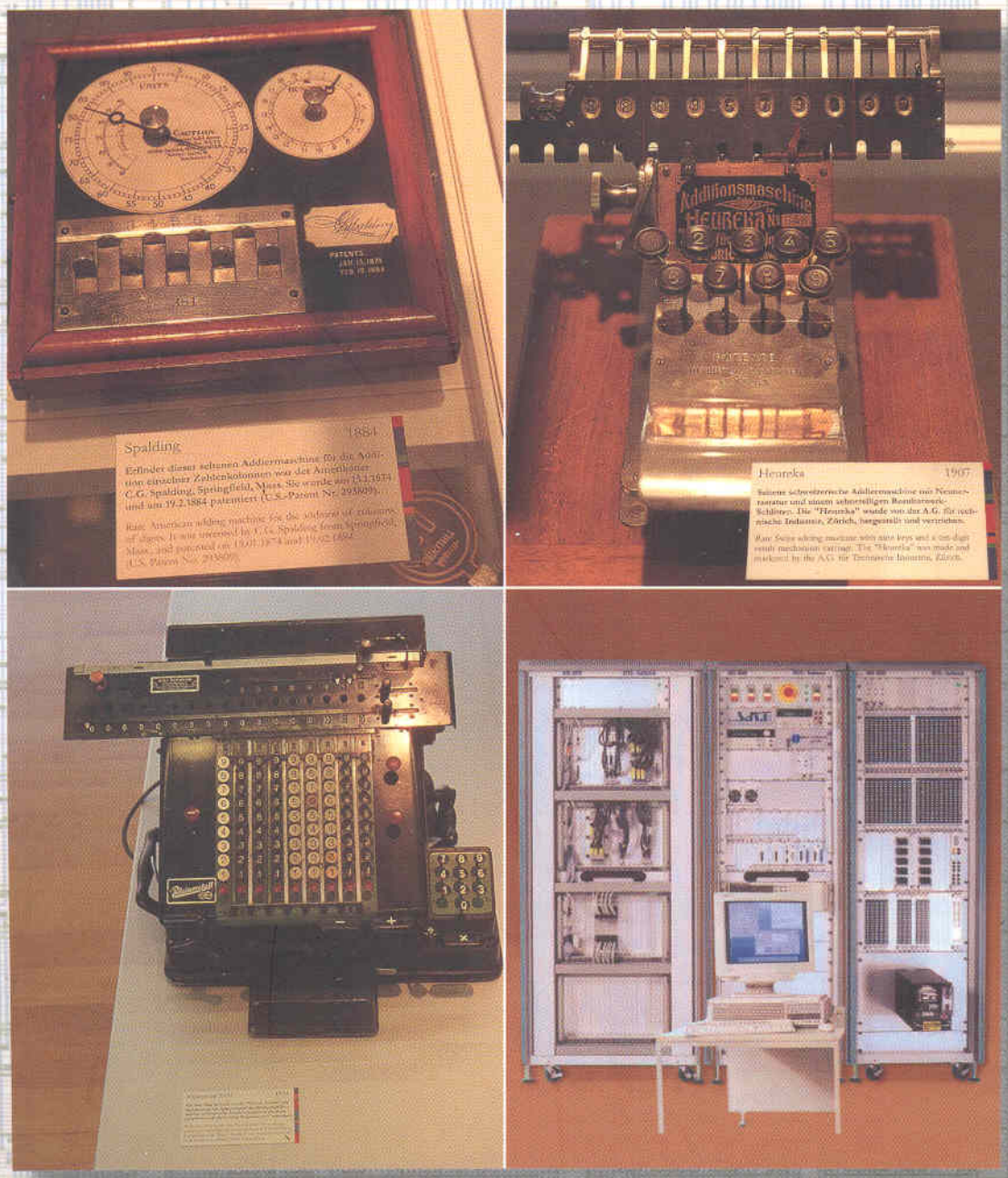
คอมพิวเตอร์

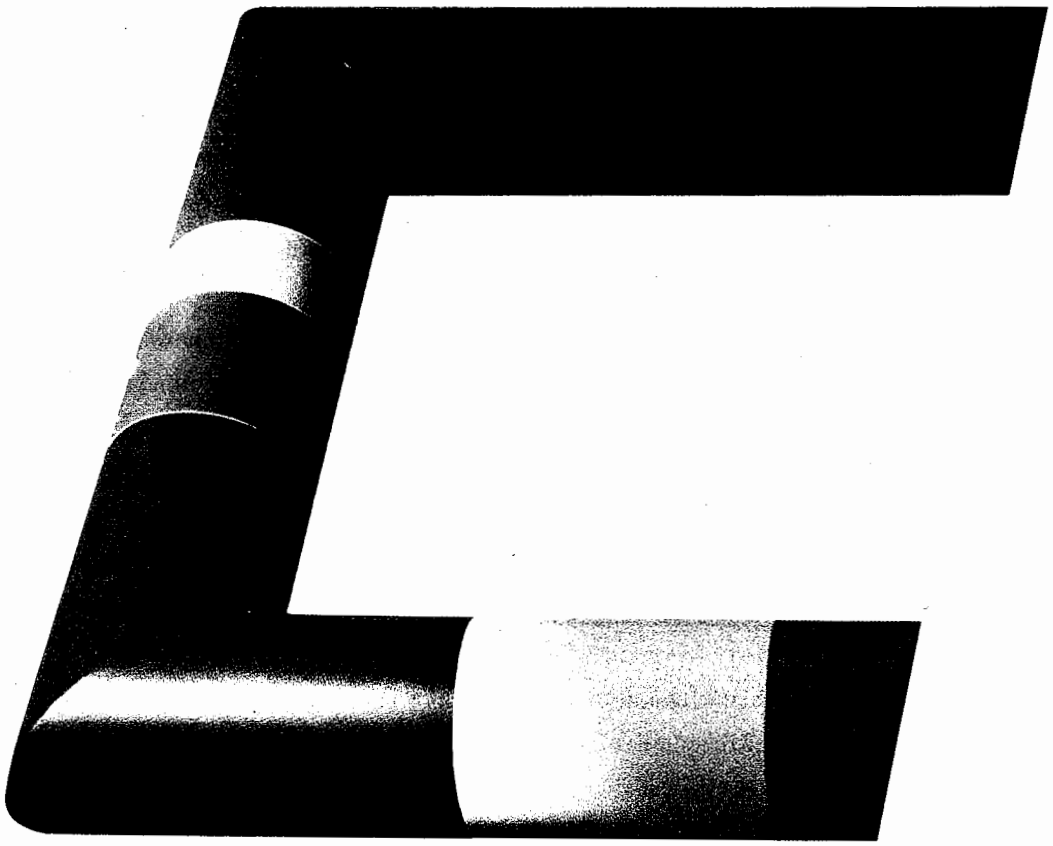
การเขียนโปรแกรมคอมพิวเตอร์ ภาษาซี



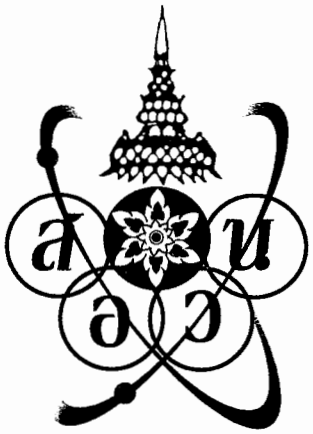
โครงการตำราวิทยาศาสตร์และคณิตศาสตร์มูลนิธิ สอวน.

คอมพิวเตอร์

(การเขียนโปรแกรมคอมพิวเตอร์ ภาษาซี)



โครงการตำราวิทยาศาสตร์และคณิตศาสตร์มูลนิธิ สอวน.



มูลนิธิส่งเสริมโอลิมปิกวิชาการและพัฒนามาตรฐานวิทยาศาสตร์ศึกษา ในพระอุปถัมภ์สมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ (สอวน.)

ความเป็นมา

ประเทศไทยส่งนักเรียนเข้าแข่งขันคณิตศาสตร์โอลิมปิกระหว่างประเทศครั้งแรกในปี พ.ศ. 2532 ที่ประเทศเยอรมนี โดยความร่วมมือระหว่างสมาคมวิทยาศาสตร์แห่งประเทศไทยในพระบรมราชูปถัมภ์ และสมาคมคณิตศาสตร์แห่งประเทศไทย ในพระบรมราชูปถัมภ์ สมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา (พระยศในขณะนั้น) ได้พระราชทานเงินส่วนพระองค์ จำนวนหนึ่งเพื่อเป็นค่าใช้จ่าย

การคัดเลือกนักเรียนเพื่อไปแข่งขันคณิตศาสตร์โอลิมปิกระหว่างประเทศ ครั้งที่ 30 สมาคมคณิตศาสตร์แห่งประเทศไทยฯ ได้จัดคณาจารย์มหาวิทยาลัยจำนวนหนึ่งมาช่วยฝึกอบรมรวมเป็นเวลาประมาณสองเดือน เพื่อให้นักเรียนได้เรียนรู้เนื้อหาเพิ่มเติมครอบคลุมหลักสูตรที่จะใช้แข่งขัน ซึ่งอยู่ในระดับชั้นปีที่ 1-2 ของมหาวิทยาลัย จากผลสำเร็จในปีแรก ทำให้รัฐบาลเห็นความสำคัญจึงได้จัดสรรงบประมาณให้กับโครงการนี้ผ่านสถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี (สสวท.) ตั้งแต่ปี พ.ศ. 2533 เป็นต้นมา สมทบกับเงินพระราชทานเป็นรายปีจากสมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ เพื่อสนับสนุนการส่งนักเรียนไทยไปแข่งขันโอลิมปิกวิชาการระหว่างประเทศ 5 สาขา จนถึงปัจจุบัน รวม 14 ปี นักเรียนไทยได้ทำชื่อเสียงได้เหรียญทองรวม 17 เหรียญ เหรียญเงิน 53 เหรียญ เหรียญทองแดง 93 เหรียญ และเกียรติคุณประกาศอีก 35 ราย รวมทั้งสิ้น 198 รางวัล จากจำนวนนักเรียนที่ส่งไปเข้าแข่งขัน 276 คน (72%)

อย่างไรก็ตาม จากการที่ประเทศไทยดำเนินการจัดส่งนักเรียนเข้าร่วมการแข่งขันคณิตศาสตร์ วิทยาศาสตร์ โอลิมปิกระหว่างประเทศ ตั้งแต่ พ.ศ.2532 จนถึงปัจจุบัน ทำให้เห็นว่ามาตรฐานการศึกษาด้านคณิตศาสตร์และวิทยาศาสตร์ของไทยยังต่ำกว่ามาตรฐานสากล การเตรียมตัวของนักเรียนยังใช้เวลาน้อยเกินไป

เพื่อให้การส่งเสริมและสนับสนุนโครงการจัดส่งผู้แทนประเทศไทยไปแข่งขันโอลิมปิกวิชาการระหว่างประเทศมีประสิทธิภาพยิ่งขึ้นสมเด็จพระเจ้า พี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ จึงมีพระดำริให้จัดตั้งมูลนิธิ สอวน. และได้รับอนุมัติจากกระทรวงมหาดไทยเมื่อวันที่ 12 ตุลาคม 2542 โดยมีวัตถุประสงค์หลัก 2 ประการ

วัตถุประสงค์

1. ส่งเสริมให้นักเรียนในระดับมัธยมศึกษาตอนปลายทั่วประเทศที่มีความสามารถทางวิทยาศาสตร์และคณิตศาสตร์ มีโอกาสได้รับการพัฒนาศักยภาพทางด้านคณิตศาสตร์ คอมพิวเตอร์ เคมี ฟิสิกส์ และชีววิทยา ตามความถนัดทั้งด้านทฤษฎีและทักษะด้านปฏิบัติ ให้สามารถคิดวิเคราะห์และแก้ปัญหาที่ซับซ้อนได้ โดยจัดให้มีศูนย์อบรมทั่วประเทศเป็นการเพิ่มเวลาฝึกอบรม เพื่อช่วยให้นักเรียนมีความพร้อมที่จะเข้ารับการคัดเลือกไปแข่งขันโอลิมปิกวิชาการระหว่างประเทศให้ได้ผลดียิ่งขึ้น ตามพระดำริของสมเด็จพระเจ้าพี่นางเธอเจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ องค์ประธานมูลนิธิ สอวน.
2. เพื่อนำประสบการณ์ที่ได้จากการแข่งขันโอลิมปิกวิชาการระหว่างประเทศมาพัฒนามาตรฐานคณิตศาสตร์และวิทยาศาสตร์ศึกษาของไทยให้สูงขึ้นเทียบเท่าระดับสากล

โครงการตำราวิทยาศาสตร์และคณิตศาสตร์มูลนิธิ สอวน.

มูลนิธิส่งเสริมโอลิมปิกวิชาการและพัฒนามาตรฐานวิทยาศาสตร์ศึกษา ในพระอุปถัมภ์สมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ (สอวน.) ได้ร่วมมือกับคณาจารย์มหาวิทยาลัยของรัฐ 20 แห่ง และกระทรวงศึกษาธิการ ดำเนินการจัดตั้งศูนย์ สอวน. ในภูมิภาค 12 ศูนย์ และในกรุงเทพฯ 1 ศูนย์ (5 โรงเรียน) เพื่อพัฒนาศักยภาพทางปัญญาของนักเรียนที่มีความพร้อมในด้านวิทยาศาสตร์ คณิตศาสตร์และคอมพิวเตอร์จากทั่วประเทศ เพื่อให้นักเรียนได้มีความรู้ความสามารถเทียบเท่ามาตรฐานสากล และพร้อมที่จะสอบคัดเลือกเป็นผู้แทนประเทศไทยไปร่วมการแข่งขันโอลิมปิกวิชาการระหว่างประเทศในสาขาวิชาต่างๆ ได้ และเพื่อขยายผลจากประสบการณ์ที่ได้เข้าแข่งขันโอลิมปิกวิชาการระหว่างประเทศมาพัฒนามาตรฐานวิทยาศาสตร์และคณิตศาสตร์ของไทยให้สูงขึ้นเทียบเท่าสากล

ในการอบรมนักเรียนของศูนย์ สอวน. มูลนิธิ สอวน.ได้พิจารณาเห็นว่าตำราที่มีความสมบูรณ์ของเนื้อหาตามหลักสูตรของ สอวน. จะช่วยพัฒนาศักยภาพของนักเรียนในศูนย์สอวน.ให้สูงยิ่งขึ้นได้ นอกจากนั้นตำราเรียนในวิชาวิทยาศาสตร์และคณิตศาสตร์ในระดับมัธยมศึกษาตอนปลายที่มีความสมบูรณ์ถูกต้องยังขาดแคลน นักเรียนและครูสามารถนำตำราที่พัฒนาขึ้นมานี้ไปใช้เป็นหนังสืออ้างอิงประกอบการเรียนการสอนได้อีกด้วย มูลนิธิ สอวน. จึงได้มี "โครงการจัดทำตำราส่งเสริมพัฒนาศักยภาพของนักเรียนด้านวิทยาศาสตร์ คณิตศาสตร์และคอมพิวเตอร์" ขึ้น เพื่อผลิตตำราคณิตศาสตร์ คอมพิวเตอร์ เคมี ชีววิทยาและฟิสิกส์ที่มีเนื้อหาหลักสูตรตามมาตรฐานสากล โดยมีวัตถุประสงค์

1. เพื่อเป็นที่ระลึกในมหามงคลสมัยที่สมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ ทรงเจริญพระชนมายุ 80 พรรษา ซึ่งพระองค์ทรงมีพระมหากรุณาธิคุณต่อโครงการจัดส่งผู้แทนประเทศไทยไปแข่งขันโอลิมปิกวิชาการระหว่างประเทศและทรงสนับสนุนการดำเนินงานเพื่อพัฒนามาตรฐานการศึกษาวิทยาศาสตร์และคณิตศาสตร์ของไทยอย่างหาที่สุดมิได้

2. เพื่อผลิตหนังสือคณิตศาสตร์ คอมพิวเตอร์ เคมี ชีววิทยาและฟิสิกส์ระดับมัธยมศึกษาตอนปลายถึงระดับชั้นปีที่ 1 ของคณะวิทยาศาสตร์ในมหาวิทยาลัยให้มีคุณภาพเทียบเท่าสากล เรียบเรียงโดยคณาจารย์จากมหาวิทยาลัยของรัฐที่มีประสบการณ์สูงและที่มีส่วนรวมในการฝึกอบรมนักเรียนในค่าย สอวน. ค่าย สสวท. และควบคุมนักเรียนไปแข่งขันโอลิมปิกวิชาการระหว่างประเทศ เนื้อหาในหนังสือเน้นกระบวนการคิดแบบวิทยาศาสตร์ เพื่อฝึกฝนให้นักเรียนสามารถคิดวิเคราะห์และแก้ปัญหาที่ซับซ้อนทั้งในเชิงทฤษฎีและประยุกต์ได้โดยจัดพิมพ์บนกระดาษอาร์ตอย่างดีและมีภาพสีประกอบคำบรรยาย

โครงการตำราวิทยาศาสตร์และคณิตศาสตร์ของมูลนิธิ สอวน. มีเป้าหมายในการผลิตตำราคณิตศาสตร์ รวม 5 เล่ม คอมพิวเตอร์ 3 เล่ม เคมี 3 เล่ม ชีววิทยา 5 เล่ม และฟิสิกส์ 3 เล่ม โดยหนังสือชุดแรกจะนำขึ้นทูลเกล้าถวายในวันที่ 6 มิถุนายน 2547 ซึ่งเป็นวันประชุมสามัญประจำปี พ.ศ. 2547 ของคณะกรรมการบริหาร และสมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์ เสด็จเป็นองค์ประธานที่ประชุม ส่วนที่เหลือคาดว่าจะจัดพิมพ์ให้แล้วเสร็จภายในปี พ.ศ. 2547 นี้ นอกจากนั้นคณะกรรมการโครงการผลิตตำราวิทยาศาสตร์และคณิตศาสตร์มูลนิธิ สอวน. มีเป้าหมายจะผลิตตำราที่มีคุณภาพสูงนี้ในระดับมัธยมศึกษาต้นโดยคณาจารย์จากมหาวิทยาลัย เพื่อวางรากฐานวิทยาศาสตร์และคณิตศาสตร์พื้นฐานและเพื่อสร้างแรงบันดาลใจและความสนใจทางด้านวิทยาศาสตร์ คณิตศาสตร์แก่เยาวชนอีกด้วย หากได้รับความอนุเคราะห์ช่วยเหลือด้านงบประมาณจากภาครัฐและเอกชน

คณะจัดทำหนังสือคณิตศาสตร์ วิทยาศาสตร์หวังว่าโครงการตำราคณิตศาสตร์ วิทยาศาสตร์ของมูลนิธิ สอวน. จะมีส่วนร่วมในการปฏิรูปการเรียนรู้และยกระดับมาตรฐานวิทยาศาสตร์และคณิตศาสตร์ของไทยให้เทียบเท่ากับระดับสากลเพื่อสนองพระดำริของสมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์

****

*คำขอบคุณ : คณะกรรมการดำเนินงานโครงการตำราฯ ขอขอบคุณพิพิธภัณฑ์ ARITHMEUM, Bonn ที่อนุญาตให้ประธานคณะกรรมการฯ เข้าถ่ายภาพในพิพิธภัณฑ์ เพื่อนำมาเผยแพร่

คณะกรรมการโครงการตำราวิทยาศาสตร์และคณิตศาสตร์มูลนิธิ สอวน.

คณะกรรมการที่ปรึกษา

1. ศาสตราจารย์ นายแพทย์ จรัส สุวรรณเวลา	รองประธานมูลนิธิ สอวน.
2. รองศาสตราจารย์ ดร.กำจัด มงคลกุล	กรรมการมูลนิธิฯ
3. ดร. คุณหญิงกษมา วรวรรณ ณ อยุธยา	กรรมการมูลนิธิฯ
4. รองศาสตราจารย์ ดร. คุณหญิงสุมณฑา พรหมบุญ	กรรมการมูลนิธิฯ
5. นายเชาว์ อรรถมานะ (อดีต)	กรรมการมูลนิธิฯ
6. นายสมนึก พิมลเสถียร	กรรมการมูลนิธิฯ
7. นายสุนทร อรุณานนท์ชัย	กรรมการมูลนิธิฯ
8. รองศาสตราจารย์บุญรักษา สุนทรธรรม	กรรมการมูลนิธิฯ

คณะกรรมการดำเนินงานโครงการตำราฯ

1. ศาสตราจารย์ศักดา ศิริพันธุ์ (ราชบัณฑิต) เลขาธิการมูลนิธิ สอวน.	ประธาน
2. รองศาสตราจารย์เย็นใจ สมวิเชียร	รองประธาน
3. รองศาสตราจารย์ ดร. ณรงค์ ปั้นนิ่ม	กรรมการ (วิชาคณิตศาสตร์)
4. รองศาสตราจารย์ยืน ภู่วรวรรณ	กรรมการ (วิชาคอมพิวเตอร์)
5. รองศาสตราจารย์ ดร. พินิติ รตะนานุกูล	กรรมการ (วิชาเคมี)
6. ผู้ช่วยศาสตราจารย์ ดร. วุทธิพันธุ์ ปรัชญพฤทธิ์	กรรมการ (วิชาฟิสิกส์)
7. ศาสตราจารย์อักษร ศรีเปล่ง	กรรมการ (วิชาชีววิทยา ประเภทพืช)
8. รองศาสตราจารย์ ดร. อุษณีย์ ยศยิ่งยวด	กรรมการ (วิชาชีววิทยา ประเภทสัตว์)

คณะผู้เขียนวิชาคอมพิวเตอร์

1. รองศาสตราจารย์ยืน ภู่วรวรรณ	ประธานคณะอนุกรรมการ
2. รองศาสตราจารย์ ดร. อนงค์นาฏ ศรีวิหค	อนุกรรมการ
3. ผู้ช่วยศาสตราจารย์อุมาพร ศิรธรานนท์	อนุกรรมการ
4. ผู้ช่วยศาสตราจารย์กัลยาณี บรรจงจิตร	อนุกรรมการ
5. ผู้ช่วยศาสตราจารย์นงนุช สุขวารี	อนุกรรมการ
6. ผู้ช่วยศาสตราจารย์กรรณิกา คงสาคร	อนุกรรมการ
7. อาจารย์ศิริกร จันทร์นวล	อนุกรรมการ
8. อาจารย์พบสิทธิ์ กมลเวช	อนุกรรมการ
9. ดร. นวลวรรณ สุนทรภิษัช	อนุกรรมการ
10. ดร. สุขุมาล กิติสิน	อนุกรรมการ
11. อาจารย์พัชรี เลิศจิตรศิลป์	อนุกรรมการ
12. อาจารย์มาริสา มัยยะ	อนุกรรมการ

คำนำ

การเขียนโปรแกรมคอมพิวเตอร์ "ภาษาซี" เล่มนี้เหมาะสมสำหรับพื้นฐานให้ผู้สนใจการเขียนโปรแกรมแบบโครงสร้าง ทั้งผู้ที่เริ่มต้นศึกษาการเขียนโปรแกรมคอมพิวเตอร์และผู้ที่เคยศึกษาการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซีอื่นมาแล้ว โดยเนื้อหาประกอบด้วยทฤษฎี คำอธิบายและตัวอย่างโปรแกรม เริ่มต้นจากความรู้พื้นฐาน กฎเกณฑ์ การเขียนข้อความสั่งควบคุม การใช้แถวลำดับ ตัวชี้ รวมทั้งการประมวลผลแฟ้ม โดยได้พยายามลำดับเนื้อหาทีละขั้นตอน เพื่อให้ผู้อ่านสามารถติดตาม เรียนรู้และเข้าใจง่าย คณะผู้เขียนหวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้จะเป็นแนวทางในการศึกษาการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซีและเป็นประโยชน์ต่อผู้อ่าน

ที่ปรึกษา

รองศาสตราจารย์ยืน ภู่วรวรรณ

วศ.บ. (วิศวกรรมไฟฟ้าสื่อสาร) เกียรตินิยม, วศ. ม. (วิศวกรรมไฟฟ้า) จุฬาลงกรณ์มหาวิทยาลัย

M.Eng (Industrial Engineering and Management) at Asian Institute of Technology

ผู้เขียน

ผู้ช่วยศาสตราจารย์อุมาพร ศิรธรานนท์

M.S. (Mathematics and Computer Science) University of Illinois at Chicago

ผู้ช่วยศาสตราจารย์กัลยาณี บรรจงจิตร

พบ.ม. (สถิติประยุกต์) สถาบันบัณฑิตพัฒนบริหารศาสตร์

ดร. นวลวรรณ สุนทรภิษัช

วศ.ด. (วิศวกรรมคอมพิวเตอร์) จุฬาลงกรณ์มหาวิทยาลัย

ISBN 974-92235-2-7

จัดพิมพ์โดย มูลนิธิ สอวน.

พิมพ์ครั้งที่ 2 พ.ศ. 2549 (2006)

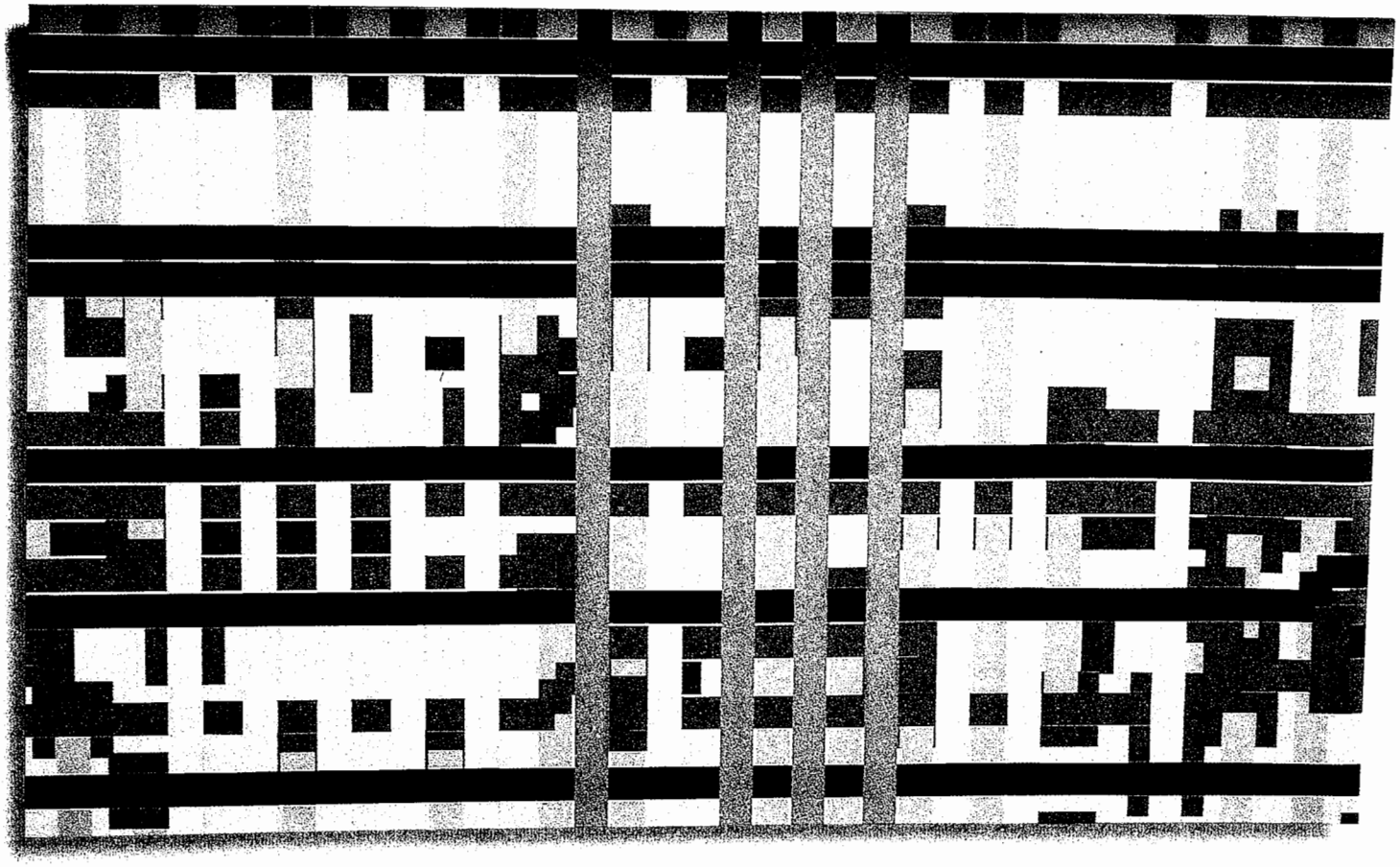
ออกแบบปก หน้ารองปกและหน้านำบทที่ 1 โดย ศาสตราจารย์ศักดา ศิริพันธุ์

ศิลปกรรม: นายปรีชา ฉัตระเนตร

พิมพ์ที่ บริษัทด่านสุทธาการพิมพ์ จำกัด

แยกสี / เพลท เอ็นอาร์ ฟิล์ม

สารบัญ



บทที่ 1
ความรู้พื้นฐานโปรแกรมภาษาซี

โปรแกรมภาษาซีเป็นโปรแกรมที่มีความยืดหยุ่นและมีขีดความสามารถสูง โปรแกรมมีขนาดเล็กทำงานได้เร็ว ลักษณะของภาษาจะอยู่ในรูปแบบของฟังก์ชัน โปรแกรมหนึ่งอาจประกอบด้วยฟังก์ชันเดียวหรือหลายฟังก์ชัน เมื่อเขียนโปรแกรมใหม่ก็อาจนำเอาฟังก์ชันจากอีกโปรแกรมหนึ่งมาใช้ได้ ถ้าโปรแกรมทั้งสองมีการทำงานบางส่วนเหมือนกัน โปรแกรมที่ใช้ในคอมพิวเตอร์ระบบหนึ่ง ยังสามารถนำไปใช้กับคอมพิวเตอร์อีกระบบหนึ่งได้ โดยอาจมีการแก้ไขเพียงเล็กน้อย ด้วยเหตุนี้โปรแกรมภาษาซีจึงได้รับความนิยมจากนักเขียนโปรแกรมเป็นอย่างมาก

1.1 โครงสร้างของโปรแกรมภาษาซี

โปรแกรมภาษาต่างๆ จะมีรูปแบบหรือโครงสร้างเฉพาะที่แตกต่างกันไป สำหรับโปรแกรมภาษาซี มีโครงสร้างและลำดับการเขียนดังนี้

- ข้อความสั่งตัวประมวลผลก่อน (preprocessor statements)
- รหัสต้นฉบับ (source code) มีลำดับการเขียนดังนี้
    - ข้อความสั่งประกาศครอบคลุม (global declaration statements)
    - ต้นแบบฟังก์ชัน (function prototypes)
    - ฟังก์ชันหลัก (main function) มีเพียงฟังก์ชันเดียว
    - ฟังก์ชัน (functions) มีได้หลายฟังก์ชัน
    - ข้อความสั่งประกาศตัวแปรเฉพาะที่ (local declaration statements)
- หมายเหตุ (comment) สามารถแทรกไว้ที่ใดก็ได้ ภายในโปรแกรม

ข้อความสั่งตัวประมวลผลก่อน (preprocessor statements)

ข้อความสั่งตัวประมวลผลก่อนขึ้นต้นด้วยเครื่องหมาย # เช่น

```
#include <stdio.h>
```

หมายความว่า ให้ตัวประมวลผลก่อนไปอ่านข้อมูลจากแฟ้ม stdio.h ซึ่งเป็นแฟ้มที่มีอยู่ในคลัง
เมื่อโปรแกรมมีการใช้ข้อความสั่งอ่านและบันทึก จะต้องใช้ข้อมูลจากแฟ้ม stdio.h
ข้อความสั่งตัวประมวลผลก่อนจะต้องเขียนไว้ตอนต้นของโปรแกรม

รหัสต้นฉบับ (source code)

รหัสต้นฉบับ หมายถึง ตัวโปรแกรมที่ประกอบด้วยข้อความสั่งและตัวฟังก์ชันต่าง ๆ

ข้อความสั่งประกาศครอบคลุม (global declaration statements)

ข้อความสั่งประกาศครอบคลุมใช้ประกาศตัวแปรส่วนกลาง โดยที่ตัวแปรส่วนกลางนี้จะสามารถถูกเรียกใช้ จากทุกส่วนของโปรแกรม

ต้นแบบฟังก์ชัน (function prototypes)

ต้นแบบฟังก์ชันใช้ประกาศฟังก์ชัน เพื่อบอกให้ตัวแปลโปรแกรมทราบถึง ชนิดของค่าที่ส่งกลับ และ ชนิดของค่าต่างๆ ที่ส่งไปกระทำการในฟังก์ชัน

ฟังก์ชันหลัก (main function **หรือ** function main())

เมื่อสั่งให้กระทำการโปรแกรม ฟังก์ชันหลักจะเป็นจุดเริ่มต้นของการกระทำการ ภายในฟังก์ชันหลักจะประกอบด้วยข้อความสั่งและข้อความสั่งที่เรียกใช้ฟังก์ชัน เมื่อมีการทำงานตามข้อความสั่งและฟังก์ชันต่างๆ แล้ว จะมีการส่งค่าและกลับมาทำงานที่ฟังก์ชันหลักจนจบฟังก์ชัน

ฟังก์ชัน (functions)

ฟังก์ชัน หมายถึง กลุ่มของข้อความสั่งที่ทำงานใดงานหนึ่งโดยเป็นอิสระจากฟังก์ชันหลัก แต่อาจมีการรับส่งค่าระหว่างฟังก์ชันและฟังก์ชันหลัก การเขียนฟังก์ชัน ขึ้นต้นด้วย ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน วงเล็บ และตามด้วยเครื่องหมายปีกกา ภายในเครื่องหมายปีกกาประกอบด้วยข้อความสั่งภาษาซี

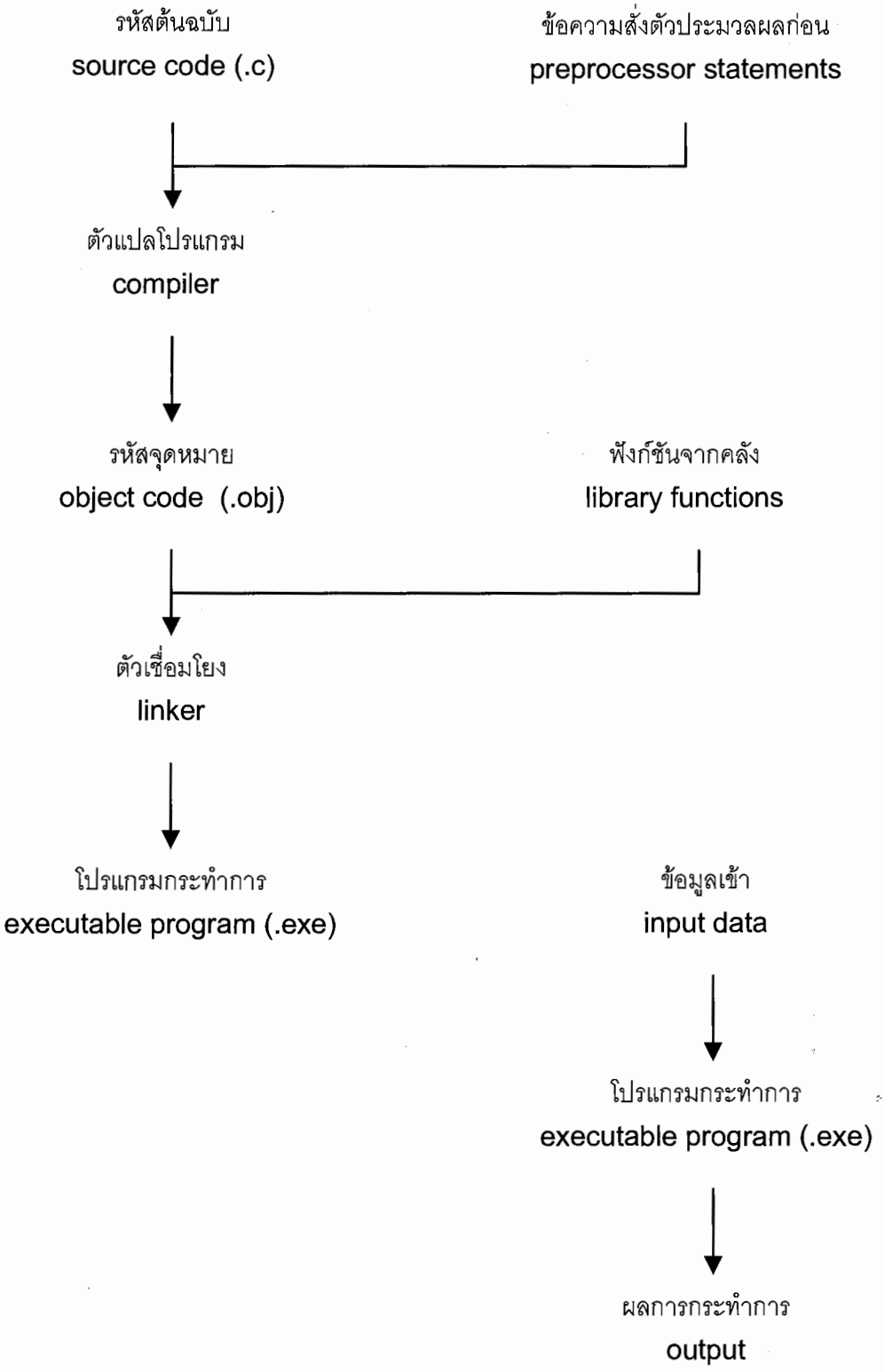
ข้อความสั่งประกาศตัวแปรเฉพาะที่ (local declaration statements)

ข้อความสั่งประกาศตัวแปรเฉพาะที่ ใช้ประกาศตัวแปรเฉพาะที่ โดยที่ตัวแปรเฉพาะที่จะสามารถถูกเรียกใช้เฉพาะภายในฟังก์ชันนั้น

การแปลและกระทำการโปรแกรม (program compilation and execution)

เมื่อได้เขียนและป้อนข้อความสั่งตัวประมวลผลก่อน (preprocessor statements) และรหัสต้นฉบับ (source code) ลงในโปรแกรมบรรณาธิกรณ์ (editor program) เสร็จแล้ว จะต้องเรียกตัวแปลโปรแกรม (compiler) มาเพื่อให้แปลภาษาซีให้เป็นภาษาเครื่อง (machine language) หากโปรแกรมเขียนได้ถูกต้องตรงตามกฎของภาษาซี ตัวแปลโปรแกรมแปลจะแปลโปรแกรมภาษาซีให้เป็นภาษาเครื่อง แล้วนำไปเก็บไว้ในแฟ้มชื่อเดียวกันแต่ส่วนขยายเป็น โอบีเจ (ชื่อแฟ้ม.obj) จากนั้นตัวเชื่อมโยง (linker) จะต้องนำฟังก์ชันจากคลัง (library function) ต่างๆที่โปรแกรมได้เรียกใช้ มารวมเข้ากับแฟ้มจุดโอบีเจ แล้วนำไปเก็บไว้ในแฟ้มชื่อเดิม แต่ส่วนขยายเป็น อีเอ็กซ์อี (ชื่อแฟ้ม.exe) ซึ่งแฟ้มนี้จะเป็นแฟ้มที่พร้อมสำหรับกระทำการ (execute) เมื่อต้องการกระทำการโปรแกรมก็สามารถป้อนข้อมูลเข้า (input data) ให้กับโปรแกรม ซึ่งจะได้ผลการกระทำการ (output)

ฟังก์ชันจากคลัง เป็นฟังก์ชันสำเร็จรูปที่มีอยู่แล้วในตัวแปลโปรแกรม ซึ่งผู้เขียนโปรแกรมสามารถเรียกใช้ได้ด้วยการเขียนชื่อฟังก์ชันไว้ในโปรแกรม



รูปที่ 1.1 การแปลและกระทำการโปรแกรม

ตัวอย่าง 1.1

แสดงโครงสร้างของโปรแกรม

โปรแกรมประกอบด้วยฟังก์ชัน main() และฟังก์ชัน sum ()

ฟังก์ชัน main() ทำหน้าที่รับค่ามาเก็บไว้ในตัวแปร a และตัวแปร b แล้วส่งค่าของตัวแปรทั้งสองไปยังฟังก์ชัน sum() เพื่อคำนวณหาผลรวม เมื่อคำนวณผลรวมแล้ว จะส่งค่าของผลรวมกลับไปยังฟังก์ชัน main() จากนั้นฟังก์ชัน main() จะแสดงค่าของผลรวม

```
#include <stdio.h>                          // คำสั่งตัวประมวลผลก่อน
int a, b, c;                                // คำสั่งประกาศครอบคลุม
int sum(int x, int y);                      // ต้นแบบฟังก์ชัน
void main()                                 // ฟังก์ชัน main()
{                                           // เริ่มต้นฟังก์ชัน main()
    scanf("%d", &a);                        // คำสั่งในฟังก์ชัน main()
    scanf("%d", &b);                        // คำสั่งในฟังก์ชัน main()
    c = sum (a, b);                         // คำสั่งในฟังก์ชัน main()
    printf("\n%d + %d = %d", a, b, c);      // คำสั่งในฟังก์ชัน main()
}                                           // จบฟังก์ชัน main()
int sum(int x, int y)                       // ฟังก์ชัน sum()
{                                           // เริ่มต้นฟังก์ชัน sum()
    return (x + y);                         // คำสั่งในฟังก์ชัน sum()
}                                           // จบฟังก์ชัน sum()
```

ข้อความสั่งหมายเหตุ (comment statement)

ข้อความสั่งหมายเหตุ คือ ข้อความที่เขียนไว้ภายในโปรแกรม เพื่อใช้อธิบายโปรแกรม โดยตัวแปลโปรแกรมจะไม่แปลข้อความสั่งหมายเหตุให้เป็นภาษาเครื่อง

การเขียนข้อความสั่งหมายเหตุในโปรแกรมทำได้ 2 แบบ ได้แก่

1. // หมายเหตุ ใช้เครื่องหมาย // หน้าข้อความหมายเหตุ ใช้ได้กับหมายเหตุที่มีขนาดความยาวไม่เกิน 1 บรรทัด
2. /* หมายเหตุ */ เขียนหมายเหตุไว้ระหว่าง /* และ */ ใช้ได้กับหมายเหตุที่มีขนาดความยาวตั้งแต่ 1 บรรทัดขึ้นไป

1.2 ตัวแปร (variables)

คอมพิวเตอร์มีส่วนประกอบที่สำคัญส่วนหนึ่งคือ หน่วยความจำ หน่วยความจำเปรียบได้กับสมองของมนุษย์ทำหน้าที่เก็บข้อมูลในขณะที่ประมวลผล ในการประมวลผลแต่ละครั้งมักต้องใช้ข้อมูลจำนวนมาก อาทิ ชื่อนิสิต คะแนนสอบวิชาคณิตศาสตร์ คะแนนสอบวิชาการเขียนโปรแกรมคอมพิวเตอร์ จำนวนหน่วยกิต คะแนนเฉลี่ย เป็นต้น ซึ่งจำเป็นจะต้องนำข้อมูลเหล่านี้ไปเก็บไว้ในหน่วยความจำ และเมื่อเก็บแล้ว จะต้องทราบตำแหน่งที่นำข้อมูลเข้าไปเก็บไว้ภายในของหน่วยความจำด้วย เพื่อให้สามารถนำข้อมูลเหล่านั้นกลับมาประมวลผลได้ ในภาษาคอมพิวเตอร์ระดับสูง เพื่อความสะดวกในการเขียนโปรแกรม การจดจำตำแหน่งที่ใช้ในการเก็บข้อมูล จะทำโดยการตั้งชื่อให้กับตำแหน่งของหน่วยความจำที่ใช้เก็บข้อมูลนั้น จากนั้นระบบปฏิบัติการจะช่วยจัดการในการหาตำแหน่งที่อยู่ที่แท้จริงของข้อมูลต่อไป

ตัวแปร เป็น ชื่อของหน่วยความจำที่ตำแหน่งใดๆ เมื่อนำข้อมูลไปเก็บไว้ในหน่วยความจำตำแหน่งนั้น จะกล่าวว่า ตัวแปรนั้นมีค่าเท่ากับข้อมูลที่เก็บไว้ ตัวแปรสามารถเก็บค่าชนิดต่างๆ ตามที่ได้ประกาศไว้ในโปรแกรมเท่านั้น เช่น

- ตัวแปรที่ใช้เก็บข้อมูลชนิดจำนวนเต็ม
- ตัวแปรที่ใช้เก็บข้อมูลชนิดจำนวนจริง
- ตัวแปรที่ใช้เก็บข้อมูลชนิดอักขระ
- ตัวแปรที่ใช้เก็บข้อมูลชนิดสายอักขระ

ตัวแปรเหล่านี้จะไม่สามารถเก็บค่าชนิดอื่นนอกเหนือจากชนิดที่ประกาศไว้ และค่าที่เก็บไว้ในตัวแปรนี้สามารถเปลี่ยนค่าได้ตลอดเวลา ขึ้นกับข้อความสั่งภายในโปรแกรม

กฎการตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรมีข้อกำหนดดังนี้

- ประกอบด้วย a ถึง z, 0 ถึง 9 และ _ เท่านั้น
- อักขระตัวแรกต้องเป็น a ถึง z และ _
- ห้ามใช้ชื่อเฉพาะ
- ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก มีความหมายที่แตกต่างกัน
- ยาวสูงสุด 31 ตัวอักษร

หากตั้งชื่อตัวแปรไม่ตรงตามข้อกำหนด ก็จะทำให้ตัวแปลโปรแกรมไม่เข้าใจ ไม่ทราบว่า ชื่อนั้นคืออะไร ซึ่งจะทำให้เกิดข้อผิดพลาดขึ้นในระหว่างการแปลโปรแกรม

ชนิดข้อมูล (data types)

ข้อมูลที่ใช้ในโปรแกรมมีหลายชนิด ซึ่งนักเขียนโปรแกรมต้องเลือกใช้ตามความเหมาะสมกับการใช้งาน ข้อมูลมีขนาดที่แตกต่างกันไปตามชนิดข้อมูล นอกจากนี้แล้ว ชนิดข้อมูลยังอาจมีขนาดที่แตกต่างกันโดยขึ้นกับเครื่องคอมพิวเตอร์และตัวแปลโปรแกรมที่ใช้ในการประมวลผล แต่โดยทั่วไปแล้วในไมโครคอมพิวเตอร์ ชนิดข้อมูลมีการใช้ในโปรแกรม และขนาดดังนี้

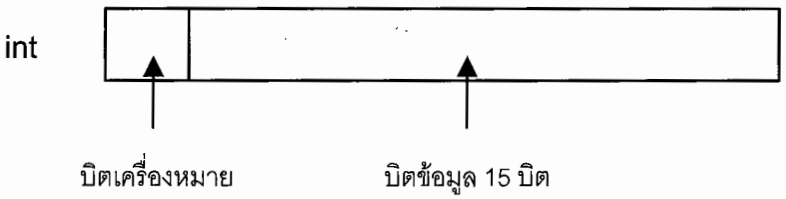
ชนิดข้อมูล	การใช้ในโปรแกรม	ขนาดข้อมูล (ไบต์)	ช่วงข้อมูล
character	char	1	-128 ถึง 127
integer	int	2	-32768 ถึง 32767
short integer	short	2	-32768 ถึง 32767
long integer	long	4	-2147483648 ถึง +2147483647
unsigned character	unsigned char	1	0 ถึง 255
unsigned integer	unsigned int	2	0 ถึง 65535
unsigned short integer	unsign short	2	0 ถึง 65535
unsigned long integer	unsign long	4	0 ถึง 4294967295
single-precision floating-point	float	4	1.2×10^{-38} ถึง 3.4×10^{38}
double-precision floating-point	double	8	2.2×10^{-308} ถึง 1.8×10^{308}

ตัวแปรชนิดตัวเลข (numeric variable types)

ตัวแปรชนิดตัวเลขแบ่งได้เป็น 2 กลุ่ม คือ

1. ตัวแปรจำนวนเต็ม (integer variables) หมายถึง ตัวแปรที่ใช้เก็บค่าที่เป็นจำนวนเต็ม ได้แก่ char, int, short, long, unsigned char, unsigned int, unsigned short และ unsigned long

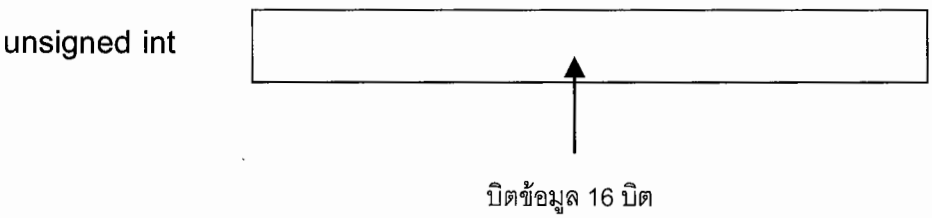
ตัวอย่าง ลักษณะการจัดเก็บตัวแปรจำนวนเต็ม ใช้เนื้อที่ 16 บิต ประกอบด้วย บิตเครื่องหมาย (sign bit) 1 บิต และบิตข้อมูล (data bits) 15 บิต



เช่น

- +15
- -2789
- 12
- -9

ตัวอย่าง ลักษณะการจัดเก็บตัวแปรจำนวนเต็มไม่มีเครื่องหมาย ใช้เนื้อที่เก็บข้อมูล 16 บิต โดยไม่มีการเก็บเครื่องหมาย



เช่น

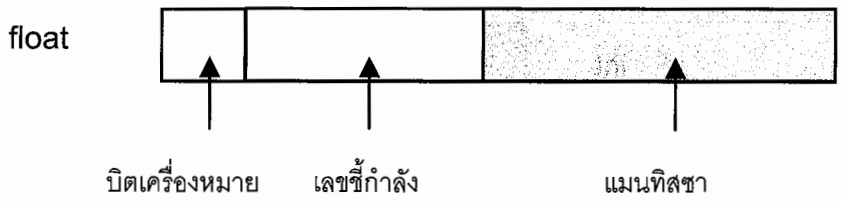
- 1578
- 2789
- 12
- 0

2. ตัวแปรจำนวนจริง (real variables) หรือ ตัวแปรจุดลอยตัว (floating-point variables) หมายถึง ตัวแปรที่ใช้เก็บค่าที่เป็นจำนวนจริง

ซึ่งเป็นตัวเลขที่เป็นเศษส่วน หรือมีจุดทศนิยม ได้แก่

- ชนิด float ใช้เนื้อที่เก็บข้อมูล 64 บิต

ตัวอย่าง ลักษณะการจัดเก็บตัวแปรจำนวนจริงชนิด float ใช้เนื้อที่เก็บข้อมูล 32 บิต ประกอบด้วย บิตเครื่องหมาย (sign bit) 1 บิต เลขชี้กำลัง (exponent) 8 บิต และแมนทิสซา (mantissa) 23 บิต



- ชนิด double ใช้เนื้อที่เก็บข้อมูล 64 บิต

ตัวแปรจำนวนจริง สามารถเขียนได้หลายแบบดังนี้

- แบบจุดตรึง (fixed point)

เช่น

- 35.058
- -400.75
- 49.
- 0.0
- +2.

- แบบจุดลอยตัว (floating point)

เช่น

- 1.234e6 (1.234×10^{6})
- 4.53e-8 (4.53×10^{-8})
- 1.8732e3 (1.8732×10^{3})
- 5.65421e-2 (5.6542×10^{-2})

การประกาศตัวแปร

การประกาศตัวแปรทำได้โดย เขียนข้อความสั่งขึ้นต้นด้วยชนิดข้อมูล ตามด้วยชื่อตัวแปร และจบข้อความสั่งประกาศตัวแปรด้วยเครื่องหมายอัฒภาค (;) ดังนี้

ชนิดข้อมูล ชื่อตัวแปร;

ถ้าต้องการประกาศตัวแปรชนิดเดียวกันหลายตัว ต้องคั่นระหว่างตัวแปรด้วยเครื่องหมายจุลภาค (,) และจบข้อความสั่งประกาศตัวแปรด้วยเครื่องหมายอัฒภาค (;) ดังนี้

ชนิดข้อมูล ชื่อตัวแปร1, ชื่อตัวแปร2, ... ;

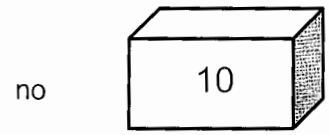
เช่น

```
int count;              // ประกาศตัวแปรชื่อ count ใช้เก็บข้อมูลชนิด integer
int m, n;               // ประกาศตัวแปรชื่อ m และ n ใช้เก็บข้อมูลชนิด integer
int no = 10;            /* ประกาศตัวแปรที่ใช้เก็บข้อมูลชนิด integer ชื่อ no และเก็บค่า 10
                           ไว้ในตัวแปรดังกล่าว */
long number;            // ประกาศตัวแปรชื่อ number ใช้เก็บข้อมูลชนิด long integer
float percent, total;   // ประกาศตัวแปรชื่อ percent และ total ใช้สำหรับเก็บข้อมูลชนิด float
```

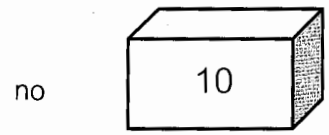
การกำหนดค่าให้ตัวแปรชนิดตัวเลข

การกำหนดค่าให้ตัวแปร ทำได้โดยการประกาศตัวแปรและกำหนดค่าให้กับตัวแปรไว้ในคำสั่งเดียวกัน หรือ อาจทำได้อีกวิธีหนึ่งคือ ประกาศตัวแปรก่อน จากนั้นจึงกำหนดค่าให้กับตัวแปรในอีกข้อความสั่งหนึ่ง เช่น

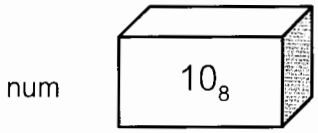
```
int no = 10;    /* ประกาศตัวแปรที่ใช้เก็บข้อมูลชนิด integer ชื่อ no และ
                   เก็บค่า 10 ไว้ในตัวแปรดังกล่าว */
```



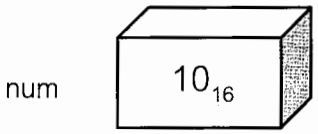

```
int no;         // ประกาศตัวแปรที่ใช้เก็บข้อมูลชนิด integer ชื่อ no
no = 10;        // เก็บค่า 10 ไว้ในตัวแปร no
```



```
int num;            // ประกาศตัวแปรชื่อ num เพื่อใช้เก็บข้อมูลชนิด integer
num = 010;          /* กำหนดให้นำค่า 10_8 เก็บไว้ใน num
                       ตัวเลขที่ขึ้นต้นด้วย 0 (ศูนย์) หมายถึง เลขฐาน 8 */
```




```
int num;            // ประกาศตัวแปรชื่อ num เพื่อใช้เก็บข้อมูลชนิด integer
num = 0x10;         /* 0x10 หมายถึง 10_16
                       กำหนดให้นำค่า 10_16 เก็บไว้ใน num
                       ตัวเลขที่ขึ้นต้นด้วย 0x หรือ 0X (ศูนย์เอ็กซ์) หมายถึง เลขฐาน 16 */
```



ตัวแปรชนิดอักขระ (character variable types)

ตัวแปรชนิดอักขระ (char) ถูกจัดเก็บไว้ในหน่วยความจำ ในรูปแบบจำนวนเต็ม ขนาด 1 ไบต์ ดังนั้น ตัวแปรชนิดอักขระจึงสามารถใช้งานได้ทั้งแบบอักขระ และ จำนวนเต็ม

ตัวอย่าง 1.2

การใช้ตัวแปรชนิดอักขระ

```
#include <stdio.h>
void main()
{
      printf("%c  %d\n", 'a', 'a');
}
```

ผลการกระทำการ

```
a 97
```

printf("%c %d\n", 'a', 'a'); เป็นฟังก์ชันที่ใช้สำหรับให้แสดงอักขระ a ในรูปแบบ char (%c) และการสั่งให้แสดงอักขระ a ในรูปแบบของ จำนวนเต็ม (%d) ส่วน \n หมายถึง ให้เลื่อนตัวชี้ตำแหน่ง (cursor) ไปที่ต้นบรรทัดใหม่

ตัวอย่าง 1.3

การใช้ตัวแปรชนิดอักขระ

```
#include <stdio.h>
void main()
{
      char c1, c2;                    // ประกาศตัวแปรชื่อ c1 และ c2 ให้เป็นชนิดอักขระ
      c1 = 'a';                       // นำ a ไปเก็บไว้ใน c1
      c2 = 'A';                       // นำ A ไปเก็บไว้ใน c2
      printf("\n c1 = %c ", c1);      /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดง
                                        ข้อความ c1 = a (ค่าของ c1 คือ a)      */
      printf("\n c2 = %c ", c2)  ;    /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดง
                                        ข้อความ c2 = A (ค่าของ c2 คือ A)      */
}
```

ตัวอย่าง 1.4

การใช้ตัวแปรชนิดอักขระ

```
#include <stdio.h>
void main()
{
      char lc, uc;              // ประกาศตัวแปรชื่อ lc และ uc ให้เป็นชนิดอักขระ
      lc = 'a';                 // ให้นำ a ไปเก็บไว้ใน lc
      uc = lc - 32;             // นำ 32 ไปลบออกจากค่าที่เก็บใน lc แล้วเก็บผลไว้ใน uc
      printf("\nThe lower case character is %c ", lc);
                                /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดงข้อความ
                                   The lower case character is a (ค่าของ lc คือ a)      */
      printf("\n Its ASCII code is %d ", lc);
                                /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดงข้อความ
                                   Its ASCII code is 97 (ค่าของ lc คือ a ซึ่งมีค่าเท่ากับ 97) */
      printf("\nThe upper case character is %c ", uc);
                                /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดงข้อความ
                                   The uppercase character is A (ค่าของ lc คือ A)      */
      printf("\n Its ASCII code is %d ", uc);
                                /* เลื่อนตัวชี้ตำแหน่งไปที่ต้นบรรทัดใหม่ แล้วแสดงข้อความ
                                   Its ASCII code is 65 (ค่าของ uc คือ A ซึ่งมีค่าเท่ากับ 65) */
}
```

1.3 ตัวคงที่ (constant)

ตัวคงที่มีลักษณะคล้ายตัวแปร แตกต่างจากตัวแปรตรงที่ ค่าที่เก็บในตัวคงที่จะคงเดิมไม่มีการเปลี่ยนแปลงจนกระทั่งจบโปรแกรม แต่ค่าที่เก็บในตัวแปรสามารถเปลี่ยนแปลงได้ตลอดเวลา

การประกาศตัวคงที่

การประกาศตัวคงที่ทำได้ 2 วิธี ดังนี้

1. ใช้คำหลัก const ตามรูปแบบดังนี้

const ชนิดข้อมูล ชื่อตัวแปร = ค่าที่เก็บในตัวแปร;

ตัวอย่าง 1.5

การประกาศตัวคงที่โดยใช้คำหลัก const

```
const int count = 120;          // กำหนดให้ count เป็นตัวคงที่ชนิด int และเก็บค่า 120
const float vat = 0.07;         // กำหนดให้ vat เป็นตัวคงที่ชนิด float และเก็บค่า 0.07
const float pi = 3.14159;       // กำหนดให้ pi เป็นตัวคงที่ชนิด float และเก็บค่า 3.14159
```

2. ใช้ตัวประมวลผลก่อน ตามรูปแบบดังนี้

#define ชื่อตัวคงที่ ค่าคงที่

ตัวอย่าง 1.6

การประกาศตัวคงที่ โดยใช้ข้อความสั่งตัวประมวลผลก่อน

```
#define COUNT 120               // กำหนดให้ COUNT เป็นตัวคงที่ชนิด int และเก็บค่า 120
#define VAT 7                   // กำหนดให้ VAT เป็นตัวคงที่ชนิด float และเก็บค่า 0.07
#define PI  3.14159             // กำหนดให้ PI เป็นตัวคงที่ชนิด float และเก็บค่า 3.14159
```

1.4 การแสดงผลและการรับค่า

ฟังก์ชัน printf()

ฟังก์ชัน printf() เป็นฟังก์ชันจากคลัง ที่มาพร้อมกับตัวแปลโปรแกรมภาษาซี ใช้สำหรับการแสดงผล มีรูปแบบดังนี้

```
printf("สายอักขระควบคุม", ตัวแปร );
```

โดยที่ สายอักขระควบคุม ประกอบด้วย 3 ส่วน คือ

- ตัวอักขระที่จะแสดง
- รูปแบบการแสดงผล ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์ (%)
- ลำดับหลีก (escape sequence)

ตัวแปร คือ ชื่อของตัวแปรที่จะแสดงผล

รูปแบบการแสดงผล (format specifiers)

การกำหนดรูปแบบการแสดงผล

- ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์ (%)
- ตามด้วยอักขระ 1 ตัว หรือหลายตัว โดยที่อักขระนั้นมีความหมายดังนี้

อักขระ	ชนิดข้อมูล	รูปแบบการแสดงผล
c	char	อักขระเดียว
d	int	จำนวนเต็มฐานสิบ
o		จำนวนเต็มฐานแปด
x		จำนวนเต็มฐานสิบหก
f	float	จำนวนที่มีทศนิยม ในรูปฐานสิบ

ตัวอย่าง 1.7

การใช้ฟังก์ชัน printf()

```
#include <stdio.h>
void main()
{
      int n;
      float score;
      n = 10;
      score = 100;
      printf("No : %d   Score : %f", n, score);
}
```

ผลการกระทำการ

No : 10 Score : 100.000000

แสดงค่าที่เก็บในตัวแปร n ในรูปแบบจำนวนเต็ม และแสดงค่าที่เก็บใน score ในรูปแบบจำนวนจริงที่มีทศนิยม

ตัวอย่าง 1.8

การใช้ฟังก์ชัน printf()

```
#include <stdio.h>
void main()
{
      int n;
      float score;
      n = 10;
      score = 100;
      printf("No : %d", n);
      printf("   Score : %f", score);
}
```

ผลการกระทำการ

No : 10 Score : 100.000000

ลำดับหลีก (escape sequence)

ในการแสดงผล บางสิ่งบางอย่างที่จะแสดง อาจไม่ใช่ตัวอักษร จึงไม่สามารถที่จะเขียนสิ่งที่จะแสดงไว้ในโปรแกรมได้ เช่น ต้องการเขียนโปรแกรมให้ส่งเสียง (แสดงผลเป็นเสียง) หรือต้องการให้เลื่อนขึ้นบรรทัดใหม่ก่อนแสดงข้อความ ดังนั้นในการเขียนโปรแกรมเพื่อแสดงผลสิ่งที่ไม่ใช่ตัวอักขระปกติ จะต้องใช้ลำดับหลีก เพื่อช่วยในการกำหนดอักขระพิเศษหรือสิ่งที่ไม่ใช่อักขระ ที่ต้องการให้โปรแกรมแสดง

ลำดับหลีก จะเขียนขึ้นต้นด้วยเครื่องหมายทับกลับหลัง (/) แล้วตามด้วยอักขระ ในการทำงาน เครื่องหมายทับกลับหลังจะบอกให้เครื่องคอมพิวเตอร์ทราบว่า ให้หลีกเลี่ยงการตีความอักขระที่ตามหลังมานี้ในลักษณะปกติ เพราะอักขระเหล่านี้จะมีความหมายพิเศษแตกต่างออกไป

ลำดับหลีกที่ใช้ในการแสดงผลสิ่งที่ไม่ใช่อักขระปกติ ได้แก่

ลำดับหลีก	ผลการกระทำการ
\n	ขึ้นบรรทัดใหม่ (new line)
\t	เลื่อนไปยังจุดตั้งระยะ (tab) ถัดไป
\a	เสียงกระดิ่ง (bell)
\b	ถอยไปหนึ่งที่ว่าง (backspace)
\f	ขึ้นหน้าใหม่ (form feed)
\\	แสดงเครื่องหมายทับกลับหลัง (backslash)
\'	แสดงเครื่องหมายฝนทอง (single quote)
\"	แสดงเครื่องหมายฟันหนู (double quote)

ตัวอย่าง 1.9

การใช้ลำดับหลีก \n

```
#include <stdio.h>                          // ข้อความสั่งตัวประมวลผลก่อน
void main()
{
      printf("Welcome to C!\n");      // ฟังก์ชันจากคลัง ใช้ในการแสดงข้อความที่อยู่ใน " "
}
```

ผลการกระทำการ

```
Welcome to C!

_
```

แสดงข้อความ Welcome to C! ณ ตำแหน่งที่ตัวชี้ตำแหน่งอยู่ แล้วตัวชี้ตำแหน่งเลื่อนไปที่ต้นบรรทัดใหม่

ตัวอย่าง 1.10

การใช้ลำดับหลีก \f และ \n

```
#include <stdio.h>                  // ข้อความสั่งตัวประมวลผลก่อน
void main()
{
      printf("\fWelcome ");         // ฟังก์ชันจากคลังใช้ในการแสดงข้อความที่อยู่ในฟันหนู (" ")
      printf("to C!\n");            /* ฟังก์ชันจากคลังใช้ในการแสดง
                                       ข้อความและอักขระที่อยู่ใน ฟันหนู (" ") */
}
```

ผลการกระทำการ

Welcome to C!

_

เลื่อนไปหน้าใหม่ แล้วแสดงข้อความ Welcome to C! ที่สดมภ์แรกของบรรทัดแรก แล้วตัวชี้ตำแหน่งเลื่อนไปที่ต้นบรรทัดใหม่ ดังนี้

ตัวอย่าง 1.11

การใช้ลำดับหลีก \n

```
#include <stdio.h>                          // ข้อความสั่งตัวประมวลผลก่อน
void main()
{
      printf("Welcome\n to\n C!\n");        /* ฟังก์ชันจากคลังใช้ในการแสดง
                                              ข้อความและอักขระที่อยู่ใน ฟันหนู (" ") */
}
```

ผลการกระทำการ

Welcome

to

C!

_

แสดงข้อความ Welcome ณ ตำแหน่งที่ตัวชี้จอภาพอยู่ แล้วตัวชี้ตำแหน่งเลื่อนไปที่ต้นบรรทัดใหม่
ที่ต้นบรรทัดใหม่ แสดงข้อความ to แล้วตัวชี้ตำแหน่งเลื่อนไปที่ต้นบรรทัดใหม่
ที่ต้นบรรทัดใหม่ แสดงข้อความ C! แล้วตัวชี้ตำแหน่งเลื่อนไปที่ต้นบรรทัดใหม่

ตัวอย่าง 1.12

การใช้รูปแบบการแสดงผลของจำนวนเต็ม

```
#include <stdio.h>
void main()
{
      int i, j;
      i = 368;
      j = 24;
      printf("%d\n", i);
      printf("%d    %d", i, j);
}
```

ผลของการกระทำการ

368

368 24

ตัวอย่าง 1.13

การใช้รูปแบบการแสดงผลของจำนวนจริงที่มีทศนิยม

```
#include <stdio.h>
void main()
{
      float i, j;
      i = 100.123456;
      j = 4.345678e2;
      printf("\n\n Fixed point format");
      printf("\n i=%f  j=%f", i, j);
}
```

ผลการกระทำการ

-บรรทัดว่าง-

-บรรทัดว่าง-

Fixed point format

-บรรทัดว่าง-

i=100.123456 j=434.567810

ฟังก์ชัน scanf()

ฟังก์ชัน scanf() เป็นฟังก์ชันจากคลัง ใช้ในการรับข้อมูลจากแป้นพิมพ์ โดยจะบอกเลขที่อยู่ของตัวแปรในหน่วยความจำ แล้วจึงนำค่าที่รับมาไปเก็บไว้ตามที่อยู่นั้น

ฟังก์ชัน scanf() มีรูปแบบดังนี้

scanf ("%รูปแบบ", &ตัวแปร);

โดยที่ &ตัวแปร หมายถึง เลขที่อยู่ (address) ของตัวแปรที่จะรับค่ามาเก็บในหน่วยความจำ

ตัวอย่าง 1.14

การรับข้อมูลโดยใช้ scanf()

```
scanf("%f", &radius);
```

รับข้อมูลชนิด float จากแป้นพิมพ์ แล้วนำไปเก็บไว้ในตัวแปร radius ซึ่งมีเลขที่อยู่คือ &radius

ตัวอย่าง 1.15

การรับข้อมูลโดยใช้ scanf()

```
scanf("%d%f%d", &no, &amount, &unit);
```

รับข้อมูล 3 ค่า ชนิด int, float และ int จากแป้นพิมพ์ แล้วนำไปเก็บไว้ในตัวแปร no, amount และ unit ซึ่งมีเลขที่อยู่ตามที่กำหนดไว้ใน &no, &amount และ &unit ตามลำดับ

ตัวอย่าง 1.16

การรับข้อมูลโดยใช้ scanf()

```
scanf("%f%f%f", &length, &width, &height);
```

รับข้อมูล 3 ค่า ชนิด float จากแป้นพิมพ์ แล้วนำไปเก็บไว้ในตัวแปร length, width และ height ซึ่งมีเลขที่อยู่คือ ตามที่กำหนดไว้ใน &length, &width และ &height ตามลำดับ

ตัวแปร	หน่วยความจำ	เลขที่อยู่ (เปลี่ยนแปลงไปในการกระทำการแต่ละครั้ง)
length	35.5	8760
width	42.0	8764
height	5.2	8768

1.5 นิพจน์ (expressions)

ในภาษาซี นิพจน์ หมายถึง สิ่งที่ประมวลผลแล้วสามารถให้เป็นค่าตัวเลขได้ ซึ่งแต่ละนิพจน์จะมีระดับความยากง่ายในการประมวลผลที่แตกต่างกัน

นิพจน์ที่มีระดับการประมวลผลแบบง่ายที่สุด จะประกอบด้วย ตัวแปรเพียงตัวเดียว หรือ ค่าคงที่ นิพจน์ที่มีลักษณะเป็นค่าคงที่ เช่น

```
100
'g'
```

นิพจน์ที่เป็นค่าคงที่ที่เป็นสัญลักษณ์ เช่น

```
#define VAT 7
#define PI 3.14159
const int a = 35;
const char ch = 'm';
```

จากข้อความสั่งในภาษาซีดังกล่าวข้างต้น ทำให้ VAT, PI, a และ ch ต่างเป็นนิพจน์ที่เป็นค่าคงที่

นิพจน์ที่มีลักษณะเป็นตัวแปร เช่น

```
int count;
float amount;
char ch;
```

จากข้อความสั่งในภาษาซีดังกล่าวข้างต้น ทำให้ count, amount และ ch ต่างเป็นนิพจน์ที่เป็นตัวแปร

ดังนั้น นิพจน์ จึงหมายถึง จำนวนใดจำนวนหนึ่งต่อไปนี้

- จำนวนเต็มจำนวนเดียว
- จำนวนจริงจำนวนเดียว
- ตัวเลขจำนวนเต็ม หรือ ตัวเลขจำนวนจริง หลายจำนวน ที่เชื่อมโยงกันด้วยตัวดำเนินการ +, -, *, / หรือ% ซึ่งเรียกว่า การคำนวณทางคณิตศาสตร์

สำหรับนิพจน์ที่มีระดับการประมวลผลที่ซับซ้อน จะประกอบด้วย นิพจน์ที่มีระดับการประมวลผลอย่างง่ายหลายนิพจน์ และเชื่อมต่อกันด้วยตัวดำเนินการ เช่น

นิพจน์ที่ประกอบด้วย 2 นิพจน์ และ 1 ตัวดำเนินการ

```
37 + 6
54 * 7
405 / 9
```

นิพจน์ที่มีความซับซ้อนมากขึ้น เช่น

```
score1 * 2 + score2  * 5 + score3 * 3
```

1.6 ข้อความสั่งกำหนดค่า (assignment statement)

ข้อความสั่งกำหนดค่า ใช้สำหรับกำหนดค่าให้กับตัวแปร มีรูปแบบดังนี้

ตัวแปร = นิพจน์;

ข้อความสั่งกำหนดค่า คือ ข้อความสั่งที่ใช้สำหรับ สั่งให้นำผลลัพธ์ของนิพจน์ที่อยู่ด้านขวาของตัวดำเนินการเท่ากับ (=) มาเก็บไว้ในตัวแปรที่อยู่ด้านซ้ายของตัวดำเนินการเท่ากับ (=)

เมื่อนำนิพจน์มาเขียนไว้ในโปรแกรมภาษาซีจะกลายเป็นข้อความกำหนดค่า ดังตัวอย่างต่อไปนี้

```
con = 10.5;
result = 25 * 6;
point = score1 * 2 + score2  * 5 + score3 * 3;
```

ตัวอย่าง 1.17

การใช้ข้อความสั่งกำหนดค่า

```
#include <stdio.h>
void main()
{
      int a, b;
      scanf("%d",&a);
      a = b;                              // นำค่าที่เก็บใน b ไปเก็บไว้ใน a
      printf("a is %d\n", a);
      printf("b is %d\n", b);
 }
```

ข้อความสั่งกำหนดค่าอาจเขียนในรูปแบบที่ซับซ้อนขึ้นดังนี้

a = b = 0;

กำหนดให้ เก็บค่าศูนย์ไว้ใน a และ b

ตัวอย่าง 1.18

การใช้ข้อความสั่งกำหนดค่าในรูปแบบที่ซับซ้อน

```
#include <stdio.h>
void main()
{
        int sum, total;
        sum = total = 0;
        printf(" sum   = %d \n", sum);
        printf(" total = %d \n", total);
}
```

ผลการกระทำการ

sum = 0

total = 0

1.7 การคำนวณทางคณิตศาสตร์

ในการเขียนโปรแกรม เพื่อทำการคำนวณทางคณิตศาสตร์ จะต้องใช้ตัวดำเนินการต่างๆ ซึ่งมีวิธีการใช้งาน และการทำงาน ดังนี้

การคำนวณ	ตัวดำเนินการ	ตัวอย่าง	การทำงาน
บวก	+	c = a + b;	นำค่าที่เก็บใน a บวกกับค่าที่เก็บใน b แล้วเก็บผลลัพธ์ไว้ใน c
ลบ	-	c = a – b;	นำค่าที่เก็บใน b ลบออกจากค่าที่เก็บใน a แล้วเก็บผลลัพธ์ไว้ใน c
คูณ	*	c = a * b;	นำค่าที่เก็บใน a คูณกับค่าที่เก็บใน b แล้วเก็บผลลัพธ์ไว้ใน c
หาร	/	c = a / b;	ให้ค่าที่เก็บใน a เป็นตัวตั้ง ค่าที่เก็บใน b เป็นตัวหาร แล้วเก็บผลหารไว้ใน c ถ้าทั้งตัวตั้งและตัวหารต่างเป็นจำนวนเต็ม ค่าที่เก็บใน c จะเป็นจำนวนเต็ม แต่ถ้าตัวตั้งหรือตัวหารตัวใดตัวหนึ่งเป็นจำนวนจริง ที่มีทศนิยม ผลลัพธ์ที่ได้จะเป็นจำนวนจริงที่มีทศนิยมด้วย
มอดูลัส	%	c = a % b;	ให้ค่าที่เก็บใน a เป็นตัวตั้ง ค่าที่เก็บใน b เป็นตัวหาร แล้วเก็บเศษไว้ใน c

ตัวอย่าง 1.19

การแสดงผลลัพธ์ของการบวก

```
#include <stdio.h>
void main()
{
      int a, b, sum;
      scanf("%d",&a);                    // รับค่าจำนวนเต็มมาเก็บไว้ใน a
      scanf("%d",&b);                    // รับค่าจำนวนเต็มมาเก็บไว้ใน a
      printf("Sum is %d\n", a + b);      // แสดงผลจากการบวก
 }
```

ตัวอย่าง 1.20

การใช้ข้อความสั่งกำหนดค่า และแสดงผลลัพธ์ของการบวก

```
#include <stdio.h>
void main()
{
      int a, b, sum;
      scanf("%d", &a);                   // รับค่าจำนวนเต็มมาเก็บไว้ใน a
      scanf("%d", &b);                   // รับค่าจำนวนเต็มมาเก็บไว้ใน b
      sum = a + b;                       /* นำค่าที่เก็บใน a บวกกับค่าที่เก็บใน b
                                            แล้วนำผลบวกไปเก็บใน sum */
      printf("Sum is %d\n", sum);        // แสดงผลจากการบวก
      sum = sum + sum;                   /* นำค่าที่เก็บใน sum บวกกัน
                                            แล้วเก็บไว้ใน sum ตามเดิม */
      printf("Sum is %d\n", sum);        // แสดงค่าที่เก็บไว้ใน sum
}
```

ตัวอย่าง 1.21

การแสดงผลลัพธ์ของการคูณ

```
#include <stdio.h>
#define GP 454
int gram, pound;
void main()
{
      scanf("%d", &pound);                        // รับค่าจำนวนเต็ม pound
      gram = pound * GP;                          // แปลงปอนด์ให้เป็นกรัม
      printf("Weight in grams = %d\n", gram);     // แสดงค่ากรัมที่ได้จากการแปลง
}
```

ตัวอย่าง 1.22

การแสดงผลลัพธ์ของการหารและมอดูลัส

```
#include <stdio.h>
#define MS 60                                   // กำหนดให้ MSเป็นตัวคงที่มีค่า 60
#define HM 60                                   // กำหนดให้ HM เป็นตัวคงที่มีค่า 60
int sec, min, hr, sec_left, min_left;
void main()
{
      scanf("%d", &sec);                        // รับค่า sec (วินาที)
      min = sec / MS;                           // แปลงวินาทีให้เป็นนาที
      sec_left = sec % MS;                      // หาเศษวินาทีจากการแปลงเป็นนาที
      hr = min / HM;                            // แปลงนาทีให้เป็นชั่วโมง
      min_left = min % HM;                      // หาเศษนาทีจากการแปลงเป็นชั่วโมง
      printf("%d seconds is equal to ", sec);
      printf("%d h, %d m, and %d s", hr, min_left, sec_left);
}
```

ตัวอย่าง 1.23

การแสดงผลลัพธ์ของการหาร

```
#include <stdio.h>
void main()
{
      printf("\n integer division");
      printf("\n 11/4 = %d", 11/4);    // การหาร โดยมีตัวตั้งและตัวหารเป็นจำนวนเต็ม
      printf("\n 3/4 = %d", 3/4);      // การหาร โดยมีตัวตั้งและตัวหารเป็นจำนวนเต็ม
      printf("\n floating point division");
      printf("\n 11./4. = %f", 11./4.);   /* การหาร โดยมีตัวตั้งและตัวหาร
                                            เป็นจำนวนจริงที่มีทศนิยม */
      printf("\n 3./4. = %f", 3./4.);      /* การหาร โดยมีตัวตั้งและตัวหาร
                                            เป็นจำนวนจริงที่มีทศนิยม */
}
```

ผลการกระทำการ

integer division

11/4 = 2

3/4 = 0

floating point division

11./4. = 2.750000

3./4. = 0.750000

ตัวอย่าง 1.24

การใช้ข้อความสั่งกำหนดค่า และการบวก

```
#include <stdio.h>
void main()
{
      int a, b, x;
      a = 20;                       // กำหนดให้เก็บค่า 20 ไว้ใน a
      x = a;                        // กำหนดให้เก็บค่าที่อยู่ใน a ไว้ใน b
      printf("x is %d\n", x);       // แสดงค่าที่เก็บใน x
      x = a +1;                     // นำค่าที่เก็บใน a บวก 1 แล้วเก็บใน x
      printf("x is %d\n", x);       // แสดงค่าที่เก็บใน x
      b = 30;                       // กำหนดให้เก็บค่า 30 ไว้ใน b
      x = a + b;                    // นำค่าที่อยู่ใน a บวกกับค่าที่อยู่ใน b แล้วเก็บไว้ใน x
      printf("x is %d\n", x);       // แสดงค่าที่เก็บใน x
}
```

ผลการกระทำการ

x is 20

x is 21

x is 50

_

ลำดับการดำเนินการในนิพจน์ที่มีตัวดำเนินการหลายตัว

ในกรณีที่นิพจน์มีตัวดำเนินการหลายตัว จะต้องดำเนินการตามลำดับต่อไปนี้

1. ()
2. * / %
3. + -

ถ้าในนิพจน์มีตัวดำเนินการที่มีลำดับเท่ากัน จะประมวลผลจากซ้ายไปขวา

ตัวอย่าง 1.25

การประมวลผลนิพจน์มีตัวดำเนินการหลายตัว

```
#include <stdio.h>
void main()
{
      int p, q, r, w, m, n, y;
      p = 30;
      q = 2;
      r = 10;
      w = 7;
      m = 20;
      n = 5;
      y = p + q * r % w - m / n;
      printf("y =  %d", y);
}
```

ผลการกระทำการ

y = 32

การประมวลผลนิพจน์ p + q * r % w – m / n จะทำตามลำดับดังนี้

1. q * r = 2 * 10 = 20
2. 20 % w = 20 % 7 = 6
3. m / n = 20 / 5 = 4
4. p + 6 = 30 + 6 = 36
5. 36 – 4 = 32

ตัวอย่าง 1.26

การใช้ข้อความสั่งกำหนดค่าในรูปแบบที่ซับซ้อน และมีตัวดำเนินการหลายตัว

```
#include <stdio.h>
void main()
{
      float sum, point, score1, score2, score3;
      scanf("%f%f%f", &score1, &score2, &score3);
      sum = point = score1 * 2 + score2 * 5 + score3 * 3;
      printf(" sum   = %f \n", sum);
      printf(" point = %f \n", point);
}
```

1.8 ตัวดำเนินการเอกภาค (unary operator)

ตัวดำเนินการเอกภาค คือ การใช้ตัวดำเนินการ กับตัวแปรตัวเดียว ในที่นี้จะแสดง การใช้ตัวดำเนินการ 2 ตัวกับตัวแปรตัวเดียว ซึ่งมีลักษณะการใช้ 2 แบบ คือ

1. ตัวดำเนินการเอกภาคเติมหลัง (postfix mode) หมายถึง ตัวดำเนินการเอกภาคอยู่หลังตัวแปร เช่น a++ หมายถึง ให้เพิ่มค่าให้ตัวแปร a ขึ้นอีก 1
2. ตัวดำเนินการเอกภาคเติมหน้า (prefix mode) หมายถึง ตัวดำเนินการเอกภาคอยู่หลังตัวแปร เช่น ++a หมายถึง ให้เพิ่มค่าให้ตัวแปร a ขึ้นอีก 1

การใช้ตัวดำเนินการเอกภาคทั้ง 2 แบบ มีการใช้งานดังนี้

การคำนวณ	ตัวดำเนินการ	ตัวอย่าง	การทำงาน
เพิ่มค่าตัวถูกดำเนินการทีละหนึ่ง	++	x++	x = x + 1
เพิ่มค่าตัวถูกดำเนินการทีละหนึ่ง	++	++ x	x = x + 1
ลดค่าตัวถูกดำเนินการทีละหนึ่ง	--	--x	x = x - 1
ลดค่าตัวถูกดำเนินการทีละหนึ่ง	--	x--	x = x - 1

ตัวอย่าง 1.27

การใช้ตัวดำเนินการเอกภาค

```
#include <stdio.h>
void main()
{
      int a, b, x;
      a = 20;                          // กำหนดให้เก็บค่า 20 ไว้ใน a
      x = a;                           // กำหนดให้เก็บค่าที่เก็บใน a ไว้ใน b
      printf("x is %d\n", x);          // แสดงค่าที่เก็บใน x
      x ++;                            // เพิ่มค่า x ขึ้นอีก 1
      printf("x is %d\n", x);          // แสดงค่า x
      b = 30;                          // กำหนดให้เก็บค่า 20 ไว้ใน a
      x = a + b;                       /* กำหนดให้นำค่าที่อยู่ใน a บวกเข้ากับค่าที่เก็บใน b
                                          แล้วนำไปเก็บไว้ใน x */
      printf("x is %d\n", x);          // แสดงค่าที่เก็บใน x
}
```

ผลการกระทำการ

x is 20

x is 21

x is 50

จากที่กล่าวมาแล้วข้างต้น จะเห็นว่าการใช้ตัวดำเนินการเอกภาคทั้ง 2 แบบมีการทำงานเหมือนกัน แต่ในความเป็นจริง ตัวดำเนินการเอกภาคเติมหลัง มีการทำงานและความหมายแตกต่างจากตัวดำเนินการเอกภาคเติมหน้า เมื่อมีการใช้งานร่วมกับการทำงานอื่นภายในคำสั่งเดียวกัน ดังนี้

- ตัวดำเนินการเอกภาคเติมหลัง (postfix mode) จะทำงานอื่นภายในข้อความสั่งเดียวกันก่อน จึงจะเพิ่มค่าให้ตัวแปร

- ตัวดำเนินการเอกภาคเติมหน้า (prefix mode) จะเพิ่มค่าให้ตัวแปรก่อนแล้วจึงจะทำงานอื่นภายในข้อความสั่งเดียวกัน

ตัวอย่าง 1.28

การใช้ตัวดำเนินการเอกภาคร่วมกับการทำงานอื่นๆ ภายในข้อความสั่งเดียวกัน

```
#include <stdio.h>
void main()
{
      int x, y;
      x = y = 0;
      printf("\nx   y");
      printf("\n%d  %d", x++, ++y);    /* พิมพ์ค่า x ก่อนแล้วจึงเพิ่มค่า x ขึ้นอีก 1
                                           เพิ่มค่า y ขึ้นอีก 1 แล้วจึงพิมพ์ค่า y   */

      printf("\n%d  %d", x++, ++y);
      printf("\n%d  %d", x++, ++y);
      printf("\n%d  %d", x++, ++y);
      printf("\n%d  %d", x++, ++y);
      y = x++;                         // เก็บค่า x ไว้ใน y ก่อน แล้วจึงเพิ่มค่า x ขึ้นอีก 1
      printf("\n%d     %d", x, y );
      y = ++x;                         // เพิ่มค่า x ขึ้นอีก 1 แล้วจึงเก็บค่า x ไว้ใน y
      printf("\n%d     %d", x, y);
}
```

ผลการกระทำการ

x	y
0	1
1	2
2	3
3	4
4	5
5	4
6	6

1.9 ตัวดำเนินการประกอบ (compound operator)

ตัวดำเนินการประกอบ เป็นการใช้ตัวดำเนินการหนึ่งตัวร่วมกับเครื่องหมายเท่ากับ การใช้ตัวดำเนินการประกอบจะช่วยให้เขียนข้อความสั่งได้สั้น และเร็วขึ้น

ตัวดำเนินการประกอบ	ตัวอย่าง	การทำงาน
+=	x+=5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= 5	x = x % 5
+=	x += y / 8	x = x + y / 8

ลำดับในการดำเนินการ

ในกรณีที่คำสั่งประกอบด้วยตัวดำเนินการประกอบ ตัวดำเนินการเอกภาคหลายตัว จะประมวลผลตามลำดับต่อไปนี้

1. ()
2. ++ --
3. * / %
4. + -
5. += *= /= -= %=

ถ้าในคำสั่งมีตัวดำเนินการที่อยู่ในลำดับเท่ากัน จะประมวลผลจากซ้ายไปขวา

ตัวอย่าง 1.29

การใช้ตัวดำเนินการประกอบ

```
#include <stdio.h>
void main()
{
      int  x = 10, y = 20;
      printf("\n %d  %d", x, y);
      ++x;
      printf("\n %d  %d", x, y);
      y = --x;
      printf("\n %d  %d", x, y);
      x = x-- + y;
      printf("\n %d  %d", x, y);
      y = x - ++x;
      printf("\n %d  %d", x, y);
}
```

ผลการกระทำการ

```
10      20
11      20
10      10
19      10
20       0
```

การทำงานของคำสั่งกำหนดค่าและคำสั่งที่เปลี่ยนแปลงค่าของตัวแปร x และ y

คำสั่ง	x	y
int x = 10, y = 20;	10	20
++x;	11	20
y = --x;	10	10
x = x-- + y;	19	10
y = x - ++x;	20	0

1.10 การแปลงชนิดข้อมูล (type cast)

การแปลงชนิดข้อมูลมีหลายวิธี แต่ที่กล่าวในที่นี้คือ การแปลงชนิดข้อมูลโดยการกำหนดชนิดไว้ที่หน้าข้อมูลนั้น

รูปแบบ

ตัวแปร = (ชนิดข้อมูล) นิพจน์;

โดยที่ (ชนิดข้อมูล) นิพจน์ อาจมีหลายชุด แล้วเชื่อมโยงกันด้วยตัวดำเนินการต่างๆ

ตัวอย่าง 1.30

การแปลงชนิดข้อมูล จากจำนวนจริงที่มีทศนิยมให้เป็นจำนวนเต็ม

```
#include <stdio.h>
void main()
{
      int x;
      x = 5.6 + 3.5;
      printf("\n %d", x);
      x = (int) 5.6 + (int) 3.5;      // แปลง 5.6 ให้เป็น 5 และแปลง 3.5 ให้เป็น 3
      printf("\n %d", x);
}
```

ผลการกระทำการ

```
9
-บรรทัดว่าง-
8
```

ตัวอย่าง 1.31

การแปลงชนิดข้อมูลจากจำนวนเต็มให้เป็นจำนวนจริงที่มีทศนิยม

```
#include <stdio.h>
void main()
{
      int x = 3, y = 2;
      float a, b;
      a = x / y;                  // ทั้งตัวตั้งและตัวหารต่างเป็นจำนวนเต็ม ผลลัพธ์จึงปัดเศษทิ้ง
      printf("\n %f", a);
      b = (float) x / y;          /* กำหนดให้ตัวตั้งเป็นจำนวนจริงที่มีทศนิยม
                                     ผลลัพธ์จึงเป็นจำนวนจริงที่มีทศนิยมด้วย  */
      printf("\n %f", b);
}
```

ผลการกระทำการ

1.000000

1.500000

1.11 การกำหนดค่าจากข้อมูลหลายชนิด (assignment with mixed types)

ถ้านิพจน์ในข้อความสั่งกำหนดค่าประกอบด้วย ตัวแปร หรือ ตัวคงที่ที่มีชนิดข้อมูลต่างกัน จะต้องแปลงให้เป็นชนิดเดียวกันก่อน แล้วจึงนำมาดำเนินการ โดยมีหลักดังนี้ คือ

ถ้าตัวแปร หรือ ตัวคงที่ มีชนิดข้อมูลที่ต่างกัน จะต้องแปลงให้เป็นชนิดเดียวกันก่อน แล้วจึงดำเนินการ โดยในการแปลงจะต้องแปลงชนิดข้อมูลที่มีขนาดเล็กกว่าให้เป็นชนิดข้อมูลที่มีขนาดใหญ่กว่า

ดังนั้น ถ้าตัวแปร หรือ ตัวคงที่ ตัวหนึ่งเป็นชนิดจำนวนเต็ม (int) ส่วนอีกตัวหนึ่งเป็นชนิดจำนวนจริง (float) จะต้องแปลงตัวแปรหรือตัวคงที่ที่เป็นจำนวนเต็ม (int) ให้เป็นจำนวนจริง (float) ก่อน แล้วจึงดำเนินการ

ตัวอย่าง 1.32

แสดงการกำหนดค่าจากข้อมูลหลายชนิด

```
#include <stdio.h>
void main()
{
      int i, j, k;
      float a, b, c;
      i = 5;
      j = 3;
      a = 2.5;
      b = 30.6;
      k = i + a;                  /* แปลงชนิดข้อมูลของ i ให้เป็น float ก่อน
                                     แล้วจึงบวก จากนั้นปัดเศษทิ้ง
                                     แล้วค่อยนำไปเก็บที่ k */

      printf("k = %d", k);
      k = a + b;                  // ปัดเศษของผลบวกทิ้ง แล้วค่อยนำไปเก็บไว้ใน k
      printf("k = %d", k);
      c = a + b;
      printf("c = %f", c);
      c = i + a;
      printf("c = %f", c);
}
```

ผลการกระทำการ

k = 7

k = 33

c = 33.100000

แบบฝึกหัดบทที่ 1

1. ให้เขียนข้อความสั่งเพื่อประกาศตัวแปรเพื่อใช้เก็บค่าต่อไปนี้

   1.1 อุณหภูมิ

   1.2 ความยาวของเส้นรอบวงกลม

   1.3 จำนวนผู้ชมภาพยนตร์

   1.4 ระดับคะแนนของนิสิตซึ่งมี 5 ระดับ คือ A, B, C, D และ F

   1.5 คะแนนเฉลี่ยของนิสิต

2. ถ้ากำหนดให้ a = 5, b = 3, c = 2, d = 0.5 ให้แสดงค่าของ y ถ้ากำหนด y มีชนิดข้อมูล int

   2.1 y = a * b + c;

   2.2 y = b + c * b;

   2.3 y = a * a + b * b + c * c;

   2.4 y = c % 5;

   2.5 y = a / c;

   2.6 y = a / d;

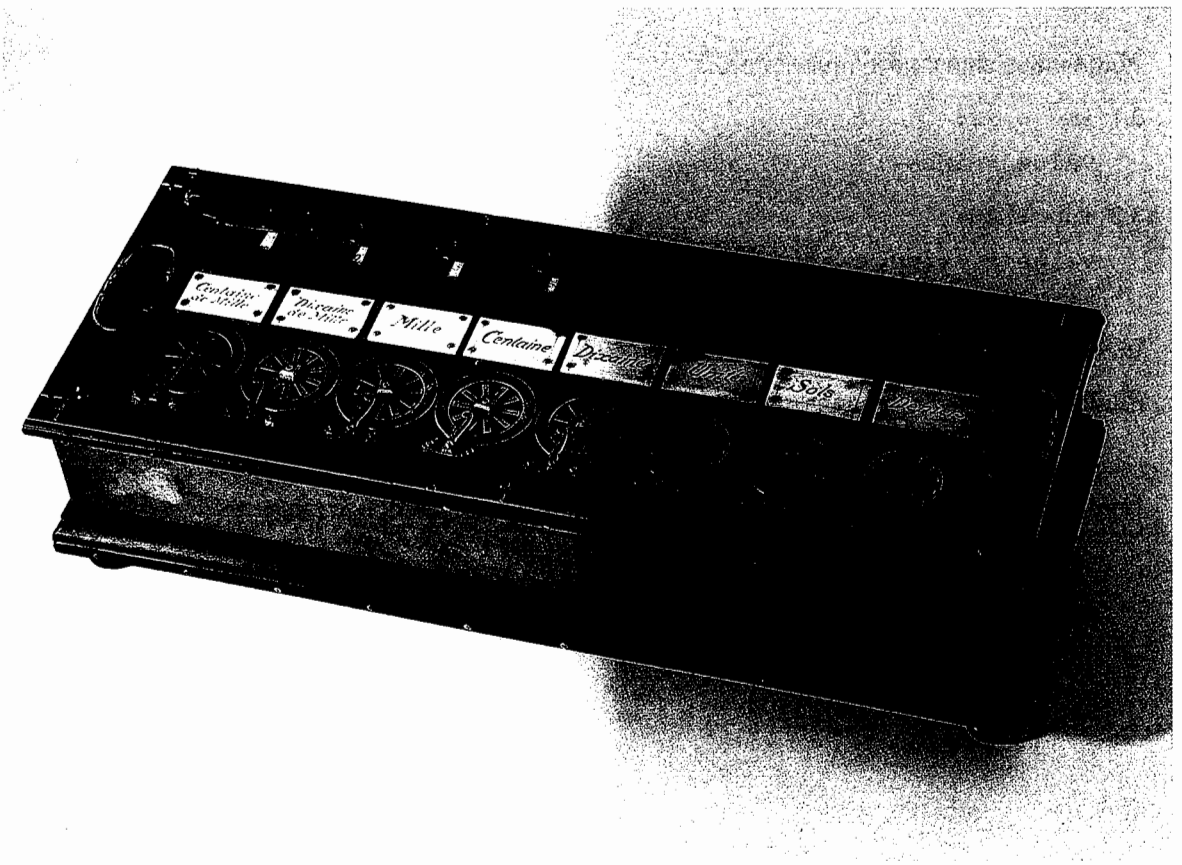
3. ให้แสดงผลของข้อความสั่งกำหนดค่าต่อไปนี้

   3.1
   ```
   inc = 20;
   value = inc++;
   ```

   3.2
   ```
   inc = 20;
   value = ++inc;
   ```

4. ให้เขียนโปรแกรมเพื่อรับค่าตัวเลขจำนวนเต็ม 2 จำนวน แล้วนำจำนวนทั้ง 2 มาหารกัน ให้ตัวแรกเป็นตัวตั้ง ตัวหลังเป็นตัวหาร ให้แสดงผลที่ได้จากการหารทั้งผลและเศษเป็นเลขจำนวนเต็ม

5. ให้เขียนโปรแกรมเพื่อรับค่ารัศมีของวงกลม แล้วคำนวณหาพื้นที่และเส้นรอบวง พร้อมทั้งแสดงผล



เครื่องบวกเลข ประดิษฐ์โดย **Blaise Pascal** ค.ศ.1642

บทที่ 2
การจัดรูปแบบในการรับและแสดงผลข้อมูล

ในการประมวลผลข้อมูลด้วยคอมพิวเตอร์นั้นจำเป็นที่จะต้องมีการรับข้อมูลเข้าจากผู้ใช้ ซึ่งมักจะรับจากแป้นพิมพ์ เมื่อคอมพิวเตอร์ได้รับข้อมูลแล้วจะนำมาเก็บในตัวแปร (variable) หลังจากนั้นจะทำการประมวลผลข้อมูลที่เก็บอยู่ในตัวแปรและแสดงผลลัพธ์ให้ผู้ใช้ทราบผ่านทางจอภาพ ดังนั้นคำสั่งที่เกี่ยวข้องกับการรับและแสดงผลข้อมูลจึงมีความจำเป็นอย่างยิ่งในการเขียนโปรแกรมทุกภาษา

สำหรับภาษาซีนั้นได้มีการเตรียมฟังก์ชันมาตรฐานที่เกี่ยวข้องกับการรับและแสดงผลไว้ใน header file ที่ชื่อว่า stdio.h ดังนั้นผู้เขียนโปรแกรมสามารถเรียกใช้ฟังก์ชันเหล่านี้ได้โดยที่จะต้องประกาศให้ตัวแปรโปรแกรม (compiler) ได้ทราบถึงแหล่งที่มาของฟังก์ชันเหล่านี้ โดยการใช้ข้อความสั่ง #include <stdio.h> ที่ส่วนต้นของโปรแกรม เนื้อหาในบทนี้จะกล่าวถึงฟังก์ชันที่ใช้ในการรับข้อมูลจากแป้นพิมพ์ และฟังก์ชันที่ใช้ในการแสดงผลข้อมูลออกทางจอภาพ

2.1 ฟังก์ชันรับข้อมูล

ในภาษาซีสามารถใช้ฟังก์ชัน scanf() ในการรับข้อมูลจากแป้นพิมพ์ซึ่งมีรูปแบบดังนี้

scanf("%รูปแบบ", &ตัวแปร);

รูปแบบ หมายถึง รูปแบบการรับข้อมูล เช่น %d ใช้กับการรับข้อมูลจำนวนเต็ม %f ใช้กับการรับข้อมูลจำนวนจริง

ในการรับข้อมูลสามารถรับข้อมูลได้ครั้งละหลายตัวแปรได้โดยจะต้องมีการระบุรูปแบบกำกับให้กับตัวแปรทุกตัว เช่น scanf("%d%d", &x, &y); หมายถึงการรับข้อมูลชนิดจำนวนเต็มมาเก็บไว้ในตัวแปร x และ ตัวแปร y ที่ตำแหน่งของตัวแปรดังกล่าวในหน่วยความจำ ตามลำดับ โดยที่ตัวแปรทั้งสองได้มีการประกาศไว้ที่ส่วนต้นของโปรแกรมหลักแล้ว โดยจะต้องใช้เครื่องหมาย & นำหน้าตัวแปรทุกครั้งสำหรับฟังก์ชัน scanf()

เนื่องจากตัวแปรมีหลายชนิด ดังนั้นจะขอแสดงรูปแบบการรับข้อมูลเข้าตามชนิดของตัวแปรดังนี้

ชนิดข้อมูล	ชนิดตัวแปร	รูปแบบสำหรับ scanf
จำนวนเต็ม	short	%hd หรือ %hi
	integer	%d หรือ % i
	long	%ld หรือ %li
	unsigned short	%hu
	unsigned int	%u
	unsigned long	%lu
จำนวนจริง	float	%f
	double	%lf
	long double	%Lf
อักขระ	char	%c
สายอักขระ	char s[ ]	%s

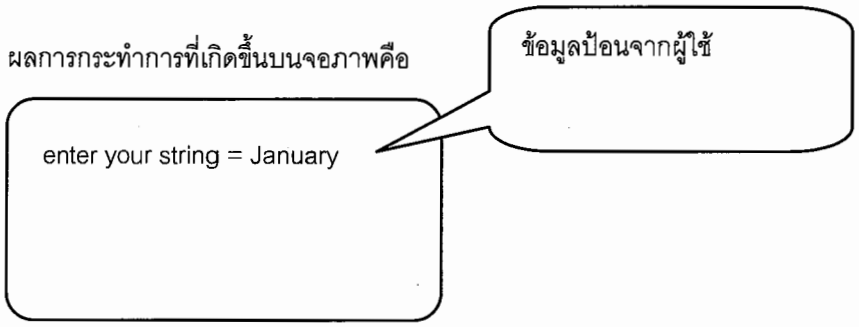
ตารางที่ 2.1 รูปแบบของตัวแปรแบบต่างๆ สำหรับฟังก์ชัน scanf()

ตัวอย่าง 2.1

การรับข้อมูลชนิดอักขระและสายอักขระ

```
#include <stdio.h>
void main()
{
    char a;
    char b[10];
    printf("enter your string = ");
    scanf("%c%s", &a, b); /* กรณีรับข้อมูลเป็นสายอักขระ ไม่ต้องมีเครื่องหมาย & หน้าตัวแปร */
}
```

ผลการกระทำการที่เกิดขึ้นบนจอภาพคือ



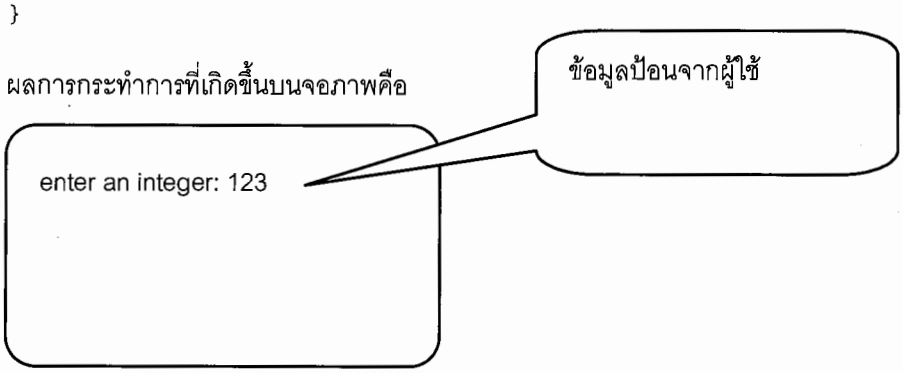
จากตัวอย่างข้างต้นจะเห็นว่าการรับข้อมูลโดยใช้ฟังก์ชัน scanf() สำหรับตัวแปรชนิดอักขระ (a) และตัวแปรชนิดสายอักขระ (b) มีความแตกต่างกันที่ตัวแปรชนิดสายอักขระไม่ต้องมีเครื่องหมาย & หน้าตัวแปร เมื่อโปรแกรมรับข้อมูลจากแป้นพิมพ์แล้ว จะส่งผลให้ตัวแปร a มีค่า 'J' เก็บอยู่ และตัวแปร b[ ] มีค่าเป็น "anuary"

ตัวอย่าง 2.2

การรับข้อมูลชนิดจำนวนเต็ม

```
#include <stdio.h>
void main()
{
    int y;
    printf("enter an integer: ");
    scanf("%d", &y);
}
```

ผลการกระทำการที่เกิดขึ้นบนจอภาพคือ

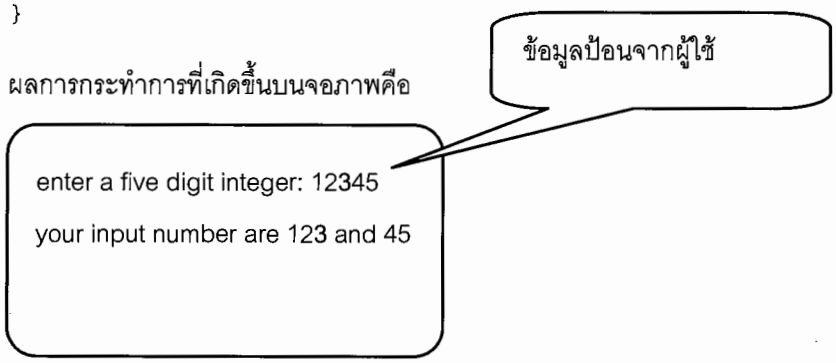


ตัวอย่าง 2.3

การรับข้อมูลชนิดจำนวนเต็มที่มีการระบุจำนวนหลัก

```
#include <stdio.h>
void main()
{
    int y, z;
    printf("enter a five digit integer: ");
    scanf("3d%d", &y, &z);
    printf("your input number are %d and %d\n", y, z);
}
```

ผลการกระทำการที่เกิดขึ้นบนจอภาพคือ



จากผลลัพธ์ที่เกิดขึ้นจะโปรแกรมทำการรับข้อมูลเข้าจากผู้ใช้โดยจะนำตัวเลข 3 ตัวแรกไปเก็บในตัวแปร y และนำตัวเลขที่เหลือไปเก็บในตัวแปร z เมื่อทำการแสดง ค่าของ y และ z โดยใช้ฟังก์ชัน printf( ) จึงได้ผลลัพธ์เป็น 123 และ 45 ตามลำดับ

นอกจากนี้ในการรับข้อมูลที่มีรูปแบบเฉพาะเช่น วันที่ อาจมีรูปแบบในการใส่ข้อมูลเป็น 23 – 9 – 47 สามารถทำได้ดังแสดงในตัวอย่าง

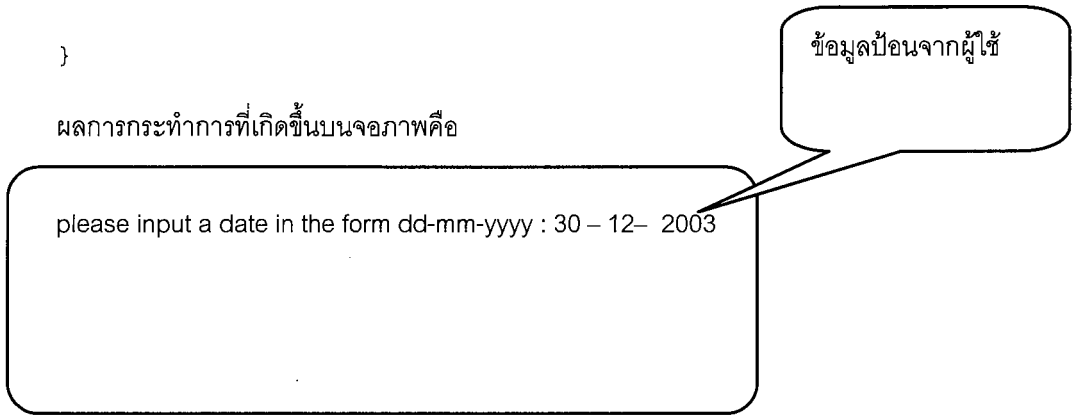
ตัวอย่าง 2.4

การรับข้อมูลรูปแบบวันที่

```
#include <stdio.h>
void main()
{
    int month;
    int day;
    int year;
    printf("please input a date in the form dd-mm-yyyy : ");
    scanf("%d%*c%d%*c%d", &day, &month, &year);

}
```

ผลการกระทำการที่เกิดขึ้นบนจอภาพคือ



จากโปรแกรมข้างต้นผู้ใช้จะใส่ข้อมูลในรูปแบบวันที่ซึ่งจะมีเครื่องหมาย – คั่นระหว่างวันเดือนปี แต่โปรแกรมจะทำการรับข้อมูลเฉพาะส่วนที่เป็นตัวเลขแสดงวันเดือนปีเท่านั้น โดยจะข้ามเครื่องหมาย – เมื่อโปรแกรมกระทำการเสร็จสิ้นลงค่าที่เก็บในตัวแปร day , month และ year คือ 30 , 12 และ 2003 ตามลำดับ

2.2 ฟังก์ชันในการแสดงผลข้อมูลออกทางจอภาพ

เมื่อต้องการแสดงข้อความออกทางจอภาพหรือแสดงผลลัพธ์ที่ได้จากการกระทำการ ภาษาซีได้สร้างฟังก์ชันมาตรฐานเตรียมไว้โดยมีชื่อว่า printf() การใช้งานฟังก์ชันนี้แบ่งเป็น 3 ลักษณะคือ การแสดงเฉพาะข้อความ การแสดงค่าของตัวแปรที่ได้จากการประมวลผล และการแสดงทั้งข้อความและค่าที่เก็บอยู่ในตัวแปร การแสดงข้อความนี้สามารถจัดรูปแบบการขึ้นบรรทัด การเว้นวรรค ได้ดังแสดงในตารางที่ 2.2

รหัสที่ใช้ในคำสั่ง printf	ความหมาย
\n	เป็นการสั่งให้เคอร์เซอร์ขึ้นบรรทัดใหม่
\t	เป็นการสั่งให้เคอร์เซอร์เลื่อนไปทางขวา 1 ช่วงแท็บ
\a	เป็นการสั่งให้เครื่องคอมพิวเตอร์ส่งเสียงระฆัง
\\	เป็นการแสดงสัญลักษณ์ \ ออกทางจอภาพ 1 ตัว
\"	เป็นการแสดงสัญลักษณ์ " ออกทางจอภาพ 1 ตัว

ตารางที่ 2.2 รูปแบบของรหัสคำสั่งสำหรับการจัดรูปแบบโดยใช้ฟังก์ชัน printf()

2.2.1 การแสดงเฉพาะข้อความ

ในการประมวลผลข้อมูลด้วยคอมพิวเตอร์นั้นจำเป็นที่โปรแกรมจะต้องสื่อสารกับผู้ใช้เพื่อให้ทราบถึงวัตถุประสงค์ของโปรแกรมหรือสั่งให้ผู้ใช้เตรียมการป้อนข้อมูล ดังนั้นโปรแกรมที่ดีจึงควรมีการแสดงข้อความต่างๆซึ่งมีรูปแบบคำสั่งดังนี้

```
printf("ข้อความ");
```

เช่น printf("This program will calculate tax\n"); โดยจะมีเครื่องหมายฟันหนู (") ปิดที่หัวและท้ายข้อความและจะสังเกตเห็นว่ามีการใช้สัญลักษณ์ \n ต่อท้ายที่ข้อความ ซึ่งเป็นการให้เคอร์เซอร์ขึ้นบรรทัดใหม่หลังจากแสดงข้อความนี้แล้ว ถ้าต้องการเว้นบรรทัดเพิ่มก็สามารถแสดงสัญลักษณ์ \n เพิ่มเติมได้ และเมื่อต้องการเว้นวรรคหรือแสดงสัญลักษณ์ต่างๆ ก็สามารถใช้รหัสที่แสดงในตารางที่ 2.2

ตัวอย่าง 2.5

โปรแกรมแสดงการแสดงข้อความในรูปแบบต่างๆ

```
#include <stdio.h>
void main()
{
    printf("Welcome to C ");
    printf("programming\n");
    printf("This is my first program\n\n");
    printf("Thank you");
    printf("\nbye bye");
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

```
Welcome to C programming
This is my first program
-บรรทัดว่าง-
Thank you
bye bye
```

จากโปรแกรมในตัวอย่าง 2.5 จะสังเกตได้ว่าฟังก์ชัน printf()จะใช้ในการแสดงข้อความบนจอภาพซึ่งจะมีการขึ้นบรรทัดใหม่เมื่อมีสัญลักษณ \n ต่อท้าย และถ้ามีสัญลักษณ์ \n ติดกันก็จะเว้นบรรทัดเพิ่ม โดยผู้เขียนโปรแกรมสามารถแทรกสัญลักษณ์ \n ไว้ที่ส่วนใดของข้อความก็ได้เพื่อที่จะให้มีการขึ้นบรรทัดใหม่

ตัวอย่าง 2.6

โปรแกรมแสดงการแสดงสัญลักษณ์ " ที่หัวและท้ายคำ"

```
#include <stdio.h>
void main()
{
    printf("Good\t Morning\n ");
    printf("\"new\" programmer\n");
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

```
Good    Morning
"new" programmer
```

จะเห็นว่าผลการกระทำการของโปรแกรมคือ จะเว้นช่วง 1 แท็บระหว่างคำว่า Good และคำว่า Morning และมีการแสดงสัญลักษณ์ฟันหนู (") ที่คำว่า new

ตัวอย่าง 2.7

โปรแกรมแสดงการแสดงสัญลักษณ์ขึ้นบรรทัดใหม่

```
#include <stdio.h>
void main()
{
    printf("1 January \n2 February \n3 March \n4 April \n");
    printf("5 May \n");
}
```

ผลการกระทำการบนจอภาพคือ

```
1 January
2 February
3  March
4  April
5  May
```

2.2.2 การแสดงค่าของตัวแปรที่ได้จากการกระทำการ

เมื่อการกระทำการโปรแกรมเสร็จสิ้นสามารถแสดงผลลัพธ์ที่เก็บอยู่ในตัวแปรได้โดยการระบุรูปแบบที่ต้องการ โดยดูได้จากตารางที่ 2.3 นอกจากนี้ยังสามารถระบุจำนวนตัวเลขหลังจุดทศนิยมที่ต้องการแสดงได้อีกด้วย เช่น printf("%10.2f", amount); ในกรณีที่ต้องการแสดงตัวเลขจากตัวแปรต่างๆในหลายบรรทัดและต้องการจัดรูปแบบให้อยู่ในสดมภ์เดียวกันก็สามารถนำการจัดรูปแบบลักษณะนี้มาใช้ได้เช่นเดียวกันดังตัวอย่าง 2.8

ชนิดข้อมูล	ชนิดตัวแปร	รูปแบบสำหรับ printf
จำนวนเต็ม	short	%d หรือ %i
	integer	%d หรือ %i
	long	%ld หรือ %li
	unsigned short	%u
	unsigned int	%u
	unsigned long	%lu
จำนวนจริง	float	%f
	double	%f
	long double	%Lf
ตัวอักขระ	char	%c
สายอักขระ	char s[ ]	%s

ตารางที่ 2.3 รูปแบบของรหัสคำสั่งสำหรับการจัดรูปแบบโดยใช้ฟังก์ชัน printf()

ตัวอย่าง 2.8

โปรแกรมแสดงตัวเลขในรูปแบบต่างๆ

```
#include <stdio.h>
void main()
{
    int a;
    double b;
    a = 1;
    b = 1040.041;
    printf("%4i = %10.2f\n", a, b);
    a = 2;
    b = 5.05;
    printf("%4i = %10.2f\n", a, b);
```

```
    a = 3;
    b = 1234567.351;
    printf("%4i = %10.2f\n", a, b);
    printf("100%%");     // ใช้สัญลักษณ์ % สองครั้งเมื่อต้องการแสดงสัญลักษณ์ % ทางจอภาพ
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

```
   1 =    1040.04
   2 =       5.05
   3 = 1234567.35
100 %
```

ตัวอย่าง 2.9

การแสดงตัวเลขที่ระบุเฉพาะจำนวนตัวเลขหลังจุดทศนิยม

```
#include <stdio.h>
void main()
{
    float a = 5;
    printf("%.3f\n", a);
    printf("%.4f\n", a);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

```
5.000
5.0000
```

ในการจัดรูปแบบผลลัพธ์ที่จะปรากฏบนจอภาพนั้นจะมีรูปแบบดังนี้

"%	flag	maximum Width	precision	size	conversion code	"

flag มีค่าที่เป็นไปได้คือ - หมายถึงการจัดชิดซ้าย

\+ หมายถึงการกำหนดให้แสดงเครื่องหมาย + หน้าตัวเลข

maximumWidth คือ การกำหนดจำนวนเสดมภ์ที่จองไว้สำหรับการแสดงผลข้อมูล

precision คือการกำหนดจำนวนตัวเลขหลังจุดทศนิยม

size จะสอดคล้องกับชนิดของตัวแปร เช่น h ใช้กับตัวแปรชนิด int ถ้าเป็น l จะใช้กับตัวแปรชนิด double เป็นต้น ดังตารางที่ 2.3

conversion code คือการกำหนดชนิดของข้อมูลในการแสดงผลเช่น

ถ้ามีค่าเป็น o (โอ) หมายถึงการแปลงค่าของข้อมูลให้เป็นเลขฐานแปด

ถ้ามีค่าเป็น x หมายถึงการแปลงค่าของข้อมูลให้เป็นเลขฐานสิบหก

ถ้ามีค่าเป็น d หมายถึงการแปลงค่าของข้อมูลให้เป็นเลขฐานสิบ

ถ้ามีค่าเป็น s หมายถึงการแสดงสายอักขระ

ถ้ามีค่าเป็น e หมายถึงการแสดงตัวเลขให้เป็นรูปแบบวิทยาศาสตร์

ถ้ามีค่าเป็น f หมายถึงการแสดงตัวเลขในรูปแบบตัวเลขจำนวนจริงที่มีจุดทศนิยม

ถ้ามีค่าเป็น p หมายถึงการแสดงค่าที่เก็บในตัวแปรชนิดตัวชี้ (ดูรายละเอียดในบทที่ 7)

นอกจากนี้ในการแสดงตัวเลขที่มีจุดทศนิยม ยังสามารถจัดรูปแบบได้หลายลักษณะ เช่น การแสดงให้อยู่ในรูปตัวเลขแบบวิทยาศาสตร์ เป็นต้น รายละเอียดของการใช้ตัวระบุรูปแบบแสดงในตารางที่ 2.4

ตัวระบุรูปแบบ	ความหมาย
e หรือ E	แสดงตัวเลขในรูปแบบวิทยาศาสตร์ เช่น 1.357911e+10
f	แสดงตัวเลขในแบบทศนิยม เช่น 2.4567
g หรือ G	แสดงตัวเลขในรูปแบบวิทยาศาสตร์หรือแสดงตัวเลขในแบบทศนิยม
L	ใส่ไว้หน้าตัวระบุรูปแบบ f เพื่อเป็นการแสดงถึงการแสดงเลขแบบ long double

ตารางที่ 2.4 การจัดรูปแบบตัวเลขที่มีจุดทศนิยม

ตัวอย่าง 2.10

การจัดรูปแบบจำนวนจริง

```
#include <stdio.h>
void main()
{
    printf("%e\n", 9876543.21);
    printf("%e\n", +9876543.21);
    printf("%e\n", -9876543.21);
    printf("%E\n", -9876543.21);
    printf("%f\n", 9876543.21);
    printf("%g\n", 9876543.21);
    printf("%G\n", 9876543.21);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

9	·	8	7	6	5	4	3	e	+	0	0	6			
9	·	8	7	6	5	4	3	e	+	0	0	6			
-	9	·	8	7	6	5	4	3	e	+	0	0	6		
9	·	8	7	6	5	4	3	E	+	0	0	6			
9	8	7	6	5	4	3	.	2	1						
9	.	8	7	6	5	4	e	+	0	0	6				
9	.	8	7	6	5	4	E	+	0	0	6				

ตัวอย่าง 2.11

การจัดรูปแบบจำนวนเต็ม

```
#include <stdio.h>
void main()
{
    printf("%5d\n", 9);
    printf("%5d\n", 98);
    printf("%5d\n", 987);
    printf("%5d\n", 9876);
    printf("%5d\n", 98765);
    printf("%5d\n", 987654);
    printf("%5d\n", -9);
    printf("%5d\n", -98);
    printf("%5d\n", -987);
    printf("%5d\n", -9876);
    printf("%5d\n", -98765);
    printf("%5d\n", -987654);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

				9										
			9	8										
		9	8	7										
	9	8	7	6										
9	8	7	6	5	4									
			-	9										
		-	9	8										
	-	9	8	7										
-	9	8	7	6										
-	9	8	7	6	5									
-	9	8	7	6	5	4								

ตัวอย่าง 2.12

แสดงสายอักขระหรือข้อมูลในแถวลำดับ

```
#include <stdio.h>
void main()
{
    char string[] = "This string is printed";
    char alpha = 'a';
    printf("%s\n", string);
    printf("%c\n", alpha);
    printf("%24s\n", string);
    printf("7s\n", string);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

T	h	i	s		s	t	r	i	n	g		i	s		p	r	i	n	t	e	d		
a																								
		T	h	i	s		s	t	r	i	n	g		i	s		p	r	i	n	t	e	d
T	h	i	s		s	t																		

จากผลการกระทำการข้างต้นจะสังเกตได้ว่าในการแสดงสายอักขระนั้น ถ้ามีการกำหนดขนาดที่น้อยกว่าความยาวของสายอักขระจะทำให้ไม่สามารถแสดงสายอักขระได้ครบทั้งหมด ซึ่งแตกต่างจากการแสดงตัวเลข และถ้ากำหนดความยาวเกินกว่าขนาดของสายอักขระโปรแกรมจะแสดงชิดขวา

ตัวอย่าง 2.13

การจัดรูปแบบโดยใช้เครื่องหมาย + เพื่อแสดงเครื่องหมายหน้าตัวเลข

```
#include <stdio.h>
void main()
{
    printf("%+08d\n", 999);
    printf("%08d\n", 999);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

+	0	0	0	0	9	9	9						
0	0	0	0	0	9	9	9						

ตัวอย่าง 2.14

การจัดรูปแบบชิดซ้ายและชิดขวา

```
#include <stdio.h>
void main()
{
    printf("%10s%10d%10c\n", "good", 3, 'x');
    // ใช้เครื่องหมาย - เมื่อต้องการจัดชิดซ้าย
    printf("%-10s%-10d%-10c%\n", "good", 3, 'x');
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

						g	o	o	d									3								x													
g	o	o	d							3									x																				

ตัวอย่าง 2.15

การแสดงค่าของตัวแปรในรูปแบบเลขฐานแปดและฐานสิบหก

```
#include <stdio.h>
void main()
{
    printf("%o\n",455);          // แสดงในรูปแบบเลขฐานแปด
    printf("%x\n",455);          // แสดงในรูปแบบเลขฐานสิบหก
    printf("%d\n",455);          // แสดงในรูปแบบเลขฐานสิบ
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

7	0	7												
1	c	7												
4	5	5												

นอกจากการใช้ฟังก์ชัน printf() ในการแสดงข้อความแล้ว ภาษาซียังได้จัดเตรียมฟังก์ชันที่ใช้ในการแสดงสายอักขระชื่อ puts() โดยมีรูปแบบการเรียกใช้ดังนี้

puts("ข้อความ");

หรือ **puts(ตัวแปรชนิดตัวชี้ที่ชี้ไปยังข้อมูลชนิดตัวอักขระ);**

ตัวอย่าง 2.16

การใช้ฟังก์ชัน gets() และ puts() ในการแสดงผลข้อความ

```
#include <stdio.h>
void main()
{
    char input[50];
    puts("Please enter sentence :");
    gets(input);        /* ฟังก์ชัน gets() ทำหน้าที่รับข้อมูลเข้าผ่านทางแป้นพิมพ์มาเก็บไว้ใน
                           ตัวแปร */
    puts("This is an output\n");
    puts(input);
}
```

ข้อมูลป้อนเข้าจากผู้ใช้

ผลการกระทำการบนจอภาพคือ

```
Please enter sentence : hello
This is an output
hello
```

2.2.3 การแสดงทั้งข้อความและค่าที่เก็บในตัวแปร

ฟังก์ชัน printf() สามารถใช้ในการแสดงทั้งข้อความและค่าที่เก็บตัวแปรได้

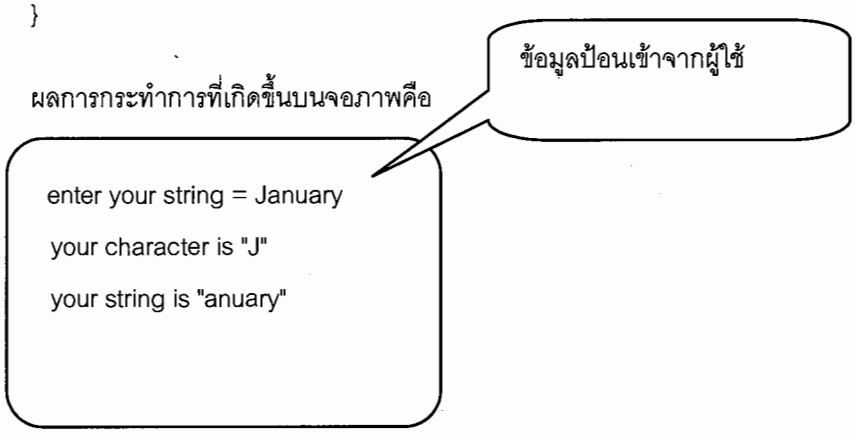
เช่น printf("The value of b = %10.2f",b); ดูตัวอย่างโปรแกรมด้านล่าง

ตัวอย่าง 2.17

การรับข้อมูลชนิดตัวอักขระและสายอักขระ

```
#include <stdio.h>
void main()
{
    char a;
    char b[10];
    printf("enter your string = ");
    scanf("%c%s", &a, b); /* กรณีรับข้อมูลเป็นสายอักขระ ไม่ต้องมีเครื่องหมาย & หน้าตัวแปร */
    printf("your character is \"%c\" ", a);
    printf("your string is \"%s\"\n", b);
}
```

ผลการกระทำการที่เกิดขึ้นบนจอภาพคือ



ตัวอย่าง 2.18

การใช้ฟังก์ชัน printf() ในการแสดงทั้งข้อความและค่าที่เก็บในตัวแปร

```
#include <stdio.h>
#define amount 5000.00
#define rate 0.06
#define period 7
void main()
{
    printf("rate:   %7.2f%%\n", rate);
    printf("amount: %7.2f\n\n", amount);
    printf("period:  %7i years", period);
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ

r	a	t	e	:			0	.	0	6	%					
a	m	o	u	n	t	:	5	0	0	0	.	0	0			
p	e	r	i	o	d	:						7	y	e	a	r	s

ตัวอย่าง 2.19

การรับและแสดงผลข้อมูลโดยใช้ฟังก์ชัน scanf() และ printf()

```
#include <stdio.h>
void main()
{
    char d1,d2,d3,d4;
    int x;
    scanf("%d%c%c%c%c", &x, &d1, &d2, &d3, &d4);
    printf("%d%c%c%c%c", x, d1, d3, d2, d4);
}
```

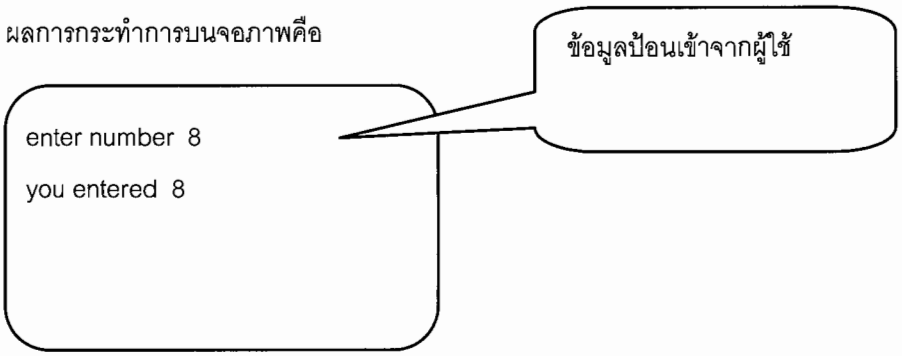
จากโปรแกรมในตัวอย่าง 2.19 ถ้าผู้ใช้ป้อนข้อมูล 97 354 ฟังก์ชัน scanf() จะนำข้อมูลเลข 97 ไปเก็บไว้ในตัวแปร x นำตัวอักขระช่องว่างไปเก็บไว้ในตัวแปร d1 นำตัวอักขระ 3 ไปเก็บไว้ในตัวแปร d2 นำตัวอักขระ 5 ไปเก็บไว้ในตัวแปร d3 และนำตัวอักขระ 4 ไปเก็บไว้ในตัวแปร d4 และเมื่อโปรแกรมทำงานถึงฟังก์ชัน printf() โดยจะทำการแสดงค่าของตัวแปรแต่ละตัวตามลำดับดังนี้ 97 534

ตัวอย่าง 2.20

การรับและแสดงผลตัวแปรชนิด unsigned

```
#include <stdio.h>
void main()
{
    unsigned x;
    printf("enter number ");
    scanf("%u", &x);
    printf("you entered %u", x);
}
```

ผลการกระทำการบนจอภาพคือ

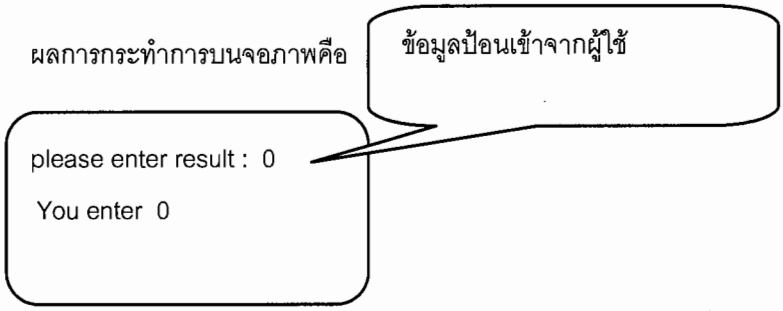


ตัวอย่าง 2.21

การรับและแสดงผลตัวแปรชนิด long

```
#include <stdio.h>
void main()
{
    long result;
    printf("please enter result : ");
    scanf("%ld", &result);
    printf("You enter %ld \n", result);
}
```

ผลการกระทำการบนจอภาพคือ



นอกจากนี้การรับข้อมูลชนิดอักขระ (char) สามารถใช้ฟังก์ชัน getchar() และใช้ฟังก์ชัน putchar() ในการแสดงผลบนจอภาพ โดยมีรูปแบบดังนี้

ตัวแปร = getchar();

เช่น data = getchar(); จะเป็นการรับข้อมูล 1 ตัวอักขระมาเก็บไว้ในตัวแปร data ส่วนการแสดงผลออกทางจอภาพจะใช้ ฟังก์ชัน putchar ซึ่งมีรูปแบบดังนี้

putchar(ตัวแปร);

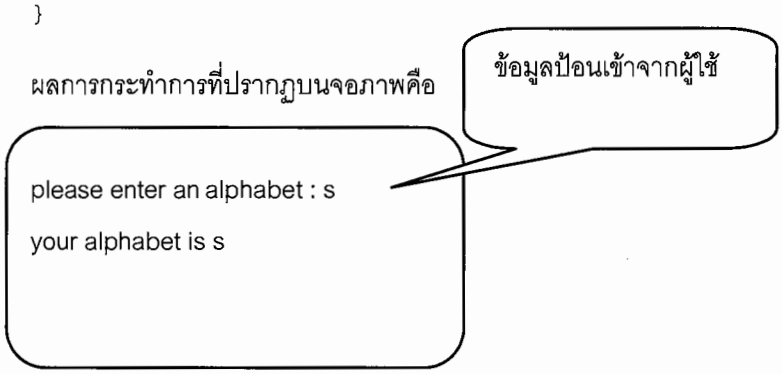
เช่น putchar(data);

ตัวอย่าง 2.22

แสดงการใช้ฟังก์ชัน getchar() และ putchar()

```
#include <stdio.h>
void main()
{
    char c;
    printf("please enter an alphabet : ");
    c = getchar();
    printf("your alphabet is ");
    putchar( c );
}
```

ผลการกระทำการที่ปรากฏบนจอภาพคือ



โดยสรุปแล้วจะพบว่าการจัดรูปแบบการรับและแสดงผลข้อมูลสามารถทำได้โดยนำรหัสตัวอักขระที่ภาษาซีได้กำหนดให้มาจัดรูปแบบภายใต้เครื่องหมายฟันหนูและตามด้วยเครื่องหมายเปอร์เซนต์ หลังจากนั้นโปรแกรมจะสามารถแสดงผลลัพธ์ตามรูปแบบที่ถูกกำหนด

แบบฝึกหัดบทที่ 2

1. จงเขียนฟังก์ชัน printf() เพื่อจัดรูปแบบตัวเลขให้มีผลการกระทำการดังแสดงในรูป

```
   4
  45
  87
 98
78
  -1
 -198
 -98
-9
-98
-980054
```

2. จงเขียนฟังก์ชัน printf() เพื่อจัดรูปแบบสายอักขระให้มีผลการกระทำการดังแสดงในรูป
โดยกำหนดตัวแปรคือ char data[ ] = "basic operation"

```
basic operation
basic
          basic operation
```

3. กำหนดค่าของตัวแปรดังนี้

```
#define commission   5000.00
#define percent      0.05
#define no_of_day    7
```

จงเขียนโปรแกรมเพื่อแสดงข้อความและตัวแปรให้มีผลการกระทำการดังรูป

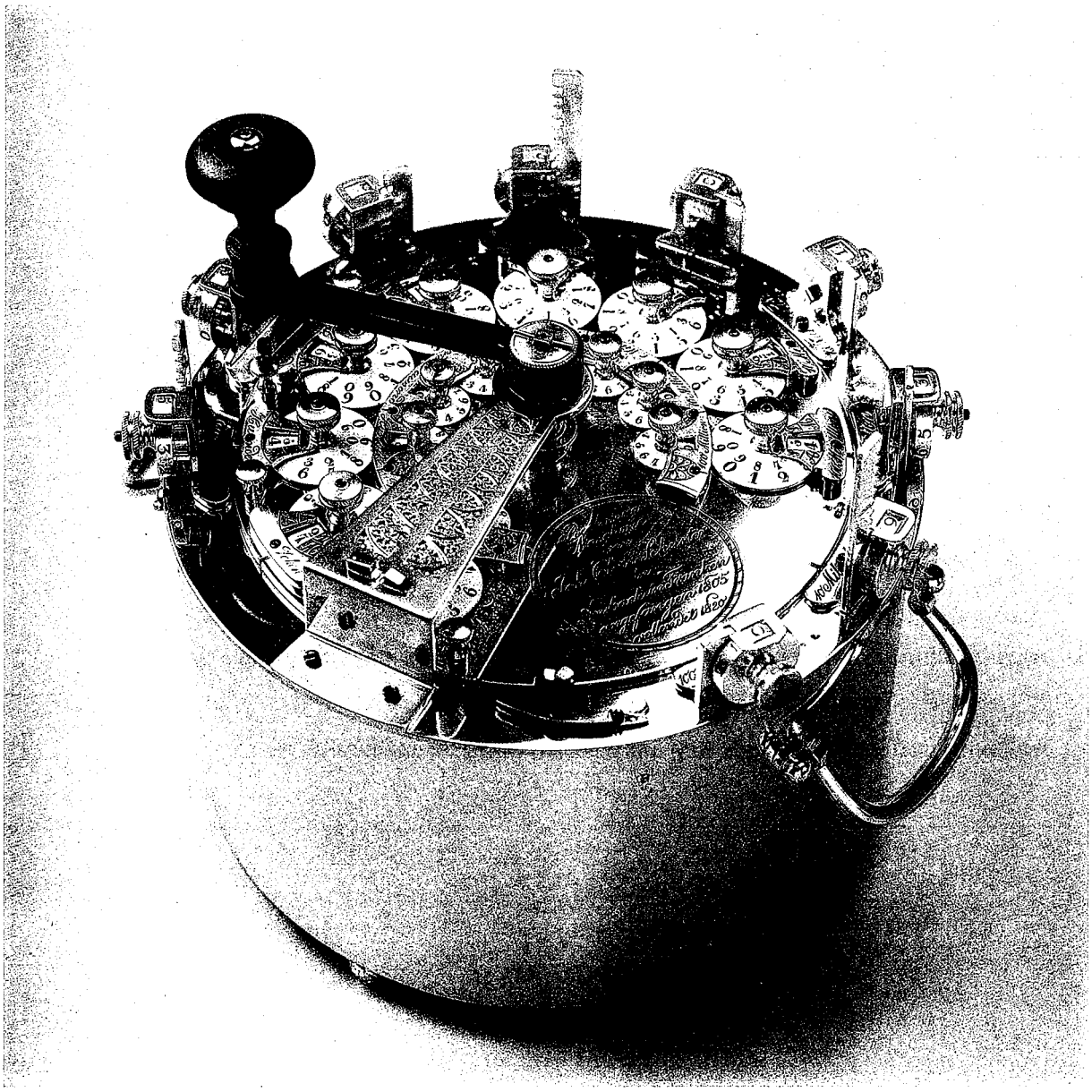
```
rate :    0.05%
comm   :5000.00

no.of.day     7days
```

4. จงเขียนโปรแกรมเพื่อแสดงค่าของจำนวน 500 ในรูปแบบเลขฐานแปด และ ฐานสิบหก ออกทางจอภาพ

5. จงเขียนโปรแกรมโดยใช้ฟังก์ชัน gets() และ puts()เพื่อรับและแสดงผลสายอักขระ "Olympic Games"

6. จงเขียนโปรแกรมโดยใช้ฟังก์ชัน getchar() และ putchar() เพื่อรับและแสดงผลตัวอักษร 'a'



เครื่องคำนวณ ประดิษฐ์โดย Johann Christoph Schuster ค.ศ.1820

บทที่ 3
การควบคุมโปรแกรม

โดยทั่วไปแล้วลำดับการกระทำการในโปรแกรมจะเป็นแบบบนลงล่าง (top-down) โดยปริยาย การกระทำการเริ่มต้นที่ฟังก์ชัน main() แล้วกระทำทีละคำสั่งตั้งแต่คำสั่งแรกจนถึงคำสั่งสุดท้ายตามลำดับ จนกระทั่งจบฟังก์ชัน แต่โดยทั่วไปแล้วลำดับการกระทำการดังกล่าวพบได้น้อยมาก ส่วนใหญ่ในฟังก์ชันจะประกอบด้วยข้อความสั่งควบคุม เพื่อควบคุมลำดับการกระทำการภายในโปรแกรม โดยอาจให้มีการเลือกทำเพียงบางข้อความสั่ง หรือ ให้ทำบางข้อความสั่งหลายครั้ง ดังนั้นจะเห็นได้ว่า ข้อความสั่งควบคุมโปรแกรมจะช่วยเพิ่มขีดความสามารถในการกระทำการให้มีประสิทธิภาพ ทำให้โปรแกรมสามารถทำงานได้หลากหลายรูปแบบ อาทิ สามารถกำหนดเงื่อนไขให้โปรแกรมสามารถเลือกกระทำการหรือไม่กระทำการบางข้อความสั่ง ในโปรแกรมภาษาซีมีข้อความสั่งที่นักเขียนโปรแกรมสามารถใช้เพื่อควบคุมลำดับของการกระทำการโปรแกรมหลายชนิด ซึ่งจะได้อธิบายถึงรายละเอียดในบทนี้

3.1 ตัวดำเนินการสัมพันธ์ (relational operator)

ตัวดำเนินการสัมพันธ์ ใช้สำหรับเปรียบเทียบนิพจน์ ผลของการเปรียบเทียบนิพจน์ที่มีตัวดำเนินการสัมพันธ์จะให้ผลลัพธ์เป็นจริงหรือเท็จ

ตัวดำเนินการสัมพันธ์ มีวิธีการใช้ดังนี้

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
==	เท่ากับ	x == y
>	มากกว่า	x > y
<	น้อยกว่า	x < y
>=	มากกว่า หรือ เท่ากับ	x >= y
<=	น้อยกว่า หรือ เท่ากับ	x <= y
!=	ไม่เท่ากับ	x != y

สำหรับนิพจน์ที่มีการใช้ตัวดำเนินการสัมพันธ์ เมื่อการเปรียบเทียบได้ผลลัพธ์เท่ากับศูนย์จะแปลความหมายว่าเท็จ แต่หากผลลัพธ์เท่ากับหนึ่ง จะแปลความหมายว่าจริง

ตัวอย่าง 3.1

การแปลความหมายของนิพจน์ เมื่อกำหนดให้ x = 5 , y = 10

นิพจน์	ผลลัพธ์	การแปลความหมาย
x ==y	0	เท็จ
x >y	0	เท็จ
x>=y	0	เท็จ
x<=y	1	จริง
x!=y	1	จริง
x*x < y*y	1	จริง
x+y >= x*y	0	เท็จ

3.2 ตัวดำเนินการตรรกะ (logical operator)

ตัวดำเนินการตรรกะใช้สำหรับเชื่อมโยงความสัมพันธ์ระหว่างนิพจน์ตรรกะตั้งแต่ 2 นิพจน์ขึ้นไป ผลจากการเชื่อมโยงความสัมพันธ์ดังกล่าว จะให้ผลลัพธ์เป็นจริงหรือเท็จ

ตัวดำเนินการตรรกะมีวิธีการใช้ดังนี้

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&&	**และ** (and)	x == y
||	**หรือ** (or)	x || y
!	**ไม่** (not)	! x

การทำงานของตัวดำเนินการและ(and) ตัวดำเนินการหรือ (or) และ ตัวดำเนินการไม่ (not)

x	y	นิพจน์	ผลลัพธ์	การแปลความหมาย
nonzero (จริง)	nonzero (จริง)	x && y	1	จริง
nonzero (จริง)	0 (เท็จ)	x && y	0	เท็จ
0 (เท็จ)	nonzero (จริง)	x && y	0	เท็จ
0 (เท็จ)	0 (เท็จ)	x && y	0	เท็จ
nonzero (จริง)	nonzero (จริง)	x || y	1	จริง
nonzero (จริง)	0 (เท็จ)	x || y	1	จริง
0 (เท็จ)	nonzero (จริง)	x || y	1	จริง
0 (เท็จ)	0 (เท็จ)	x || y	0	เท็จ
nonzero (จริง)		!x	0	เท็จ
0 (เท็จ)		!x	1	จริง

3.3 ลำดับในการดำเนินการ

ในนิพจน์ที่ประกอบด้วยตัวดำเนินการต่างๆหลายชนิด จะต้องดำเนินการตามลำดับดังต่อไปนี้

1. ()
2. ++ -- + - ! (ตัวดำเนินการเอกภาค)
3. * / %
4. + -
5. < <= > >=
6. == !=
7. &&
8. ||
9. = += -= *= /= %=

ในกรณีที่นิพจน์มีตัวดำเนินการที่อยู่ในลำดับเดียวกันหลายตัว จะต้องดำเนินการตัวดำเนินการที่อยู่ด้านซ้ายของนิพจน์ก่อน

ตัวอย่าง 3.2

นิพจน์	การดำเนินการ	ผลลัพธ์	การแปลความหมาย
'e' + 1 == 'f'	('e' + 1) == 'f'	1	จริง
3 < k + 3 && 0	(3 < (k + 3)) && 0	0	เท็จ
20 * 3 / 5 || 0 && 5	((20 * 3) / 5) || (0 && 5)	1	จริง

3.4 ข้อความสั่งให้เลือกทำ

3.4.1 ข้อความสั่ง if

ข้อความสั่ง if เป็นข้อความสั่งที่ให้ตรวจสอบผลลัพธ์จากนิพจน์ ถ้าผลลัพธ์เป็นจริงให้ทำงานตามข้อความสั่งที่อยู่ภายในข้อความสั่ง if

ข้อความสั่ง if รูปแบบที่ 1

```
if(นิพจน์)
    ข้อความสั่ง A;
```

ใช้สำหรับกรณีที่ข้อความสั่งภายในข้อความสั่ง if มีเพียงข้อความสั่งเดียว ถ้านิพจน์ให้ผลลัพธ์เป็นจริง จะกระทำการข้อความสั่ง A แต่ถ้านิพจน์เป็นเท็จ จะไม่กระทำการข้อความสั่ง A แต่จะจบการทำงานในข้อความสั่ง if แล้วไปกระทำการข้อความสั่งที่อยู่ถัดไป

ตัวอย่าง 3.3

การใช้ข้อความสั่ง if เพื่อเปรียบเทียบค่า 2 ค่า

```
#include <stdio.h>
void main()
{
      int x, y, z;
      y = 1;
      z = 5;
      x = y * z;
      if(x == 5)                  // เปรียบเทียบ x มีค่าเท่ากับ 5 หรือไม่
          x++;                    // ถ้า x มีค่าเท่ากับ 5  เพิ่มค่า x ขึ้นอีก 1
      printf("x = %d", x);
}
```

ผลการกระทำการ

x = 6

ตัวอย่าง 3.4

การใช้ข้อความสั่ง if เพื่อเปรียบเทียบค่า 2 ค่า

```
#include <stdio.h>
void main()
{
      int x;
      x = 7;
      if(x != 3)                  // เปรียบเทียบค่าของ x และ 3
           x += 3;                // ถ้า x ไม่เท่ากับ 3  จะเพิ่มค่า x อีก 3
      printf("x = %d", x);
}
```

ผลการกระทำการ

x = 10

ตัวอย่าง 3.5

การใช้ข้อความสั่ง if

```
#include <stdio.h>
void main()
{
      int age;
      float income;
      scanf("%f%d", &income, &age);
      if(income >= 20000 && age <= 30)
          printf("\n income >= 20000  baht and age <= 30 years");
}
```

ถ้าค่าที่เก็บใน income มีค่ามากกว่าหรือเท่ากับ 20000 และ ค่าที่เก็บใน age มีค่าน้อยกว่าหรือเท่ากับ 30 เป็นจริง ให้แสดงข้อความ income >= 20000 baht and age <= 30 years

แต่ถ้าเงื่อนไขใดเงื่อนไขหนึ่ง หรือ ทั้งสองเงื่อนไขดังกล่าวเป็นเท็จ ฟังก์ชัน printf("\n income >= 20000 baht and age <= 30 years"); จะไม่ถูกกระทำ

ตัวอย่าง 3.6

การใช้ข้อความสั่ง if เพื่อแสดงระดับคะแนน

```
#include <stdio.h>
void main()
{
      float lecture, average, lab;
      scanf("%f%f%f", &lecture, &average, &lab);
      if(lecture > average && lab >  50)
          printf("\n Grade = G");
      if(lecture >= average || lab >=  50)
          printf("\n Grade = P");
      if(lecture < average || lab <  50)
          printf("\n Grade = F");
      if(lecture > 100 || average > 100 || lab > 100)
          printf("\n Grade is not valid.");
}
```

ถ้าค่าที่เก็บใน lecture มีค่ามากกว่าค่าที่เก็บใน average และ ค่าที่เก็บใน lab มีค่ามากกว่า 50 ขึ้นไป ให้แสดงข้อความ Grade = G

ถ้าค่าที่เก็บใน lecture มีค่ามากกว่าหรือเท่ากับค่าที่เก็บใน average หรือ ค่าที่เก็บใน lab มีค่าตั้งแต่ 50 ขึ้นไป ให้แสดงข้อความ Grade = P

ถ้าค่าที่เก็บใน lecture มีค่าน้อยกว่าค่าที่เก็บใน average หรือ ค่าที่เก็บใน lab มีค่าน้อยกว่า 50 ให้แสดงข้อความ Grade = F

ถ้าค่าที่เก็บใน lecture หรือ average หรือ lab ค่าใดค่าหนึ่งมีค่ามากกว่า 100 ให้แสดงข้อความ Grade is not valid.

ตัวอย่าง 3.7

การใช้ข้อความสั่ง if เพื่อหาค่าสูงสุด

```
#include <stdio.h>
void main()
{
      int a, b, c, max;
      printf(Enter three integer : ");
      scanf("%d%d%d", &a, &b, &c);
      max = c;
      if(a > max)
           max = a;
      if(b > max)
           max = b;
      printf("Maximum is : %d\n", max);
}
```

ให้ป้อนค่า 3 ค่า ไปเก็บไว้ใน a, b, และ c ตามลำดับ

กำหนดให้ค่าที่เก็บไว้ใน max มีค่าเท่ากับค่าที่เก็บไว้ใน c

ถ้า ค่าที่เก็บใน a มีค่ามากกว่าค่าที่เก็บใน max ให้นำค่าที่เก็บใน a ไปไว้ใน max

ถ้า ค่าที่เก็บใน b มีค่ามากกว่าค่าที่เก็บใน max ให้นำค่าที่เก็บใน b ไปไว้ใน max

แล้วให้แสดงข้อความ Maximum is และแสดงค่าที่เก็บใน max

ข้อความสั่ง if รูปแบบที่ 2

```
if (นิพจน์)
{
    ข้อความสั่ง 1;
    ข้อความสั่ง 2;
    ...
    ข้อความสั่ง n;
}
```

ใช้สำหรับกรณีที่ข้อความสั่งภายในข้อความสั่ง if มีมากกว่า 1 ข้อความสั่ง ข้อความสั่งเหล่านั้นจะต้องอยู่ภายในเครื่องหมายปีกกา ({ }) ถ้านิพจน์ให้ผลลัพธ์เป็นจริง จะกระทำการข้อความสั่ง 1 ถึงข้อความสั่ง n แต่ถ้านิพจน์เป็นเท็จจะไม่กระทำการข้อความสั่ง 1 ถึงข้อความสั่ง n

ตัวอย่าง 3.8

การใช้ข้อความสั่ง if

```
#include <stdio.h>
void main()
{
      int x, y;
      scanf("%d%d", &x, &y);
      if(x > y)
      {
           x = x - y;
           printf("x is greater than y");
      }
}
```

รับค่าจำนวนเต็ม 2 ค่า นำมาเก็บไว้ใน x และ y
เปรียบเทียบค่า x และ y ถ้าค่าที่เก็บใน x มากกว่าค่าที่เก็บใน y
จะนำผลลัพธ์ของ x – y ไปเก็บไว้ใน x และแสดงข้อความ x is greater than y
แต่ถ้าค่าที่เก็บใน x น้อยกว่าหรือเท่ากับค่าที่เก็บใน y
จะไม่กระทำการข้อความสั่ง x = x – y; และฟังก์ชัน printf("x is greater than y");

3.4.2 ข้อความสั่ง if-else

ข้อความสั่ง if-else เป็นข้อความสั่งที่ให้ตรวจสอบผลลัพธ์จากนิพจน์ ถ้าผลลัพธ์เป็นจริงให้ทำงานตามข้อความสั่งที่อยู่ภายในข้อความสั่ง if แต่ไม่กระทำการข้อความสั่งที่อยู่ภายใน else หากผลลัพธ์เป็นเท็จ จะไม่กระทำการข้อความสั่งที่อยู่ภายในข้อความสั่ง if แต่จะกระทำการข้อความสั่งที่อยู่ภายใน else

ข้อความสั่ง if-else รูปแบบที่ 1

```
if (นิพจน์)
    ข้อความสั่ง A;
else
    ข้อความสั่ง B;
```

ใช้สำหรับกรณีที่ข้อความสั่งภายในข้อความสั่ง if ที่จะถูกกระทำการมีเพียงข้อความสั่งเดียว
ถ้านิพจน์เป็นจริง จะกระทำการข้อความสั่ง A แต่ถ้านิพจน์เป็นเท็จ จะกระทำการข้อความสั่ง B
ข้อความสั่ง A หรือ ข้อความสั่ง B เพียงข้อความสั่งเดียวเท่านั้นที่จะถูกกระทำ เมื่อนิพจน์เป็นจริง หรือ เท็จ

ตัวอย่าง 3.9

การใช้ข้อความสั่ง if-else

```
#include <stdio.h>
void main()
{
      int x, y;
      scanf("%d%d", &x, &y)
      if(x > y)
            printf("x is greater than y");
      else
            printf("x is less than or equal to y");
}
```

รับค่าจำนวนเต็ม 2 ค่า นำมาเก็บไว้ใน x และ y
เปรียบเทียบค่า x และ y
ถ้าค่าที่เก็บใน x มากกว่าค่าที่เก็บใน y จะแสดงข้อความ x is greater than y
แต่ถ้าค่าที่เก็บใน x น้อยกว่าหรือเท่ากับค่าที่เก็บใน y
จะแสดงข้อความ x is less than or equal to y

ข้อความสั่ง if-else **รูปแบบที่** 2

```
if (นิพจน์)
{
        ข้อความสั่ง 1;
        ข้อความสั่ง 2;
        ...
        ข้อความสั่ง n;
}
else
{
        ข้อความสั่ง n + 1 ;
        ข้อความสั่ง n + 2;
        ...
        ข้อความสั่ง m;
}
```

ใช้สำหรับกรณีที่ข้อความสั่งที่จะถูกกระทำมีมากกว่า 1 ข้อความสั่ง เมื่อนิพจน์เป็นจริง กลุ่มของข้อความสั่งที่จะถูกกระทำต้องอยู่ภายในเครื่องหมายปีกกา หรือกรณีที่ข้อความสั่งที่จะถูกกระทำมีมากกว่า 1 ข้อความสั่ง เมื่อนิพจน์เป็นเท็จ กลุ่มของข้อความสั่งที่จะถูกกระทำต้องอยู่ภายในเครื่องหมายปีกกา

ถ้านิพจน์ให้ผลลัพธ์เป็นจริง จะกระทำการข้อความสั่ง 1 ถึงข้อความสั่ง n แต่ถ้านิพจน์เป็นเท็จ จะกระทำการข้อความสั่ง n+1 ถึงข้อความสั่ง m

ตัวอย่าง 3.10

การใช้ข้อความสั่ง if-else เพื่อคำนวณระดับคะแนน และนับจำนวนของนิสิตในแต่ละระดับคะแนน

```
#include <stdio.h>
void main()
{
      int x, p, f;
      char grade;
      p = 0;
      f = 0;
      scanf("%d", &x);
      if(x >= 50)
      {
           grade = 'P';
           p++;
      }
      else
      {
            grade = 'F';
            f++;
      }
}
```

ถ้า x มีค่ามากกว่าหรือเท่ากับ 50 เป็นจริง จะนำค่าคงที่ P ไปเก็บไว้ใน grade แล้วเพิ่มค่า p ขึ้นอีกหนึ่ง แต่ถ้า x มีค่าน้อยกว่า 50 จะนำค่าคงที่ F ไปเก็บไว้ใน grade แล้วเพิ่มค่า f ขึ้นอีกหนึ่ง

3.4.3 ข้อความสั่ง if ซ้อน

ข้อความสั่ง if ซ้อน หมายถึง การนำข้อความสั่ง if หรือ if-else ไปใส่ไว้ภายในข้อความสั่ง if หรือ if-else ซึ่งข้อความสั่งที่อยู่ภายในนี้ อาจเป็นข้อความสั่ง if หรือ if-else ก็ได้ ข้อความสั่ง if หรือ if-else ที่ซ้อนอยู่ภายใน อาจจะอยู่หลัง if หรือหลัง else ก็ได้ และอาจซ้อนกันได้โดยไม่จำกัดจำนวนข้อความสั่ง

ข้อความสั่ง if ซ้อนรูปแบบที่ 1 ข้อความสั่ง if-else อยู่หลัง if

```
if (นิพจน์ 1)                    // 1
    if (นิพจน์ 2)                // 2
        if ...                   // 3
            ... ;                // 4
        else ...                 // 5
                ... ;            // 6
    else ...                     // 7
            ... ;                // 8
else ...                         // 9
    ... ;                        // 10
```

ถ้านิพจน์ 1 ให้ผลลัพธ์เป็นจริง จะกระทำการข้อความสั่ง if(นิพจน์ 2) ในบรรทัดที่ 2
ในการกระทำการบรรทัดที่ 2 ถ้านิพจน์ 2 ให้ผลลัพธ์เป็นจริง จะกระทำการข้อความสั่ง if ในบรรทัดที่ 3
แต่ถ้านิพจน์ 2 ให้ผลลัพธ์เป็นเท็จ จะกระทำการข้อความสั่งหลัง else ในบรรทัดที่ 7
แต่ถ้านิพจน์ 1 ให้ผลลัพธ์เป็นเท็จ จะกระทำการข้อความสั่งหลัง else ในบรรทัดที่ 9 และ 10

ตัวอย่าง 3.11

การใช้ข้อความสั่ง if ซ้อน if

```
#include <stdio.h>                                          // 1
void main()                                                 // 2
{
      float  x, y, z;                                       // 3
      scanf("%f%f", &x, &y);                                // 4
      if(x > 0)                                             // 5
            if(y != 0)                                      // 6
            {
                  z = x / y;                                // 7
                  printf("\n  %f / %f  =  %f ", x, y, z);   // 8
            }
}                                                           // 9
```

บรรทัดที่ 4 รับค่า 2 ค่าจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน x และ y ตามลำดับ

บรรทัดที่ 5 ถ้า x > 0 เป็นจริง จะไปที่บรรทัดที่ 6

แต่ถ้า x > 0 เป็นเท็จ จะไปที่บรรทัดที่ 9

ซึ่งจะกระทำการตามข้อความสั่งที่อยู่ถัดจากข้อความสั่ง if

โดยที่ข้อความสั่ง z = x / y; และ printf("\n %f / %f = %f ", x, y, z); จะไม่ถูกกระทำ

บรรทัดที่ 6 ตรวจสอบว่า y ไม่เท่ากับศูนย์ เป็นจริงหรือไม่ ถ้าเป็นจริง ไปที่บรรทัดที่ 7

บรรทัดที่ 7 กำหนดให้นำผลลัพธ์ของ x / y เก็บไว้ใน z

บรรทัดที่ 8 แสดงค่า x, y, และ z

ตัวอย่าง 3.12

การใช้ข้อความสั่ง if-else

```
#include <stdio.h>
void main()
{
      float  x, y, z;
      scanf("%f%f", &x, &y);
      if(x > 0)
            if(y != 0)
            {
                  z = x / y;
                  printf("z = %f", z);
            }
            else
                  printf("can't be divided by zero");
      else
      {
            z = x * y;
            printf("z = %f", z);
      }
}
```

ถ้า x > 0 เป็นจริง จะไปตรวจสอบว่า y ไม่เท่ากับศูนย์ เป็นจริงหรือไม่ ถ้าเป็นจริง จะกระทำการข้อความสั่ง z = x / y; และฟังก์ชัน printf("z = %f", z); แต่ถ้า y ไม่เท่ากับศูนย์ เป็นเท็จ จะไปกระทำการฟังก์ชัน printf("can't be divided by zero");

แต่ถ้า x > 0 เป็นเท็จ จะกระทำการที่ข้อความสั่ง else ซึ่งประกอบด้วยข้อความสั่ง z = x * y; และ printf ("z = %f", z); โดยที่ข้อความสั่ง z = x / y; และฟังก์ชัน printf("can't be divided by zero"); จะไม่ถูกกระทำ

ข้อความสั่ง if-else **รูปแบบที่** 2 ข้อความสั่ง if-else อยู่หลัง else

```
if (นิพจน์ 1)
    ข้อความสั่ง A;
else if (นิพจน์ 2)
        ข้อความสั่ง B;
    else if ...
            ....
        else
            ข้อความสั่ง N;
```

ถ้านิพจน์ 1 ให้ผลลัพธ์เป็นจริง จะทำงานตามข้อความสั่ง A
แต่ถ้านิพจน์ 1 ให้ผลลัพธ์เป็นเท็จ จะตรวจสอบผลลัพธ์ของนิพจน์ 2
ถ้านิพจน์ 2 ให้ผลลัพธ์เป็นจริง จะทำงานตามข้อความสั่ง B
แต่ถ้านิพจน์ 2 ให้ผลลัพธ์เป็นเท็จ ให้ไปตรวจสอบผลลัพธ์ของนิพจน์ถัดไป
จนกระทั่งสุดท้าย หากไม่มีนิพจน์ใด เป็นจริงเลย จึงจะไปทำงานตามข้อความสั่ง N

ตัวอย่าง 3.13

การใช้ข้อความสั่ง if ซ้อน โดยรับค่าคะแนน แล้วนำไปคำนวณหาระดับคะแนน

```
#include <stdio.h>                                                    // 1
void main()                                                           // 2
{
      float score;                                                    // 3
      char grade;                                                     // 4
      scanf("%f", &score);                                            // 5
      if(score > 100)                                                 // 6
            printf("Score must be less than or equal to 100");        // 7
      else if(score >= 80)                                            // 8
             grade = 'G';                                             // 9
          else if(score >= 50)                                        // 10
                 grade = 'P';                                         // 11
               else                                                   // 12
                     grade = 'F';                                     // 13
      printf("\n Grade = %c", grade);                                 // 14
}
```

บรรทัดที่ 5 รับค่าคะแนนจากแป้นพิมพ์

บรรทัดที่ 6 ถ้า score มีค่ามากกว่า 100

ให้แสดงข้อความ Score must be less than or equal to 100

บรรทัดที่ 8 ถ้า score มีค่าตั้งแต่ 80 ขึ้นไป ไปทำตามข้อความสั่งในบรรทัดที่ 9 มิฉะนั้นไปที่บรรทัดที่ 10

บรรทัดที่ 9 ให้เก็บ G ไว้ใน grade แล้วไปที่ข้อความสั่งในบรรทัด 14

บรรทัดที่ 10 ตรวจสอบผลลัพธ์ของนิพจน์ score >= 50 ถ้า x มีค่าตั้งแต่ 50 ขึ้นไป ให้กระทำการข้อความสั่งใน บรรทัดที่ 11 แต่ถ้า score มีค่าน้อยกว่า 50 ให้กระทำการบรรทัดที่ 13

บรรทัดที่ 11 ให้เก็บ P ไว้ใน grade แล้วไปที่ข้อความสั่งในบรรทัด 14

บรรทัดที่ 13 ให้เก็บ F ไว้ใน grade

บรรทัดที่ 14 แสดงข้อความ Grade = และ แสดงค่าของ grade

ตัวอย่าง 3.14

การใช้ข้อความสั่ง if ซ้อน

```
#include <stdio.h>                                    // 1
void main()                                           // 2
{
      float score;                                    // 3
      char grade;                                     // 4
      int point;                                      // 5
      scanf("%f",&score);                             // 6
      if(score >= 80)                                 // 7
      {
          grade = 'G';                                // 8
          point = 4;                                  // 9
      }
      else if(score >= 50)                            // 10
      {
            grade = 'P';                              // 11
            point = 2;                                // 12
      }
          else                                        // 13
          {
              grade = 'F';                            // 14
              point = 0;                              // 15
           }
}
```

เมื่อข้อความสั่งภายในข้อความสั่ง if ที่จะถูกกระทำมีมากกว่าหนึ่งข้อความสั่ง จะต้องใส่ข้อความสั่งเหล่านั้นไว้ภายในเครื่องหมายปีกกา

บรรทัดที่ 6 รับค่าจำนวนจริง จากแป้นพิมพ์ แล้วนำไปเก็บใน score

บรรทัดที่ 7 ถ้า score มีค่าตั้งแต่ 80 ขึ้นไป ให้ไปกระทำการที่บรรทัดที่ 8

บรรทัดที่ 8 ให้เก็บค่า G ไว้ใน grade

บรรทัดที่ 9 ให้เก็บค่า 4 ไว้ใน point

บรรทัดที่ 10 แต่ถ้านิพจน์ score >= 80 มีค่าเป็นเท็จ จะตรวจสอบผลลัพธ์ของนิพจน์ score >= 50 ถ้า score มีค่าตั้งแต่ 50 ขึ้นไป ให้ไปกระทำการบรรทัด 11

บรรทัดที่ 11 ให้เก็บค่า P ไว้ใน grade

บรรทัดที่ 12 ให้เก็บค่า 2 ไว้ใน point

บรรทัดที่ 13 แต่ถ้า score มีค่าน้อยกว่า 50 ให้ไปกระทำการบรรทัดที่ 14

บรรทัดที่ 14 ให้เก็บค่า F ไว้ใน grade

บรรทัดที่ 15 ให้เก็บค่า 0 ไว้ใน point

ตัวอย่าง 3.15

การใช้ข้อความสั่ง if ซ้อน เพื่อหาค่าสูงสุด

```
#include <stdio.h>                                          // 1
void main()                                                 // 2
{
      int a, b, c, max;                                     // 3
      printf("Enter three integer : ");                     // 4
      scanf("%d%d%d", &a, &b, &c);                          // 5
      max = c;                                              // 6
      if(a > max)                                           // 7
         max = a;                                           // 8
      else if(b > max)                                      // 9
            max = b;                                        // 10
      printf("Maximum is : %d\n", max);                     // 11
}
```

บรรทัดที่ 4 แสดงข้อความ Enter three integer :

บรรทัดที่ 5 ให้ป้อนค่า 3 ค่า ไปเก็บไว้ใน a, b, และ c ตามลำดับ

บรรทัดที่ 6 กำหนดให้ค่าที่เก็บไว้ในตัวแปร max มีค่าเท่ากับค่าที่เก็บไว้ในตัวแปร c

บรรทัดที่ 7 ถ้า ค่าที่เก็บใน a มีค่ามากกว่าค่าที่เก็บใน max เป็นจริง ไปกระทำการข้อความสั่งในบรรทัดที่ 8 แต่ถ้าค่าที่เก็บใน a มีค่ามากกว่าค่าที่เก็บใน max เป็นเท็จ จะไปกระทำการข้อความสั่งในบรรทัดที่ 9

บรรทัดที่ 8 ให้นำค่าที่เก็บใน a ไปไว้ใน max แล้วไปกระทำการข้อความสั่งในบรรทัดที่ 11

บรรทัดที่ 9 ตรวจสอบว่า ค่าที่เก็บใน b มากกว่าค่าที่เก็บใน max หรือไม่ ถ้าค่าที่เก็บใน b มีค่ามากกว่าค่าที่เก็บใน max ไปกระทำการข้อความสั่งในบรรทัดที่ 10 ถ้าค่าที่เก็บใน b มีค่าน้อยกว่าหรือเท่ากับค่าที่เก็บใน max ให้ไปกระทำการข้อความสั่งในบรรทัดที่ 11

บรรทัดที่ 10 ให้นำค่าที่เก็บใน b ไปไว้ใน max แล้วไปกระทำการบรรทัดที่ 11

บรรทัดที่ 11 ให้แสดงข้อความ Maximum is และแสดงค่าที่เก็บใน max

ตัวอย่าง 3.16

การใช้ข้อความสั่ง if ซ้อน

```
#include <stdio.h>                                    // 1
void main()                                           // 2
{
      int a, b, c, max;                               // 3
      scanf("%f %f", &x, &y);                         // 4
      if(x > 0)                                       // 5
         if(y > 0)                                    // 6
             z = x / y;                               // 7
         else                                         // 8
             z = x - y;                               // 9
      else                                            // 10
         if(x < 0)                                    // 11
             if(y > 0)                                // 12
                 z = x * y;                           // 13
             else                                     // 14
                 z = x + y;                           // 15
         printf("z = %f", z);                        // 16
}
```

บรรทัดที่ 4 รับค่า x และ y

บรรทัดที่ 5 ถ้า x > 0 ไปกระทำการข้อความสั่งในบรรทัดที่ 6 มิฉะนั้นไปที่บรรทัดที่ 10

บรรทัดที่ 6 ถ้า y > 0 ไปกระทำการข้อความสั่งในบรรทัดที่ 7 มิฉะนั้นไปที่บรรทัดที่ 8

บรรทัดที่ 7 เมื่อกระทำการข้อความสั่ง z = x / y; แล้ว จะไปกระทำการข้อความสั่งในบรรทัดที่ 16

บรรทัดที่ 8 else คู่กับ if ในบรรทัดที่ 6

บรรทัดที่ 9 เมื่อทำข้อความสั่ง z = x – y; แล้ว จะไปกระทำการข้อความสั่งในบรรทัดที่ 16
บรรทัดที่ 10 else คู่กับ if ในบรรทัดที่ 5
บรรทัดที่ 11 ถ้า x < 0 ไปกระทำการข้อความสั่งในบรรทัดที่ 12 มิฉะนั้นไปที่บรรทัด 16
บรรทัดที่ 12 ถ้า y > 0 ไปกระทำการข้อความสั่งในบรรทัดที่ 13
บรรทัดที่ 13 เมื่อทำข้อความสั่ง z = x * y แล้ว จะไปกระทำการข้อความสั่งในบรรทัดที่ 16
บรรทัดที่ 14 else คู่กับ if ในบรรทัดที่ 12
บรรทัดที่ 15 เมื่อทำข้อความสั่ง z = x + y; แล้ว จะไปกระทำการข้อความสั่งในบรรทัดที่ 16
บรรทัดที่ 16 แสดงข้อความ z = และ ค่าของ z

3.4.4 ข้อความสั่ง switch

ข้อความสั่ง switch เป็นข้อความสั่งที่ให้เลือกทำข้อความสั่ง หรือกลุ่มข้อความสั่งใดๆ โดยพิจารณาจากค่าของนิพจน์ ถ้าค่าของนิพจน์มีค่าเท่ากับค่าใด ก็จะเริ่มกระทำการข้อความสั่งที่อยู่หลังคำหลัก case นั้น และกระทำการข้อความสั่งอื่นที่ตามมาทั้งหมด

รูปแบบ

```
switch (นิพจน์)
{
        case ค่าที่ 1 :  ข้อความสั่ง 11;
                         ข้อความสั่ง 12;
                         ...
                         ข้อความสั่ง 1n;
        case ค่าที่ 2 :  ข้อความสั่ง 21;
                         ข้อความสั่ง 22;
                         ...
                         ข้อความสั่ง 2m;
        ...              ...
                         ...
        case ค่าที่ j :  ข้อความสั่ง j1;
                         ข้อความสั่ง j2;
                         ...
                         ข้อความสั่ง jp;
        default :        ข้อความสั่ง k1;
                         ข้อความสั่ง k2;
                         ...
                         ข้อความสั่ง kq;
```

โดยที่ค่าของนิพจน์จะต้องเป็นเลขจำนวนเต็มหรืออักขระเท่านั้น

ข้อความสั่ง switch จะตรวจสอบนิพจน์ แล้วนำผลไปเปรียบเทียบกับค่าที่ตามหลังคำหลัก case ถ้าเท่ากับค่าที่ตามหลังคำหลัก case ค่าใด ก็จะกระทำการที่ข้อความสั่งนั้น จากนั้นจะไปกระทำการที่ข้อความสั่งถัดไปทุกข้อความสั่งที่ตามมา ในกรณีที่ผลลัพธ์ของนิพจน์ไม่ตรงกับค่าที่ตามหลังคำหลัก case จะไปกระทำการที่ข้อความสั่งที่อยู่หลัง default

ผลลัพธ์ของนิพจน์ ต้องเป็นชนิด long, int, หรือ char อย่างใดอย่างหนึ่งเท่านั้น

ตัวอย่าง 3.17

แสดงการใช้ข้อความสั่ง switch โดยรับค่าจากแป้นพิมพ์ แล้วนำมาตรวจสอบว่า เป็นเลขคู่ หรือเลขคี่

```
#include <stdio.h>                                  // 1
int value;                                          // 2
void main()                                         // 3
{
      scanf("%d", &value);                          // 4
      switch (value % 2)                            // 5
      {
          case 0 :                                  // 6
               printf("Even integer\n");            // 7
          case 1 :                                  // 8
               printf("Odd integer\n");             // 9
      }
}
```

บรรทัดที่ 4 รับค่าจำนวนเต็มจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน value

บรรทัดที่ 5 คำนวณหาค่าของ value % 2

ถ้า value % 2 มีค่าเท่ากับ 0 จะกระทำการบรรทัดที่ 6 ถึงบรรทัดที่ 9

โดยจะแสดง ข้อความ Even integer

Odd integer

ถ้า value % 2 มีค่าเท่ากับ 1 จะกระทำการบรรทัดที่ 8 ถึงบรรทัดที่ 9

โดยจะแสดง ข้อความ Odd integer

3.4.5 ข้อความสั่ง break

ข้อความสั่ง break ใช้สำหรับควบคุมการกระทำการ โดยบังคับให้หยุดกระทำการข้อความสั่ง switch เพื่อป้องกันไม่ให้กระทำการข้อความสั่งอื่นที่ตามมาภายในข้อความสั่ง switch

ตัวอย่าง 3.18

การใช้ข้อความสั่ง switch และ break

```
#include <stdio.h>                                    // 1
int value;                                            // 2
void main()                                           // 3
{
      scanf("%d", &value);                            // 4
      switch(value % 2)                               // 5
      {
          case 0 :                                    // 6
                printf("Even integer\n");             // 7
                break;                                // 8
          case 1 :                                    // 9
                printf("Odd integer\n");              // 10
      }
}
```

บรรทัดที่ 4 รับค่าจำนวนเต็มจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน value

บรรทัดที่ 5 คำนวณหาค่าของ value % 2

ถ้า value % 2 มีค่าเท่ากับ 0 จะกระทำการบรรทัดที่ 6 ถึง 8

โดยจะแสดง ข้อความ Even integer

ถ้า value % 2 มีค่าเท่ากับ 1 จะกระทำการบรรทัดที่ 9 ถึง 10

โดยจะแสดง ข้อความ Odd integer

บรรทัดที่ 8 สั่งให้หยุดกระทำการข้อความสั่ง switch

ตัวอย่าง 3.19

แสดงการใช้ข้อความสั่ง switch และ break

```
#include <stdio.h>                                          // 1
int num;                                                    // 2
void main()                                                 // 3
{
      puts ("Enter 1 or 2 : ");                             // 4
      scanf("%d", &num);                                    // 5
      switch (num)                                          // 6
      {
          case 1 :                                          // 7
                printf("One\n");                            // 8
                break;                                      // 9
          case 2 :                                          // 10
                printf("Two\n");                            // 11
                break;                                      // 12
      }
      puts ("End of program");                              // 13
}
```

บรรทัดที่ 4 แสดงข้อความ Enter 1 or 2 :

บรรทัดที่ 5 รับค่าจำนวนเต็มจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน num

บรรทัดที่ 6 ถ้า num มีค่าเท่ากับ 1 จะกระทำการบรรทัดที่ 7 ถึง 9
ถ้า num มีค่าเท่ากับ 2 จะกระทำการบรรทัดที่ 10 ถึง 12

บรรทัดที่ 7-9 แสดง ข้อความ One
แล้วไปกระทำการบรรทัดที่ 13

บรรทัดที่ 10-12 แสดง ข้อความ Two
แล้วไปกระทำการบรรทัดที่ 13
ถ้า num มีค่าไม่เท่ากับ 1 หรือ 2 จะไปกระทำการบรรทัดที่ 13

บรรทัดที่ 13 แสดงข้อความ End of program

ตัวอย่าง 3.20

การใช้ข้อความสั่ง switch และ break

```
#include <stdio.h>                                          // 1
int num;                                                    // 2
void main()                                                 // 3
{
      puts ("Enter 1 or 2 : ");                             // 4
      scanf("%d",&num);                                     // 5
      switch (num)                                          // 6
      {
          case 1 :                                          // 7
                 printf("One\n");                          // 8
                 break;                                    // 9
          case 2 :                                          // 10
                 printf("Two\n");                          // 11
                 break;                                    // 12
          default :                                         // 13
                 printf("Out of range");                   // 14
      }
      puts ("End of program");                              // 15
}
```

บรรทัดที่ 4 แสดงข้อความ Enter 1 or 2 :

บรรทัดที่ 5 รับค่าจำนวนเต็มจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน num

บรรทัดที่ 6 ถ้า num มีค่าเท่ากับ 1 จะกระทำการบรรทัดที่ 7 ถึง 9

ถ้า num มีค่าเท่ากับ 2 จะกระทำการบรรทัดที่ 10 ถึง 12

ถ้า num มีค่าไม่เท่ากับ 1 หรือ 2 จะไปกระทำการบรรทัดที่ 13, 14 และ 15

บรรทัดที่ 7-9 แสดง ข้อความ One แล้วไปกระทำการบรรทัดที่ 15

บรรทัดที่ 10-12 แสดง ข้อความ Two แล้วไปกระทำการบรรทัดที่ 15

บรรทัดที่ 13-14 แสดง ข้อความ Out of range แล้วไปกระทำการบรรทัดที่ 15

บรรทัดที่ 15 แสดงข้อความ End of program

ตัวอย่าง 3.21

การใช้ข้อความสั่ง switch และ break

```
#include <stdio.h>                                          // 1
int g, p, f;                                               // 2
char grade;                                                // 3
void main()                                                // 4
{
      g = p = f = 0;                                       // 5
      puts ("Enter the letter grade : ");                  // 6
      scanf("%c", &grade);                                 // 7
      switch (grade)                                       // 8
      {
          case 'G' : case 'g' :                            // 9
               ++g;                                        // 10
               break;                                      // 11
          case 'P' : case 'p' :                            // 12
               ++p;                                        // 13
               break;                                      // 14
          case 'F' : case 'f' :                            // 15
               ++f;                                        // 16
               break;                                      // 17
          default :                                        // 18
                printf("Out of range ");                   // 19
      }
      puts ("End of program");                             // 20
}
```

บรรทัดที่ 5 ให้เก็บค่าศูนย์ไว้ใน g, p, และ f

บรรทัดที่ 6 แสดงข้อความ Enter the letter grade :

บรรทัดที่ 7 รับอักขระจากแป้นพิมพ์ แล้วนำมาเก็บไว้ใน grade

บรรทัดที่ 8 ถ้า grade มีค่าเท่ากับ G หรือ g จะกระทำการบรรทัดที่ 9 ถึง 11

ถ้า grade มีค่าเท่ากับ P หรือ p จะกระทำการบรรทัดที่ 12 ถึง 14

ถ้า grade มีค่าเท่ากับ F หรือ f จะกระทำการบรรทัดที่ 15 ถึง 17

ถ้า grade มีค่าไม่เท่ากับ G หรือ g หรือ P หรือ p หรือ F หรือ f จะไปกระทำการบรรทัดที่ 19

บรรทัดที่ 9-11 เพิ่มค่า g ขึ้นอีก 1
แล้วไปกระทำการบรรทัดที่ 20
บรรทัดที่ 12-14 เพิ่มค่า p ขึ้นอีก 1
แล้วไปกระทำการบรรทัดที่ 20
บรรทัดที่ 15-17 เพิ่มค่า f ขึ้นอีก 1
แล้วไปกระทำการบรรทัดที่ 20
บรรทัดที่ 19 แสดง ข้อความ Out of range
แล้วไปกระทำการบรรทัดที่ 20
บรรทัดที่ 20 แสดงข้อความ End of program

3.5 การวนซ้ำ

การวนซ้ำ หมายถึง การควบคุมให้กระทำการบางข้อความสั่งซ้ำหลายรอบ ซึ่งจะช่วยให้การเขียนโปรแกรมทำได้ง่าย สะดวก ไม่ต้องเขียนข้อความสั่งเดิมหลายครั้ง ทำให้โปรแกรมมีความกระชับ สามารถตรวจสอบความผิดพลาดได้ง่าย

ข้อความสั่งวนซ้ำ มี 3 ชนิด ได้แก่

- ข้อความสั่ง for
- ข้อความสั่ง do-while
- ข้อความสั่ง while

โดยที่แต่ละข้อความสั่ง มีรูปแบบและวิธีการใช้งานที่แตกต่างกัน นักเขียนโปรแกรมสามารถเลือกใช้ได้ตามความเหมาะสมกับลักษณะของการใช้งานในโปรแกรม

3.5.1 ข้อความสั่ง for

ข้อความสั่ง for เป็นข้อความสั่งที่สั่งให้กระทำการข้อความสั่ง หรือกลุ่มของข้อความสั่ง วนซ้ำหลายรอบ โดยมีจำนวนรอบในการวนซ้ำที่แน่นอน

รูปแบบ

```
for (ค่าเริ่มต้น; เงื่อนไข; ค่าเพิ่มหรือค่าลด)
    ข้อความสั่ง;
```

โดยที่

ค่าเริ่มต้น เงื่อนไข และ ค่าเพิ่มหรือค่าลด เป็นนิพจน์

ข้อความสั่ง หมายถึง ข้อความสั่งที่จะถูกกระทำซ้ำ ซึ่งอาจจะมีเพียงข้อความสั่งเดียว หรือ หลายข้อความสั่งก็ได้

ข้อความสั่ง for มีขั้นตอนการทำงานดังนี้

1. คำนวณหาค่าเริ่มต้นของตัวแปร ที่ใช้ควบคุมการวนซ้ำ ซึ่งอยู่ในรูปของข้อความสั่งกำหนดค่า
2. คำนวณหาผลลัพธ์จากเงื่อนไข ที่อยู่ในรูปของนิพจน์ความสัมพันธ์ ซึ่งจะให้ผลเป็นจริง(ศูนย์) หรือเท็จ (ค่าที่ไม่เป็นศูนย์)
3. ถ้าผลจากข้อ 2 มีค่าเป็นเท็จ หรือ ศูนย์ ไปที่ข้อ 7
4. ถ้าผลจากข้อ 2 มีค่าเป็นจริง หรือ ค่าที่ไม่ใช่ศูนย์ ข้อความสั่งที่อยู่ภายในข้อความสั่ง for จะถูกกระทำ
5. คำนวณหาค่าใหม่ของตัวแปรที่ใช้ควบคุมการวนซ้ำ
6. กลับไปที่ข้อ 2
7. จบการกระทำการข้อความสั่ง for และข้อความสั่งแรกที่อยู่ถัดจากข้อความสั่ง for จะถูกกระทำในลำดับต่อไป

ตัวอย่าง 3.21

การวนซ้ำเพื่อแสดงตัวเลข 1, 2, 3, 4, 5

```
#include <stdio.h>
int n;
void main()
{
      for(n = 1; n <= 5; n++)
            printf("\n%d", n);
}
```

ผลการกระทำการ

```
1
2
3
4
5
```

ตัวอย่าง 3.22

การวนซ้ำเพื่อแสดงตัวเลข 10, 20, 30, 40, 50

```
#include <stdio.h>
int n;
void main()
{
     for(n = 10; n <= 50; n+=10)
          printf("\n%d", n);
}
```

ผลการกระทำการ

10

20

30

40

50

ตัวอย่าง 3.23

การวนซ้ำเพื่อแสดงตัวเลข 5, 4, 3, 2, 1

```
#include <stdio.h>
int n;
void main()
{
     for(n = 5; n > 0; n--)
          printf("\n%d",n);
}
```

ผลการกระทำการ

5

4

3

2

1

นอกจากการใช้ข้อความสั่ง for ตามรูปแบบดังกล่าวข้างต้นแล้ว ข้อความสั่ง for ยังมีรูปแบบการใช้แบบอื่น ดังตัวอย่างต่อไปนี้

ตัวอย่าง 3.24

การละค่าเริ่มต้นในข้อความสั่ง for จะต้องกำหนดค่าเริ่มต้นไว้ก่อนข้อความสั่ง for

```
#include <stdio.h>
int n;
void main()
{
      n = 5;
      for( ; n > 0; n--)
           printf("%d   ",n);
}
```

ผลการกระทำการ

```
5  4  3  2  1
```

ตัวอย่าง 3.25

การละค่าเริ่มต้นและค่าเพิ่มหรือค่าลด

```
#include <stdio.h>
int n;
void main()
{
      n = 5;
      for( ; n <= 100; )
      {
            printf("%d  ",n);
            n = n + 5;
      }
}
```

ผลการกระทำการ

```
5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95  100
```

ตัวอย่าง 3.26

การใช้ข้อความสั่งอื่นแทนค่าเริ่มต้น ข้อความสั่งนั้นจะถูกกระทำเพียงครั้งเดียว

```
#include <stdio.h>
int n;
void main()
{
      n = 5;
      for( "Executing the for statement" ; n > 0; n--)
          printf("\n%d",n);
}
```

ผลการกระทำการ

Executing the for statement

5

4

3

2

1

ตัวอย่าง 3.27

การละส่วนการเปลี่ยนค่าตัวแปรที่ใช้ควบคุมการวนซ้ำ

```
#include <stdio.h>
int n;
void main()
{
      n = 5;
      for( "Executing the for statement" ; n > 0;)
             printf("\n%d", n--);
}
```

ผลการกระทำการ

Executing the for statement

5

4

3

2

1

ตัวอย่าง 3.28

การกำหนดให้ทุกนิพจน์อยู่ในเครื่องหมายวงเล็บของข้อความสั่ง for

```
#include <stdio.h>
void main()
{
      int n;
      for( n = 5; n <= 100; printf("%d  ", n), n = n + 5);
}
```

ผลการกระทำการ

5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

3.5.2 ข้อความสั่ง while

ข้อความสั่ง while เป็นข้อความสั่งวนซ้ำ ที่สั่งให้กระทำการข้อความสั่งที่อยู่ภายในข้อความสั่ง while หลายรอบจนกระทั่งเงื่อนไขเป็นเท็จ หรือ ศูนย์ จึงจะจบการวนซ้ำ

รูปแบบ

```
while (เงื่อนไข)
      ข้อความสั่ง;
```

โดยที่ เงื่อนไข ต้องอยู่ในรูปของนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และข้อความสั่งที่อยู่ภายในข้อความสั่ง while อาจมีเพียงข้อความสั่งเดียว หรือ หลายข้อความสั่ง

ข้อความสั่ง while มีขั้นตอนการทำงานดังนี้

1. คำนวณหาค่าของเงื่อนไข
2. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จ หรือ ศูนย์ ไปที่ข้อ 5
3. ถ้าค่าของเงื่อนไข มีค่าเป็นจริง หรือ ค่าที่ไม่ใช่ศูนย์ ข้อความสั่งที่อยู่ภายในข้อความสั่ง while จะถูกกระทำ
4. กลับไปที่ข้อ 1
5. จบการกระทำการข้อความสั่ง while และข้อความสั่งแรกที่อยู่ถัดจากข้อความสั่ง while จะถูกกระทำในลำดับต่อไป

ตัวอย่าง 3.29

โปรแกรมแสดงตัวเลข 1, 2, 3, 4, 5

```
#include <stdio.h>
int n;
void main()
{
        n = 1;
        while(n <= 5)
        {
                printf("\n%d", n);
                n++;
        }
}
```

ผลการกระทำการ

1

2

3

4

5

ข้อสังเกต

for(; เงื่อนไข ;) ให้ผลการทำงานเช่นเดียวกับ while(เงื่อนไข)

3.5.3 ข้อความสั่ง do-while

ข้อความสั่ง do-while เป็นข้อความสั่งวนซ้ำ ที่กระทำข้อความสั่งภายในข้อความสั่ง do-while หนึ่งรอบ แล้วจึงจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นเท็จ จะจบการทำงานทันที

รูปแบบ

```
do
        ข้อความสั่ง;
while(เงื่อนไข);
```

โดยที่

เงื่อนไขอยู่ในรูปของนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ

ข้อความสั่งภายในข้อความสั่ง do-while อาจมีเพียงข้อความสั่งเดียว หรือ หลายข้อความสั่ง ก็ได้

ข้อความสั่ง do-while มีขั้นตอนการทำงานดังนี้

1. กระทำข้อความสั่งที่อยู่ภายในข้อความสั่ง do-while
2. คำนวณหาค่าของเงื่อนไข
3. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จหรือ ศูนย์ ไปที่ข้อ 5
4. ถ้าค่าของเงื่อนไข มีค่าเป็นจริง หรือ ค่าที่ไม่ใช่ศูนย์ กลับไปที่ข้อ 1
5. จบการทำงานข้อความสั่ง do-while และข้อความสั่งแรกที่อยู่ถัดจากข้อความสั่ง do-while จะถูกกระทำในลำดับต่อไป

ตัวอย่าง 3.30

โปรแกรมแสดงตัวเลข 1, 2, 3, 4, 5

```
#include <stdio.h>
int n;
void main()
{
      n = 1;
      do
      {
            printf("\n%d", n);
            n++;
      } while(n <= 5);
}
```

ผลการกระทำการ

1

2

3

4

5

3.6 การวนซ้ำซ้อน

การวนซ้ำซ้อน หมายถึง การควบคุมให้กระทำบางข้อความสั่งหลายรอบ และในการทำงานแต่ละรอบ ก็จะควบคุมให้กระทำการข้อความสั่งที่อยู่ภายในนั้นอีกหลายรอบ

การวนซ้ำซ้อน อาจเขียนได้โดยใช้ข้อความสั่งวนซ้ำใดๆ ซ้อนกัน 2 ข้อความสั่ง เช่น

- ข้อความสั่ง for ซ้อนกับข้อความสั่ง for
- ข้อความสั่ง while ซ้อนกับข้อความสั่ง while
- ข้อความสั่ง while-do ซ้อนกับข้อความสั่ง while-do
- ข้อความสั่ง while ซ้อนกับข้อความสั่ง for

หลักการทำงานของการวนซ้ำซ้อน จะเริ่มที่การวนซ้ำรอบนอกก่อน 1 รอบ ในระหว่างนั้น จะไปกระทำการวนซ้ำรอบในอีกหลายๆ รอบ จากนั้นจึงจะไปกระทำการวนซ้ำรอบนอก และวนซ้ำรอบในอีกหลายๆ รอบ เช่นนี้เรื่อยไป จนกระทั่งจบการวนซ้ำรอบนอก

ตัวอย่าง 3.31

การวนซ้ำซ้อน

```
#include <stdio.h>
int n, m;
void main()
{
      for(n = 1; n <= 2; n++)                  // จุดเริ่มต้นรอบนอก
          for(m = 1; m <= 3; m++)              // จุดเริ่มต้นรอบใน
             printf("\n%d    %d",n,m);         // จุดจบรอบในและจุดจบรอบนอก
}
```

ผลการกระทำการ

```
1    1
1    2
1    3
2    1
2    2
2    3
```

การวนซ้ำจะเริ่มที่รอบนอกรอบที่ 1 ก่อน จากนั้นจะทำวนซ้ำรอบใน 3 รอบ แล้วจึงจบการวนซ้ำรอบนอกรอบที่ 1 จากนั้นจะเริ่มทำรอบนอกรอบที่ 2 แล้วจึงทำวนซ้ำรอบในอีก 3 รอบ

ตัวอย่าง 3.32

การคำนวณคะแนนเฉลี่ยของนิสิต

```
#include <stdio.h>                                                    // 1
void main()                                                           // 2
{
      int m, n, nostudent, noscore;                                   // 3
      float score, total, average;                                    // 4
      scanf("%d%d", &nostudent, &noscore);                            // 5
      for(n = 1; n <= nostudent; n++)                                 // 6
      {
          total = 0;                                                  // 7
          printf("student %d\n", n);                                  // 8
          for(m = 1; m <= noscore; m++)                               // 9
          {
                 scanf("%f", &score);                                 // 10
                 printf("%f ", score);                                // 11
                 total = total + score;                               // 12
           }
          average = total / noscore;                                  // 13
          printf("\nThe average of student %d is %f n",n,average);    // 14
      }
}
```

บรรทัดที่ 5 รับค่าจำนวนนิสิต และจำนวนวิชา

บรรทัดที่ 6 จุดเริ่มต้นสำหรับการคำนวณคะแนนของนิสิตแต่ละคน

บรรทัดที่ 7 กำหนดให้เก็บศูนย์ไว้ใน total เมื่อเริ่มต้นคำนวณคะแนนของนิสิตแต่ละคน

บรรทัดที่ 8 แสดงข้อความ student และค่าของ n หมายถึงนิสิตคนที่...

บรรทัดที่ 9 จุดเริ่มต้นสำหรับการคำนวณในแต่ละวิชา ของนิสิต

บรรทัดที่ 10 รับค่าคะแนนในแต่ละวิชา

บรรทัดที่ 11 แสดงค่าคะแนนในแต่ละวิชาจากที่รับเข้ามา

บรรทัดที่ 12 คำนวณผลรวมคะแนนของนิสิตแต่ละคน

บรรทัดที่ 13 คำนวณหาคะแนนเฉลี่ยของนิสิตแต่ละคน

บรรทัดที่ 14 แสดงข้อความ The average of student ... is .... และคะแนนเฉลี่ยของนิสิตแต่ละคน

แบบฝึกหัดบทที่ 3

1. `for(x = 0; x < 100, x++);` เมื่อจบข้อความสั่ง x มีค่าเท่าใด

2. `for(x = 2; x < 20; x += 3);` เมื่อจบข้อความสั่ง x มีค่าเท่าใด

3. ให้เขียนโปรแกรมเพื่อแสดงเลขคู่ตั้งแต่ 2 ถึง 100 และให้หาผลรวมของเลขดังกล่าว โดยใช้คำสั่ง

   3.1 for

   3.2 while

   3.3 while-do

4. ให้เขียนโปรแกรม เพื่อรับค่า A, B, C และ m จากแป้นพิมพ์ แล้วนำมาคำนวณหาค่า Y โดยมีเงื่อนไขต่อไปนี้

   กำหนดให้ A, B, C, m เป็นเลขจำนวนเต็ม

   Y = Am2 + Bm + C เมื่อ m > 5

   Y = Am2 - Bm - C เมื่อ m = 5

   Y = Am2 + Bm เมื่อ m < 5

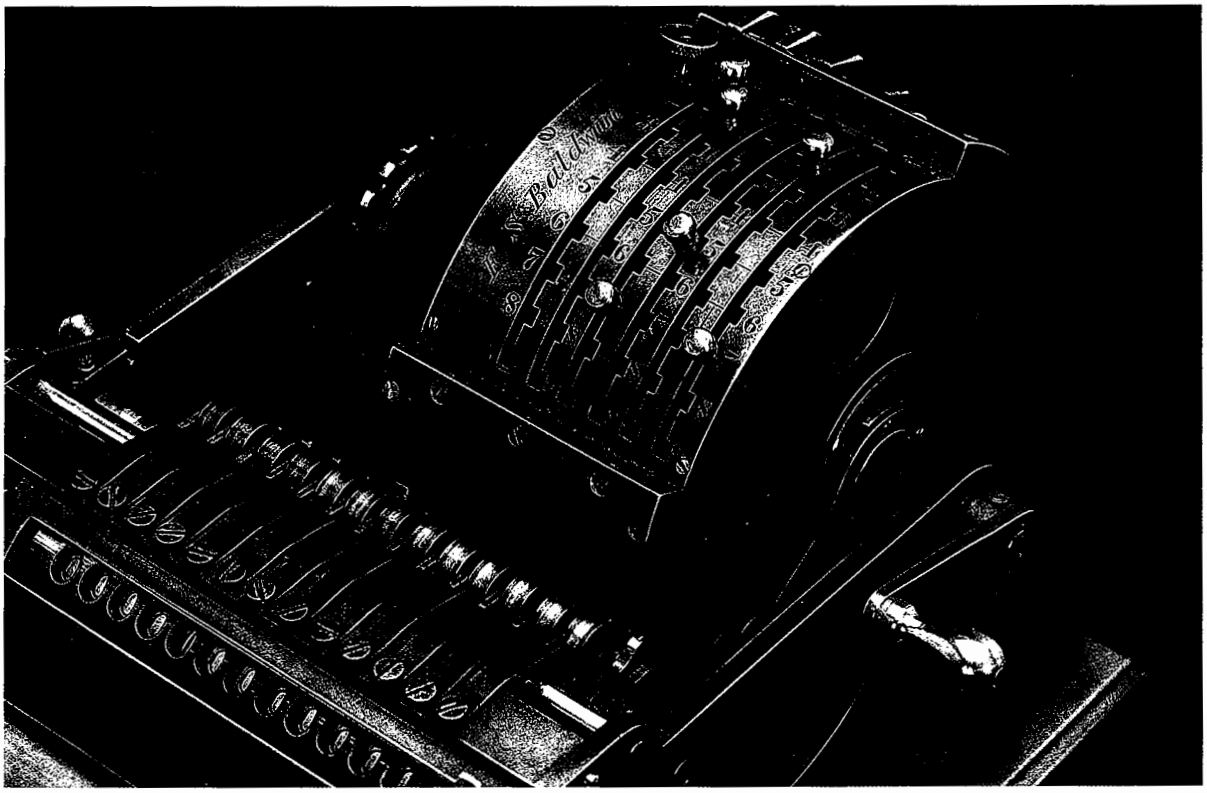
5. ให้เขียนโปรแกรม เพื่อรับค่ายอดขายรวม (Total sales) ของพนักงานจากแป้นพิมพ์ แล้วนำมาคำนวณหาค่านายหน้า (Sales commission) ตามเงื่อนไขต่อไปนี้

ยอดขายรวม (บาท)	ค่านายหน้า (%)
< 10000	0
10000 <= ยอดขายรวม < 25000	7
>=25000	10

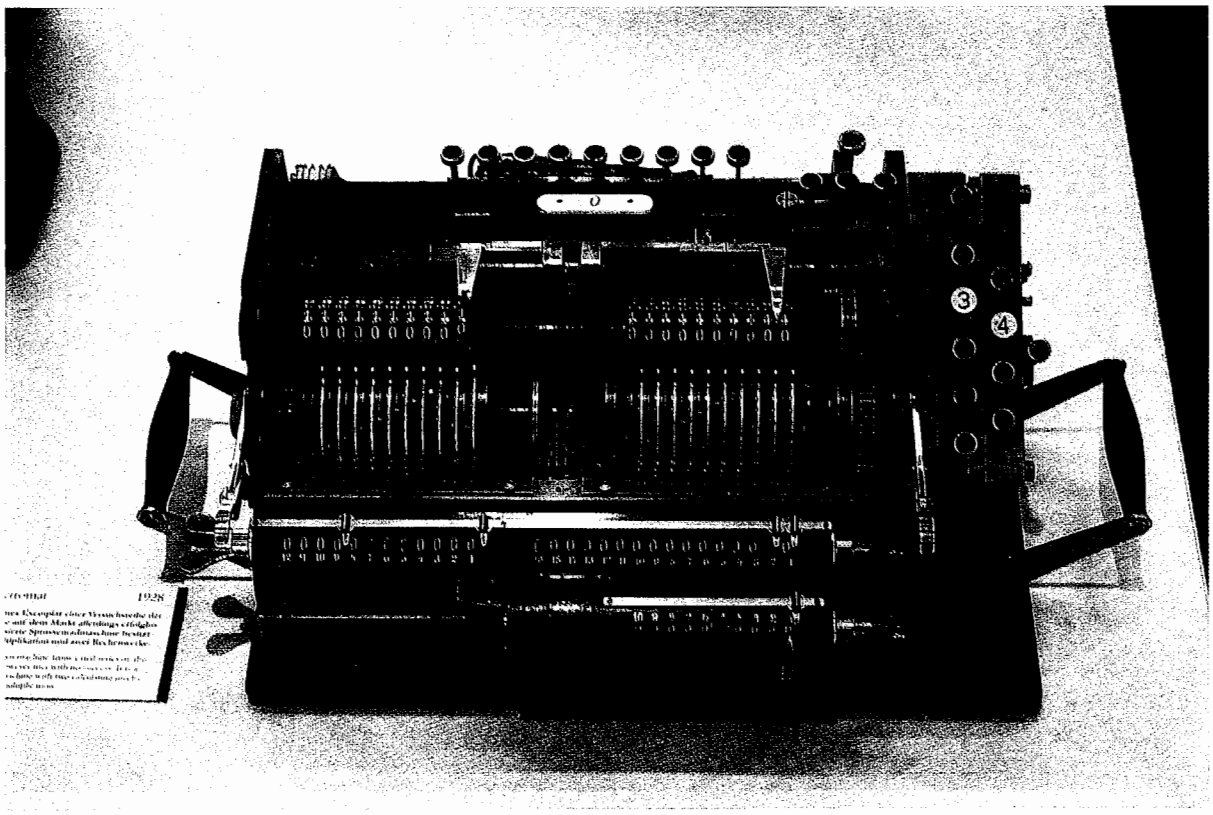
   โปรแกรมจะจบการกระทำการเมื่อยอดขายรวมมีค่าน้อยกว่าศูนย์

   รูปแบบผลลัพธ์

No.	Total Sales (Baht)	Sales Commission (Baht)
1	xxxxxxx.xx	xxxxxx.xx
2	xxxxxxx.xx	xxxxxx.xx
...	...	...
...	...	...
n	xxxxxxxx.xx	xxxxxx.xx



เครื่องวงล้อหมุด จดทะเบียนเครื่องประดิษฐ์โดย Frank S. Baldwin ค.ศ.1875



เครื่องคำนวณ

บทที่ 4
ฟังก์ชัน

ฟังก์ชัน (function) คือ กลุ่มของคำสั่ง มีวัตถุประสงค์เพื่อแบ่งโปรแกรมเป็นส่วนย่อยๆ แต่ละฟังก์ชันเขียนขึ้นเพื่อให้ทำงานอย่างใดอย่างหนึ่ง นอกจากฟังก์ชัน main() แล้ว โปรแกรมภาษาซีอาจประกอบด้วยฟังก์ชันอื่นกี่ฟังก์ชันก็ได้ อาจมีการส่งข้อมูลไปและกลับระหว่างฟังก์ชัน ฟังก์ชันอาจเรียกใช้ฟังก์ชันอื่นหรือตัวเองได้ โดยการทำงานของโปรแกรมจะเริ่มทำงานที่ข้อความสั่งแรกในฟังก์ชัน main() ก่อน

4.1 ฟังก์ชันที่เรียกใช้ฟังก์ชันอื่น

รูปแบบของฟังก์ชัน

```
ชนิดข้อมูล  ชื่อฟังก์ชัน (การประกาศพารามิเตอร์ – ถ้ามี)
{
        การประกาศตัวแปรเฉพาะที่
        กลุ่มข้อความสั่ง
        [return]
}
```

ชนิดข้อมูล	คือ ชนิดของข้อมูลที่ฟังก์ชันนี้จะส่งกลับ ถ้าฟังก์ชันไม่ส่งค่ากลับ ชนิดข้อมูลคือ void
การประกาศพารามิเตอร์	คือ การกำหนดชนิดของตัวแปรที่เป็นตัวรับค่า ถ้ามีมากกว่า 1 ตัว ใช้จุลภาค (,) คั่นระหว่างแต่ละตัว
การประกาศตัวแปรเฉพาะที่	คือ การประกาศตัวแปรที่ใช้เฉพาะในฟังก์ชันนี้
return	คือ ข้อความสั่งที่ใช้ในการส่งค่ากลับ ค่าที่ส่งกลับคือค่าหลัง return

รูปแบบของการเรียกใช้ฟังก์ชัน

ชื่อฟังก์ชัน(a1, a2, ..., an)

ชื่อฟังก์ชัน	คือ ชื่อฟังก์ชันที่ต้องการเรียกใช้
a1, a2, ..., an	คือ นิพจน์ที่ต้องการส่งค่าไปใช้ในฟังก์ชัน เรียกว่าอาร์กิวเมนต์ (argument) จำนวนอาร์กิวเมนต์ และจำนวนพารามิเตอร์ต้องเท่ากัน และตัวที่อยู่ที่ตำแหน่งตรงกันต้องมีชนิดเดียวกันด้วย

การส่งผ่านข้อมูลระหว่างอาร์กิวเมนต์ และพารามิเตอร์

ใช้วิธีเรียกด้วยมูลค่า (call by value) โดยมีการทำงานคือ

1. คำนวณนิพจน์ที่เป็นอาร์กิวเมนต์
2. คัดลอกค่าอาร์กิวเมนต์ให้กับพารามิเตอร์ ที่อยู่ในลำดับตรงกัน
3. ทำงานตามข้อความสั่งในฟังก์ชันจนจบ หรือเมื่อพบข้อความสั่ง return
4. ส่งกลับค่าที่อยู่ในข้อความสั่ง return (ถ้ามี)

การส่งผ่านข้อมูลระหว่างอาร์กิวเมนต์และพารามิเตอร์โดยวิธีเรียกด้วยมูลค่า ทำให้ไม่สามารถเปลี่ยนค่าของอาร์กิวเมนต์โดยฟังก์ชันได้

ต้นแบบฟังก์ชัน (function prototype)

ลำดับการกำหนดฟังก์ชันอาจกำหนดฟังก์ชันที่ถูกเรียกก่อนฟังก์ชันที่เป็นตัวเรียก ดังตัวอย่าง

ตัวอย่าง 4.1

รูปแบบของฟังก์ชัน

```
#include <stdio.h>
void fun()          /* ฟังก์ชันที่ถูกเรียก */
{
    ...
    ...
}
void main()         /* ฟังก์ชันที่เป็นตัวเรียก */
{
    ...
    ...
    fun();          /* เรียกฟังก์ชัน fun() จาก ฟังก์ชัน main() */
    ...
    ...
}
```

นอกจากนี้สามารถกำหนดฟังก์ชันที่เป็นตัวเรียกก่อนฟังก์ชันที่ถูกเรียกได้ ในกรณีที่กำหนดฟังก์ชันที่เป็นตัวเรียกไว้ก่อน ต้องมีต้นแบบฟังก์ชันก่อนฟังก์ชันที่เป็นตัวเรียก โดยส่วนมากมักกำหนดต้นแบบฟังก์ชันของฟังก์ชันทั้งหมดในโปรแกรม หลังข้อความสั่งตัวประมวลผลก่อน (preprocessor)

รูปแบบของต้นแบบฟังก์ชัน

ชนิดข้อมูล ชื่อฟังก์ชัน(ชนิดของพารามิเตอร์);

ชนิดข้อมูล	คือ ชนิดของข้อมูลที่ฟังก์ชันนี้จะส่งกลับ
ชนิดของพารามิเตอร์	คือ ชนิดข้อมูลของพารามิเตอร์แต่ละตัว ถ้ามีมากกว่า 1 ตัว ใช้จุลภาค (,) คั่นระหว่างแต่ละตัว กรณีไม่มีพารามิเตอร์ให้ใส่ void

ตัวอย่าง 4.2

การใช้ต้นแบบฟังก์ชัน

```
#include <stdio.h>

void fun(void);          /* ต้นแบบฟังก์ชัน */

void main()              /* ฟังก์ชันที่เป็นตัวเรียก */
{
    ...
    ...
    fun();
    ...
    ...
}

void fun()               /* ฟังก์ชันที่ถูกเรียก */
{
    ...
    ...
}
```

ตัวอย่าง 4.3

ฟังก์ชันที่ไม่มีการส่งค่ากลับ

```
#include <stdio.h>
void underline()
{
    printf("-------------\n");
}
void main()
{
    underline();
    printf("C PROGRAMMING \n");
    underline();
}
```

ในตัวอย่างกำหนดฟังก์ชันที่ถูกเรียก (underline()) ก่อน ฟังก์ชันที่เป็นตัวเรียก (main())

ผลการกระทำการ

```
-------------
C PROGRAMMING
-------------
```

ตัวอย่าง 4.4

การใช้ต้นแบบฟังก์ชัน

```
#include <stdio.h>
void underline(void);          /* ต้นแบบฟังก์ชัน */
void main()
{
    underline();
    printf("C PROGRAMMING \n");
    underline();
}
void underline()
{
    printf("-------------\n");
}
```

ในตัวอย่างกำหนดฟังก์ชันที่เป็นตัวเรียกก่อนฟังก์ชันที่ถูกเรียกต้องมีต้นแบบฟังก์ชัน

ผลการกระทำการ

```
-------------
C PROGRAMMING
-------------
```

ตัวอย่าง 4.5

ฟังก์ชันที่มีการส่งค่าไปแต่ไม่ส่งค่ากลับ

```
#include <stdio.h>
void underline(int);
void main()
{
    int limit;
    limit = 2;
    underline(limit);    /* คัดลอกค่าของ limit คือ 2 ไปไว้ใน num ในฟังก์ชัน underline()
                            แล้วไปกระทำการตามคำสั่งในฟังก์ชัน underline() จนจบ */
    printf("Kasetsart University \n");
    limit = 3;
    underline(limit);    /* คัดลอกค่าของ limit คือ 3 ไปไว้ใน num ในฟังก์ชัน underline()
                            แล้วไปกระทำการตามคำสั่งในฟังก์ชัน underline() จนจบ */
}
void underline(int num)
{
    int count;
    for(count = 1; count <= num; count++)
        printf("--------------------\n");
}
```

ผลการกระทำการ

```
--------------------
--------------------
Kasetsart University
--------------------
--------------------
--------------------
```

ตัวอย่าง 4.6

โปรแกรมรับข้อมูลเป็นความกว้างและความยาวของรูปสี่เหลี่ยม แล้วพิมพ์พื้นที่ของรูปสี่เหลี่ยมนี้

```
#include <stdio.h>
void area(float, float);
void main()
{
    float w, h;
    printf("Please enter width : ");
    scanf("%f", &w);
    printf("\n Please enter height : ");
    scanf("%f", &h);
    area(w, h);                          /* หาพื้นที่ของรูปสี่เหลี่ยม */
}
void area(float width, float height)   /* รับข้อมูลซึ่งเป็นความกว้างและความยาว */
{
    float result;
    result = width * height;
    printf("The area of the square is %f \n", result);
}
```

ตัวอย่าง 4.7

ฟังก์ชันที่มีการส่งค่าไปและกลับโดยใช้ต้นแบบฟังก์ชัน

```
#include <stdio.h>
int square(int);
void main()
{
    int i = 3;
    int k;
    k = square(i);          /* คัดลอกค่าของ i ไปไว้ใน x ในฟังก์ชัน square() */
                            /* หลังฟังก์ชัน square() ส่งค่ากลับ ค่านั้นจะถูกให้กับตัวแปร k */
    printf("%d \n", k);
}
int square(int x)
{
    return (x*x);           /* ส่งค่า x * x กลับไปไว้ในตำแหน่งที่เรียกฟังก์ชัน square() */
}
```

ผลการกระทำการ

```
9
```

ตัวอย่าง 4.8

ฟังก์ชันที่มีการส่งค่าไปและกลับ โดยไม่ใช้ต้นแบบฟังก์ชัน

```
#include <stdio.h>
int cube(int x)          /* ฟังก์ชันเพื่อหาค่ากำลังสามของพารามิเตอร์ชนิดจำนวนเต็ม */
{
    return (x*x*x);      /* ส่งค่า x * x * x กลับไปไว้ในตำแหน่งที่เรียกฟังก์ชัน cube() */
}
void main()
{
    int i = 2;
    while(i < 500)
    {
        i = cube(i);     /* คัดลอกค่าของ i ไปไว้ใน x ในฟังก์ชัน cube() */
        printf("%d\n", i);
    }
}
```

ผลการกระทำการ

```
8
512
```

ตัวอย่าง 4.9

การหาค่าสัมบูรณ์โดยใช้ฟังก์ชัน

```
#include <stdio.h>
int abs(int);
void main()
{
    int x;
    scanf("%d", &x);
    while(x != 0)
    {
        printf("The absolute value of %d is %d \n", x, abs(x));
        scanf("%d", &x);
    }
}
```

```
int abs(int i)              /* หาค่าสัมบูรณ์ (absolute value) ของ x */
{
    if (i < 0)
        return (-i);
    else
        return (i);
}
```

จะอ่านข้อมูลเข้าเป็นจำนวนเต็ม แล้วพิมพ์ค่าของเลขจำนวนนั้นและค่าสมบูรณ์ของเลขจำนวนนั้นจนกว่าข้อมูลที่รับเข้าจะมีค่าเป็น 0 จึงจบการทำงาน

ตัวอย่าง 4.10

ฟังก์ชันหาค่ากำลังสอง

```
#include <stdio.h>
float xquare(float);
void main()
{
    float i = 2.4;
    while(i < 500)
    {
        i = xsquare(i);
        printf("%.2f\n", i);
    }
}
float xsquare(float x)
{
    return(x*x);
}
```

ผลการกระทำการ

5.76

33.18

1100.75

ตัวอย่าง 4.11

ฟังก์ชันหาค่าแฟคทอเรียล (factorial)

```
#include <stdio.h>
int fact(int);
void main()
{
    int k = 4;
    printf("%d\n", fact(k));
}
int fact(int n)
{
    int i;
    int prod = 1;
    if(n > 1)
        for(i = 2; i <= n; i++)
            prod = prod * i;
    return(prod);
}
```

ผลการกระทำการ

```
24
```

ตัวอย่าง 4.12

ฟังก์ชันหาผลรวมของ $1^3 + 2^3 + ... + n^3$

```
#include <stdio.h>
int sum_of_cube(int);
void main()
{
    int n;
    for(n = 1; n <= 4; n++)
        printf("%d\n", sum_of_cube(n));
}
int sum_of_cube(int n)                    /* 1^3+ 2^3 + ... + n^3 */
{
    int i, sum;
    sum = 0;
    for(i = 1; i <= n; i++)
        sum = sum + i * i * i;
    return(sum);
}
```

ผลการกระทำการ

1

9

36

100

ตัวอย่าง 4.13

แสดงการส่งผ่านข้อมูลด้วยวิธีเรียกด้วยมูลค่า

```
#include <stdio.h>
void try_to_modify(int);
void main()
{
    int i = 7, j;
    j = try_to_modify(i);
    printf("%d %d \n", i, j);
}
void try_to_modify(int k)
{
    printf("%d \n", k);
    k = k + 22;
    printf("%d \n", k);
    return(k);
}
```

ผลการกระทำการ

7

29

7 29

หมายเหตุ ไม่สามารถเปลี่ยนค่าของอาร์กิวเมนต์ โดยฟังก์ชันอื่นๆ ได้

ในตัวอย่างข้างต้นจึงไม่สามารถเปลี่ยนค่าของอาร์กิวเมนต์ คือ i โดยฟังก์ชัน try_to_modify()

ตัวอย่าง 4.14

แสดงการส่งผ่านข้อมูลด้วยวิธีเรียกด้วยมูลค่า

```
#include <stdio.h>
void change(int, int);
void main()
{
    int a, b;
    a = 17;
    b = 29;
    printf("before calling function change() a = %d  b = %d \n", a, b);
    change(a,b);
    printf("after calling function change() a = %d  b = %d \n", a, b);
}
void change(int x, int y)
{
    int temp;
    printf("in function change() x = %d  y = %d \n", x, y);
    temp = x;
    x = y;
    y = temp;
    printf("in function change() x = %d  y = %d \n", x, y);
}
```

ผลการกระทำการ

```
before calling function change() a = 17  b = 29
in function change() x = 17  y = 29
in function change() x = 29  y = 17
after calling function change() a = 17  b = 29
```

4.2 ฟังก์ชันเรียกซ้ำ (recursive function)

ฟังก์ชันในภาษาซีสามารถเรียกตัวเองได้ซึ่งเรียกว่าฟังก์ชันเรียกซ้ำ รูปแบบของฟังก์ชันเหมือนกับฟังก์ชันธรรมดา ส่วนมากเขียนฟังก์ชันเรียกซ้ำเพื่อแก้ปัญหาชนิดที่มีการกำหนดแบบเรียกซ้ำ (recursively defined) เช่น การหาค่า n! (n เป็นจำนวนเต็มบวก) ซึ่งเป็นผลคูณของเลขจำนวนเต็มตั้งแต่ 1 ถึง n

โดย $n! = n * (n-1)!$ ถ้า $n > 0$

$0! = 1$

สมมติ n มีค่าเท่ากับ 5

$5! = 5 * 4!$

$4! = 4 * 3!$

$3! = 3 * 2!$

$2! = 2 * 1!$

$1! = 1 * 0!$

$0! = 1$

เพราะฉะนั้น

$1! = 1 * 1 = 1$

$2! = 2 * 1 = 2$

$3! = 3 * 2 = 6$

$4! = 4 * 6 = 24$

$5! = 5 * 24 = 120$

ตัวอย่าง 4.15

ฟังก์ชันเรียกซ้ำหาค่าแฟคทอเรียล

```
#include <stdio.h>
int factorial(int);
void main()
{
    printf("%d \n", factorial(4));
}
int factorial(int n)                          /* คำนวณ n! */
{
    if(n <= 1)
        return (1);
    else
        return (n * factorial(n-1));          /* ฟังก์ชัน factorial() เรียกตัวเอง */
}
```

การทำงานคือ factorial(4) ส่งค่า (4 * factorial(3)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

factorial(3) ส่งค่า (3 * factorial(2)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

factorial(2) ส่งค่า (2 * factorial(1)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

factorial(1) ส่งค่า (1) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

ดังนั้น factorial(2) มีค่า 2 * factorial(1) คือ 2 * 1 เท่ากับ 2

factorial(3) มีค่า 3 * factorial(2) คือ 3 * 2 เท่ากับ 6

factorial(4) มีค่า 4 * factorial(3) คือ 4 * 6 เท่ากับ 24

ผลการกระทำการ

24

ตัวอย่าง 4.16

ฟังก์ชันคำนวณไฟโบนาซชี่

อนุกรมไฟโบนาซชี่ (Fibonacci series) ได้แก่ 0, 1, 1, 2, 3, 5, 8, 13, 21, ... โดยเริ่มจาก 0 และ 1 เลขไฟโบนาซชี่ลำดับต่อไปคือ ผลบวกของเลขไฟโบนาซชี่ 2 ตัวก่อนหน้า

เลขไฟโบนาซชี่ กำหนดโดย

fibonacci(0) = 0

fibonacci(1) = 1

fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)

```
#include <stdio.h>
long fib(long);
void main()
{
    long num, answer;
    printf("Enter a number : ");
    scanf("%ld", &num);
    answer = fib(num);
    printf("Fibonacci(%2ld) = %2ld \n", num, answer);
}
long fib(long n)
{
    if(n == 0 || n == 1)
        return (n);
    else
        return (fib(n-1) + fib(n-2));       /* ฟังก์ชัน fib() เรียกตัวเอง */
}
```

ถ้า num มีค่า 4

การทำงานคือ fib(4) ส่งค่า (fib(3) + fib(2)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

fib(3) ส่งค่า (fib(2) + fib(1)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

fib(2) ส่งค่า (fib(1) + fib(0)) กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

fib(1) ส่งค่า 1 กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

fib(0) ส่งค่า 0 กลับไปยังตำแหน่งที่เรียกฟังก์ชันนี้

ดังนั้น fib(2) ได้ผลลัพธ์ fib(1) + fib(0) คือ 1+0 เท่ากับ 1

fib(3) ได้ผลลัพธ์ fib(2) + fib(1) คือ 1+1 เท่ากับ 2

fib(4) ได้ผลลัพธ์ fib(3) + fib(2) คือ 2+1 เท่ากับ 3

ผลการกระทำการ

Fibonacci(4) = 3

4.3 ฟังก์ชันจากคลัง (library function)

นอกเหนือจากการกำหนดฟังก์ชันเองแล้ว สามารถเรียกใช้ฟังก์ชันจากคลังได้ การเรียกฟังก์ชันจากคลังเหมือนการเรียกฟังก์ชันที่กำหนดเอง แต่ไม่ต้องกำหนดฟังก์ชันนั้น ๆ ในโปรแกรม โดยต้องมีข้อความสั่งตัวประมวลผลก่อน #include ตอนต้นของโปรแกรม ตามรูปแบบดังนี้

`#include <ชื่อแฟ้มข้อมูลที่เก็บการทำงานของฟังก์ชัน>`

ตัวอย่าง 4.17

```
#include <stdio.h>      /* ใช้เมื่อต้องการเรียกฟังก์ชันในการรับและแสดงผล */
#include <math.h>       /* ใช้เมื่อต้องการเรียกฟังก์ชันการทำงานทางคณิตศาสตร์ */
#include <string.h>     /* ใช้เมื่อต้องการเรียกฟังก์ชันการทำงานกับสายอักขระ (string) */
```

ในที่นี้ขอยกตัวอย่างฟังก์ชันการทำงานทางคณิตศาสตร์ ในแฟ้มข้อมูล math.h

ฟังก์ชัน	ความหมาย
abs(i)	ค่าสัมบูรณ์ของ i i เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนเต็ม
fabs(x)	ค่าสัมบูรณ์ของ x x เป็นตัวแปร หรือค่าคงที่ชนิดจำนวนจริง
sin(x)	ค่า sine ของมุม x x มีหน่วยเป็นเรเดียน

ฟังก์ชัน	ความหมาย
cos(x)	ค่า cosine ของมุม x x มีหน่วยเป็นเรเดียน
tan(x)	ค่า tangent ของมุม x x มีหน่วยเป็นเรเดียน
exp(x)	ค่า e^x x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
pow(x, y)	ค่า x^y x และ y เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
sqrt(x)	ค่ารากที่สองของ x x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
log(x)	ค่าลอการิทึมธรรมชาติ (natural logarithm) ของ x (ฐาน e) x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
log10(x)	ค่าลอการิทึม (logarithm) ของ x (ฐาน 10) x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
ceil(x)	ค่าจำนวนเต็มต่ำสุดที่มีค่าไม่น้อยกว่า x x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
floor(x)	ค่าจำนวนเต็มสูงสุดที่มีค่าไม่มากกว่า x x เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง
fmod(x, y)	เศษที่ได้ (เป็นจำนวนจริง) หลังการหารของ x/y x และ y เป็นตัวแปรหรือค่าคงที่ชนิดจำนวนจริง

ตัวอย่าง 4.18

โปรแกรมเพื่อพิมพ์รากที่สองของ 1.0, 2.5, 4.0, 5.5, ..., 10.0 ตามลำดับ

```
#include <stdio.h>
#include <math.h>
void main()
{
    float x;
    printf(" x    square root of x \n");
    for(x = 1.0; x <= 10.0; x = x + 1.5)
    printf("%5.2f %10.2f \n", x, sqrt(x));
}
```

ผลการกระทำการ

x	square root of x
1.00	1.00
2.50	1.58
4.00	2.00
5.50	2.35
7.00	2.65
8.50	2.92
10.00	3.16

ตัวอย่าง 4.19

แสดงการใช้ฟังก์ชัน ceil() และ floor()

```
#include <stdio.h>
#include <math.h>
void main()                    /* ตัวอย่างฟังก์ชัน ceil() และ floor() */
{
    float cf1 = -7.9, cf2 = 7.9, cf3 = 17.2;
    printf("ceiling of %.2f is %.2f \n", cf1, ceil(cf1));
    printf("ceiling of %.2f is %.2f \n", cf2, ceil(cf2));
    printf("ceiling of %.2f is %.2f \n", cf3, ceil(cf3));
    printf("floor of %.2f is %.2f \n", cf1, floor(cf1));
    printf("floor of %.2f is %.2f \n", cf2, floor(cf2));
    printf("floor of %.2f is %.2f \n", cf3, floor(cf3));
}
```

ผลการกระทำการ

ceiling of –7.90 is –7.00

ceiling of 7.90 is 8.00

ceiling of 17.20 is 18.00

floor of –7.90 is –8.00

floor of 7.90 is 7.00

floor of 17.20 is 17.00

ตัวอย่าง 4.20

แสดงการใช้ฟังก์ชัน fmod()

```
#include <stdio.h>
#include <math.h>
void main()
{
    float x, y;
    x = 14.0;
    y = 3.0;
    printf("remainder of %.2f / %.2f is %.2f \n", x, y, fmod(x,y));
    x = 14.3;
    y = 3.7;
    printf("remainder of %.2f / %.2f is %.2f \n", x, y, fmod(x,y));
}
```

ผลการกระทำการ

remainder of 14.00 / 3.00 is 2.00

remainder of 14.30 / 3.70 is 3.20

แบบฝึกหัดบทที่ 4

1. เขียนฟังก์ชันเพื่อคำนวณพื้นที่วงกลม โดยให้พารามิเตอร์คือ รัศมีของวงกลม
   จากสูตร พื้นที่วงกลม = ¶ * รัศมี^{2}

2. เขียนโปรแกรมเพื่อคำนวณค่าจอดรถ ตามอัตราดังนี้
   1 ชั่วโมงแรก ฟรี
   ชั่วโมงต่อ ๆ ไป ชั่วโมงละ 30 บาท
   เศษของชั่วโมง คิดเป็น 1 ชั่วโมง
   ให้ข้อมูลเข้าคือ จำนวนชั่วโมงและนาทีที่จอดรถ

3. เขียนโปรแกรมเพื่อรับข้อมูลชนิดจำนวนเต็มเป็นจำนวนข้อมูล รับข้อมูลตามจำนวนที่ระบุ แล้วเขียนฟังก์ชันเพื่อหาค่าต่ำสุด สูงสุด และค่าเฉลี่ย

4. เขียนโปรแกรมรับข้อมูลหน่วยวัดซึ่งมีหน่วยเป็นนิ้ว เขียนฟังก์ชันแปลงนิ้วเป็นฟุต (1 ฟุต = 12 นิ้ว) ฟังก์ชันแปลงนิ้วเป็นเซนติเมตร (2.54 ซม. = 1 นิ้ว) และฟังก์ชันแปลงนิ้วเป็นหลา (1 หลา = 36 นิ้ว)

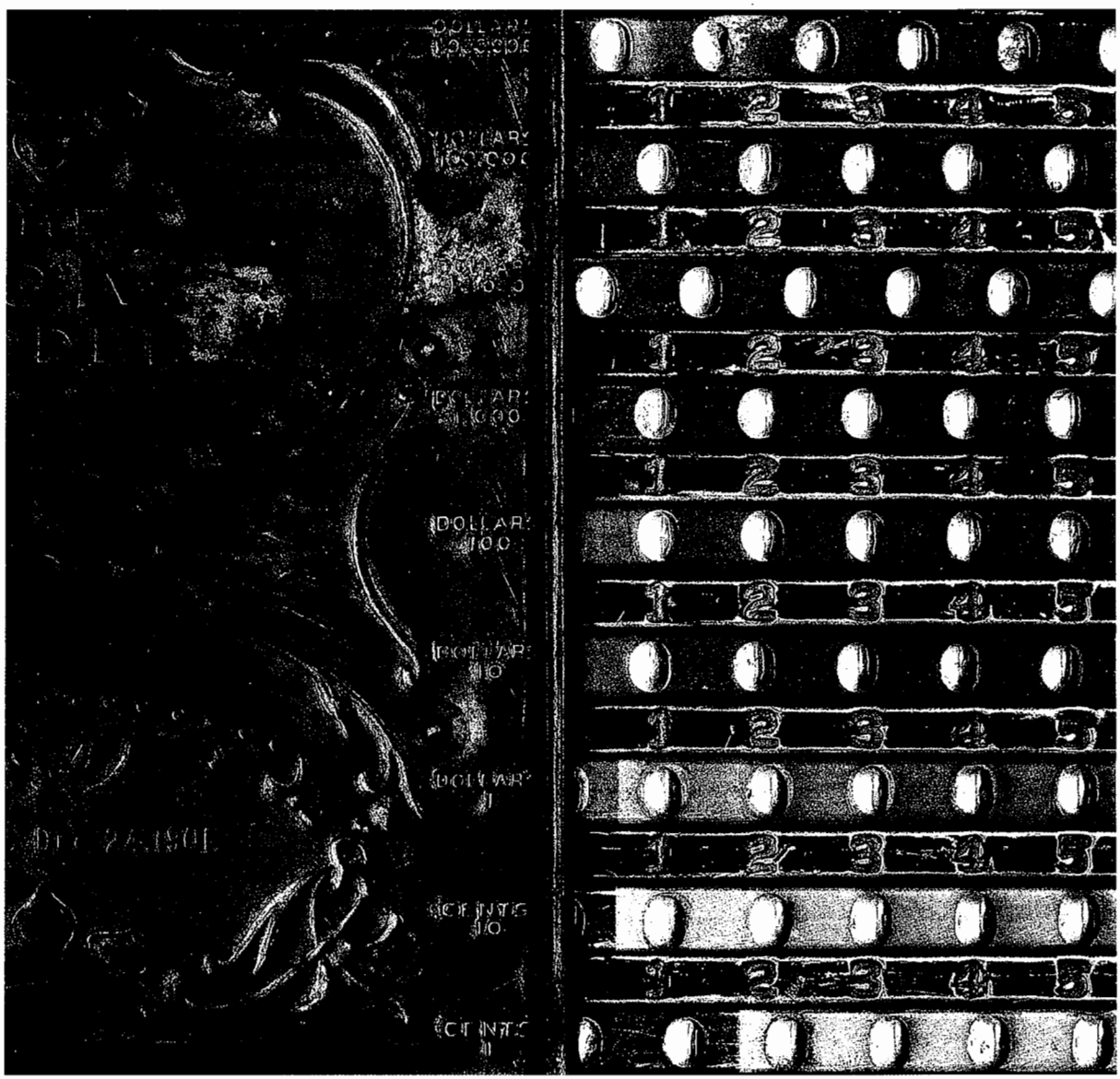
5. เขียนโปรแกรมเพื่อรับข้อมูลเป็นคะแนนสอบของนักเรียน แล้วแสดงระดับคะแนนของนักเรียน ตามเกณฑ์ต่อไปนี้

ตั้งแต่ 90 คะแนนขึ้นไป	ให้ระดับคะแนน A
80 - 89 คะแนน	ให้ระดับคะแนน B
70 - 79 คะแนน	ให้ระดับคะแนน C
60 - 69 คะแนน	ให้ระดับคะแนน D
50 คะแนนลงมา	ให้ระดับคะแนน F

   โดยให้โปรแกรมประกอบด้วยฟังก์ชันอย่างน้อย 3 ฟังก์ชัน

6. เขียนโปรแกรมเพื่อรับข้อมูลชนิดจำนวนเต็ม เรียกฟังก์ชันเรียกซ้ำเพื่อแสดงข้อมูลในลำดับตรงกันข้าม เช่น ถ้าข้อมูลเข้าคือ 9378 ผลลัพธ์คือ 8739

เครื่องคำนวณ Millionär จดทะเบียนเครื่องประดิษฐ์โดย Otto Steiger คศ.1893



เครื่องคำนวณอย่างง่ายผลิตโดย C.E. Locke ค.ศ. 1900

บทที่ 5
แถวลำดับ

แถวลำดับ (array) คือ เนื้อที่ในหน่วยความจำที่เรียงต่อกัน ใช้ในการเก็บข้อมูลชนิดเดียวกันหลายจำนวน

การประกาศตัวแปรเป็นแถวลำดับจะทำให้ตัวแปรนั้นเก็บข้อมูลได้หลายจำนวนและสามารถเข้าถึงข้อมูลได้โดยใช้ตัวแปรชื่อเดียวกันตามด้วยดรรชนีกำกับ (subscript) ที่บอกถึงตำแหน่งของข้อมูลในแถวลำดับนั้น

5.1 การประกาศตัวแปรแถวลำดับ

ประกาศตัวแปรแถวลำดับโดยใช้ข้อความสั่งประกาศตัวแปร ตามรูปแบบดังนี้

ชนิดข้อมูล ชื่อตัวแปรแถวลำดับ[n];

โดยที่

ชนิดข้อมูล คือ ชนิดข้อมูลที่เก็บในตัวแปรแถวลำดับ

n คือ จำนวนสมาชิกของตัวแปรแถวลำดับนี้ เป็นจำนวนเต็มบวก

ตัวอย่าง 5.1

การประกาศตัวแปรแถวลำดับเพื่อเก็บข้อมูลชนิดต่าง ๆ

```
int n[10];
```

ประกาศตัวแปรแถวลำดับชื่อ n ประกอบด้วยสมาชิก 10 หน่วย และแต่ละหน่วยใช้เก็บข้อมูลชนิดจำนวนเต็ม

```
char a[20];
```

ประกาศตัวแปรแถวลำดับชื่อ a ประกอบด้วยสมาชิก 20 หน่วย และแต่ละหน่วยใช้เก็บข้อมูลชนิดอักขระ

```
float g[5];
```

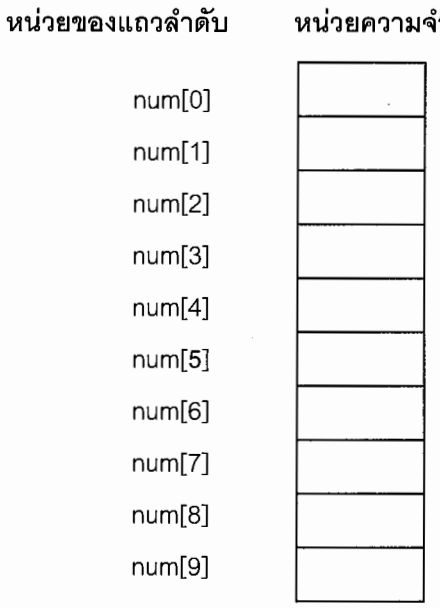
ประกาศตัวแปรแถวลำดับชื่อ g ประกอบด้วยสมาชิก 5 หน่วย และแต่ละหน่วยใช้เก็บข้อมูลชนิดจำนวนจริง

ตัวอย่าง 5.2

การประกาศตัวแปรแถวลำดับ

```
int num[10];
```

ประกาศตัวแปรแถวลำดับชื่อ num ประกอบด้วยสมาชิก 10 หน่วย และแต่ละหน่วยเก็บเลขจำนวนเต็ม โดยลักษณะของตัวแปรแถวลำดับในหน่วยความจำจะเรียงต่อกันดังนี้



5.2 การอ้างถึงสมาชิกแต่ละหน่วยของตัวแปรแถวลำดับ

สามารถอ้างถึงสมาชิกแต่ละหน่วยของแถวลำดับ ดังนี้

ชื่อตัวแปรแถวลำดับ[ดรรชนีกำกับ]

ดรรชนีกำกับ หมายถึง นิพจน์หรือตัวแปรชนิดจำนวนเต็ม มีค่าตั้งแต่ 0 จนถึงจำนวนสมาชิกลบหนึ่ง

ตัวอย่าง 5.3

การอ้างถึงสมาชิกในแถวลำดับ

```
int n[5];
n[4] = 25;
```

สามารถอ้างถึงสมาชิกแต่ละหน่วยของตัวแปรแถวลำดับ n[ ] โดยใช้ n[0], n[1], n[2], n[3] และ n[4]

ตัวอย่าง 5.4

โปรแกรมเพื่ออ่านข้อมูลเข้าเป็นจำนวนเต็ม 10 จำนวน แล้วหาผลรวมของเลขเหล่านั้น

```
#include <stdio.h>
void main()
{
    int num[10], sum, i;
    for(i = 0; i < 10; i++)
        scanf("%d", &num[i]);
    sum = 0;
    for(i = 0; i < 10; i++)
        sum = sum + num[i];
    printf("sum is %d\n", sum);
}
```

ตัวอย่าง 5.5

โปรแกรมเพื่ออ่านข้อมูลชนิดจำนวนจริง 100 จำนวน หาค่าเฉลี่ย และจำนวนข้อมูลที่มีค่าสูงกว่าค่าเฉลี่ย

```
#include <stdio.h>
void main()
{
    int n, count, k;
    float avg, d, sum = 0;
    float large;
    float list[100];
    /* อ่านค่าจำนวนข้อมูลเก็บไว้ในตัวแปร n */
    scanf("%d", &n);
    /* อ่านข้อมูลจำนวนจริงเก็บไว้ในแถวลำดับและหาผลรวม */
    for(count = 0; count < n; count++)
    {
        scanf("%f", &list[count]);
        sum = sum + list[count];
    }
    /* คำนวณและแสดงค่าเฉลี่ย */
    avg = sum / n;
    printf("The average is %f \n", avg);
    /* นับจำนวนข้อมูลที่มีค่าสูงกว่าค่าเฉลี่ย */
    k = 0;
```

```
    for(count = 0; count < n; count++)
        if(list[count] > avg)
            k = k + 1;
    printf("number above average is %d\n",k);
    /* หาค่าสูงสุด */
    large = list[0];
    for(count = 1; count < n; count++)
        if(list[count] > large)
            large = list[count];
    printf("The largest number is %f\n", large);
}
```

ตัวอย่าง 5.6

โปรแกรมเพื่ออ่านข้อมูลชนิดจำนวนจริง 25 จำนวน แล้วพิมพ์ตัวเลขเหล่านั้นในลำดับตรงกันข้ามกับที่อ่านเข้ามา

```
#include <stdio.h>
void main()
{
    int i;
    float a[25];
    for(i = 0; i < 25; i++)
        scanf("%f", &a[i]);
    for(i = 24; i >= 0; i--)
        printf("%f\n", a[i]);
}
```

ตัวอย่าง 5.7

โปรแกรมเพื่ออ่านข้อมูลเข้าเป็นคะแนนสอบ ซึ่งเป็นจำนวนจริงแล้วเก็บไว้ในตัวแปรแถวลำดับ score[ ] อ่านจนกว่าข้อมูลเข้าจะมีค่าน้อยกว่าศูนย์ แล้วพิมพ์จำนวนนักเรียนที่เข้าสอบทั้งหมด ซึ่งมีไม่เกิน 500 คน

```
#include <stdio.h>
void main()
{
    int i = 0;
    float score[500], temp;
    scanf("%f", &temp);
```

```
        while(temp >= 0)
        {
            score[i] = temp;
            i = i + 1;
            scanf("%f", &temp);
        }
        printf("Number of students taking the exam is %d", i);
}
```

5.3 ตัวแปรแถวลำดับเพื่อเก็บข้อมูลชนิดอักขระ

ในการเก็บข้อมูลที่เป็นสายอักขระ (string) ต้องเก็บแต่ละอักขระไว้ในตัวแปรแถวลำดับ โดยที่อักขระหนึ่งตัวคือสมาชิกหน่วยหนึ่งของแถวลำดับซึ่งใช้เนื้อที่ 1 ไบต์ และจบสายอักขระด้วย '\0'

ตัวอย่าง 5.8

การเก็บสายอักขระในหน่วยความจำ

สายอักขระ "string" จะเก็บในตัวแปรแถวลำดับดังนี้

หน่วยของแถวลำดับ	หน่วยความจำ
0	s
1	t
2	r
3	i
4	n
5	g
6	\0

ตัวอย่าง 5.9

การเก็บสายอักขระในหน่วยความจำ

สายอักขระ "c" จะเก็บในตัวแปรแถวลำดับดังนี้

หน่วยของแถวลำดับ	หน่วยความจำ
0	c
1	\0

การกำหนดตัวแปรแถวลำดับ เพื่อเก็บสายอักขระทำโดยใช้คำสั่งประกาศตัวแปรแถวลำดับดังนี้

char ชื่อตัวแปรลำดับ[n];

n คือ จำนวนเต็มบวก ที่มีค่ามากกว่าหรือเท่ากับ จำนวนอักขระบวกด้วยหนึ่ง (สำหรับ '\0')

การอ่านหรือแสดงสายอักขระให้ใช้ %s และใช้ชื่อตัวแปรแถวลำดับที่เก็บสายอักขระนี้

ตัวอย่าง 5.10

การกำหนดค่าให้กับสมาชิกของแถวลำดับ

```
#include <stdio.h>
void main()
{
    char message[14];
    message[0] = 'I';
    message[1] = ' ';
    message[2] = 'a';
    message[3] = 'm';
    message[4] = ' ';
    message[5] = 'f';
    message[6] = 'i';
    message[7] = 'n';
    message[8] = 'e';
    message[9] = '.';
    message[10] = '\0';
    printf("%s", message);
}
```

ผลการกระทำการ

I am fine.

ตัวอย่าง 5.11

การรับและแสดงสายอักขระ

```
#include <stdio.h>
void main()
{
    char string[64];
    scanf("%s", string);
    printf("%s", string);
}
```

ถ้าข้อมูลเข้าตั้งแต่สดมภ์แรกคือ Computer Science

string[0] มีค่าเป็น 'C' string[1] มีค่าเป็น 'o' string[2] มีค่าเป็น 'm' string[3] มีค่าเป็น 'p'
string[4] มีค่าเป็น 'u' string[5] มีค่าเป็น 't' string[6] มีค่าเป็น 'e' string[7] มีค่าเป็น 'r'
string[8] มีค่าเป็น '\0'

ผลการกระทำการ

Computer

ฟังก์ชัน scanf() จะอ่านสายอักขระจนกว่าจะเจอช่องว่าง จึงจะถือว่าจบสายอักขระนั้น ในกรณีที่ระบุจำนวนสดมภ์ก็จะอ่านจนถึงช่องว่างเท่านั้น ถ้าเจอช่องว่างก่อนอ่านครบตามจำนวนสดมภ์ที่ระบุ

ตัวอย่าง 5.12

การรับและแสดงสายอักขระ

```
#include <stdio.h>
void main()
{
    char string[60];
    scanf("%7s", string);
    printf("%s", string);
}
```

ถ้าข้อมูลเข้าคือ Computer Science

ผลการกระทำการ

Compute

ถ้าข้อมูลเข้าคือ the end

ผลการกระทำการ

the

5.4 ตัวแปรแถวลำดับสองมิติ

ตัวแปรแถวลำดับ 2 มิติ ใช้เก็บข้อมูลชนิดต่าง ๆ โดยอาจมองเป็นรูปตารางข้อมูล การประกาศตัวแปรแถวลำดับ 2 มิติ มีรูปแบบดังนี้

ชนิดข้อมูล ชื่อตัวแปรแถวลำดับ[nrows][ncolumns];

ชนิดข้อมูล	คือ ชนิดของข้อมูลที่เก็บในตัวแปรแถวลำดับ
nrows	หมายถึง จำนวนแถวเป็นจำนวนเต็มบวก
ncolumns	หมายถึง จำนวนสดมภ์เป็นจำนวนเต็มบวก

จำนวนหน่วยของตัวแปรแถวลำดับ คือ nrows*ncolumns

ตัวอย่าง 5.13

การประกาศตัวแปรแถวลำดับสองมิติ

```
float a[3][5];
```

ประกาศตัวแปรแถวลำดับ a[ ][ ] มีสมาชิก 3*5=15 หน่วย แต่ละหน่วยเก็บจำนวนจริง

คือ **สดมภ์** 0 1 2 3 4

แถว

0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

```
int t[5][4];
```

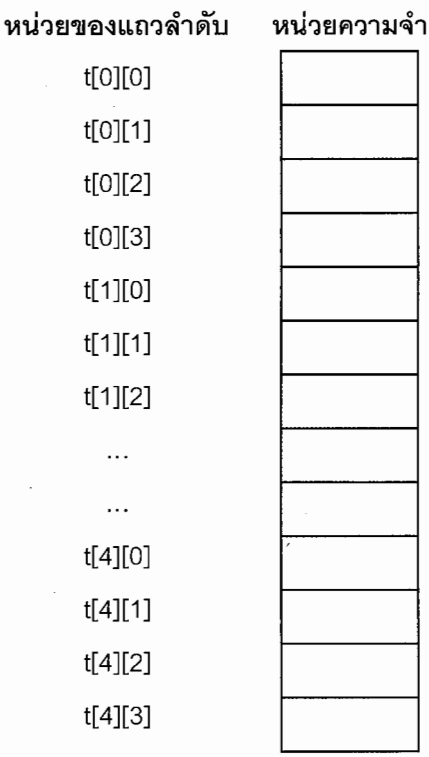
ประกาศตัวแปรแถวลำดับ t[ ][ ] มีสมาชิก 5*4=20 หน่วย แต่ละหน่วยเก็บจำนวนเต็ม

คือ **สดมภ์** 0 1 2 3

แถว

0	t[0][0]	t[0][1]	t[0][2]	t[0][3]
1	t[1][0]	t[1][1]	t[1][2]	t[1][3]
2	t[2][0]	t[2][1]	t[2][2]	t[2][3]
3	t[3][0]	t[3][1]	t[3][2]	t[3][3]
4	t[4][0]	t[4][1]	t[4][2]	t[4][3]

การเก็บข้อมูลของตัวแปรแถวลำดับ 2 มิติ ในหน่วยความจำ จะเก็บตามลำดับแถวดังนี้



ตัวอย่าง 5.14

โปรแกรมเพื่ออ่านข้อมูลชนิดจำนวนเต็มเก็บไว้ในตัวแปรแถวลำดับ b[ ][ ] ที่มีการจองเนื้อที่ 5 แถว และ 4 สดมภ์ แล้วหาผลรวมของข้อมูลดังกล่าว

```
#include <stdio.h>
void main()
{
    int b[5][4];
    int i, j, sum;
    sum = 0;
    for(i = 0; i < 5; i++)
        for(j = 0; j < 4; j++)
        {
            scanf("%d", &b[i][j]);
            sum = sum + b[i][j];
        }
    printf("The sum is %d\n", sum);
}
```

ตัวอย่าง 5.15

โปรแกรมเพื่ออ่านคะแนนสอบ (จำนวนจริง) จำนวน 5 ครั้งของนักเรียน 50 คนมาเก็บไว้ในตัวแปรแถวลำดับ score[ ][ ] แล้วหาคะแนนรวมของแต่ละคน

```
#include <stdio.h>
void main()
{
    float score[50][5];          /* 50 แถวแทนจำนวนนักเรียน 5 สดมภ์แทนจำนวนครั้งที่สอบ */
    float total;
    int i, j;
    for(i = 0; i < 50; i++)
        for(j = 0; j < 5; j++)
            scanf("%f", &score[i][j]);
    for(i = 0; i < 50; i++)
    {
        total = 0;               /* คะแนนรวมของนักเรียนแต่ละคน */
        for(j = 0; j < 5; j++)
            total = total + score[i][j];
        printf("student #%d gets %f points\n", i+1, total);
    }
}
```

กำหนดให้ข้อมูลเข้าคือ คะแนนสอบแต่ละครั้งของนักเรียนคนที่ 1 คนที่ 2 จนถึงคนที่ 50 ตามลำดับ

ตัวอย่าง 5.16

โปรแกรมเพื่ออ่านข้อมูลเกี่ยวกับจำนวนคนในบ้าน 6 หลัง เพื่อเก็บในแถวลำดับดังนี้

	อายุ (ปี)			
	0-5	6-12	13-19	20 ขึ้นไป
บ้านหลังที่ 1	.	.	.	.
บ้านหลังที่ 2	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
บ้านหลังที่ 6	.	.	.	.

แล้วหาจำนวนเด็กอายุ 6-12 ปี ทั้งหมด

```
#include <stdio.h>
void main()
{
    int s[6][4], sum, i, k;
    sum = 0;
    for(i = 0; i < 6; i++)
        for(k = 0; k < 4; k++)
            scanf("%d", &s[i][k]);
    /* คำนวณจำนวนเด็กอายุ 6-12 ปี */
    for(i = 0; i < 6; i++)
        sum = sum + s[i][1];
    printf("Total children age 6-12 years is %d\n", sum);
}
```

5.5 ตัวแปรแถวลำดับสามมิติ

ตัวแปรแถวลำดับ 3 มิติ ใช้เก็บข้อมูลชนิดต่าง ๆ โดยอาจมองเป็นรูปตารางข้อมูลหลายตาราง การประกาศตัวแปรแถวลำดับ 3 มิติ มีรูปแบบดังนี้

ชนิดข้อมูล ชื่อตัวแปรแถวลำดับ[ntabs][nrows][ncolumns];

ชนิดข้อมูล	คือ ชนิดของข้อมูลที่เก็บในตัวแปรแถวลำดับ
ntabs	หมายถึง จำนวนตารางเป็นจำนวนเต็มบวก
nrows	หมายถึง จำนวนแถวเป็นจำนวนเต็มบวก
ncolumns	หมายถึง จำนวนสดมภ์เป็นจำนวนเต็มบวก

จำนวนหน่วยของตัวแปรแถวลำดับคือ ntabs*nrows*ncolumns

ตัวอย่าง 5.17

การประกาศตัวแปรแถวลำดับสามมิติ

```
int t[2][3][4];
```

ประกาศตัวแปรแถวลำดับ t[ ][ ][ ] มีสมาชิก 2*3*4 = 24 หน่วย มีจำนวน 2 ตาราง แต่ละตารางมี 3 แถว และ 4 สดมภ์ โดยที่แต่ละหน่วยเก็บข้อมูลชนิดจำนวนเต็มดังนี้

ตารางที่ 0

t[0][0][0]	t[0][0][1]	t[0][0][2]	t[0][0][3]
t[0][1][0]	t[0][1][1]	t[0][1][2]	t[0][1][3]
t[0][2][0]	t[0][2][1]	t[0][2][2]	t[0][2][3]

ตารางที่ 1

t[1][0][0]	t[1][0][1]	t[1][0][2]	t[1][0][3]
t[1][1][0]	t[1][1][1]	t[1][1][2]	t[1][1][3]
t[1][2][0]	t[1][2][1]	t[1][2][2]	t[1][2][3]

ตัวอย่าง 5.18

โปรแกรมเพื่ออ่านข้อมูลชนิดจำนวนเต็มเก็บในตัวแปรแถวลำดับ tab[ ][ ][ ] แล้วหาผลรวมของข้อมูลในแถวลำดับ

```
#include <stdio.h>
void main()
{
    int tab[3][4][5], sum = 0;
    int i, j, k;
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
            for(k = 0; k < 5; k++)
            {
                scanf("%d", &tab[i][j][k]);
                sum = sum + tab[i][j][k];
            }
    printf("The sum is %d\n", sum);
}
```

5.6 การกำหนดค่าเริ่มต้นให้กับแถวลำดับ

การกำหนดค่าเริ่มต้นให้กับตัวแปรแถวลำดับสามารถทำได้ตามรูปแบบดังนี้

ชนิดข้อมูล ชื่อตัวแปรแถวลำดับ = { ค่าเริ่มต้น };

ถ้ามีค่าเริ่มต้นมากกว่า 1 ค่า ให้คั่นด้วยจุลภาค

จำนวนค่าเริ่มต้นอาจมีน้อยกว่าหรือเท่ากับจำนวนหน่วยของตัวแปรแถวลำดับก็ได้

ตัวอย่าง 5.19

กำหนดค่าเริ่มต้นให้กับตัวแปรแถวลำดับ

```
void main()
{
    int num[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    …
    …
}
```

กำหนดค่าเริ่มต้นให้กับตัวแปรแถวลำดับ num[ ] ดังนี้

num[0] มีค่าเริ่มต้น 0 num[1] มีค่าเริ่มต้น 1
num[2] มีค่าเริ่มต้น 2 num[3] มีค่าเริ่มต้น 3
num[4] มีค่าเริ่มต้น 4 num[5] มีค่าเริ่มต้น 5
num[6] มีค่าเริ่มต้น 6 num[7] มีค่าเริ่มต้น 7
num[8] มีค่าเริ่มต้น 8 num[9] มีค่าเริ่มต้น 9

การให้ค่าเริ่มต้นกับตัวแปรแถวลำดับที่เก็บข้อมูลชนิดอักขระ สามารถทำโดยกำหนดค่าให้กับแต่ละหน่วยของตัวแปรแถวลำดับ โดยกำหนดค่าเริ่มต้นในปีกกา หรือเขียนเป็นสายอักขระในฟันหนู (" ") ก็ได้

ตัวอย่าง 5.20

การกำหนดค่าเริ่มต้นชนิดอักขระให้กับแถวลำดับ

```
void main()
{
    char fname[10] = {'s', 'w', 'e', 'e', 't', '\0'};
    char last[10] = "Honey";
    …
    …
}
```

กำหนดค่าเริ่มต้นให้ตัวแปรแถวลำดับ fname[ ] ดังนี้

fname[0] มีค่าเริ่มต้น 's' fname[1] มีค่าเริ่มต้น 'w'
fname[2] มีค่าเริ่มต้น 'e' fname[3] มีค่าเริ่มต้น 'e'
fname[4] มีค่าเริ่มต้น 't' fname[5] มีค่าเริ่มต้น '\0'

กำหนดค่าเริ่มต้นให้ตัวแปรแถวลำดับ last[ ] ดังนี้

last[0] มีค่าเริ่มต้น 'H' last[1] มีค่าเริ่มต้น 'o'
last[2] มีค่าเริ่มต้น 'n' last[3] มีค่าเริ่มต้น 'e'
last[4] มีค่าเริ่มต้น 'y' last[5] มีค่าเริ่มต้น '\0'

ในกรณีที่ไม่กำหนดจำนวนสมาชิกของตัวแปรแถวลำดับในการให้ค่าเริ่มต้น ตัวแปลโปรแกรมจะกำหนดจำนวนสมาชิกของตัวแปรแถวลำดับให้ โดยนับจากจำนวนค่าเริ่มต้น

ตัวอย่าง 5.21

การกำหนดค่าเริ่มต้นให้กับแถวลำดับโดยไม่ระบุจำนวนสมาชิก

```
void main()
{
    int guess[] = {1, 2, 5, 7, 9};
    …
    …
}
```

ตัวแปลโปรแกรมจะกำหนดตัวแปรแถวลำดับ guess[ ] ว่ามีสมาชิกทั้งหมด 5 หน่วย และให้ค่าเริ่มต้นดังนี้

guess[0] มีค่าเริ่มต้น 1 guess[1] มีค่าเริ่มต้น 2

guess[2] มีค่าเริ่มต้น 5 guess[3] มีค่าเริ่มต้น 7

guess[4] มีค่าเริ่มต้น 9

การกำหนดค่าเริ่มต้นกับตัวแปรแถวลำดับ 2 มิติ ทำได้ด้วยการกำหนดค่าเริ่มต้นของแต่ละแถวในปีกกา หรือถ้าเป็นข้อมูลชนิดอักขระ อาจกำหนดในฟันหนูก็ได้ โดยจะให้ค่าเริ่มต้นเรียงตามลำดับแถว และในแต่ละแถวเรียงตามลำดับสดมภ์

ตัวอย่าง 5.22

การกำหนดค่าเริ่มต้นให้กับตัวแปรแถวลำดับ 2 มิติ

```
void main()
{
    int emp[3][7] = { {10, 20, 30, 40, 50, 60, 70},        /* แถว 0 */
                      {11, 21, 31, 41, 51, 61, 71},        /* แถว 1 */
                      {12, 22, 32, 42, 52, 62, 72}         /* แถว 2 */
                    };
    …
    …
}
```

กำหนดค่าเริ่มต้นให้ตัวแปรแถวลำดับ emp[ ][ ] ดังนี้

emp[0][0] มีค่าเริ่มต้น 10 emp[0][1] มีค่าเริ่มต้น 20 emp[0][2] มีค่าเริ่มต้น 30

emp[0][3] มีค่าเริ่มต้น 40 emp[0][4] มีค่าเริ่มต้น 50 emp[0][5] มีค่าเริ่มต้น 60

emp[0][6] มีค่าเริ่มต้น 70

emp[1][0] มีค่าเริ่มต้น 11 emp[1][1] มีค่าเริ่มต้น 21 emp[1][2] มีค่าเริ่มต้น 31
emp[1][3] มีค่าเริ่มต้น 41 emp[1][4] มีค่าเริ่มต้น 51 emp[1][5] มีค่าเริ่มต้น 61
emp[1][6] มีค่าเริ่มต้น 71
emp[2][0] มีค่าเริ่มต้น 12 emp[2][1] มีค่าเริ่มต้น 22 emp[2][2] มีค่าเริ่มต้น 32
emp[2][3] มีค่าเริ่มต้น 42 emp[2][4] มีค่าเริ่มต้น 52 emp[2][5] มีค่าเริ่มต้น 62
emp[2][6] มีค่าเริ่มต้น 72

ตัวอย่าง 5.23

การกำหนดค่าเริ่มต้นชนิดอักขระ

```
void main()
{
    char paint[2][5] = { {'r', 'e', 'd', '\0'},
                         {'p', 'i', 'n', 'k', '\0'}
                       };
    ...
    ...
}
```

หรือ

```
void main()
{
    char paint[2][5] = {"red", "pink"};
    ...
    ...
}
```

กำหนดค่าเริ่มต้นให้ตัวแปรแถวลำดับ paint[ ][ ] ดังนี้

paint[0][0] มีค่าเริ่มต้น 'r' paint[0][1] มีค่าเริ่มต้น 'e'
paint[0][2] มีค่าเริ่มต้น 'd' paint[0][3] มีค่าเริ่มต้น '\0'
paint[1][0] มีค่าเริ่มต้น 'p' paint[1][1] มีค่าเริ่มต้น 'i'
paint[1][2] มีค่าเริ่มต้น 'n' paint[1][3] มีค่าเริ่มต้น 'k'
paint[1][4] มีค่าเริ่มต้น '\0'

การกำหนดค่าเริ่มต้นกับตัวแปรแถวลำดับ 3 มิติ ทำโดยกำหนดค่าเริ่มต้นของแต่ละตารางในปีกกา สำหรับแต่ละตารางกำหนดค่าเริ่มต้นของแต่ละแถวในปีกกา โดยจะให้ค่าเริ่มต้นเรียงตามลำดับตาราง ในแต่ละตารางเรียงตามลำดับแถว และในแต่ละแถวเรียงตามลำดับสดมภ์

ตัวอย่าง 5.24

การกำหนดค่าเริ่มต้นให้กับตัวแปรแถวลำดับ 3 มิติ

```
void main() .
{
    int ta[2][3][4] = {
                        {                           /* ตารางที่ 0 */
                          {1, 2, 3, 4},             /* แถวที่ 0 */
                          {5, 6, 7, 8},             /* แถวที่ 1 */
                          {9, 10, 11, 12}           /* แถวที่ 2 */
                        },
                        {                           /* ตารางที่ 1 */
                          {13, 14, 15, 16},         /* แถวที่ 0*/
                          {17, 18, 19, 20},         /* แถวที่ 1 */
                          {21, 22, 23, 24}          /* แถวที่ 2 */
                        }
                      };
    ...
    ...
}
```

กำหนดค่าเริ่มต้นให้ตัวแปรแถวลำดับ ta[ ][ ][ ] ดังนี้

ในตารางที่ 0

ta[0][0][0] มีค่าเริ่มต้น 1 ta[0][0][1] มีค่าเริ่มต้น 2 ta[0][0][2] มีค่าเริ่มต้น 3
ta[0][0][3] มีค่าเริ่มต้น 4 ta[0][1][0] มีค่าเริ่มต้น 5 ta[0][1][1] มีค่าเริ่มต้น 6
ta[0][1][2] มีค่าเริ่มต้น 7 ta[0][1][3] มีค่าเริ่มต้น 8 ta[0][2][0] มีค่าเริ่มต้น 9
ta[0][2][1] มีค่าเริ่มต้น 10 ta[0][2][2] มีค่าเริ่มต้น 11 ta[0][2][3] มีค่าเริ่มต้น 12

ในตารางที่ 1

ta[1][0][0] มีค่าเริ่มต้น 13 ta[1][0][1] มีค่าเริ่มต้น 14 ta[1][0][2] มีค่าเริ่มต้น 15
ta[1][0][3] มีค่าเริ่มต้น 16 ta[1][1][0] มีค่าเริ่มต้น 17 ta[1][1][1] มีค่าเริ่มต้น 18
ta[1][1][2] มีค่าเริ่มต้น 19 ta[1][1][3] มีค่าเริ่มต้น 20 ta[1][2][0] มีค่าเริ่มต้น 21
ta[1][2][1] มีค่าเริ่มต้น 22 ta[1][2][2] มีค่าเริ่มต้น 23 ta[1][2][3] มีค่าเริ่มต้น 24

5.7 ฟังก์ชันเพื่อทำงานกับสายอักขระ

การเรียกใช้ฟังก์ชันจากคลังเพื่อทำงานกับสายอักขระ จำเป็นจะต้องมีข้อความสั่งตัวประมวลผลก่อน #include <string.h> อยู่ที่ต้นของโปรแกรม

5.7.1 ฟังก์ชัน strcat()

ฟังก์ชัน strcat() ใช้สำหรับนำสายอักขระ 2 สายมาเชื่อมต่อกัน โดยมีรูปแบบดังนี้

strcat(สายอักขระ1, สายอักขระ2)

สายอักขระ1 หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระ

สายอักขระ2 หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระ หรือ สายอักขระในฟันหนู (" ")

การทำงานคือ จะลบ '\0' จากสายอักขระ1 แล้วเติมด้วยอักขระจากสายอักขระ2 ดังนั้นสายอักขระ1 ต้องมีจำนวนหน่วยเพียงพอที่จะเก็บอักขระทั้งหมด

ตัวอย่าง 5.25

การใช้ฟังก์ชัน strcat()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str1[21] = "What school do ";
    char str2[] = "you go to?";
    strcat(str1, str2);
    printf("%s \n", str1);
}
```

ผลการกระทำการ

What school do you go to?

ตัวอย่าง 5.26

การใช้ฟังก์ชัน strcat()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str1[21] = "What school do ";
    strcat(str1, "you go to?");
    printf("%s \n", str1);
}
```

ผลการกระทำการ

What school do you go to?

5.7.2 ฟังก์ชัน strcmp()

ฟังก์ชัน strcmp() ใช้สำหรับเปรียบเทียบสายอักขระ 2 สาย แล้วส่งค่ากลับเป็นจำนวนเต็ม ขึ้นกับผลของการเปรียบเทียบ โดยมีรูปแบบดังนี้

strcmp(**สายอักขระ**1, **สายอักขระ**2)

สายอักขระ1 และสายอักขระ2 หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระหรือสายอักขระใน " "

ฟังก์ชันนี้จะส่งค่ากลับดังนี้

ผลการเปรียบเทียบ	**ค่าที่ส่งกลับ**
สายอักขระ1 < สายอักขระ2	จำนวนเต็มลบ
สายอักขระ1 เหมือนกับ สายอักขระ2	ศูนย์
สายอักขระ1 > สายอักขระ2	จำนวนเต็มบวก

ตัวอย่าง 5.27

การใช้ฟังก์ชัน strcmp()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str1[] = "name1";
    printf("%d \n", strcmp("name2", str1));
    printf("%d \n", strcmp("name", "name"));
}
```

ผลการกระทำการ

1

0

5.7.3 ฟังก์ชัน strcpy()

ฟังก์ชัน strcpy() ใช้สำหรับคัดลอกสายอักขระไปไว้ในตัวแปรแถวลำดับ โดยมีรูปแบบดังนี้

strcpy(สายอักขระปลายทาง, สายอักขระต้นทาง)

สายอักขระปลายทาง หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระ

สายอักขระต้นทาง หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระ หรือสายอักขระใน " "

ฟังก์ชันนี้จะคัดลอกสายอักขระจากสายอักขระต้นทางไปไว้ที่สายอักขระปลายทาง

ตัวอย่าง 5.28

การใช้ฟังก์ชัน strcpy()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str1[20];
    strcpy(str1, "Have a nice day");
    printf("%s \n", str1);
}
```

ผลการกระทำการ

Have a nice day

5.7.4 ฟังก์ชัน strlen()

ฟังก์ชัน strlen() ใช้สำหรับนับจำนวนอักขระทั้งหมดในสายอักขระ (ไม่รวม '\0') โดยมีรูปแบบดังนี้

strlen(สายอักขระ)

สายอักขระ หมายถึง ชื่อตัวแปรแถวลำดับเพื่อเก็บอักขระหรือสายอักขระใน " "

ตัวอย่าง 5.29

การใช้ฟังก์ชัน strlen()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str[] = "Computer Science";
    printf("%d \n", strlen(str));
    printf("%d \n", strlen("C Programming"));
}
```

ผลการกระทำการ

16

13

ตัวอย่าง 5.30

การใช้ฟังก์ชัน strlen()

```
#include <stdio.h>
#include <string.h>
void main()
{
    int i;
    char st[25];
    scanf("%s", st);
    i = strlen(st);
    printf("There are %d characters. \n", i);
}
```

5.8 การส่งตัวแปรแถวลำดับไปยังฟังก์ชัน

ในโปรแกรมที่มีการทำงานกับตัวแปรแถวลำดับ สามารถส่งอาร์กิวเมนต์เป็นตัวแปรแถวลำดับให้กับฟังก์ชันได้ โดยอาจส่งหน่วย (element) ของตัวแปรแถวลำดับ หรือส่งตัวแปรแถวลำดับทั้งหมด

5.8.1 การส่งหน่วยของตัวแปรแถวลำดับเป็นอาร์กิวเมนต์

สามารถส่งอาร์กิวเมนต์ในการเรียกฟังก์ชันเป็นหน่วยของตัวแปรแถวลำดับได้เหมือนการส่งอาร์กิวเมนต์เป็นตัวแปรอื่น ๆ โดยอาร์กิวเมนต์และพารามิเตอร์ที่ตำแหน่งตรงกันต้องมีชนิดเดียวกัน และใช้วิธีการส่งผ่านข้อมูลโดยวิธีเรียกด้วยมูลค่า ดังนั้นฟังก์ชันไม่สามารถเปลี่ยนค่าของอาร์กิวเมนต์ได้

ตัวอย่าง 5.31

การส่งอาร์กิวเมนต์เป็นหน่วยของตัวแปรแถวลำดับ

```
#include <stdio.h>
void try_to_modify(int);
void main()
{
    int p[7] = {1, 2, 3, 4, 5, 6, 7};
    printf("value of p[4] is %d \n", p[4]);
    try_to_modify(p[4]);              /* ส่งอาร์กิวเมนต์เป็นหน่วยของตัวแปรแถวลำดับ */
    printf("value of p[4] after calling function is %d\n", p[4]);
}
void try_to_modify(int a)
{
    a = a + 1;
}
```

ผลการกระทำการ

value of p[4] is 5

value of p[4] after calling function is 5

ตัวอย่าง 5.32

การส่งอาร์กิวเมนต์เป็นหน่วยของตัวแปรแถวลำดับ

```
#include <stdio.h>
void twice(int);
void main()
{
    int a[4] = {3, 5, 7, 9};
    int k;
    for(k = 0; k < 4; k++)
        twice(a[k]);                  /* หน่วยของตัวแปรแถวลำดับเป็นอาร์กิวเมนต์ */
    printf("\n");
    for(k = 0; k < 4; k++)
        printf("%4d", a[k]);
}
void twice(int b)
{
    printf("%4d", 2 * b);
}
```

ผลการกระทำการ

6 10 14 18

3 5 7 9

5.8.2 การส่งตัวแปรแถวลำดับเป็นอาร์กิวเมนต์

การเรียกใช้ฟังก์ชันสามารถส่งอาร์กิวเมนต์เป็นตัวแปรแถวลำดับได้ โดยเขียนชื่อตัวแปรแถวลำดับไว้ในตำแหน่งที่ต้องการ สิ่งที่ส่งไปคือ เลขที่อยู่ (address) เริ่มต้นของตัวแปรแถวลำดับ ดังนั้นถ้ามีการเปลี่ยนแปลงค่าของตัวแปรแถวลำดับในฟังก์ชันค่านั้นจะถูกเปลี่ยน

การส่งอาร์กิวเมนต์ที่เป็นตัวแปรแถวลำดับ ทำได้โดย

1. เขียนชื่อตัวแปรแถวลำดับที่ต้องการให้เป็นอาร์กิวเมนต์ ในข้อความสั่งเรียกใช้ฟังก์ชัน

2. ในฟังก์ชันที่ถูกเรียกต้องกำหนดพารามิเตอร์ที่อยู่ในตำแหน่งเดียวกับอาร์กิวเมนต์ให้เป็นตัวแปรแถวลำดับ (โดยไม่จำเป็นต้องระบุจำนวนสมาชิก สำหรับตัวแปรแถวลำดับ 1 มิติ) และมีชนิดของข้อมูลเหมือนอาร์กิวเมนต์ ชื่อของตัวแปรแถวลำดับในฟังก์ชันไม่จำเป็นต้องเหมือนชื่อตัวแปรแถวลำดับที่เป็นอาร์กิวเมนต์

3. ถ้าเป็นตัวแปรแถวลำดับ 2 มิติ ต้องระบุจำนวนสดมภ์เมื่อกำหนดตัวแปรแถวลำดับในฟังก์ชัน

4. ถ้าเป็นตัวแปรแถวลำดับ 3 มิติ ต้องระบุจำนวนแถว และจำนวนสดมภ์ เมื่อกำหนดตัวแปรแถวลำดับในฟังก์ชัน

ตัวอย่าง 5.33

การส่งตัวแปรแถวลำดับเป็นอาร์กิวเมนต์ไปยังฟังก์ชัน

```
#include <stdio.h>
void funct(int []); /* ต้นแบบฟังก์ชัน */
void main()
{
    int arrayp[20]; /* กำหนดตัวแปรแถวลำดับชื่อ arrayp */
        ...
        ...
    funct(arrayp);   /* เรียกฟังก์ชัน funct() โดยมีตัวแปรแถวลำดับ arrayp[ ] เป็นอาร์กิวเมนต์ */
        ...
        ...
}
void funct(int arraya[])  /* กำหนดพารามิเตอร์เป็นตัวแปรแถวลำดับชื่อ arraya */
{
    ...
    ...
}
```

ตัวอย่าง 5.34

โปรแกรมหาผลรวมของข้อมูลในตัวแปรแถวลำดับ

```
#include <stdio.h>
float sum(float [], int);
void main()
{
    float item[100];
    int i;
    for(i = 0; i < 100; i++)
        scanf("%f", &item[i]);
    printf("Sum is %f\n", sum(item,100));
}
float sum(float a[], int n)
{
    int i;
    float s = 0.0;
    for(i = 0; i < n; i++)
        s = s + a[i];
    return (s);
}
```

ตัวอย่าง 5.35

โปรแกรมหาความยาวของสายอักขระในตัวแปรแถวลำดับ

```
#include <stdio.h>
int len(char []);
void main()
{
    int i;
    char st[25];
    scanf("%s", st);
    i = len(st);
    printf("There are %d characters. \n", i);
}
/* ฟังก์ชันเพื่อหาจำนวนอักขระในตัวแปรแถวลำดับ */
int len(char s[])
{
    int  i;
    for(i = 0; s[i] != '\0'; i++);
    return (i);
}
```

ตัวอย่าง 5.36

การคัดลอกข้อมูลระหว่างตัวแปรแถวลำดับ

```
#include <stdio.h>
void copy(char [], char[]);
void main()
{
    char line[90], destination[90];
    scanf("%s", line);
    copy(line, destination);
    printf("%s\n", destination);
}
/* ฟังก์ชันเพื่อคัดลอกข้อมูลจากตัวแปรแถวลำดับ source[ ] ไปยังตัวแปรแถวลำดับ dest[ ] */
void copy(char source[], char dest[])
{
    int i = 0;
    while(source[i] != '\0')
    {
        dest[i] = source[i];          /* คัดลอกข้อมูลจาก source[i] ไปยัง dest[i] */
        i++;
    }
    dest[i] = '\0';
}
```

ตัวอย่าง 5.37

การหาผลรวมของข้อมูลในตัวแปรแถวลำดับ 2 มิติ

```
#include <stdio.h>
int sum(int [][8]);
void main()
{
    int num[7][8];
    int i, k;
    for(i = 0; i < 7; i++)
        for(k = 0; k < 8; k++)
            scanf("%d", &num[i][k]);
    printf("Sum is %d \n", sum(num));
}
```

```
/* ฟังก์ชันเพื่อหาผลรวมของข้อมูลในตัวแปรแถวลำดับ v[7][8] */
int sum(int v[][8])
{
    int i, j, s = 0;
    for(i = 0; i < 7; i++)
        for(j = 0; j < 8; j++)
            s = s + v[i][j];
    return (s);
}
```

ตัวอย่าง 5.38

การหาผลรวมของข้อมูลในตัวแปรแถวลำดับ 3 มิติ

```
#include <stdio.h>
long sum(int [][9][4]);
void main()
{
    int i, j, k;
    int num[7][9][4];
    long total;
    for(i = 0; i < 7; i++)
        for(j = 0; j < 9; j++)
            for(k = 0; k < 4; k++)
                    scanf("%d", &num[i][j][k]);
    total = sum(num);
    printf("Sum is %ld \n", total);
}
/* ฟังก์ชันเพื่อหาผลรวมของข้อมูลในตัวแปรแถวลำดับ a[7][9][4] */
long sum(int a[][9][4])
{
    int i, j, k;
    long s = 0;
    for(i = 0; i < 7; i++)
        for(j = 0; j < 9; j++)
            for(k = 0; k < 4; k++)
                s = s + a[i][j][k];
    return (s);
}
```

ตัวอย่าง 5.39

การส่งตัวแปรแถวลำดับไปยังฟังก์ชันเพื่อเพิ่มค่าของสมาชิกแต่ละตัวในแถวลำดับขึ้นอีก 1

```
#include <stdio.h>
void modify(int []);
void main()
{
    int p[7] = {1, 2, 3, 4, 5, 6, 7};
    int k;
    printf("values of original array \n");
    for(k = 0; k < 7; k++)
        printf("%4d", p[k]);
    printf("\n");
    modify(p);
    printf("values of array after calling function modify() \n");
    for(k = 0; k < 7; k++)
        printf("%4d", p[k]);
    printf("\n");
}
void modify(int a[])
{
    int t;
    for(t = 0; t < 7; t++)
        a[t] = a[t] + 1;
}
```

ผลการกระทำการ

values of original array

1 2 3 4 5 6 7

values of array after calling function modify()

2 3 4 5 6 7 8

แบบฝึกหัดบทที่ 5

1. เขียนโปรแกรมเพื่อรับข้อมูลชนิดจำนวนเต็ม 100 จำนวน เก็บไว้ในตัวแปรแถวลำดับ แล้วหาค่าสูงสุด ค่าต่ำสุด ค่าเฉลี่ยของข้อมูล

2. เขียนโปรแกรมเพื่อรับข้อมูลชนิดจำนวนจริง 50 จำนวน เก็บไว้ในตัวแปรแถวลำดับ a[ ] แสดงผลของข้อมูลใน a[ ] และสลับค่าของสมาชิกของ a[ ] ดังนี้

   สลับค่าของ a[0] กับ a[1] a[2] กับ a[3] a[4] กับ a[5] ... a[48] กับ a[49] แล้วแสดงผลข้อมูลใน a[ ]

3. เขียนโปรแกรมเพื่อรับข้อมูลชนิดจำนวนเต็ม 50 จำนวน
   ก. แสดงผลข้อมูล 2 จำนวนที่มีผลรวมคือ 37
   ข. หาค่าสูงสุดของข้อมูลที่เป็นเลขคู่
   ค. หาจำนวนข้อมูลที่เป็นเลขคู่และจำนวนข้อมูลที่เป็นเลขคี่

4. เขียนโปรแกรมเพื่อคำนวณคะแนนรวมของนักเรียน 100 คน โดยคำนวณจากคะแนนการบ้าน 5 ครั้ง คะแนนเต็มครั้งละ 10 คะแนน คะแนนสอบกลางภาค คะแนนเต็ม 100 คะแนน และคะแนนสอบไล่ คะแนนเต็ม 100 คะแนน โดยให้คะแนนรวมการบ้านคิดเป็น 20% คะแนนสอบกลางภาคคิดเป็น 30% และคะแนนสอบไล่คิดเป็น 50% ของคะแนนรวม ให้อ่านข้อมูลสำหรับนักเรียนแต่ละคน ได้แก่ คะแนนการบ้าน 5 ครั้ง คะแนนสอบกลางภาค และคะแนนสอบไล่

5. ห้างสรรพสินค้าแห่งหนึ่งมี 7 สาขา แต่ละสาขามี 12 แผนก ให้รับข้อมูลเข้าเป็นรายได้ของแต่ละสาขาเรียงตามลำดับแผนก เขียนโปรแกรมเพื่อ
   ก. คำนวณรายได้ของแต่ละสาขา
   ข. คำนวณรายได้ของแต่ละแผนก
   ค. คำนวณรายได้ทั้งหมดของห้างสรรพสินค้า

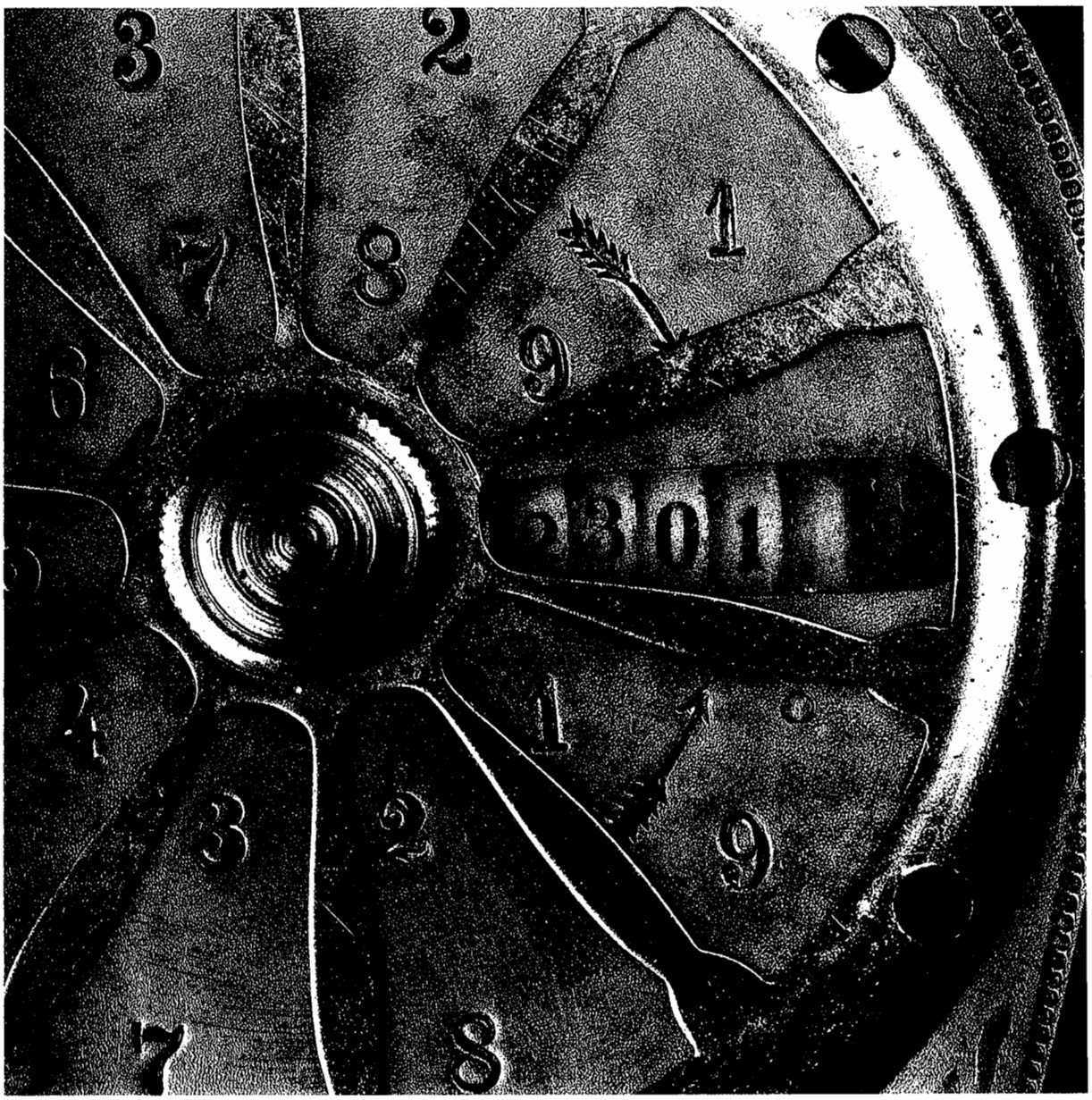
6. บริษัทแห่งหนึ่งมี 3 สาขา แต่ละสาขามี 5 แผนก ให้รับข้อมูลเข้าเป็นรายจ่ายในแต่ละวัน (วันจันทร์ อังคาร พุธ พฤหัสบดี ศุกร์) ของแต่ละสาขาเรียงตามลำดับแผนก เขียนโปรแกรมเพื่อ
   ก. คำนวณรายจ่ายของแต่ละสาขาในวันอังคาร
   ข. คำนวณรายจ่ายรวมของแผนกที่ 4
   ค. คำนวณรายจ่ายของแต่ละสาขา แล้วระบุว่าสาขาไหนมีรายจ่ายสูงสุด

7. เขียนฟังก์ชันเพื่อลบอักขระตัวสุดท้ายออกจากสายอักขระ โดยมีพารามิเตอร์ของฟังก์ชันคือ ตัวแปรแถวลำดับเก็บข้อมูลชนิดอักขระ

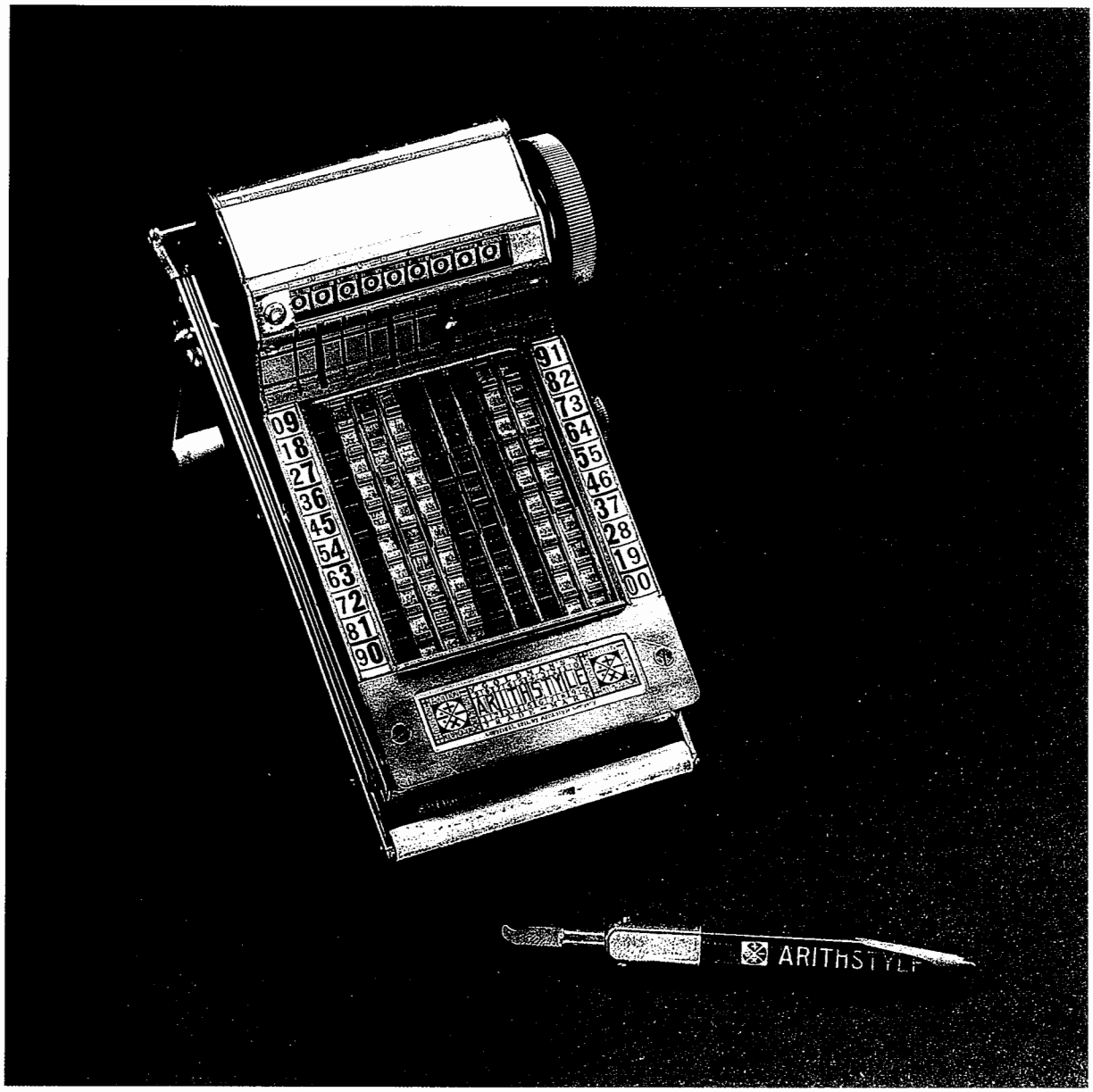
8. เขียนฟังก์ชันเพื่อลบอักขระตัวแรกออกจากสายอักขระ โดยพารามิเตอร์ของฟังก์ชันคือ ตัวแปรแถวลำดับเก็บข้อมูลชนิดอักขระ

9. เขียนโปรแกรมเพื่อแปลงสายอักขระเลขโรมัน (Roman number) เป็นเลขฐาน 10 โดยสัญลักษณ์ที่ใช้แทนเลขโรมัน มีดังนี้

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



เครื่องคำนวณ สร้างโดย Justin W. Bamberger ค.ศ.1905



เครื่องบวกเลขประดิษฐ์โดย Henry Goldman ผลิตขายตั้งแต่ปี ค.ศ. 1906

บทที่ 6
ตัวแปรโครงสร้าง

จากเนื้อหาในบทที่ 1 ถึง 5 ได้กล่าวถึงตัวแปร ซึ่งเป็นตัวแปรที่ใช้เก็บข้อมูลชนิดใดชนิดหนึ่งเท่านั้น แต่ในความเป็นจริง จะมีข้อมูลที่ประกอบด้วยชนิดข้อมูลที่หลากหลายและมีความสัมพันธ์กัน จำเป็นต้องเก็บไว้ด้วยกัน เช่น ในระบบทะเบียนประวัติของพนักงานจะต้องเก็บข้อมูล ชื่อพนักงาน ที่อยู่ หมายเลขประจำตัวพนักงาน เป็นต้น ซึ่งอาจเขียนโครงสร้างของทะเบียนประวัติพนักงานได้ ดังนี้

ชื่อพนักงาน
ที่อยู่
หมายเลขประจำตัว

เมื่อเขียนโปรแกรมจะต้องนำโครงสร้างทะเบียนประวัติพนักงานมาประกาศเป็นตัวแปรซึ่งจะอยู่ในรูปของตัวแปรโครงสร้างที่จะกล่าวถึงในบทนี้

ตัวแปรโครงสร้าง (structured variable) หมายถึง กลุ่มของตัวแปร ซึ่งตัวแปรเหล่านั้นอาจเป็นชนิดเดียวกันหรือต่างชนิด อยู่รวมกันภายใต้ชื่อเดียวกัน ในบางภาษา เช่น ปาสกาล เรียกว่า ระเบียน (record)

ประโยชน์ของตัวแปรโครงสร้างคือ ทำให้สามารถรวบรวมข้อมูลที่สัมพันธ์กันให้อยู่ภายใต้ชื่อตัวแปรโครงสร้างเดียวกัน

6.1 การประกาศตัวแปรโครงสร้าง

ก่อนการประกาศตัวแปรโครงสร้าง ต้องประกาศโครงสร้างก่อน การประกาศโครงสร้างมีวัตถุประสงค์เพื่อกำหนดรูปแบบโครงสร้าง โดยไม่มีการจองเนื้อที่ในหน่วยความจำ

รูปแบบของการประกาศโครงสร้าง

```
struct ชื่อโครงสร้าง
{
    ชนิดข้อมูล ชื่อตัวแปร1;          /* สมาชิกตัวที่ 1 */
    ชนิดข้อมูล ชื่อตัวแปร2;          /* สมาชิกตัวที่ 2 */
      ...
      ...
};
```

ตัวอย่าง 6.1

โครงสร้างของไพ่

```
struct card
{
    int num;
    char suit;      /* โพของไพ่แทนด้วยตัวอักษร h, s, d, c */
};
```

กำหนดรูปแบบโครงสร้างชื่อ card ซึ่งประกอบด้วยสมาชิกคือ num และ suit

การประกาศตัวแปรโครงสร้าง ตามรูปแบบของโครงสร้างที่ได้ประกาศไว้แล้ว ทำโดยใช้ข้อความสั่งตามรูปแบบดังนี้

`struct` ชื่อโครงสร้าง รายชื่อตัวแปรโครงสร้าง;

ข้อความสั่งข้างต้นจะจองเนื้อที่ในหน่วยความจำให้กับตัวแปรที่ประกาศตามขนาดที่กำหนดไว้ในโครงสร้าง

ตัวอย่าง 6.2

การประกาศตัวแปรโครงสร้าง

```
struct card g1, g2;
```

ประกาศตัวแปรโครงสร้าง g1 และ g2 ให้มีโครงสร้างเหมือน card ที่ได้ประกาศไว้ในตัวอย่าง 6.1 หรืออาจใช้การประกาศดังตัวอย่าง

```
struct
{
    int num;
    char suit;
} g1, g2;
```

ข้อความสั่งข้างต้นกำหนดโครงสร้าง และประกาศตัวแปรโครงสร้างในคำสั่งเดียวกัน

การอ้างถึงสมาชิกแต่ละตัวของตัวแปรโครงสร้างทำโดยใส่ชื่อตัวแปรโครงสร้างคั่นด้วยมหัพภาพ (.) ตามด้วยชื่อสมาชิก

รูปแบบของการอ้างถึงสมาชิกของโครงสร้าง

ชื่อตัวแปรโครงสร้าง.ชื่อสมาชิก

ตัวอย่าง 6.3

การอ้างถึงสมาชิกของตัวแปรโครงสร้าง

```
struct card
{
    int num;
    char suit;
} g1, g2;
```

กำหนดค่าให้กับสมาชิกของตัวแปรโครงสร้าง g1 และ g2 โดยใช้ข้อความสั่งกำหนดค่าดังนี้

g1.num = 9;

g1.suit = 'h';

g2.num = 10;

g2.suit = 'd';

สมาชิกในตัวแปรโครงสร้างเดียวกัน ไม่สามารถมีชื่อซ้ำกันได้ แต่สมาชิกของตัวแปรโครงสร้างที่ต่างกัน อาจมีชื่อซ้ำกันได้

ตัวอย่าง 6.4

การประกาศตัวแปรโครงสร้างที่มีสมาชิกชื่อเดียวกันแต่อยู่ต่างโครงสร้าง

```
struct S1
{
    float f;
    char c;
    int i;
};
struct S2
{
    char c;
    float x;
};
struct S1 a;
struct S2 b;
```

การอ้างถึงสมาชิกของ ตัวแปรโครงสร้าง a ทำได้ดังนี้

a.f

a.c

และ a.i

การอ้างถึงสมาชิกของ ตัวแปรโครงสร้าง b ทำได้ดังนี้

b.c และ b.x

สมาชิกของตัวแปรโครงสร้างอาจเป็นตัวแปรแถวลำดับได้

ตัวอย่าง 6.5

สมาชิกของตัวแปรโครงสร้างชนิดแถวลำดับ

```
struct employee
{
    char name[40];
    char address[50];
    long emp_id;
    float salary;
};                                  /* ประกาศโครงสร้าง */
struct employee emp1;               /* ประกาศตัวแปรโครงสร้างชื่อ emp1 */
```

การอ้างถึงสมาชิกของตัวแปรโครงสร้าง emp1 ทำได้ดังนี้

emp1.name, emp1.address, emp1.emp_id และ emp1.salary

ตัวอย่าง 6.6

การอ้างถึงสมาชิกของตัวแปรโครงสร้างที่เป็นแถวลำดับ

```
#include <stdio.h>
void main()
{
    struct date
    {
        int day;
        char mon_name[4];
        int year;
    };
    struct date d;
    d.day = 4;
    d.year = 2004;
    d.mon_name[0] = 'A';        /* กำหนดค่าให้กับสมาชิกของโครงสร้างที่เป็นแถวลำดับ */
    d.mon_name[1] = 'p';
    d.mon_name[2] = 'r';
    d.mon_name[3] = '\0';
    printf("%d %s %d \n", d.day, d.mon_name, d.year);
}
```

หากต้องการเปรียบเทียบ ชื่อเดือน สามารถใช้ข้อความสั่งดังนี้

```
if (strcmp(d.mon_name, "Aug")) == 0) ...
```

หากต้องการเปรียบเทียบ ตัวอักษรตัวแรกของชื่อเดือนสามารถใช้ข้อความสั่งดังนี้

```
if (d.mon_name[0] == 'A') ...
```

นอกจากนี้ ตัวแปรโครงสร้างอาจมีสมาชิกเป็นตัวแปรโครงสร้างได้

ตัวอย่าง 6.7

การประกาศตัวแปรโครงสร้างที่มีสมาชิกเป็นตัวแปรโครงสร้าง

```
struct date
{
    int day;
    int month;
    int year;
};
struct employee
{
    char first_name[20];
    char last_name[20];
    int str_num;
    char str_name[20];
    char city[20];
    long zipcode;
    double salary;
    struct date birthdate;          /* วัน เดือน ปี เกิด */
    struct date hiredate;           /* วัน เดือน ปี ที่บรรจุเป็นพนักงาน */
};
struct employee emp1;
```

ประกาศตัวแปรโครงสร้าง emp1 ซึ่งมีสมาชิกเป็นตัวแปรโครงสร้างคือ birthdate และ hiredate

ในการอ้างถึงเดือนเกิด ทำได้ดังนี้ emp1.birthdate.month

ในการอ้างถึงปีที่บรรจุเป็นพนักงาน ทำได้ดังนี้ emp1.hiredate.year

ตัวอย่าง 6.8

โปรแกรมคำนวณอายุโดยคำนวณจากปีเกิดและปีปัจจุบัน

```
#include <stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
struct info
{
    char name[40];
    long id;
    struct date birthdate;
};
void main()
{
    struct info person;
    int y, age;
    printf("Please enter first name, last name \n");
    gets(person.name);
    printf("Please enter id \n");
    scanf("%ld", &person.id);
    printf("Please enter day, month, year of birth \n");
    scanf("%d%d%d", &person.birthdate.day, &person.birthdate.month,
          &person.birthdate.year);
    printf("Please enter current year \n");
    scanf("%d", &y);
    age = y - person.birthdate.year;
    printf("%s is %d years old. \n", person.name, age);
}
```

ถ้า a และ b เป็นตัวแปรโครงสร้างที่มีโครงสร้างเหมือนกัน และถ้าต้องการให้สมาชิกทุกตัวของ a มีค่าเท่ากับสมาชิกทุกตัวของ b สำหรับตัวแปลโปรแกรมภาษาซีบางตัว เช่น เทอร์โบซี อาจใช้ข้อความสั่ง a = b;

6.2 การกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้าง

การกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้างในคำสั่งประกาศตัวแปรทำได้ โดยกำหนดค่าเริ่มต้นในระหว่างปีกกาหลังการประกาศตัวแปรโครงสร้าง

ตัวอย่าง 6.9

การกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้าง

```
struct card
{
    int num;
    char suit;
};
void main()
{
    struct card g1 = {7,'s'};
    …
    …
}
```

กำหนดให้ g1.num มีค่าเป็น 7 และ g1.suit มีค่าเป็น 's'

การประกาศโครงสร้างไว้ก่อนฟังก์ชันเพื่อให้สามารถกำหนดตัวแปรโครงสร้างให้มีรูปแบบเหมือนโครงสร้างนี้ได้ในทุกฟังก์ชันที่ตามมา โดยไม่ต้องกำหนดโครงสร้างในฟังก์ชันอีก

ตัวอย่าง 6.10

การประกาศโครงสร้างไว้ก่อนฟังก์ชัน main() และการกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้าง

```
struct date
{
    int day;
    int month;
    int year;
    char mon_name[4];
};
void main()
{
    struct date d = {13, 4, 2004, "Apr"};
    …
    …
}
```

กำหนดค่าเริ่มต้น d.day เป็น 13 d.month เป็น 4 d.year เป็น 2004 d.mon_name[0] เป็น 'A' d.mon_name[1] เป็น 'p' d.mon_name[2] เป็น 'r' และ d.mon_name[3] เป็น '\0'

ตัวอย่าง 6.11

การประกาศโครงสร้างไว้ก่อนฟังก์ชัน main() และการกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้าง

```
#include <stdio.h>
struct address
{
    int str_num;
    char street[30];
    char city[20];
    long zip;
};
struct person
{
    char name[20];
    struct address addr;
    char sex[7];
};
void main()
{
    struct person member = {"Somsak  Jaidee",
                            {123, "Sukumvit 39", "Bangkok", 10110},
                            "Male"
                           };
    ...
    ...
}
```

ตัวอย่าง 6.12

การกำหนดค่าเริ่มต้นให้กับตัวแปรโครงสร้าง

```
#include <stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
```

```
struct employee
{
    char name[40];
    char address[40];
    long zipcode;
    double salary;
    struct date birthdate;
    struct date hiredate;
};
void main()
{
    struct employee emp1 = {"Somjai Sukdee",
                            "50 Paholyotin Rd.",
                            10903,
                            15000,
                            {9, 9, 2527},
                            {1, 3, 2547}
                           };
        ...
        ...
}
```

6.3 ตัวแปรแถวลำดับที่มีสมาชิกคือตัวแปรโครงสร้าง

นอกจากการประกาศตัวแปรแต่ละตัวให้เป็นตัวแปรโครงสร้างแล้ว ยังสามารถประกาศตัวแปรแถวลำดับให้มีสมาชิกเป็นตัวแปรโครงสร้างได้

ตัวอย่าง 6.13

การประกาศตัวแปรแถวลำดับที่มีสมาชิกเป็นตัวแปรโครงสร้าง

```
struct card
{
    int num;
    char suit;
};
struct card deck[52];
```

กำหนดตัวแปรแถวลำดับ deck[ ] ซึ่งมีสมาชิก 52 ตัว และแต่ละตัวเป็นแบบโครงสร้าง การอ้างถึงสมาชิกของตัวแปรแถวลำดับ มีรูปแบบดังนี้

ชื่อตัวแปรแถวลำดับ[n].ชื่อสมาชิกของโครงสร้าง

n คือ ดรรชนีกำกับ

การอ้างถึงสมาชิกของตัวแปรแถวลำดับ deck[ ] ทำได้ดังนี้

deck[0].num , deck[0].suit , deck[1].num, deck[1].suit, ..., deck[51].num, deck[51].suit

ตัวอย่าง 6.14

การอ้างถึงสมาชิกของตัวแปรโครงสร้าง

```
struct address
{
    int str_num;              /* บ้านเลขที่ */
    char street[20];          /* ถนน */
    char city[20];            /* จังหวัด */
    long zipcode;             /* รหัสไปรษณีย์ */
};
struct person
{
    char name[20];
    struct address addr;
};
void main()
{
    struct person people[100];
    ...
    ...
}
```

ประกาศตัวแปรแถวลำดับ people[ ] ซึ่งมีสมาชิกจำนวน 100 หน่วย

การอ้างถึงสมาชิกของตัวแปรแถวลำดับ people[ ] ทำได้ดังนี้

people[0].name	อ้างถึงชื่อของสมาชิกตัวแรกในตัวแปรแถวลำดับ
people[0].addr.str_num	อ้างถึงบ้านเลขที่ของสมาชิกตัวแรกในตัวแปรแถวลำดับ
people[0].addr.street	อ้างถึงชื่อถนนของสมาชิกตัวแรกในตัวแปรแถวลำดับ
people[0].addr.city	อ้างถึงชื่อจังหวัดของสมาชิกตัวแรกในตัวแปรแถวลำดับ
people[0].addr.zipcode	อ้างถึงรหัสไปรษณีย์ของสมาชิกตัวแรกในตัวแปรแถวลำดับ

ตัวอย่าง 6.15

โปรแกรมเพื่ออ่านนามสกุล หมายเลขประจำตัวและระดับคะแนน ('A', 'B', 'C', 'D', 'F') ของนักเรียน 100 คน แล้วนับจำนวนนักเรียนที่ได้ระดับคะแนน 'A'

```
#include <stdio.h>
struct student
{
    char last_name[20];
    long student_id;
    char grade;
};
void main()
{
    struct student class_room[100];
    int i, count;
    for(i = 0; i < 100; i++)
        scanf("%s%ld%c", class_room[i].last_name,
              &class_room[i].student_id, &class_room[i].grade);
    count = 0;
    for(i = 0; i < 100; i++)
        if(class_room[i].grade == 'A')
           count = count + 1;
    printf("%d students get A's \n", count);
}
```

6.4 การส่งตัวแปรโครงสร้างไปยังฟังก์ชัน

การส่งอาร์กิวเมนต์เป็นตัวแปรโครงสร้างและส่งค่ากลับเป็นโครงสร้าง มีวิธีการส่งผ่านข้อมูลระหว่างอาร์กิวเมนต์และพารามิเตอร์ด้วยวิธีเรียกด้วยมูลค่า ทำให้ไม่สามารถเปลี่ยนค่าของอาร์กิวเมนต์โดยฟังก์ชันได้

ตัวอย่าง 6.16

การคำนวณแต้มระดับคะแนน โดยส่งอาร์กิวเมนต์เป็นตัวแปรโครงสร้าง

```
#include <stdio.h>
struct student
{
    long id;
    char grade;
};
float gpa(struct student); /* ต้นแบบฟังก์ชัน */
```

```
void main()
{
    struct student person;
    float grade_point;
    scanf("%ld%c", &person.id, &person.grade);
    grade_point = gpa(person);        /* ตัวแปรโครงสร้างเป็นอาร์กิวเมนต์ */
    printf("%ld gets %c so G.P.A. is %f \n", person.id, person.grade,
           grade_point);
}
float gpa(struct student p)           /* ตัวแปรโครงสร้างเป็นพารามิเตอร์ */
{
    if(p.grade == 'A')
        return (4.0);
    else if(p.grade == 'B')
        return (3.0);
    else if(p.grade == 'C')
        return (2.0);
    else if(p.grade == 'D')
        return (1.0);
    else return (0.0);
}
```

ตัวอย่าง 6.17

การส่งผ่านข้อมูลระหว่างฟังก์ชันโดยวิธีเรียกด้วยมูลค่า

```
#include <stdio.h>
struct example
{
    float f;
    int i;
    char c;
};
void try_to_change(struct example);
void main()
{
    struct example a;
    a.f = 1.23;
    a.i = 19;
    a.c = 'A';
    printf("%.2f %d %c \n", a.f, a.i, a.c);
    try_to_change(a);
    printf("%.2f %d %c \n", a.f, a.i, a.c);
}
```

```
void try_to_change(struct example b)
{
    b.f = 3.45;
    b.i = 157;
    b.c = 'Z';
    printf("%.2f %d %c \n", b.f, b.i, b.c);
}
```

ผลการกระทำการ

1.23 19 A

3.45 157 Z

1.23 19 A

เนื่องจากไม่สามารถเปลี่ยนค่าของตัวแปรโครงสร้างโดยฟังก์ชัน try_to_change() ค่าของ a.f, a.i และ a.c ก่อนและหลังการเรียกฟังก์ชันจึงคงเดิม

ตัวอย่าง 6.18

โปรแกรมกำหนดค่าและพิมพ์ค่าของสมาชิกของตัวแปรโครงสร้าง โดยเรียกใช้ฟังก์ชัน assign_values() ซึ่งส่งค่ากลับเป็นโครงสร้าง

```
#include <stdio.h>
struct example
{
    float f;
    int i;
    char c;
};
struct example assign_values(float, int, char);
void main()
{
    struct example a;
    float data1;
    int data2;
    char data3;
    data1 = 1.27;
    data2 = 49;
    data3 = 'y';
    a = assign_values(data1, data2, data3);
    printf("%.2f %d %c \n", a.f, a.i, a.c);
}
```

```
/* ฟังก์ชัน assign_value() คืนค่าเป็นโครงสร้าง */
struct example assign_values(float p1, int p2, char p3)
{
    struct example b;
    b.f = p1;
    b.i = p2;
    b.c = p3;
    return (b);
}
```

ผลการกระทำการ

```
1.27 49 y
```

ตัวอย่าง 6.19

โปรแกรมเพื่อหาผลบวกของจำนวนเชิงซ้อน (complex number) สองจำนวน

```
#include <stdio.h>
struct complex_num
{
    float re_part;
    float im_part;
};
struct complex_num add(struct complex_num, struct complex_num);
void main()
{
    struct complex_num cp1, cp2, cp3;
    scanf("%f%f", &cp1.re_part, &cp1.im_part);
    scanf("%f%f", &cp2.re_part, &cp2.im_part);
    cp3 = add(cp1, cp2);
    printf("%f %f", cp3.re_part, cp3.im_part);
}
/* ฟังก์ชัน add() คืนค่าเป็นโครงสร้าง */
struct complex_num add(struct complex_num x, struct complex_num y)
{
    struct complex_num z;
    z.re_part = x.re_part + y.re_part;
    z.im_part = x.im_part + y.im_part;
    return (z);
}
```

ตัวอย่าง 6.20

โปรแกรมเพื่ออ่านชื่อ หมายเลขประจำตัว และระดับคะแนน ('A', 'B', 'C', 'D', 'F') ของนักเรียน 100 คน แล้ว หาจำนวนนักเรียนที่ได้ระดับคะแนน 'C'

```
#include <stdio.h>
struct student
{
    char last_name[20];
    long student_id;
    char grade;
};
int fair(struct student []);
void main()
{
    struct student course[100];
    int i;
    for(i = 0; i < 100; i++)
        scanf("%s%ld%c", course[i].last_name, &course[i].student_id,
              &course[i].grade);
    printf("%d\n", fair(course));
}
/* ฟังก์ชัน fair() มีพารามิเตอร์เป็นแถวลำดับ ซึ่งมีสมาชิกเป็นโครงสร้าง */
int fair(struct student class_room[])
{
    int i, count = 0;
    for(i = 0; i < 100; i++)
        if(class_room[i].grade == 'C')
            count = count + 1;
    return (count);
}
```

6.5 ข้อความสั่ง typedef

ข้อความสั่ง typedef ใช้เพื่อสร้างชนิดของข้อมูลชื่อใหม่ โดยสามารถนำชนิดของข้อมูลที่มีอยู่เดิมมาตั้งชื่อใหม่ได้ มีรูปแบบดังนี้

typedef ชนิดของข้อมูล ชื่อที่กำหนด;

ตัวอย่าง 6.21

การใช้งาน typedef

```
typedef int BAHT;
```

ทำให้สามารถใช้ BAHT แทน int ได้ดังนี้

```
BAHT  cost, tax;          /* ใช้แทนข้อความสั่ง int cost, tax; */
```

ข้อความสั่งข้างต้นประกาศตัวแปร cost และ tax ให้เป็นตัวแปรชนิดจำนวนเต็ม

```
typedef float FEET;
FEET  height, width;
```

ประกาศตัวแปร height และ width ให้เป็นชนิดจำนวนจริง

ตัวอย่าง 6.22

การใช้ typedef กับโครงสร้าง

```
typedef struct
{
    int num;
    char suit;
} CARD;
CARD t, deck[52];
```

typedef ใช้กำหนด CARD ให้เป็นโครงสร้างที่ประกอบด้วยสมาชิกคือ num และ suit แล้วใช้ CARD ในการประกาศตัวแปร t และตัวแปรแถวลำดับ deck[ ]

ประโยชน์ของ typedef คือ

1. ทำให้คำสั่งประกาศตัวแปรกระชับขึ้น เช่น ในการประกาศตัวแปรโครงสร้าง
2. ทำให้โปรแกรมอ่านง่ายขึ้น สามารถตั้งชื่อชนิดของข้อมูลที่สื่อความหมายได้
3. ทำให้โปรแกรมสามารถใช้ได้กับเครื่องหลายระบบ (portability) โดยเปลี่ยนเฉพาะ typedef เช่น int คอมพิวเตอร์บางเครื่องมีขนาด 2 ไบต์ บางเครื่องมีขนาด 4 ไบต์ ถ้าเดิมใช้กับเครื่องขนาด 4 ไบต์ เมื่อนำโปรแกรมไปใช้ในเครื่องขนาด 2 ไบต์ ทำให้ต้องเปลี่ยน int ให้เป็น short แต่ถ้าใช้ typedef ก็เปลี่ยนเฉพาะที่ typedef เท่านั้น

ตัวอย่าง 6.23

การใช้ typedef ในการประกาศตัวแปรโครงสร้าง เพื่อใช้ในการคำนวณคะแนนเฉลี่ยของนักเรียนแต่ละคน

```
#include <stdio.h>
typedef struct
{
    char name[30];
    float test[5];
    float avg;
} STUDENT;
void main()
{
    STUDENT s[100];
    int i, k;
    float sum;
    for(i = 0; i < 100; i++)
    {
        scanf("%s", s[i].name);
        sum = 0;
        for(k = 0; k < 5; k++)
        {
            scanf("%f", &s[i].test[k]);
            sum = sum + s[i].test[k];
        }
        s[i].avg = sum / 5;
        printf("%s receives average test score of %f \n", s[i].name,
               s[i].avg);
    }
}
```

ตัวอย่าง 6.24

การใช้ typedef ในการประกาศตัวแปรโครงสร้าง

```
#include <stdio.h>
typedef struct
{
    char name[20];
    int age;
} PERSON;
PERSON assign_values(void);     /* ฟังก์ชัน assign_values() ส่งค่ากลับเป็นโครงสร้าง */
int oldest(PERSON []);          /* ฟังก์ชัน oldest() มีพารามิเตอร์เป็นแถวลำดับที่มีสมาชิกเป็น
                                   โครงสร้าง */

void main()
{
    PERSON p[50];
    int i;
    for(i = 0; i < 50; i++)
       p[i] = assign_values();  /* ฟังก์ชัน assign_values() ส่งค่ากลับเป็นโครงสร้าง */
    printf("The oldest person is %d years old \n", oldest(p));
}
PERSON assign_values()
{
    PERSON man;
    scanf("%s%d", man.name, &man.age);
    return (man);
}
/* หาอายุสูงสุด */
int oldest(PERSON g[])
{
    int k, high = 0;
    for(k = 0; k < 50; k++)
        if(g[k].age > high)
            high = g[k].age;
    return (high);
}
```

แบบฝึกหัดบทที่ 6

1. กำหนดโครงสร้างเพื่อเก็บข้อมูลของนักศึกษาประกอบด้วย

student id	เป็นชนิดจำนวนเต็ม
first name	เป็นสายอักขระ
last name	เป็นสายอักขระ
total credits completed	เป็นชนิดจำนวนเต็ม
grade point average	เป็นชนิดจำนวนจริง

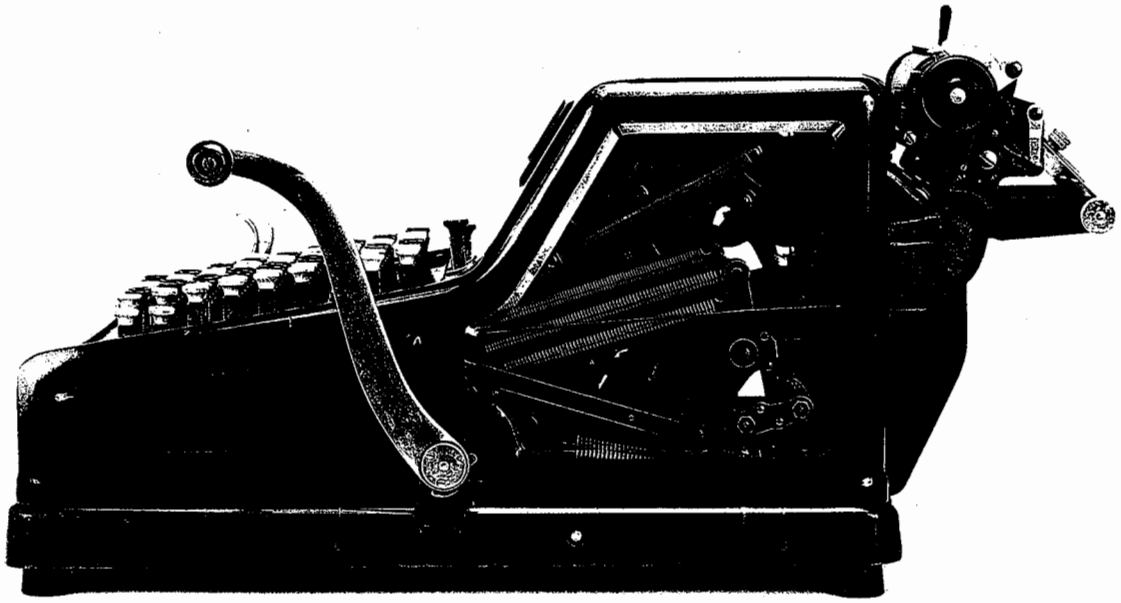
2. ประกาศตัวแปรแถวลำดับที่มีสมาชิกเป็นโครงสร้างตามข้อ 1 โดยให้แถวลำดับมีสมาชิก 100 หน่วย

3. เขียนโปรแกรมเพื่อรับข้อมูลเป็นจำนวนนักศึกษา (ไม่เกิน 200 คน) ในวิชาการโปรแกรมภาษา C อ่านชื่อ เลขประจำตัว ระดับคะแนน (A, B+, B, C+, C, D+, D, F) กำหนดและเรียกฟังก์ชัน class_avg() เพื่อคำนวณแต้มระดับคะแนนเฉลี่ยของนักศึกษาในวิชานี้ โดย ระดับคะแนน A มีแต้มระดับคะแนน 4.0 ระดับคะแนน B+ มีแต้มระดับคะแนน 3.5 ระดับคะแนน B มีแต้มระดับคะแนน 3.0 ระดับคะแนน C+ มีแต้มระดับคะแนน 2.5 ตามลำดับ

4. ห้างสรรพสินค้าแห่งหนึ่ง มีพนักงาน 120 คน เขียนโปรแกรมเพื่อรับข้อมูลสำหรับพนักงานแต่ละคน ได้แก่ รหัสประจำตัวพนักงาน ชื่อ-นามสกุล ยอดขายต่อเดือน คำนวณยอดขายเฉลี่ยต่อเดือนของพนักงานในห้างสรรพสินค้า แสดงจำนวนพนักงานที่มียอดขายต่ำกว่ายอดขายเฉลี่ย แสดงรหัสประจำตัวพนักงาน ชื่อ-นามสกุลและยอดขายของพนักงานที่มียอดขายต่ำกว่ายอดขายเฉลี่ย

5. บริษัทแห่งหนึ่งมีพนักงาน 90 คน เขียน์โปรแกรมเพื่อรับข้อมูลสำหรับพนักงานแต่ละคนได้แก่ รหัสประจำตัว ชื่อ-นามสกุล วัน-เดือน-ปีที่บรรจุ และอัตราเงินเดือน ให้ปรับอัตราเงินเดือนพนักงานขึ้นกับอายุงานตามเงื่อนไขดังนี้

อายุงาน ต่ำกว่า 5 ปี	อัตราเงินเดือนคงเดิม
อายุงาน 5 – 10 ปี	ปรับเงินเดือนขึ้น 3%
อายุงาน 11 –15 ปี	ปรับเงินเดือนขึ้น 5%
อายุงาน 16 ปีขึ้นไป	ปรับเงินเดือนขึ้น 10%



เครื่องบวกเลข Continental สร้างโดย Wanderer-Werke ค.ศ.1916

บทที่ 7
ตัวชี้

ในบทนี้จะกล่าวถึงตัวชี้ (pointer) ซึ่งเป็นลักษณะเด่นของเครื่องมือดำเนินการที่สำคัญที่สุดในภาษาซี ตัวชี้หรือพอยน์เตอร์มีประโยชน์มากในการจัดการโครงสร้างข้อมูลแบบพลวัต (dynamic data structure) เช่น รายการเชื่อมโยง (linked list) เป็นต้น ซึ่งรายละเอียดของเนื้อหาในส่วนนี้จะสามารถศึกษาได้ในวิชาโครงสร้างข้อมูล นอกจากนี้ตัวแปรชนิดตัวชี้ยังมีประโยชน์ในแง่ของประสิทธิภาพการทำงาน ช่วยให้โปรแกรมทำงานได้เร็วขึ้นและประหยัดเนื้อที่ในหน่วยความจำ ซึ่งเนื้อหาในส่วนนี้จะแสดงในหัวข้อการใช้งานตัวชี้กับฟังก์ชัน

7.1 การประกาศตัวแปรชนิดตัวชี้

ตัวแปรชนิดตัวชี้จะเก็บค่าเลขที่อยู่ของหน่วยความจำ ซึ่งแตกต่างจากตัวแปรชนิดอื่นที่จะเก็บค่าที่แท้จริง เช่นตัวแปรชนิดจำนวนจริง (float) จะเก็บค่าจำนวนจริงที่มีค่าสูงสุดถึง 3.4×10^{38} และเก็บความละเอียดของทศนิยมได้ถึง 7 ตำแหน่ง เป็นต้น

รูปแบบในการประกาศใช้ตัวแปรชนิดตัวชี้คือ

ชนิดข้อมูล *ชื่อตัวแปร ; หรือ
ชนิดข้อมูล* ชื่อตัวแปร ;

เช่น int *point; เป็นการประกาศตัวแปรชื่อ point ซึ่งมีชนิดเป็นตัวชี้ที่จะทำหน้าที่เก็บตำแหน่งในหน่วยความจำที่มีข้อมูลชนิดเลขจำนวนเต็มอยู่ ซึ่งเครื่องหมายดอกจัน (*) จะเป็นสิ่งที่ทำให้เราทราบว่าตัวแปร point เป็นตัวชี้

การกำหนดค่าเริ่มต้นให้แก่ตัวแปรชนิดตัวชี้สามารถกำหนดให้มีค่าเป็น NULL หรือ 0 ซึ่งหมายความว่า ตัวแปรนี้ยังไม่มีการเก็บตำแหน่งที่อยู่ใด ๆ นอกจากนี้ยังสามารถกำหนดให้ตัวแปรนี้อ้างอิงไปยังตำแหน่งในหน่วยความจำที่ต้องการซึ่งรายละเอียดในส่วนนี้จะกล่าวถึงในหัวข้อต่อไป

ห้ามมีเว้นวรรคระหว่างเครื่องหมายดอกจันและตัวแปร เช่น int * error เป็นการประกาศที่ผิด

การประกาศตัวแปรชนิดตัวชี้หลายตัวในเวลาเดียวกันสามารถทำได้ เช่น int *ptr1, *ptr2; นอกจากนี้ถ้าต้องการหลีกเลี่ยงการใช้เครื่องหมายดอกจันทุกครั้งที่ประกาศตัวแปรชนิดตัวชี้สามารถทำได้ด้วยการใช้คำหลัก typedef เช่น

```
typedef int*  Ptype;
Ptype  ptr1, ptr2;          // ข้อความสั่งนี้ให้ความหมายเหมือนกับการประกาศ  int *ptr1, *ptr2;
```

7.2 ตัวดำเนินการสำหรับตัวชี้

ตัวดำเนินการสำหรับตัวชี้มีอยู่ด้วยกัน 2 ตัวคือ ตัวดำเนินการเลขที่อยู่ และตัวดำเนินการอ้างอิง

7.2.1 ตัวดำเนินการเลขที่อยู่ (address operator)

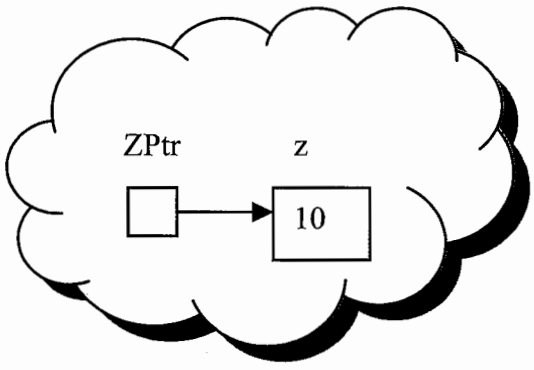
ตัวดำเนินการนี้จะใช้เครื่องหมาย & นำหน้าตัวแปร ซึ่งจะหมายความถึงตำแหน่งที่อยู่ของตัวแปรนั้นในหน่วยความจำ

ตัวอย่าง 7.1

การอ้างอิงตัวชี้

```
#include <stdio.h>
void main()
{
    int z = 10;
    int *ZPtr;
    ZPtr = &z ;
}
```

ข้อความสั่ง ZPtr = &z ; มีวัตถุประสงค์เพื่อให้ตัวแปรชนิดตัวชี้ ZPtr เก็บเลขที่อยู่ของตัวแปร z ซึ่งอาจแสดงในรูปแบบตัวชี้ดังรูป 7.1



รูปที่ 7.1 ตัวแปรชนิดตัวชี้ในหน่วยความจำ

7.2.2 ตัวดำเนินการอ้างอิง (dereferencing operator)

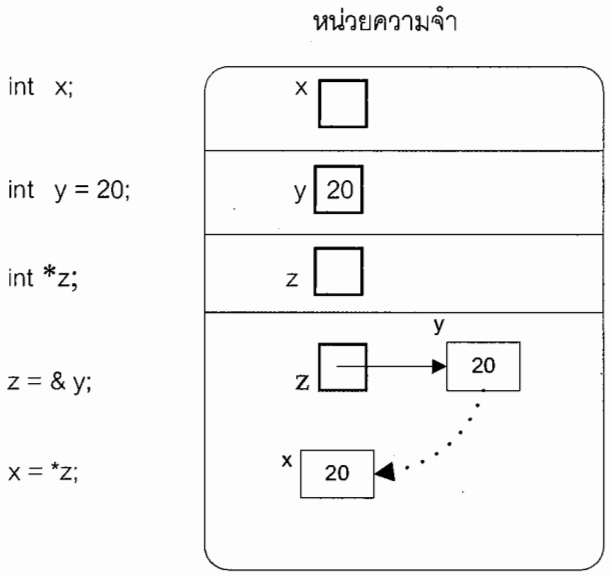
ตัวดำเนินการนี้จะใช้เครื่องหมายดอกจัน (*) นำหน้าตัวแปรชนิดตัวชี้ ซึ่งจะเป็นอ้างถึงค่าข้อมูลที่ตัวแปรชนิดตัวชี้นั้นชี้ไป ซึ่งค่าของข้อมูลที่ตัวชี้อ้างอิงนี้สามารถถูกเปลี่ยนแปลงค่าได้ เช่น การเพิ่มค่าอีก 1 สามารถทำได้โดยใช้คำสั่ง (*ptr)++ จะเป็นการเพิ่มค่าที่ตัวชี้ ptr ชี้ไป การแสดงค่าที่ตัวชี้ชี้ไปก็สามารถทำได้โดยใช้คำสั่ง printf("%d", *ZPtr); จะส่งผลให้โปรแกรมแสดงค่า 10 ออกทางจอภาพตามรูปที่ 7.1

ตัวอย่าง 7.2

การใช้งานตัวชี้

```
#include <stdio.h>
void main()
{
    int x;
    int y = 20;
    int *z;
    z = &y;
    x = *z;
    printf("%d", x);
}
```

จากส่วนของโปรแกรมข้างต้นเมื่อสั่งให้โปรแกรมทำงานจะมีการแสดงค่าของตัวแปร x ซึ่งมีค่าเป็น 20 ซึ่งเท่ากับค่าที่เก็บอยู่ในตัวแปร y ลำดับเหตุการณ์ที่เกิดขึ้นแสดงดังรูปที่ 7.2



รูปที่ 7.2 ลำดับเหตุการณ์ที่เกิดขึ้นในหน่วยความจำ

นอกจากนี้สามารถสั่งให้ตัวแปรชนิดตัวชี้ชี้ไปที่เดียวกันกับตัวแปรตัวชี้ตัวอื่น เช่น ถ้าตัวแปร next มีชนิดเป็นตัวชี้ เราสามารถสั่งให้ชี้ไปที่เดียวกันกับ ที่ตัวแปร z ชี้ไปโดยใช้ข้อความสั่ง next = z;

การใช้ตัวดำเนินการแบบอ้างอิงกับตัวแปรชนิดตัวชี้ที่ยังไม่ได้มีการกำหนดค่าจะทำให้เกิดข้อผิดพลาดขึ้นได้

เพื่อให้เห็นภาพรวมของการใช้ตัวแปรชนิดตัวชี้กับตัวดำเนินการทั้งสองรูปแบบจึงได้แสดงตัวอย่างโปรแกรมดังนี้

ตัวอย่าง 7.3

การแสดงค่าที่เก็บในตัวชี้

```
#include <stdio.h>
void main()
{
    int y;
    int *yPtr;
    y = 10;
    yPtr = &y;
    printf("The address of y is %p\n", &y);  /* ใช้ %p เมื่อต้องการพิมพ์ค่าของ
                                                ตำแหน่งหน่วยความจำที่เก็บอยู่ */
    printf("The value of yPtr is %p\n", yPtr);  // ในตัวแปรชนิดตัวชี้
    printf("The value of y is %d\n", y);
    printf("The value of *yPtr is %d\n", *yPtr);
}
```

ผลการกระทำการ

```
The address of y is 0012FF85
The value of yPtr is 0012FF85
The value of y is 10
The value of *yPtr is 10
```

จากผลลัพธ์จะพบว่าค่าที่เก็บอยู่ในตัวแปร yptr มีค่าตรงกับเลขที่อยู่ของตัวแปร y คือ 0012FF85 อันเป็นผลเนื่องมาจากข้อความสั่ง yPtr = &y; ซึ่งเป็นการสั่งให้ตัวชี้ yPtr ชี้ไปที่ตำแหน่งของตัวแปร y ในหน่วยความจำ

ค่าที่แสดงตำแหน่งในหน่วยความจำบนจอภาพสำหรับฟังก์ชัน printf("The address of y is %p\n", &y); และ printf("The value of yPtr is %p\n", yPtr); จะแตกต่างกันไปในแต่ละครั้งของการกระทำการโปรแกรมโดยขึ้นกับเลขที่อยู่ของหน่วยความจำที่ระบบปฏิบัติการจัดสรรให้

7.3 การจองเนื้อที่ในหน่วยความจำ

ภาษาซีได้มีการจัดเตรียมฟังก์ชันสำหรับการจองเนื้อที่ในหน่วยความจำใน 2 ลักษณะ คือฟังก์ชัน malloc() และฟังก์ชัน calloc() ซึ่งฟังก์ชันทั้งสองจะคืนค่าเป็นเลขที่อยู่ของหน่วยความจำที่ได้จัดสรรให้แก่ผู้ใช้ในขนาดที่กำหนด หรือคืนค่าเป็น NULL เมื่อไม่สามารถจัดสรรเนื้อที่ให้ได้

รูปแบบการเรียกใช้ฟังก์ชันดังกล่าว คือ

ตัวชี้ = malloc(sizeof(ชนิดข้อมูล)); และ
ตัวชี้ = calloc(จำนวนชุด, sizeof(ชนิดข้อมูล));

ฟังก์ชัน sizeof() เป็นฟังก์ชันที่คำนวณขนาดของชนิดข้อมูลเพื่อใช้ในการจองเนื้อที่
ฟังก์ชัน calloc() จะเป็นการจองเนื้อที่หลายชุดที่ต่อเนื่องกันตามที่ผู้ใช้ระบุ

ตัวอย่าง 7.4

การจองเนื้อที่ในหน่วยความจำ

```
int *ptr;
if (ptr = malloc(sizeof(int))) == NULL)
     return 0;                    // จบโปรแกรม
else
    *ptr = 23;
```

จากตัวอย่างส่วนของโปรแกรมมีการนำตัวชี้ ptr มารับค่าตำแหน่งที่อยู่ในหน่วยความจำที่ฟังก์ชัน malloc() ทำการจองเนื้อที่ให้ โดยมีการตรวจสอบว่าการจองเนื้อที่ดังกล่าวทำสำเร็จหรือไม่ ถ้าทำไม่สำเร็จจะออกจากโปรแกรม

7.4 การคืนเนื้อที่ให้แก่หน่วยความจำ

เมื่อไม่ต้องการใช้เนื้อที่ในหน่วยความจำแล้วผู้เขียนโปรแกรมควรคืนเนื้อที่ส่วนนั้นให้แก่หน่วยความจำโดยใช้รูปแบบดังนี้

free(ตำแหน่งของหน่วยความจำที่ตัวชี้ชี้ไป);

ตัวอย่าง 7.5

การคืนเนื้อที่ให้หน่วยความจำ

```
#include <stdio.h>
void main()
{
    int *ptr;
    if (ptr = malloc(sizeof(int))) == NULL)
        return 0;
    else
        *ptr = 20;
    free(ptr);
}
```

เมื่อไม่ต้องการใช้เนื้อที่ในหน่วยความจำส่วนใดแล้ว และไม่มีการคืนเนื้อที่ให้แก่หน่วยความจำเลย อาจทำให้เนื้อที่ในหน่วยความจำไม่เพียงพอสำหรับการใช้งานในภายหลัง

ข้อควรระวังในการใช้ตัวชี้

1. ห้ามจองเนื้อที่ในหน่วยความจำให้กับตัวแปรที่ไม่ใช่ตัวชี้ เช่น

   ```
   int  x;
   &x = malloc(sizeof(int));
   ```

2. ห้ามใช้ฟังก์ชัน free() ในการคืนเนื้อที่ให้กับหน่วยความจำ ในกรณีที่ไม่ได้มีการใช้ฟังก์ชัน malloc() ในการจองเนื้อที่มาก่อนสำหรับตัวชี้นั้น เช่น

   ```
   int  *ptr;
   int  x;
   ptr = &x;
   free(ptr);
   ```

3. ไม่ควรใช้ฟังก์ชัน malloc() ในการจองเนื้อที่ที่คืนค่าให้กับตัวชี้ตัวเดียวกันต่อเนื่องกัน เพราะการจองเนื้อที่ครั้งก่อนหน้าจะไม่สามารถถูกอ้างอิงและนำไปใช้งานได้ เช่น

   ```
   ptr = malloc(sizeof(int));
   ptr = malloc(sizeof(int));
   ```

7.5 การใช้ตัวแปรชนิดตัวชี้กับค่าคงที่

ตัวแปรชนิดตัวชี้สามารถถูกระบุคุณสมบัติเพิ่มเติมได้โดยการนำคำหลัก const ซึ่งหมายถึงค่าคงที่มาประกอบซึ่งจะมีความหมายแตกต่างกันไปดังนี้

1. const int *Ptr; หรือ int const *Ptr หมายถึงการกำหนดให้ตัวชี้ Ptr ชี้ไปที่ข้อมูลตัวเลขจำนวนเต็มที่มีค่าคงที่ ไม่สามารถเปลี่ยนแปลงค่าได้ แต่ตัวชี้ Ptr สามารถเปลี่ยนไปชี้ที่อื่นได้
2. int *const Ptr; หมายถึงการบังคับให้ตัวชี้ Ptr ชี้ไปที่ใดที่หนึ่งและห้ามเปลี่ยนตำแหน่งการชี้ แต่ค่าของข้อมูลที่เก็บอยู่ในตำแหน่งนั้นสามารถเปลี่ยนแปลงได้
3. const int *const Ptr; หมายถึงการการบังคับให้ตัวชี้ Ptr ชี้ไปที่ใดที่หนึ่งและห้ามเปลี่ยนตำแหน่งการชี้และค่าของข้อมูลที่เก็บอยู่ในตำแหน่งนั้นไม่สามารถเปลี่ยนแปลงได้

7.6 การเรียกใช้ฟังก์ชันที่มีการส่งตัวแปรแบบอ้างอิง

การเขียนโปรแกรมที่ดีควรจะมีการแบ่งส่วนชุดคำสั่งการทำงานเป็นกลุ่ม หรือเรียกอีกอย่างหนึ่งว่าฟังก์ชัน โดยส่วนของโปรแกรมหลักจะสามารถสั่งให้ฟังก์ชันทำงานได้โดยการเรียกชื่อฟังก์ชันและส่งตัวแปรที่บรรจุข้อมูลไปยังฟังก์ชันเพื่อให้ฟังก์ชันทำการประมวลผลข้อมูลที่ส่งไป รูปแบบการส่งตัวแปรไปยังฟังก์ชันมี 2 รูปแบบด้วยกันคือ การส่งตัวแปรเฉพาะค่า หรือการเรียกด้วยมูลค่า (call by value) และการส่งตัวแปรแบบอ้างอิงหรือการเรียกด้วยการอ้างอิง (call by reference) ซึ่งวิธีการเรียกโดยมูลค่านี้ได้อธิบายไว้ในเนื้อหาบทที่ 4 ในที่นี้จะขออธิบายถึงการส่งตัวแปรแบบอ้างอิงซึ่งต้องอาศัยตัวแปรชนิดตัวชี้ในการดำเนินการ วัตถุประสงค์ของการส่งตัวแปรแบบอ้างอิงคือการยินยอมให้ฟังก์ชันสามารถทำการเปลี่ยนแปลงค่าของตัวแปรนี้อย่างถาวร หมายความว่าเมื่อฟังก์ชันได้ทำงานเสร็จแล้วค่าของตัวแปรนี้จะมีค่าไม่เหมือนเดิมเมื่อเทียบกับค่าของตัวแปรก่อนเรียกฟังก์ชัน ซึ่งจะได้แสดงให้ดูในตัวอย่าง

วิธีการส่งตัวแปรแบบอ้างอิงไปยังฟังก์ชันใช้รูปแบบดังนี้

ชนิดข้อมูล ชื่อฟังก์ชัน(&ตัวแปรชนิดตัวชี้);

การเขียนส่วนหัวของฟังก์ชันใช้รูปแบบดังนี้

ชนิดข้อมูล ชื่อฟังก์ชัน(ชนิดข้อมูล *ตัวแปรชนิดตัวชี้);

ตัวอย่าง 7.6

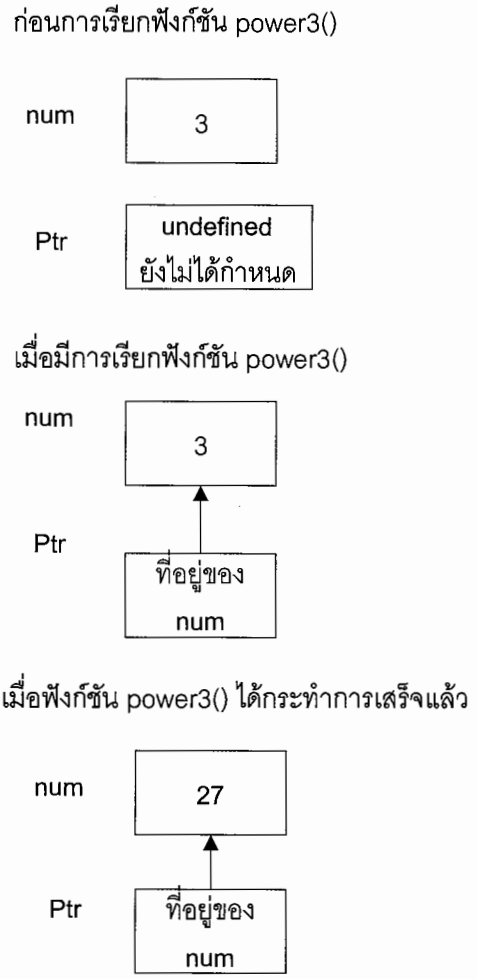
การเรียกฟังก์ชันโดยใช้ตัวชี้

```
#include <stdio.h>
void power2 (int *);
void main()
{
    int num = 3;
    printf("The value of num before calling function is %d \n", num);
    power3(&num);
    printf("The value of num after calling function is %d \n", num);
}
void power3(int *Ptr)
{
    *Ptr = *Ptr * *Ptr * *Ptr;
}
```

ผลการกระทำการ

```
The value of num before calling function is 3
The value of num after calling function is  27
```

จากโปรแกรมในข้างต้นจะเห็นได้ว่าส่วนของโปรแกรมหลักได้ทำการเรียกฟังก์ชัน power3() โดยส่งเลขที่อยู่ของตัวแปร num ไปให้ฟังก์ชันกระทำการ สังเกตที่ส่วนหัวของฟังก์ชัน จะเห็นว่าตัวแปรที่มารับค่ามีชนิดเป็นตัวชี้ที่อ้างอิงไปยังไปยังข้อมูลชนิดจำนวนเต็ม รูปที่ 7.3 จะแสดงถึงลำดับเหตุการณ์ที่เกิดขึ้นในหน่วยความจำ



รูปที่ 7.3 ลำดับเหตุการณ์การเรียกฟังก์ชันโดยการส่งตัวแปรแบบอ้างอิงในหน่วยความจำ

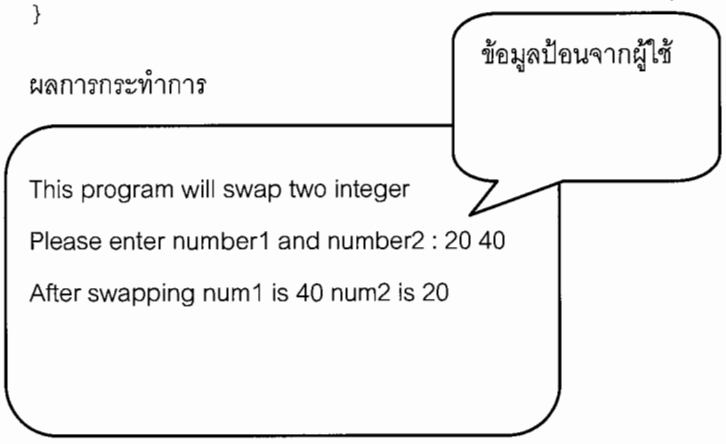
จากรูปที่ 7.3 จะสังเกตได้ว่าขณะที่โปรแกรมหลักเรียกฟังก์ชัน power3() โดยส่งเลขที่อยู่ของตัวแปร num ไปให้กับตัวแปรชนิดตัวชี้ชื่อ Ptr ที่ส่วนหัวของฟังก์ชัน จะเกิดการอ้างอิงขึ้นในหน่วยความจำ (สังเกตที่ลูกศร) หลังจากนั้นเมื่อชุดข้อความสั่งในฟังก์ชันกระทำการกับตัวแปรชนิดตัวชี้ Ptr ก็เปรียบเสมือนกับการกระทำการที่ตัวแปร num นั่นเองดังนั้นเมื่อฟังก์ชัน power3() ทำงานเสร็จสิ้นลง และทำการแสดงค่าตัวแปร num ออกทางจอภาพ จะได้ค่าใหม่ที่เกิดจากการนำตัวเลขที่เคยเก็บในตัวแปร num ยกกำลังสาม

ตัวอย่าง 7.7

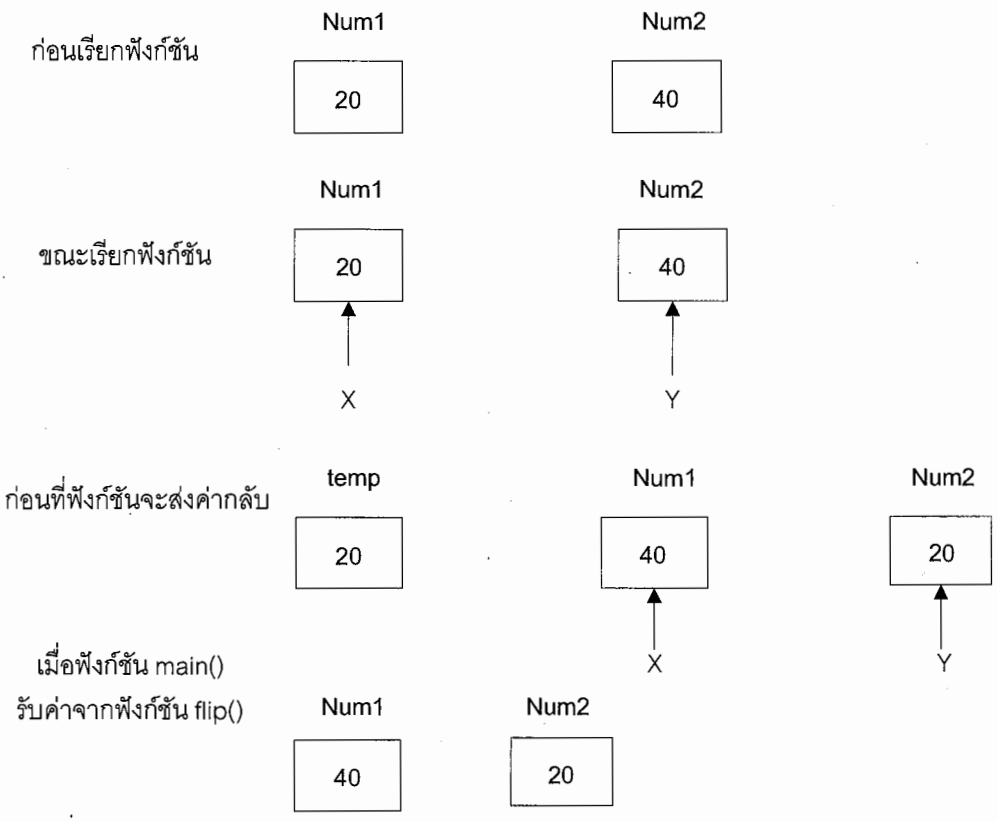
การสลับค่าของข้อมูล

```
#include <stdio.h>
void flip(int *, int *);
void main()
{
    int num1;
    int num2;
    printf("This program will swap two integer\n");
    printf("Please enter number1 and number2 : ");
    scanf("%d%d",&num1, &num2);
    flip(&num1, &num2);
    printf("After swapping num1 is %d num2 is %d\n", num1, num2);
}
void flip(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

ผลการกระทำการ



โปรแกรมข้างต้นมีวัตถุประสงค์ในการสลับค่าของตัวแปรที่ผู้ใช้ป้อนเข้ามาโดยการเรียกใช้ฟังก์ชัน flip() ซึ่งรับพารามิเตอร์เป็นตัวแปรชนิดตัวชี้ที่ชี้ไปที่ตัวแปร num1 และ num2 หลังจากนั้นฟังก์ชัน flip() จะทำการสลับค่าในตัวแปรดังกล่าวโดยใช้ตัวชี้ x และ ตัวชี้ y เพื่อให้เข้าใจง่ายขึ้นจึงแสดงรายละเอียดดังรูปที่ 7.4



รูปที่ 7.4 ลำดับการทำงานเมื่อมีการเรียกฟังก์ชันโดยใช้ตัวชี้

ตัวอย่าง 7.8

การเพิ่มค่าที่ตัวชี้อ้างอิงโดยการเรียกฟังก์ชัน

```
#include <stdio.h>
void increment (int *count_ptr)
{
    (*count_ptr)++;
}
void main()
{
    int count = 0;
    while (count < 10)
        increment(&count);
}
```

โปรแกรมข้างต้นมีวัตถุประสงค์ในการเพิ่มค่าขอตัวแปร count โดยการส่งเลขที่อยู่ของตัวแปร count ไปให้แก่ตัวชี้ count_ptr ที่ฟังก์ชัน increment() ฟังก์ชันดังกล่าวจะทำการเพิ่มค่าของ count ทีละหนึ่ง ฟังก์ชัน main() จะทำการเรียกใช้ฟังก์ชัน increment() จำนวน 10 ครั้ง เมื่อโปรแกรมทำงานเสร็จสิ้นลง ตัวแปร count จะมีค่าเป็น 10

จากหลักการส่งตัวแปรไปยังฟังก์ชันจะเห็นว่า การอ้างอิงเลขที่อยู่ของตัวแปรที่เก็บข้อมูลในลักษณะนี้จะทำให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพทั้งในแง่ความเร็วในการประมวลผลและเป็นการประหยัดเนื้อที่ในหน่วยความจำเป็นอย่างมากเมื่อเทียบกับวิธีการส่งตัวแปรแบบเรียกด้วยมูลค่า เนื่องจากการส่งตัวแปรแบบเรียกด้วยมูลค่าจะเกิดกลไกในการคัดลอกข้อมูลจากตัวแปรของผู้เรียกฟังก์ชัน ไปยังตัวแปรที่ส่วนหัวของฟังก์ชัน ซึ่งกลไกนี้จะกินเวลานานถ้าข้อมูลมีขนาดใหญ่ เช่น ข้อมูลชนิดโครงสร้างที่สร้างจากคำหลัก struct ผลที่เกิดตามมาคือจะเกิดการจองเนื้อที่เพื่อเก็บข้อมูลที่เหมือนกันอีกชุดสำหรับฟังก์ชันซึ่งเป็นการสิ้นเปลืองเนื้อที่ อย่างไรก็ตามการส่งตัวแปรแบบเรียกด้วยมูลค่ายังคงมีความสำคัญเนื่องจากบางครั้งอาจไม่ต้องการให้ฟังก์ชันเปลี่ยนแปลงค่าในตัวแปรนั้น

การใช้ตัวชี้อ้างถึงสมาชิกแต่ละตัวของตัวแปรโครงสร้างสามารถทำได้โดยใช้รูปแบบดังนี้

ตัวชี้->ชื่อสมาชิก;

ตัวอย่างต่อไปนี้แสดงการเรียกใช้ฟังก์ชันโดยการส่งตัวแปรแบบอ้างอิงสำหรับข้อมูลชนิดโครงสร้าง

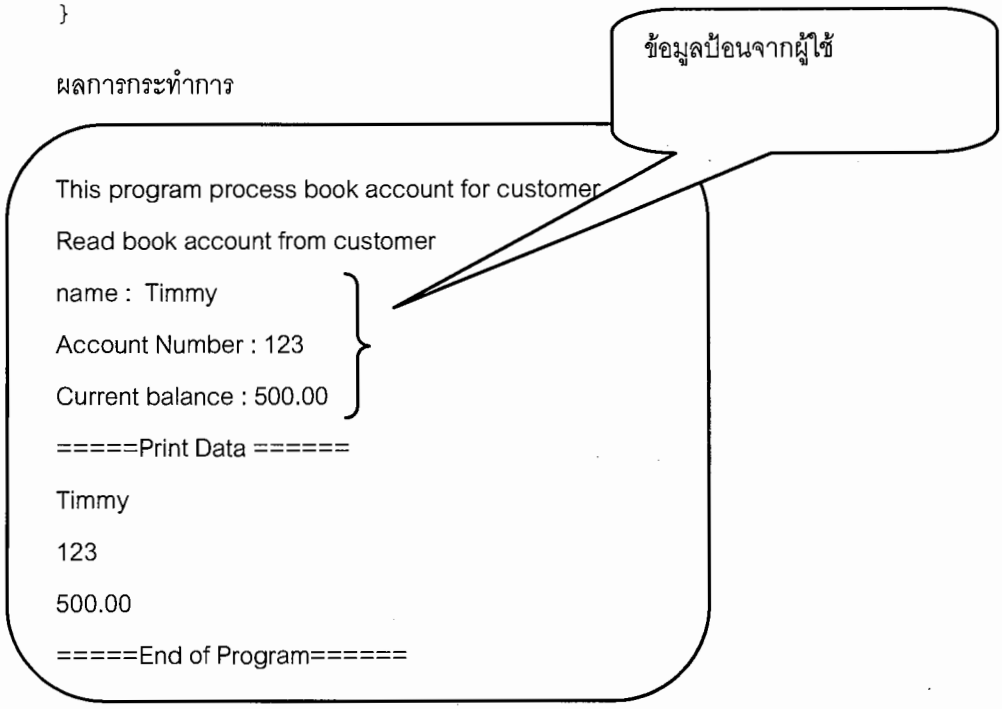
ตัวอย่าง 7.9

การเรียกฟังก์ชันโดยใช้ตัวชี้อ้างอิงไปยังตัวแปรโครงสร้าง

```
#include <stdio.h>
typedef struct {
    char name[50];
    int accNo;
    float newBalance;
} bookAcc;
bookAcc *readData();
void printData(bookAcc *);
void main()
{
    bookAcc *customer;
    printf("This program process book account for customer\n");
    customer = readData();
    printData(customer);
    printf("=====End of Program======");
}
```

```
bookAcc *readData()
{
    bookAcc tempAcc;
    bookAcc *AccPtr;
    printf("Read book account from customer\n");
    printf("name : ");
    scanf("%[^\n]", tempAcc.name);
    printf("Account Number : ");
    scanf("%d", &tempAcc.accNo);
    printf("Current balance : ");
    scanf("%f", &tempAcc.newBalance);
    AccPtr = (bookAcc *)malloc(sizeof(struct bookAcc));
    AccPtr = &tempAcc
    return AccPtr;
}
void printData(bookAcc *AccPtr)
{
    printf("=====Print Data ======");
    printf("%s\n", AccPtr->name);
    printf("%d\n", AccPtr->accNo);
    printf("%.2f\n",AccPtr->newBalance);
}
```

ผลการกระทำการ



จากตัวอย่างข้างต้น ฟังก์ชัน readData() จะทำการรับข้อมูลเข้าจากผู้ใช้และคืนค่าเป็นเลขที่อยู่ในหน่วยความจำที่ข้อมูลนั้นเก็บอยู่ (สังเกตที่เครื่องหมายดอกจันหน้าชื่อฟังก์ชัน) ส่วนของฟังก์ชัน main() จะต้องมีตัวแปรชนิดตัวชี้มารับค่าที่ได้จากการกระทำการของฟังก์ชัน readData()

7.7 การใช้ตัวชี้กับตัวแปรชนิดแถวลำดับ

ตัวแปรชนิดตัวชี้สามารถช่วยอำนวยความสะดวกในการเข้าถึงข้อมูลที่เก็บอยู่ในแถวลำดับ (array)ได้ โดยในขั้นตอนแรกจะต้องกำหนดให้ตัวแปรชนิดตัวชี้ชี้ไปยังสมาชิกตัวแรกของตัวแปรแถวลำดับเสียก่อน ซึ่งมี 2 รูปแบบดังนี้

รูปแบบที่ 1

ตัวแปรชนิดตัวชี้ = ตัวแปรแถวลำดับ;

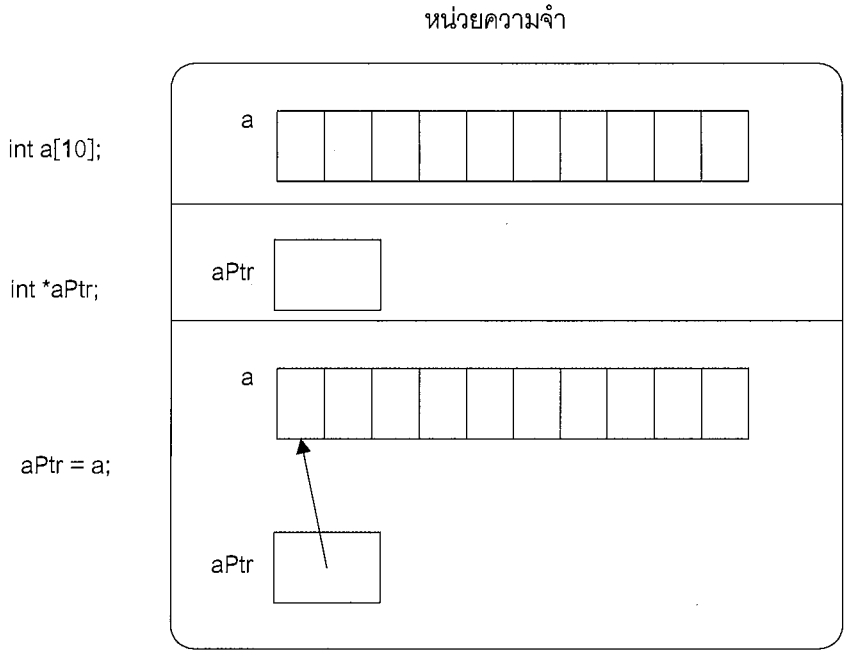
รูปแบบที่ 2

ตัวแปรชนิดตัวชี้ = &ตัวแปรแถวลำดับ[0];

ตัวอย่าง 7.10

การใช้ตัวชี้ในการอ้างถึงแถวลำดับ

```
int a[10];
int *aPtr;
aPtr = a;
```



รูปที่ 7.5 ลำดับเหตุการณ์ใช้ตัวแปรชนิดตัวชี้ในการอ้างอิงตัวแปรชนิดแถวลำดับ

หลังจากข้อความสั่ง aPtr = a; สามารถใช้ตัวแปรชนิดตัวชี้นี้ในการเข้าถึงข้อมูลในแต่ละสมาชิกของแถวลำดับได้ดังนี้

1. การเพิ่มค่าตัวชี้หนึ่งลำดับ (increment pointer)

คือการสั่งให้ตัวชี้เลื่อนไปยังสมาชิกตัวถัดไปหนึ่งลำดับ โดยใช้เครื่องหมายบวกบวก เช่น aPtr++; หรือ aPtr = aPtr+1;

2. การลดค่าตัวชี้หนึ่งลำดับ (decrement pointer)

คือการสั่งให้ตัวชี้เลื่อนไปยังสมาชิกตัวก่อนหน้าหนึ่งลำดับ โดยใช้เครื่องหมายลบลบ เช่น aPtr--; หรือ aPtr = aPtr-1;

3. การเพิ่มหรือลดค่าตัวชี้มากกว่าหรือน้อยกว่าหนึ่งลำดับ (add/substract pointer)

คือการสั่งให้ตัวชี้ชี้ข้ามไปยังสมาชิกตัวถัดไปหรือตัวก่อนหน้าตามตัวเลขที่ระบุ เช่น aPtr = aPtr+2; หมายถึงการสั่งให้ตัวชี้ aPtr ชี้ไปยังสมาชิกที่อยู่ถัดไป 2 ลำดับ เช่น ถ้าเดิมตัวชี้ aPtr ชี้อยู่ที่ a[0] เมื่อใช้ข้อความสั่ง aPtr = aPtr+2; จะทำให้ตัวชี้นี้ชี้ไปยังแถวลำดับ a[2]

4. การนำตัวแปรชนิดตัวชี้มาลบกัน

คือการนำตัวแปรชนิดตัวชี้สองตัวมาลบกันซึ่งผลลัพธ์ที่ได้คือ ระยะห่างระหว่างสมาชิกในตัวแปรแถวลำดับ+1 หรือจำนวนสมาชิกที่อยู่ระหว่างตัวชี้ทั้งสองบวกด้วย 1 นั่นเอง เช่น ถ้าตัวชี้ aPtr ชี้อยู่ที่ a[2] และตัวแปรชนิดตัวชี้ bPtr ชี้อยู่ที่ a[5]

ข้อความสั่ง x = bPtr-aPtr; จะได้ผลลัพธ์เป็น 3

5. การเปรียบเทียบตัวแปรชนิดตัวชี้

คือการตรวจสอบว่า ตัวชี้ใดชี้ไปที่สมาชิกลำดับที่มากกว่า น้อยกว่า หรือชี้ไปที่สมาชิกตัวเดียวกัน โดยการเปรียบเทียบนี้จะคืนค่าจริงหรือเท็จ ดังนั้นจึงมักนำไปใช้ร่วมกับข้อความสั่ง if หรือข้อความสั่ง while ตัวอย่างเช่น

```
if (aPtr == bPtr)
    printf("finish ");
```

เป็นการตรวจสอบว่า ตัวแปรชนิดตัวชี้ aPtr ชี้ไปที่สมาชิกตัวเดียวกันกับตัวชี้ bPtr หรือไม่

คำสั่ง

```
if (aPtr > bPtr)
    printf("aPtr is behind bPtr");
```

ตัวอย่าง 7.11

การประมวลผลแถวลำดับโดยไม่ใช้ตัวชี้

```
#include <stdio.h>
int array[] = {4, 5, 8, 9, 8, 1, 0, 1, 9, 3};
int index;
void main()
{
    index = 0;
    while (array[index] != 0)
        ++index;
     printf("Number of elements before zero %d\n", index);

}
```

โปรแกรมข้างต้นมีวัตถุประสงค์ในการนับจำนวนตัวเลขที่อยู่หน้าเลข 0 โดยไม่ใช้ตัวชี้ จะสังเกตเห็นว่าการเข้าถึงสมาชิกในแถวลำดับทำได้โดยการอ้างถึงชื่อตัวแปรแถวลำดับที่ระบุสมาชิกที่ต้องการ (array[index]) และการเลื่อนไปหาสมาชิกตัวถัดไปทำได้โดยการเพิ่มค่าของตัวแปร index อีก 1

ตัวอย่าง 7.12

การประมวลผลแถวลำดับโดยใช้ตัวชี้

```
#include <stdio.h>
int array[] = {4, 5, 8, 9, 8, 1, 0, 1, 9, 3};
int *array_ptr;
void main()
{
    array_ptr = array;
    while (*array_ptr != 0)
        ++array_ptr;
    printf("Number of elements before zero %d\n", array_ptr - array);
}
```

จากโปรแกรมข้างต้นใช้ตัวชี้ชื่อ array_ptr ในการประมวลผลแถวลำดับและเข้าถึงข้อมูลสมาชิกแต่ละตัวในแถวลำดับด้วย *array_ptr หลังจากนั้นทำการเลื่อนตัวชี้ด้วยข้อความสั่ง ++array_ptr;

ตัวอย่าง 7.13

การนำตัวแปรชนิดตัวชี้มาประมวลผลกับข้อมูลชนิดแถวลำดับ

```
#include <stdio.h>
#include<stdlib.h>
void main()
{
    char sentence[10];
    char *Ptr;
    int count;
    Ptr = sentence;
    for (count = 0; count <10;count++)
    {
        *Ptr = getchar();
        ++Ptr;
    }
    for (count = 0; count<10; count++)
    {
        --Ptr;
        putchar(*Ptr);
    }
}
```

โปรแกรมในตัวอย่าง 7.13 เป็นการนำตัวแปรชนิดตัวชี้ชื่อ Ptr เพื่อรับข้อมูลจากแป้นพิมพ์ทีละอักขระโดยใช้ฟังก์ชัน getchar() และนำมาใส่ในแถวลำดับชื่อ sentence[ ] เมื่อรับข้อมูลครบทั้งสิบตัวแล้ว โปรแกรมจะทำการแสดงผลลัพธ์โดยแสดงอักขระออกทางจอภาพทีละตัวโดยใช้ฟังก์ชัน putchar()

ตัวอย่าง 7.14

การส่งผ่านตัวแปรแถวลำดับไปยังฟังก์ชันโดยใช้ตัวชี้

```
#include <stdio.h>
long SumArray( int *tbl, int ct );
void main()
{
    int Y[5] = { 5, 7, 9, 11, 13 };
    long s;
    s = SumArray( Y, 5 );
    printf("sum is %d", s);
}
```

```
long SumArray( int *tbl, int ct )
{
    int i;
    long summ = 0;
    for( i = 0; i < ct; i++ )
    {
        summ += *tbl;    /* สะสมค่า */
        ++tbl;           /* เลื่อนไปยังสมาชิกลำดับถัดไป */
    }
    return (summ);
}
```

7.8 การใช้ตัวชี้กับสายอักขระ

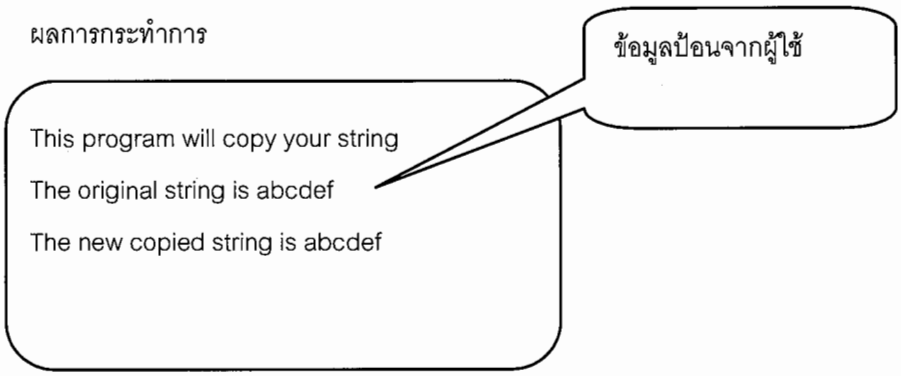
ตัวแปรชนิดตัวชี้สามารถนำมาประมวลผลกับสายอักขระได้โดยสามารถสั่งให้ตัวชี้ชี้ไปที่สายอักขระได้เช่นเดียวกับตัวแปรแถวลำดับโดยไม่ต้องมีเครื่องหมาย & อยู่หน้าตัวชี้ ดังตัวอย่างต่อไปนี้

ตัวอย่าง 7.15

โปรแกรมคัดลอกสายอักขระ

```
#include <stdio.h>
void strcopy(char *s1,char *s2)
{
    while (*s2 != '\0')
    {
        *s1 = *s2;
        s1++;
        s2++;
    }
}
void main()
{
    printf("This program will copy your string \n");
    scanf("%s", str1);
    strcopy(str1, str2);
    printf("The original string is%s\n", str1);
    printf("The new copied string is %s\n", str2);
}
```

ผลการกระทำการ



แบบฝึกหัดบทที่ 7

1. จงใช้ตัวชี้เพื่อเขียนโปรแกรมหาผลรวมของเลขคู่ที่เก็บอยู่ในแถวลำดับ 1 มิติ ขนาด 10 หน่วย

2. จงเขียนโปรแกรมเพื่อแสดงการจัดเรียงคำ 3 คำในทุกรูปแบบที่เป็นไปได้ โดยรับข้อมูลแต่ละคำจากผู้ใช้ เช่น คำที่รับจากผู้ใช้เป็นดังนี้ red green blue

   ผลการกระทำการของโปรแกรม คือ

   red green blue

   green red blue

   red blue green

   green blue red

   blue red green

   blue green red

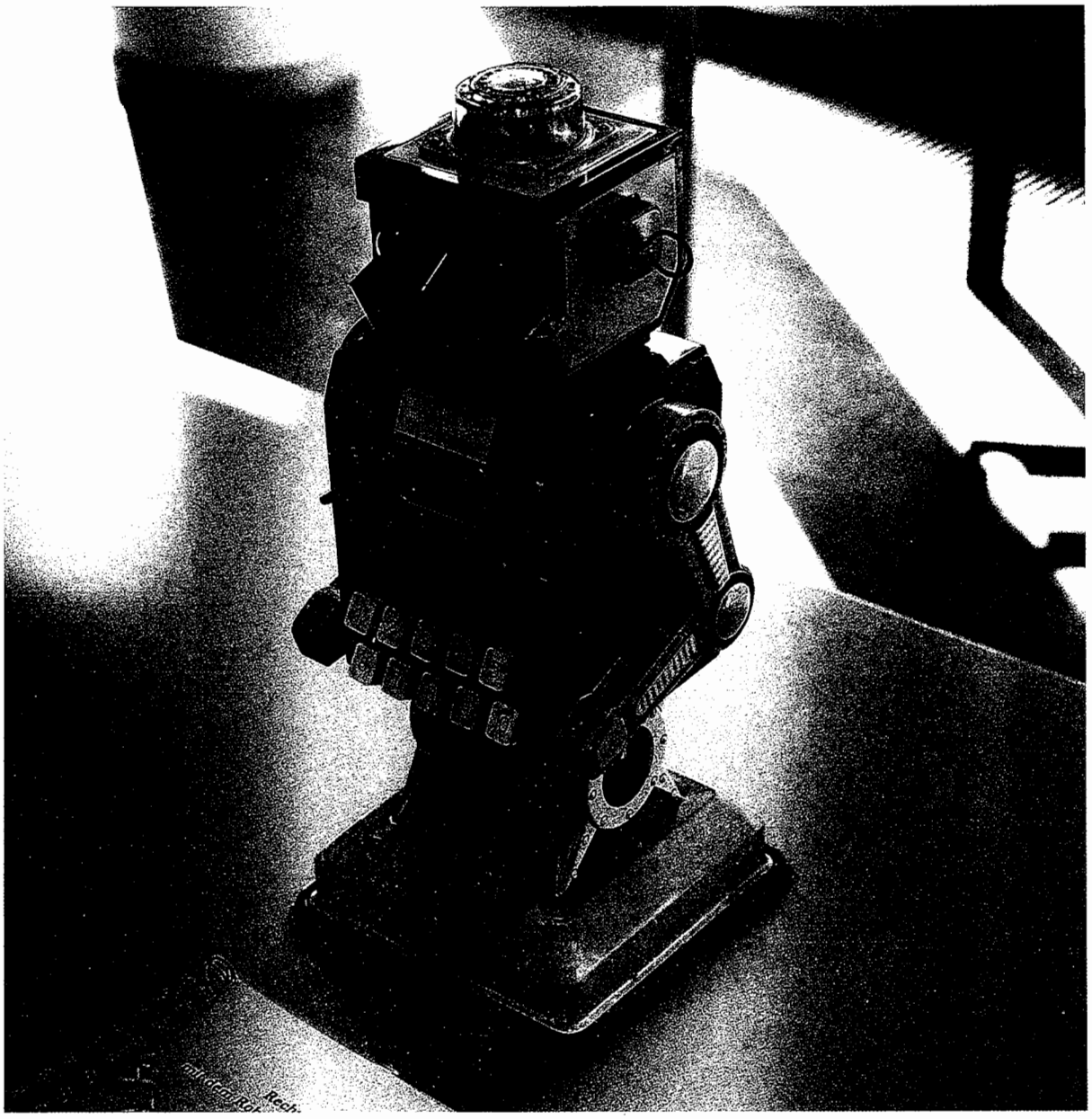
3. จงใช้ตัวชี้เพื่อเขียนโปรแกรมรับข้อมูลสายอักขระและทำการแปลงสายอักขระให้มีลักษณะเป็นตัวพิมพ์ใหญ่สลับด้วยตัวพิมพ์เล็ก เช่น

   สายอักขระที่รับจากผู้ใช้เป็นดังนี้ mathematics

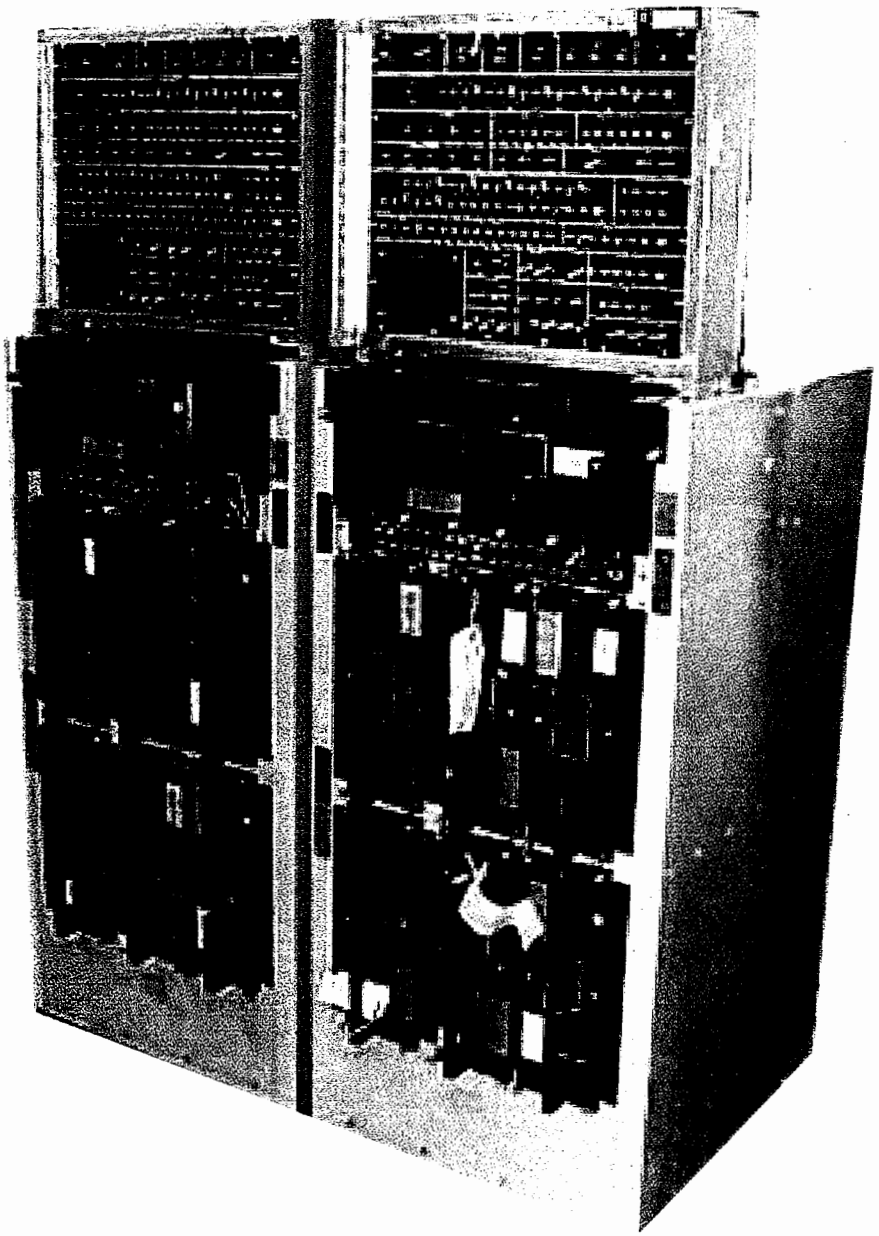
   ผลการกระทำการของโปรแกรม คือ MaThEmAtIcS

4. จงเขียนฟังก์ชัน writeToScreen(char *Ptr) ที่เลียนแบบการทำงานของฟังก์ชัน puts() โดยผลการกระทำการของฟังก์ชันนี้คือการพิมพ์ข้อความที่รับเข้ามาเป็นพารามิเตอร์ออกทางจอภาพโดยไม่ใช่ฟังก์ชัน puts()

5. จงเขียนฟังก์ชัน concatStr(char *s1, char *s2) ที่เลียนแบบการทำงานของฟังก์ชัน strcat() เพื่อทำการต่อสายอักขระ s1 และ สายอักขระ s2 เข้าด้วยกัน



หุ่นยนต์ของเล่นจากประเทศญี่ปุ่น ทำงานโดยใช้แบตเตอรี่
มีแป้นสำหรับให้ป้อนตัวเลข และช่องแสดงผล



คอมพิวเตอร์เมนเฟรมที่ใช้ในปัจจุบัน

บทที่ 8
แฟ้ม

ในการประมวลผลข้อมูลด้วยคอมพิวเตอร์นั้นนอกจากการรับข้อมูลเข้าทางแป้นพิมพ์แล้วยังสามารถจัดเตรียมข้อมูลเก็บไว้ในแฟ้ม (file) เพื่อให้โปรแกรมทำการเปิดแฟ้ม อ่านข้อมูลจากแฟ้มเพื่อทำการประมวลผล และแสดงผลลัพธ์ซึ่งอาจแสดงผลออกทางจอภาพหรือเก็บลงในแฟ้มข้อมูล แฟ้มแบ่งได้เป็น 2 ชนิดคือ แฟ้มข้อความ (text file) และแฟ้มชนิดไบนารี (binary file) ในการประมวลผลแฟ้มข้อมูลนั้น จะต้องทำให้ตัวแปลโปรแกรม (compiler) ทราบถึงชุดข้อความสั่งงานมาตรฐานที่ภาษาซีได้เตรียมไว้ให้ โดยจะต้องเขียนข้อความสั่งตัวประมวลผลก่อน #include <stdio.h> ไว้ที่ส่วนต้นของโปรแกรม สำหรับเนื้อหาในบทนี้จะกล่าวถึงการนำแฟ้มข้อมูลมาประมวลผลโดยจะเริ่มจาก การเปิดและปิดแฟ้ม การอ่านและเขียนข้อมูลลงแฟ้มตามลำดับ

8.1 การเปิดและปิดแฟ้ม

การเริ่มต้นใช้งานแฟ้มข้อมูลในโปรแกรมภาษาซีจะเริ่มต้นจากการประกาศตัวแปรที่ใช้อ้างอิงไปยังแฟ้ม โดยที่ตัวแปรนี้จะต้องมีชนิดเป็นตัวชี้ที่อ้างอิงไปยังเนื้อที่ในดิสก์ที่เก็บแฟ้มข้อมูลนั้นอยู่ โดยมีรูปแบบดังนี้

```
FILE *ตัวชี้แฟ้ม;
```

เช่น FILE *fp; เป็นการประกาศใช้ตัวแปรชื่อ fp ให้มีชนิดเป็นตัวชี้ที่ชี้ไปที่ตำแหน่งที่อยู่ที่มีแฟ้มข้อมูล การเปิดแฟ้มข้อมูลจะต้องใช้ฟังก์ชัน fopen() โดยมีรูปแบบการใช้งานดังนี้

```
ตัวชี้แฟ้ม = fopen(ชื่อแฟ้มข้อมูล, วิธีการเปิดแฟ้ม);
```

ชื่อแฟ้มข้อมูลจะเขียนในรูปแบบสายอักขระ เช่น "datafile.dat" ส่วนวิธีการเปิดแฟ้มจะเขียนในรูปแบบสายอักขระเช่นเดียวกัน รูปแบบของวิธีการเปิดแฟ้มข้อความแสดงในตารางที่ 8.1

วิธีการเปิดแฟ้ม	ในกรณีที่มีแฟ้มอยู่แล้ว	ในกรณีที่ยังไม่มีแฟ้มอยู่เดิม
"r"	เปิดแฟ้มสำหรับอ่านอย่างเดียว	เกิดความผิดพลาดไม่สามารถเปิดได้
"w"	เปิดแฟ้มใหม่สำหรับเขียน	จะสร้างแฟ้มใหม่
"a"	เปิดแฟ้มสำหรับเขียนต่อท้าย	จะสร้างแฟ้มใหม่
"r+"	เปิดแฟ้มเพื่ออ่านและเขียน	เกิดความผิดพลาด ไม่สามารถเปิดได้
"w+"	เปิดแฟ้มใหม่เพื่ออ่านและเขียน	จะสร้างแฟ้มใหม่
"a+"	เปิดแฟ้มสำหรับอ่านและเขียนต่อท้าย	จะสร้างแฟ้มใหม่

ตารางที่ 8.1 วิธีการเปิดแฟ้มข้อความแบบต่าง ๆ

เมื่อเปิดแฟ้มโดยใช้วิธีการเปิด "w" และ "w+" จะเป็นการสร้างแฟ้มใหม่เสมอ ดังนั้นถ้าตั้งชื่อแฟ้มตรงกับแฟ้มที่มีอยู่เดิมจะทำให้ลบข้อมูลที่มีอยู่เดิมในแฟ้มนั้น เมื่อการเปิดแฟ้มข้อมูลทำได้สำเร็จฟังก์ชัน fopen() จะคืนค่าเป็นตำแหน่งที่อยู่มาเก็บไว้ในตัวชี้แฟ้ม แต่ถ้าเปิดแฟ้มไม่สำเร็จ fopen() จะคืนค่า NULL

สำหรับแฟ้มชนิดไบนารีจะมีวิธีการเปิดแฟ้มแสดงในตารางที่ 8.2

วิธีการเปิดแฟ้ม	ในกรณีที่มีแฟ้มอยู่แล้ว	ในกรณีที่ยังไม่มีแฟ้มอยู่เดิม
"rb"	จะเปิดแฟ้มสำหรับอ่านอย่างเดียว	เกิดความผิดพลาดไม่สามารถเปิดได้
"wb"	จะเปิดแฟ้มใหม่สำหรับเขียน	จะสร้างแฟ้มใหม่
"ab"	จะเปิดแฟ้มสำหรับเขียนต่อท้าย	จะสร้างแฟ้มใหม่
"r+b"	จะเปิดแฟ้มเพื่ออ่านและเขียน	เกิดความผิดพลาดไม่สามารถเปิดได้
"w+b"	จะเปิดแฟ้มใหม่เพื่ออ่านและเขียน	จะสร้างแฟ้มใหม่
"a+b"	จะเปิดแฟ้มสำหรับอ่านและเขียนต่อท้าย	จะสร้างแฟ้มใหม่

ตารางที่ 8.2 วิธีการเปิดแฟ้มชนิดไบนารีแบบต่างๆ

เมื่อเปิดแฟ้มข้อมูลได้สำเร็จจะเกิดกลไกการระบุตำแหน่งในแฟ้มที่จะทำการอ่านหรือเขียนข้อมูลเรียกตัวชี้ระบุตำแหน่งนี้ว่าตัวระบุตำแหน่งในแฟ้ม (file position marker) ในกรณีที่เปิดแฟ้มโดยใช้วิธีการเปิด "r", "w", "r+", "w+" ตัวระบุตำแหน่งในแฟ้ม จะชี้อยู่ที่ตำแหน่งเริ่มต้นของแฟ้มข้อมูล แต่ถ้าใช้วิธีการเปิด "a" หรือ "a+" ตัวระบุตำแหน่งในแฟ้มจะชี้อยู่ที่ตำแหน่งท้ายแฟ้มข้อมูล เมื่อมีการอ่านหรือเขียนข้อมูลในแฟ้ม ตัวระบุตำแหน่งในแฟ้มจะเลื่อนตำแหน่งไปเรื่อยๆ ซึ่งตำแหน่งของตัวระบุตำแหน่งในแฟ้มนี้จะสามารถถูกกำหนดโดยนักเขียนโปรแกรมได้

การปิดแฟ้มข้อมูลทำได้โดยใช้ฟังก์ชัน fclose() มีรูปแบบดังนี้

```
fclose(ตัวชี้แฟ้ม);
```

ฟังก์ชัน fclose() จะคืนค่า 0 ถ้าการปิดแฟ้มข้อมูลทำได้สำเร็จ

ตัวอย่าง 8.1

การเปิดและปิดแฟ้ม

```
#include <stdio.h>
void main()
{
      FILE *fptr;
      if ((fptr = fopen("datafile.dat","r"))== NULL)
            printf("can not open file\n");
      else
      {
            printf("The file is opened now\n");
            fclose(fptr);
      }
}
```

ตัวอย่างข้างต้นเป็นการเปิดแฟ้มชื่อ datafile.dat โดยโปรแกรมจะทำการตรวจสอบว่ามีแฟ้มนั้นอยู่ในดิสก์หรือไม่ ถ้าไม่พบแฟ้มนี้จะแสดงข้อความ can not open file ทางจอภาพ ในกรณีที่พบแฟ้มจะทำการเปิดแฟ้มและแสดงข้อความ The file is opened now และทำการปิดแฟ้มโดยใช้ฟังก์ชัน fclose()

8.2 การอ่านและเขียนแฟ้ม

8.2.1 การอ่านและเขียนข้อมูลทีละอักขระจากแฟ้ม

เมื่อทำการเปิดแฟ้มเรียบร้อยแล้วสามารถอ่านข้อมูลจากแฟ้มทีละอักขระซึ่งทำได้โดยใช้ฟังก์ชัน fgetc() ซึ่งมีรูปแบบดังนี้

ตัวแปรชนิดอักขระ = fgetc(ตัวชี้แฟ้ม);

เช่น

c = fgetc(fptr);

โดยที่ c เป็นตัวแปรชนิดอักขระ

fptr เป็นตัวชี้แฟ้ม

ฟังก์ชัน fgetc() จะคืนค่าอักขระให้แก่ตัวแปร c ในกรณีที่อ่านข้อมูลไปจนถึงจุดจบของแฟ้ม ฟังก์ชัน fgetc() จะคืนค่า EOF (End Of File)

การเขียนข้อมูลชนิดอักขระลงในแฟ้มข้อมูลสามารถทำได้โดยใช้ฟังก์ชัน fputc() โดยมีรูปแบบดังนี้

fputc(ตัวอักขระ,ตัวชี้แฟ้ม);

เช่น

fputc('x',fptr); หรือ

fputc(var,fptr);

เมื่อ var เป็นตัวแปรชนิด char และมีข้อมูลชนิดอักขระเก็บอยู่ภายใน

ตัวอย่าง 8.2

การใช้งานฟังก์ชัน fgetc() และ fputc()

```
#include <stdio.h>
#include <stdlib.h>                    // ตัวประมวลผลก่อนสำหรับฟังก์ชัน exit()
void main()
{
      FILE *f_read, *f_write;
      char data;
      if (f_read  = fopen("input.dat","r") == NULL)
      {
            printf("can not open file input.dat \n");
            exit(1);                   //จบโปรแกรมและคืนค่า 1
      }
      if (f_write = fopen("output.dat", w") == NULL)
      {
            printf("can not open file output.dat \n");
            exit(1);
      }
      while ((data = fgetc(f_read)) != EOF)
      {
            putchar(data);             // แสดงตัวอักขระออกทางจอภาพ
            fputc(data, f_write);      // เขียนอักขระลงแฟ้มข้อมูล
      }
      fclose(f_read);
      fclose(f_write);
}
```

จากตัวอย่าง 8.2 มีการเปิดใช้แฟ้มจำนวน 2 แฟ้มคือ input.dat (เปิดเพื่ออ่าน) และ output.dat (เปิดเพื่อเขียน) หลังจากนั้นโปรแกรมจะทำการอ่านข้อมูลทีละอักขระจากแฟ้ม input.dat และพิมพ์อักขระทางจอภาพโดยใช้ฟังก์ชัน putchar() หลังจากนั้นจะเขียนอักขระลงในแฟ้มข้อมูลโดยใช้ฟังก์ชัน fputc() กระบวนการอ่านข้อมูลจากแฟ้ม input.dat และเขียนข้อมูลลงแฟ้ม output.dat นี้จะกระทำไปจนกว่าจะถึงจุดจบของแฟ้มเมื่อตัวอักขระที่อ่านจากแฟ้มมีค่าเป็น EOF หลังจากนั้นโปรแกรมจะทำการปิดแฟ้มทั้งสองโดยใช้ฟังก์ชัน fclose()

EOF หมายถึงจุดสิ้นสุดของแฟ้ม (End Of File) ซึ่งได้ประกาศไว้เป็นค่าคงที่ในแฟ้ม stdio.h โดยมีค่าเป็น -1

ฟังก์ชัน feof()

การตรวจสอบจุดจบของแฟ้มสามารถทำได้โดยใช้ฟังก์ชัน feof() ซึ่งฟังก์ชันนี้จะคืนค่าที่ไม่ใช่ 0 เมื่อโปรแกรมเข้าถึงจุดสิ้นสุดของแฟ้ม และจะคืนค่า 0 เมื่อยังไม่ถึงจุดสิ้นสุดของแฟ้ม ฟังก์ชัน feof() มีรูปแบบการใช้งานดังนี้

feof(ตัวชี้แฟ้ม);

ตัวอย่าง 8.3

การใช้ฟังก์ชัน feof()

```
#include <stdio.h>
void main()
{
    FILE *fp;
    …
    …
    while (!feof(fp))
        fgetc(fp);
    …
    …
}
```

จากตัวอย่างข้างต้นถ้าแฟ้มที่ตัวชี้ fp อ้างถึงยังไม่ถึงจุดสิ้นสุดของแฟ้มจะทำการอ่านข้อมูลทีละอักขระ ซึ่งการตรวจสอบในลักษณะนี้ดีกว่าการใช้ค่าคงที่ EOF เนื่องจากการอ่านแฟ้มชนิดไบนารีอาจได้ค่าเป็น –1 ซึ่งเป็นค่าของข้อมูลตามปกติไม่ใช่จุดสิ้นสุดของแฟ้มที่แท้จริง

ฟังก์ชัน ferror()

เป็นฟังก์ชันที่ใช้ในการตรวจสอบสถานะความผิดพลาดของการประมวลผลแฟ้มที่ตัวชี้นั้นอ้างถึงในครั้งล่าสุด มีรูปแบบคำสั่งดังนี้

ferror(ตัวชี้แฟ้ม);

ตัวอย่าง 8.4

การใช้ฟังก์ชัน ferror() ในการประมวลผลแฟ้ม

```
#include <stdio.h>
void main()
{
    FILE  *F;
    …
    …
    while (!ferror(F))
    {
        fgetc(F);
        if (ferror(F))
        {
            printf("Error processing file\n");
            break;
        }
    }
}
```

จากตัวอย่างโปรแกรมข้างต้นเมื่อฟังก์ชัน ferror() คืนค่าที่มากกว่า 1 (จริง) โปรแกรมจะแสดงข้อความ Error processing file และหยุดการทำงาน

ตัวอย่าง 8.5

โปรแกรมคัดลอกแฟ้ม

```
#include <stdio.h>
void main()
{
    FILE *first;
    FILE *second;
    char   c;
    if ((first = fopen("first.dat", "rb")) == NULL)
    {
        printf(" Error opening source file\n");
        return;
    }
    if ((second = fopen("second.dat", "wb")) == NULL)
    {
        printf(" Error opening destination file\n");
        return;
    }
```

```
    while (!feof(first))
    {
        c = fgetc(first);
        if (ferror(first))
        {
            printf("Error reading from source file\n");
            return;
        }
        if (!feof(first))
            fputc(c, second);
        if (ferror(second))
        {
            printf("Error writing to destination file\n");
            return;
        }
    }
    fclose(first);
    fclose(second);
}
```

โปรแกรมในตัวอย่างข้างต้นทำการเปิดแฟ้มต้นฉบับ (first.dat) และแฟ้มปลายทาง (second.dat) ถ้ามีความผิดพลาดในการเปิดแฟ้มใดก็จะแสดงข้อความให้ผู้ใช้ทราบและจบการทำงาน ในกรณีที่ไม่มีความผิดพลาดเกิดขึ้นจะเริ่มอ่านข้อมูลจากแฟ้มต้นฉบับโดยใช้ฟังก์ชัน fgetc() และเขียนข้อมูลดังกล่าวลงในแฟ้มปลายทาง ในระหว่างการอ่านและเขียนข้อมูลจะมีการตรวจสอบข้อผิดพลาดตลอดเวลา ถ้ามีความผิดพลาดเกิดขึ้นจะจบการทำงานทันที เมื่ออ่านและเขียนข้อมูลจนครบโปรแกรมจะทำการปิดแฟ้มทั้งสองโดยใช้ฟังก์ชัน fclose()

ตัวอย่าง 8.6

โปรแกรมตรวจสอบข้อมูลในแฟ้ม

```
#include <stdio.h>
void main()
{
    FILE *first;
    FILE *second;
    char  c1,c2, equal;
    unsigned long lg;
    if ((first = fopen("first.dat", "rb")) == NULL)
    {
        printf(" Error opening source file\n");
        return;
    }
```

```
    if ((second = fopen("second.dat", "rb")) == NULL)
    {
        printf("Error opening destination file\n");
        return;
    }
    equal = 1;
    lg = 0;
    while (!feof(first))
    {
        c1 = fgetc(first);
        if (ferror(first))
        {
            printf("Error reading from first file\n");
            return;
        }
        c2 = fgetc(second);
        if (ferror(second))
        {
            printf("Error reading from second file\n");
            return;
        }
        if (c1 != c2)
        {
            printf("not equivalence at byte %lu", lg);
            equal = 0;
            break;
        }
        lg++;
    }
    if (equal)
        printf("These two files are equivalence\n");
    fclose(first);
    fclose(second);
}
```

โปรแกรมในตัวอย่างข้างต้นจะทำงานในลักษณะเดียวกับตัวอย่าง 8.5 แต่จะทำการตรวจสอบอักขระที่อ่านขึ้นมาจากแฟ้มทั้งสองว่าตรงกันหรือไม่และเพิ่มค่าตัวแปร lg ทีละหนึ่ง เพื่อให้ทราบถึงตำแหน่งข้อมูลในแฟ้มที่ถูกตรวจสอบเมื่อพบอักขระที่ต่างกันโปรแกรมจะแสดงข้อความบอกให้ผู้ใช้ทราบ ทำการปิดแฟ้มและจบการทำงาน ในกรณีที่เป็นอักขระเดียวกันจะอ่านข้อมูลต่อไปจนจบแฟ้ม

8.2.2 การอ่านและเขียนสายอักขระลงในแฟ้ม

ในการประมวลผลแฟ้มข้อความนั้นบางครั้งข้อมูลมีลักษณะเป็นกลุ่ม เช่น เป็นระเบียนที่ประกอบไปด้วยข้อมูลอักขระหลายตัวมาประกอบกันเช่น ชื่อพนักงาน ตำแหน่งของพนักงาน เป็นต้น การอ่านหรือเขียนข้อมูลทีละอักขระอาจจะไม่เหมาะสม ดังนั้นเพื่อให้เกิดความสะดวกภาษาซีได้จัดเตรียมฟังก์ชันที่ช่วยให้สามารถอ่านและเขียนข้อมูลทีละหลายอักขระได้ โดยใช้ฟังก์ชัน fgets() ซึ่งจะทำหน้าที่อ่านสายอักขระจากแฟ้มโดยที่ fgets() จะคืนค่า NULL เมื่ออ่านข้อมูลถึงจุดสิ้นสุดของแฟ้ม นอกจากนี้ฟังก์ชัน fgets() ยังเติมสัญลักษณ์ '\0' ปิดท้ายสายอักขระที่อ่านมาอีกด้วย ส่วนฟังก์ชัน fputs() จะทำหน้าที่เขียนสายอักขระลงในแฟ้ม

รูปแบบการใช้งานฟังก์ชัน fgets() มีดังนี้

fgets(ตัวชี้หรือเลขที่อยู่ของตัวแปร, จำนวนอักขระ, ตัวชี้แฟ้ม);

เช่น

fgets(record, 80, f_read)

หมายถึงการเขียนสายอักขระที่เก็บอยู่ในตัวแปร record[ ] ซึ่งได้มีการประกาศไว้เป็นแถวลำดับที่บรรจุอักขระ

ตัวเลข 80 คือขนาดของสายอักขระที่ต้องสอดคล้องกับแถวลำดับ record[ ]

f_read คือตัวชี้แฟ้ม

ส่วนฟังก์ชัน fputs() ใช้สำหรับเขียนข้อมูลลงแฟ้ม มีรูปแบบการใช้งานดังนี้

fputs(ตัวชี้หรือที่อยู่ของตัวแปร, ตัวชี้แฟ้ม);

เช่น fputs(record, fptr); โดยที่ตัวแปร record[ ] ได้มีการประกาศไว้เป็นแถวลำดับที่บรรจุอักขระ และ fptr เป็นตัวชี้แฟ้ม

ตัวอย่าง 8.7

โปรแกรมการอ่านและเขียนสายอักขระลงในแฟ้มข้อมูล

```
#include <stdio.h>
void main()
{
    FILE *fptr;
    char record[80];
    fptr = fopen("inputdata.dat", "r");
    while(fgets(record, 80, fptr) != NULL )
        fputs(record, stdout); // stdout คืออุปกรณ์แสดงผลซึ่งหมายถึงจอภาพโดยปริยาย
    fclose (fptr);
}
```

จากโปรแกรมในตัวอย่าง 8.7 จะมีการเปิดแฟ้มชื่อ inputdata.dat เพื่ออ่านข้อมูล ("r") โดยโปรแกรมจะทำการอ่านสายอักขระมาเก็บไว้ในตัวแปรชื่อ record [ ] ซึ่งเป็นแถวลำดับที่เก็บตัวอักขระได้ทั้งสิ้น 79 ตัว

ตัวอย่าง 8.8

โปรแกรมการเขียนสายอักขระลงในแฟ้ม

```
#include <stdio.h>
void  main()
{
    FILE *fPtr;
    char  record[80];
    fPtr = fopen("payroll.dat","a");
    while (fgets(record, 80, stdin) != NULL) // stdin คืออุปกรณ์รับข้อมูลซึ่งหมาย
                                             // ถึงแป้นพิมพ์โดยปริยาย
    {
        puts(record);
        printf("append data to file y/n ?");
        gets(ans);
        if (*ans == 'y')
        {
            fputs(record, fPtr);
            fputc('\n', fPtr);
        }
        printf("enter next record");
    }
    fclose(fPtr);
}
```

จากตัวอย่างโปรแกรมข้างต้นเป็นการรับข้อความจากผู้ใช้ผ่านทางแป้นพิมพ์โดยใช้ฟังก์ชัน fgets(record, 80, stdin) หลังจากนั้นจะถามผู้ใช้ว่าต้องการเขียนข้อความดังกล่างลงในแฟ้มข้อมูลหรือไม่ ถ้าผู้ใช้ตอบ y โปรแกรมจะเขียนข้อความนั้นลงในแฟ้มชื่อ payroll.dat

8.2.3 การอ่านและเขียนข้อมูลแบบมีรูปแบบ

การอ่านและเขียนข้อมูลในลักษณะนี้จะทำการจัดรูปแบบของข้อมูลให้อยู่ในชนิดที่ผู้ใช้ต้องการได้ ฟังก์ชันในการอ่านข้อมูลจากแฟ้มที่จะขอกล่าวถึงในที่นี้คือ fscanf() ซึ่งจะอ่านข้อมูลจากแฟ้มโดยมีรูปแบบดังนี้

fscanf(ตัวชี้แฟ้ม, รูปแบบ, &ตัวแปร);

สำหรับตัวแปรแถวลำดับไม่ต้องมีเครื่องหมาย &

เช่น

```
int account;
char name[30];
fscanf(filePtr, "%d%s", &account, name);
```

การเขียนข้อมูลลงในแฟ้มทำได้โดยใช้คำสั่ง fprintf โดยมีรูปแบบดังนี้

fprintf(ตัวชี้แฟ้ม, รูปแบบ, ตัวแปรที่ระบุตำแหน่งของข้อมูล);

เช่น `fprintf(filePtr, "%d%s", account, name);`

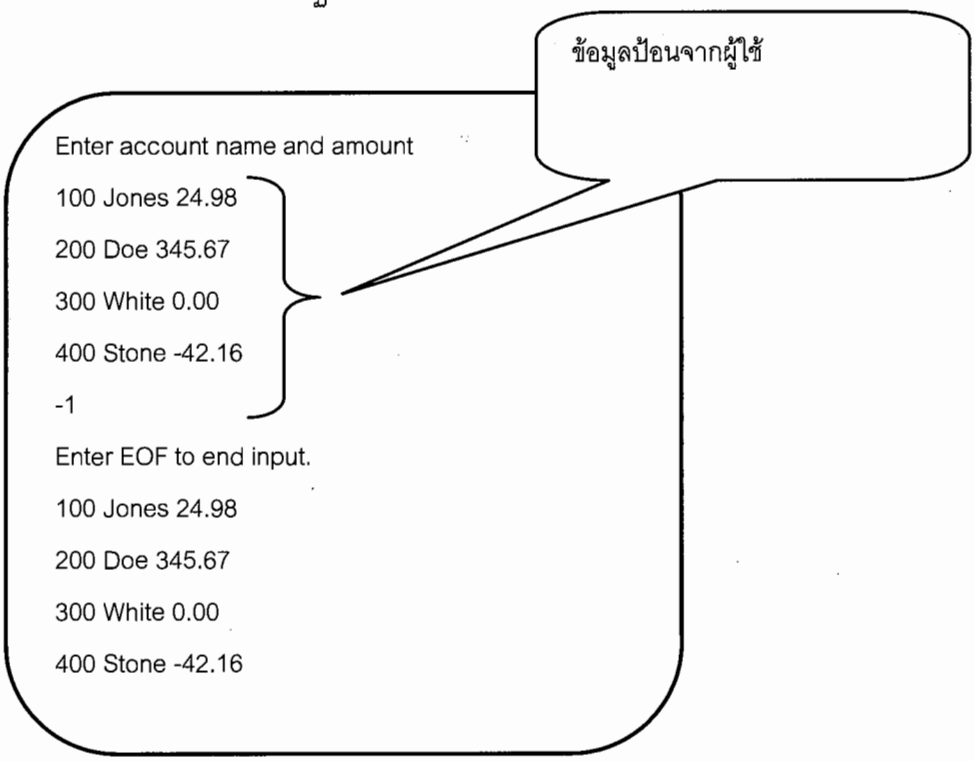
ตัวอย่าง 8.9

การประมวลผลแฟ้มโดยใช้คำสั่งการอ่านและเขียนข้อมูลอย่างมีรูปแบบ

```
#include <stdio.h>
void main()
{
    int account;
    char name[30];
    double amt;
    FILE *ptr;
    if ((ptr = fopen("customer.dat", "w")) == NULL)
        printf("could not open file\n");
    else
    {
        printf("Enter account name and amount\n");
        printf("Enter -1 to end input\n");
        scanf("%d%s%lf", &account, name, &amt);
```

```
            while (!feof(stdin))
            {
                fprintf(ptr, "%d%s%2f\n", account, name, amt);
                scanf("%d%s%1f", &account, name, &amt);
            }
            fclose(cfPtr);
        }
    }
```

ผลการกระทำการ ที่ปรากฏบนจอภาพคือ



ตัวอย่าง 8.9 มีวัตถุประสงค์ในการรับข้อมูลจากผู้ใช้ผ่านทางแป้นพิมพ์และเขียนข้อมูลดังกล่าวลงในแฟ้มชื่อ customer.dat

ตัวอย่าง 8.10

การอ่านข้อมูลจากแฟ้ม customer.dat เพื่อแสดงออกทางจอภาพ

```
#include <stdio.h>
void main()
{
    int account;
    char name[30];
    double amt;
    FILE *ptr;
    if ((ptr = fopen("customer.dat", "r")) == NULL)
        printf("could not open file\n");
    else
    {
        printf("%-10s%-13s%s\n", "Account", "Name", "amount");
        fscanf(ptr, "%d%s%lf", &account, name, &amt);
        while(!feof(ptr))
        {
            printf("%-10d%-13s%7.2f\n", account, name, amt);
            fscanf(ptr, "%d%s%lf", &account, name, &amt);
        }
        fclose(ptr);
    }
}
```

8.2.4 การอ่านและเขียนข้อมูลสำหรับแฟ้มชนิดไบนารี

แฟ้มชนิดไบนารีคือ แฟ้มที่เก็บข้อมูลในรูปแบบเลขฐานสอง การเปิดแฟ้มชนิดไบนารีเพื่ออ่านข้อมูลจะต้องกำหนดวิธีการเปิดเป็น "rb" ส่วนการเปิดแฟ้มเพื่อเขียนจะต้องกำหนดวิธีการเปิดเป็น "wb" ผู้เขียนโปรแกรมสามารถอ่านและเขียนข้อมูลลงในแฟ้มชนิดไบนารีได้โดยใช้ฟังก์ชัน fread() และ fwrite() ตามลำดับ

ฟังก์ชัน fread()

มีรูปแบบดังนี้

ขนาดข้อมูล fread(ที่อยู่ของข้อมูล, จำนวนข้อมูล, ขนาดข้อมูล, ตัวชี้แฟ้ม);

ฟังก์ชัน fwrite()

มีรูปแบบดังนี้

ขนาดข้อมูล fwrite(ที่อยู่ของข้อมูล, จำนวนข้อมูล, ขนาดข้อมูล, ตัวชี้แฟ้ม);

ตัวอย่าง 8.11

การอ่านและเขียนข้อมูลจากแฟ้มชนิดไบนารีโดยใช้ฟังก์ชัน fread() และ fwrite()

```
#include <stdio.h>
void main()
{
    FILE *Ptr;
    int x;
    if (Ptr = fopen("input.dat", "wb")) == NULL)
    {
        printf("Error opening file input.dat\n");
        return;
    }
    x = 100;
    if (fwrite(&x, sizeof(int), 1, Ptr) != 1)
    {
        printf("Error writing file input.dat\n");
        return;
    }
    fclose(Ptr);
    if (Ptr = fopen("input.dat", "rb")) == NULL)
    {
        printf("Error opening file input.dat\n");
        return;
    }
    if (fread(&x, sizeof(int), 1, Ptr) != 1)
    {
        printf("Error reading file input.dat\n");
        return;
    }
    printf("x is %d", x);
    fclose(Ptr);
}
```

ตัวอย่าง 8.12

โปรแกรมเขียนข้อมูลจากแถวลำดับลงในแฟ้มและอ่านข้อมูลเพื่อแสดงผลทางจอภาพ

```
#include <stdio.h>
double y[5] = { 5.50, 10.50, 12.6, 23.6, 45.4};
void main()
{
    int x;
    FILE *Ptr;
    if (Ptr = fopen("output.dat", "wb")) == NULL)
    {
        printf("Error opening file output.dat\n");
        return;
    }
    for (x = 0; x<10; x++)
        if (fwrite(&y[x], sizeof(double), 1, Ptr) != 1)
        {
            printf("Error writing file\n");
            return;
        }
    fclose(Ptr);
    if (Ptr = fopen("output.dat", "rb")) == NULL)
    {
        printf("Error opening file output.dat\n");
        return;
    }
    for (x = 1 ; x< 10; x++)
        if (fread(&z[x], sizeof(double),1, Ptr) != 1)
        {
            printf("Error reading file\n");
            return;
        }
        else
            printf("%f\n",z[x]);
    fclose(Ptr);
}
```

ตัวอย่าง 8.13

การอ่านและเขียนตัวแปรโครงสร้างลงในแฟ้มข้อมูล

```
#include <stdio.h>
struct Dimension
{
    float  x;
    float  y;
} record;
void main()
{
    FILE *Ptr;
    printf("please enter dimension : ");
    scanf("%f%f", &record.x, &record.y);
    if ((Ptr = fopen("output", "w")) == NULL)
    {
        printf("Error opening file\n");
        return;
    }
    fwrite(&record, sizeof(record), 1, Ptr);
    fclose();
}
```

8.3 การเข้าถึงข้อมูลในแฟ้มแบบสุ่ม (random access file)

นอกจากการเข้าถึงข้อมูลในแฟ้มแบบเรียงลำดับจากต้นแฟ้มไปยังท้ายแฟ้มแล้ว ภาษาซียังได้สร้างฟังก์ชันที่ใช้ในการเข้าถึงแฟ้มแบบสุ่มซึ่งหมายถึงการเข้าถึงข้อมูล ณ ตำแหน่งใดๆที่ผู้เขียนโปรแกรมระบุโดยใช้ฟังก์ชัน fseek() ซึ่งมีรูปแบบการใช้งานดังนี้

fseek(ตัวชี้แฟ้ม, จำนวนไบต์นับจากต้นแฟ้ม, ตำแหน่งเริ่มต้น);

โดยที่ตำแหน่งเริ่มต้นเป็นค่าคงที่ดังนี้

ตำแหน่งเริ่มต้น	ความหมาย
SEEK_SET	ค้นหาจากต้นแฟ้ม
SEEK_CUR	ค้นหาจากตำแหน่งปัจจุบันของตัวระบุตำแหน่ง
SEEK_END	ค้นหาจากท้ายแฟ้ม

ค่าของตำแหน่งเริ่มต้นเป็นค่าคงที่ได้มีการประกาศไว้แล้วในแฟ้ม stdio.h ฟังก์ชัน fseek() จะคืนค่าเป็น 0 เมื่อการค้นหาทำไม่สำเร็จ

ตัวอย่าง 8.14

การเข้าถึงข้อมูลในแฟ้ม ณ ตำแหน่งที่ 100 ไบต์นับจากต้นแฟ้ม

```
fseek(Ptr, 100, SEEK_SET);
```

เมื่อโปรแกรมกระทำการข้อความสั่งดังกล่าวเสร็จ ตัวระบุตำแหน่งในแฟ้มจะเลื่อนไป 100 ไบต์นับจากต้นแฟ้ม

นอกจากนี้นักเขียนโปรแกรมสามารถใช้ฟังก์ชัน ftell() ในการหาตำแหน่งปัจจุบันของตัวระบุตำแหน่งในแฟ้มโดยมีรูปแบบการใช้งานดังนี้

ftell(ตัวชี้แฟ้ม);

ถ้าการหาตำแหน่งปัจจุบันทำไม่สำเร็จฟังก์ชัน ftell() จะคืนค่า –1

ตัวอย่าง 8.15

การใช้ฟังก์ชัน fseek()

```
#include <stdio.h>
void main()
{
    long loc;
    FILE *Ptr;
    if (( Ptr = fopen("binary.dat", "rb")) == NULL)
    {
        printf("Error opening file binary.dat\n");
        return;
    }
    printf("enter byte to seek : ");
    scanf("%d", &loc);
    if (fseek(Ptr, loc, SEEK_SET))
    {
        printf(" Error on seeking file\n");
        return;
    }
    printf("data at location %ld is %d", loc, getc(Ptr));
    fclose(Ptr);
}
```

ตัวอย่าง 8.16

การใช้ฟังก์ชัน ftell() และ fseek() ในการคัดลอกข้อมูลจากท้ายแฟ้มไปยังต้นแฟ้มลงในแฟ้มปลายทาง

```
#include <stdio.h>
void main()
{
    FILE *source, *dest;
    long loc;
    char c;
    if ((source = fopen("input.dat", "rb")) == NULL)
    {
        printf("Error opening file input.dat\n");
        return;
    }
    if ((dest = fopen("output.dat", "wb"))== NULL)
    {
        printf("Error opening file output.dat\n");
        return;
    }
    fseek(source, 0L, SEEK_END); // ค้นหาตำแหน่งท้ายแฟ้ม
    loc = ftell(source);
    loc = loc -1;                    // เลื่อนตำแหน่งถอยหลัง 1 ไบต์เพื่อเว้นการอ่านค่า EOF
    while (loc >=0L)
    {
        fseek(source, loc, SEEK_END);
        c = fgetc(source);
        fputc(c, dest);
        loc--;
    }
    fclose(source);
    fclose(dest);
}
```

8.4 ฟังก์ชันที่เกี่ยวข้องกับการประมวลผลแฟ้ม

นอกเหนือจากการอ่านและเขียนข้อมูลลงแฟ้มแล้วภาษาซีได้จัดเตรียมฟังก์ชันต่างๆ ดังนี้

ฟังก์ชัน remove()

การลบแฟ้มข้อมูลสามารถทำได้โดยใช้ฟังก์ชัน remove() ซึ่งมีรูปแบบดังนี้

remove(ชื่อแฟ้ม);

ฟังก์ชัน remove จะคืนค่าเป็น 0 เมื่อการลบแฟ้มทำได้สำเร็จ

ฟังก์ชัน rewind()

ฟังก์ชัน rewind() ใช้สำหรับย้ายตำแหน่งของตัวระบุตำแหน่งในแฟ้มไปยังต้นแฟ้ม มีรูปแบบการใช้งานดังนี้

rewind(ตัวชี้แฟ้ม);

ฟังก์ชัน rewind จะไม่มีการคืนค่าใดๆกลับ

ฟังก์ชัน fflush()

ฟังก์ชัน fflush() ใช้สำหรับการล้างหน่วยความจำชั่วคราวที่ใช้กับแฟ้มข้อมูล มีรูปแบบการใช้งานดังนี้

fflush(ตัวชี้แฟ้ม);

ฟังก์ชัน fflush() จะคืนค่า 0 ถ้าการล้างหน่วยความจำชั่วคราวทำได้สำเร็จ และคืนค่า –1 เมื่อมีข้อผิดพลาดเกิดขึ้น

ตัวอย่าง 8.17

การใช้ฟังก์ชัน remove()

```
#include <stdio.h>
void  main()
{
    char fileName[50];
    printf("please enter filename to delete ");
    gets(fileName);
    remove(fileName);
    printf("delete successful\n");
}
```

ตัวอย่าง 8.18

การใช้ฟังก์ชัน rewind() เพื่อแสดงอักขระในแฟ้มออกทางจอภาพ 2 ครั้ง

```
#include <stdio.h>
void  main()
{
    FILE *Ptr;
    if (Ptr = fopen("input.dat", "r"))== NULL) {
        printf("Error opening file\n");
        return;
    }
    while (!feof(Ptr))
        putchar(getc(Ptr));
    rewind(Ptr);
    while (!feof(Ptr))
        putchar(getc(Ptr));
    fclose(Ptr);
}
```

โดยสรุปแล้วจะเห็นได้ว่าภาษาซีได้จัดเตรียมฟังก์ชันในการประมวลผลร่วมกับแฟ้มข้อมูลในลักษณะต่างๆ กัน นักเรียนสามารถเปิดแฟ้มที่มีอยู่เดิมหรือสร้างแฟ้มใหม่เพื่อทำการบันทึกข้อมูลลงในแฟ้ม ในขณะเดียวกันสามารถอ่านข้อมูลจากแฟ้มเพื่อนำมาประมวลผลและปรับปรุงข้อมูลในแฟ้มได้

แบบฝึกหัดบทที่ 8

1. จงเขียนโปรแกรมเพื่อพิมพ์ข้อมูลที่เก็บอยู่แฟ้มข้อความที่ผู้ใช้ระบุผ่านทางส่วนต่อประสานรายคำสั่ง (command line interface)

2. จงเขียนโปรแกรมเพื่อนับจำนวนอักขระ a ถึง z ที่อยู่ในแฟ้มข้อความ ว่ามีอย่างละกี่จำนวน

3. จงเขียนโปรแกรมเพื่อรับข้อมูลส่วนบุคคลและเขียนข้อมูลดังกล่าวลงแฟ้ม ข้อมูลส่วนบุคคลประกอบด้วย ชื่อ นามสกุล เพศ หมายเลขโทรศัพท์ ที่อยู่อีเมล์

4. จากข้อ 3 จงเขียนฟังก์ชัน search(char *name) เพื่อใช้ในการค้นหาข้อมูลส่วนบุคคลที่มีชื่อตามที่ผู้ใช้ระบุ

5. จงเขียนโปรแกรมเพื่อนับจำนวนตัวเลขทั้งหมดในแฟ้มข้อความว่ามีทั้งสิ้นกี่จำนวน

บรรณานุกรม

Al Kelley and Ira Pohl, "A Book on C : Programming in C", Second Edition, The Benjamin/Cummings Publishing Company, Inc., 1990.

Annegret Kehrbaum and Bernhard Korte, "Calculi", Westdeutscher Verlag, 1995.

Byron S. Gottfried, "Theory and Problem of Programming with C", McGraw-Hill, 1990.

Behrouz A. Forouzan, Richard F. Gilberg, "Computer Science: A Structured Programming Approach Using C", Second Edition, Brooks/Cole, 2001.

Harvey M. Deitel and Paul J. Deitel, "C How to Program", Third Edition, Prentice Hall Inc., 2001.

Jeri R. Hanly and Elliot B. Koffman, "C Program Design for Engineers", Addison Wesley, 1997.

Lawrence H. Miller, Alexander E. Quilici, "The Joy of C", John Wiley & Sons, Inc., 1997.

Peter G. Aitken, Bradley Jones, "Teach Yourself C in 21 Days", Second Edition, Sams Publishing, 1994.

Tomasz Muldner, "C for Java Programmers", Addison Wesley, 2000.

ดัชนี

ดัชนี

คอมพิวเตอร์
ISBN 974 - 92235 - 2 - 7

9 789749 223529

C 112
5500 250.00 บาท

มูลนิธิส่งเสริมโอลิมปิกวิชาการและพัฒนามาตรฐานวิทยาศาสตร์ศึกษา
ในพระอุปถัมภ์สมเด็จพระเจ้าพี่นางเธอ เจ้าฟ้ากัลยาณิวัฒนา กรมหลวงนราธิวาสราชนครินทร์
สำนักงานตั้งอยู่ในบริเวณ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ถนนพญาไท ปทุมวัน กทม. 10330
โทร. 0-2252-8916, 0-2252-8917, แฟกซ์. 0-2252-8917