	program
var	Program
variable	Pi

3. การประกาศและข้อความสั่งข้างล่างนี้มีที่ใดบ้างเขียนไม่ถูกต้อง

```
Const
   MyPi = 3.14159;
var
   C, R : Real;
begin
   MyPi := C / (2 * R * R)
```

4. จงเขียนรายการและนิยามกฎสำหรับอันดับของการประเมินผลของนิพจน์คำนวณ

5. ถ้าขนาดเฉลี่ยของครอบครัวคือ 2.8 และค่านี้เก็บในตัวแปรชื่อ Family Size จงเขียนข้อความสั่ง Pascal เพื่อแสดงผลความจริงนี้ในวิธีที่อ่านได้ง่าย (ปล่อยให้ cursor อยู่บนบรรทัดเดิม)

6. นิพจน์ชุดใดบ้างซึ่งประเมินผลแล้วให้ค่าเหมือนกัน

   a) A + B * C

   b) (A + B) * C

   c) A + (B * C)

7. แบบชนิดข้อมูลมาตรฐานของ Turbo Pascal มี 5 ชนิด ได้แก่อะไรบ้าง

8. สมมติว่า A และ B เป็นตัวแปรชนิด Integer จงบอกชนิดของนิพจน์ต่อไปนี้

   a) A div B

   b) A mod B

   c) A * B

   d) A / B

9. จงอธิบายและบอกความแตกต่างของข้อผิดพลาดต่อไปนี้

   syntax errors, runtime errors และ logic errors

โครงการการเขียนโปรแกรม (Programming Projects)

1. จงเขียนโปรแกรมเปลี่ยนอุณหภูมิซึ่งมีหน่วยเป็นองศาฟาเรนไฮต์ ให้เป็นองศาเซลเซียส

```
Problem input
Fahrenheit : Integer          {temperature in degree Fahrenheit}
```

Problem output

Celsius : Real {temperature in degree Celsius}

Relevant formula

Celsius = (5.0/9.0) x (Fahrenheit = 32.0)

2. จงเขียนโปรแกรมอ่าน data item สองตัว พิมพ์ผลบวก ผลต่าง ผลคูณ และผลหารของเลขสองตัวนี้

Problem inputs

X, Y : Integer {two items}

Problem outputs

Sum : Integer {sum of X and Y}

Difference : Integer {difference of X and Y}

Product : Integer {product of X and Y}

Quotient : Real {quotient of X divided by Y}

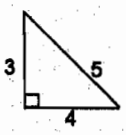
3. จงเขียนโปรแกรมอ่านน้ำหนักของวัตถุหนึ่งชิ้น ซึ่งมีหน่วยเป็น pounds จากนั้นคำนวณและพิมพ์น้ำหนักของวัตถุชิ้นนี้ ให้มีหน่วยเป็น kilograms และ gram,

(สูตร 1 pound เท่ากับ 0.453592 kilogram 453.59237 grams)

4. จงเขียนโปรแกรมอ่านความยาวและความกว้างของสนามหญ้ารูปสี่เหลี่ยมผืนผ้า และความยาว ความกว้าง ของบ้านรูปสี่เหลี่ยมผืนผ้าตั้งอยู่ในสนามหญ้า แล้วคำนวณเวลามีหน่วยเป็นนาที ในการตัดหญ้าที่อัตรา 2.3 ตารางเมตรต่อนาที

5. ทฤษฎีบทพีทาโกรัส (Pythagorean Theorem) กล่าวว่า ผลบวกกำลังสองของด้านสองด้านของสามเหลี่ยมมุมฉากจะเท่ากับกำลังสองของด้านตรงข้ามมุมฉาก ตัวอย่าง เช่น ด้านสองด้านของสามเหลี่ยมมุมฉากรูปหนึ่งยาวเท่ากัน 3 และ 4 เพราะฉะนั้นด้านตรงข้ามมุมฉากต้องยาวเท่ากับ 5

$3^2 + 4^2 = 9 + 16 = 25 = 5^2$



เลขจำนวนเต็ม 3, 4 และ 5 รวมกันเป็นสามพิทาโกเรียน (Phythagorean triple) ซึ่งจำนวน triple เช่นนี้ มีมากจนนับไม่ถ้วน กำหนดเลขจำนวนเต็มบวกสองตัว m และ n เมื่อ m > n สามพิทาโกเรียน สามารถสร้างขึ้นได้โดยใช้สูตรต่อไปนี้

side 1 = $m^2 - n^2$

side 2 = 2mn

hypotenuse = $m^2 + n^2$

จงเขียนโปรแกรม อ่านค่าของ m และ n แล้วพิมพ์ค่าของสามพิทาโกเรียน ซึ่งสร้างขึ้นโดยใช้สูตรข้างต้น