บทที่ 1
ระบบคอมพิวเตอร์

ปัจจุบันคอมพิวเตอร์ได้มีบทบาทในชีวิตประจำวันเป็นอย่างมาก ตั้งแต่ตื่นนอนจนเข้านอน ข่าวสารในแต่ละวันได้นำระบบคอมพิวเตอร์ (Computer System) มาช่วยในการเผยแพร่มากขึ้น การเดินทางด้วยระบบขนส่งมวลชนที่นำระบบคอมพิวเตอร์มาใช้ในการควบคุม งานสาธารณสุขมีระบบจัดการทะเบียน การบัญชี การเงิน ธนาคารต่าง ๆ ได้นำระบบเครือข่ายมาใช้ในการบริการลูกค้า ห้างร้านบริษัทในธุรกิจต่าง ๆ ได้มีการแข่งขันมากมายโดยนำเทคโนโลยีสารสนเทศมาพัฒนาผลิตภัณฑ์ รวมทั้งสถานศึกษานำระบบต่าง ๆ เข้ามาใช้ตั้งแต่โรงเรียนอนุบาลไปจนถึงมหาวิทยาลัย ระบบลงทะเบียน ข้อสอบออนไลน์ การบันทึกผลการเรียน รายงานที่เกี่ยวข้องต่างต้องใช้โปรแกรมเพื่อจัดการข้อมูล การเรียนรู้เรื่องการเขียนโปรแกรมเบื้องต้น จำเป็นต้องรู้ถึงการทำงานของระบบคอมพิวเตอร์ ในบทนี้อธิบายถึงระบบคอมพิวเตอร์ ฮาร์ดแวร์ (Hardware) ซอฟต์แวร์ (Software) ส่วนบุคลากร (Peopleware) ความสัมพันธ์ของฮาร์ดแวร์ ซอฟต์แวร์และผู้ใช้ ภาษาการโปรแกรมคอมพิวเตอร์ สารบบแฟ้ม (File directory) และการทำงานต่าง ๆ ในระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ซึ่งระบบคอมพิวเตอร์จะต้องมีการสื่อสารกับมนุษย์ หรือบุคลากรที่เกี่ยวข้องกับระบบคอมพิวเตอร์ กระทำผ่านฮาร์ดแวร์และซอฟต์แวร์

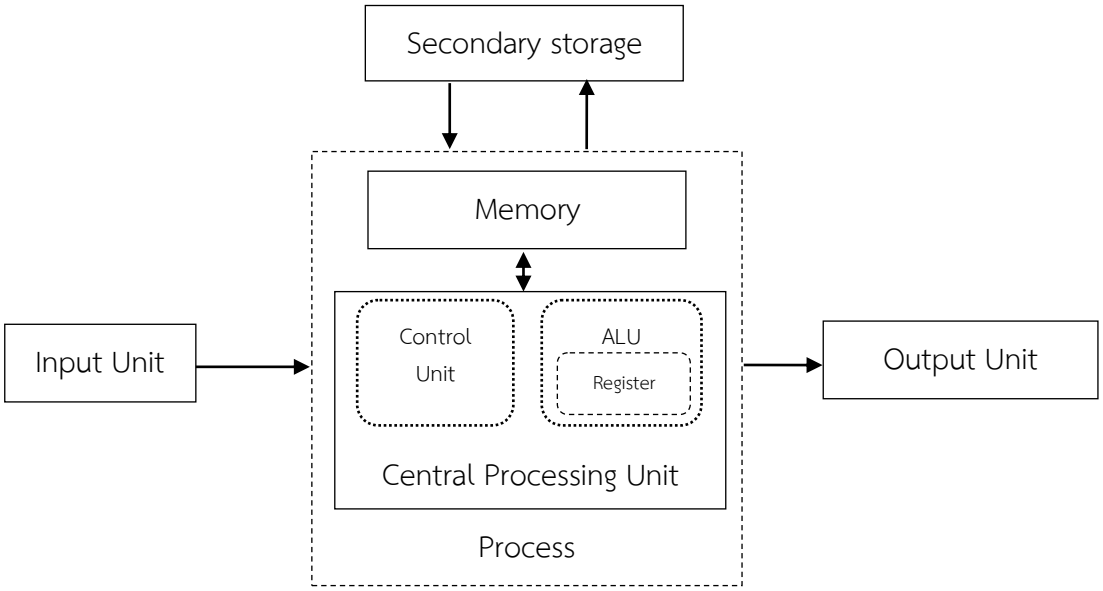
ฮาร์ดแวร์ หมายถึง อุปกรณ์อิเล็กทรอนิกส์ต่างๆ ลักษณะทางกายภาพที่สามารถจับต้องได้ ซึ่งอุปกรณ์เหล่านี้จะมีหน้าที่เกี่ยวกับการประมวลผลข้อมูล (Data processing) โดย ฮาร์ดแวร์อาจประกอบไปด้วยอุปกรณ์อิเล็กทรอนิกส์ย่อยๆ ซึ่งมีหน้าที่เฉพาะ เช่น หน่วยรับเข้า (Input Unit) ทำหน้าที่รับข้อมูลเข้าสู่ระบบ หน่วยประมวลผลกลาง (Central Processing Unit) ทำหน้าที่ประมวลผล หน่วยส่งออก (Output Unit) ทำหน้าที่แสดงผล เป็นต้น

ซอฟต์แวร์ หมายถึง ชุดคำสั่งหรือโปรแกรมที่ใช้สั่งงานให้อุปกรณ์คอมพิวเตอร์สามารถทำงานได้ โดยโปรแกรมบรรจุลำดับการทำงานของอุปกรณ์คอมพิวเตอร์อย่างละเอียด โดยโปรแกรมเหล่านี้ได้รับการบรรจุไว้ล่วงหน้า ซึ่งช่วยให้คอมพิวเตอร์สามารถที่จะรับข้อมูล เก็บข้อมูล และทำการตัดสินใจทางคณิตศาสตร์ การเข้าถึงข้อมูลในรูปแบบและลำดับที่เหมาะสม เช่น โปรแกรมประมวลผลคำ โปรแกรมแปลภาษา โปรแกรมสำเร็จรูปที่ใช้ในงานด้านต่าง ๆ เป็นต้น

บุคลากร หมายถึง บุคคลที่เกี่ยวข้องกับระบบงานทั้งในองค์กรและนอกองค์กร เช่น ผู้จัดการโครงการ (Project Manager) ผู้บริหารข้อมูล (Data Administrator) นักวิเคราะห์ระบบ (Systems Analyst) นักเขียนโปรแกรม (Programmer) วิศวกรระบบ (System Engineer) เจ้าหน้าที่ฝ่ายเทคนิค (Technical Support) และผู้ใช้ (User) เป็นต้น

ฮาร์ดแวร์

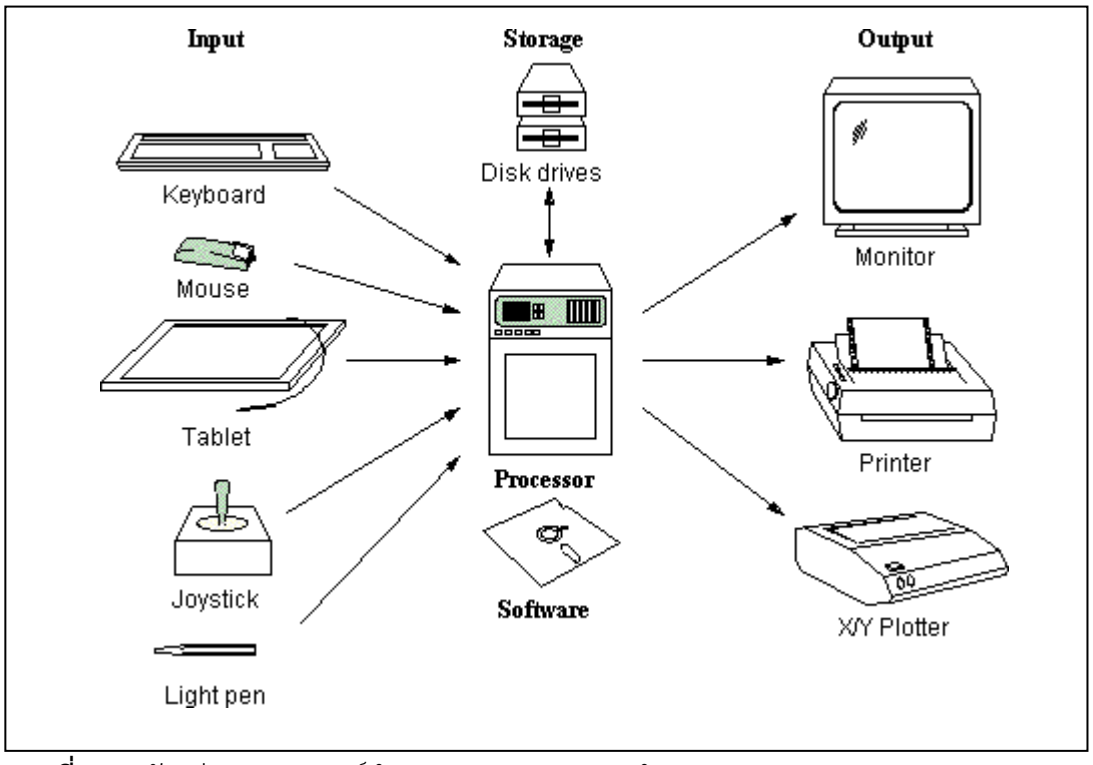
ฮาร์ดแวร์เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ที่มีองค์ประกอบสำคัญ 5 ส่วน (University of Rhode Island. 2014) คือ หน่วยประมวลผลกลาง (Central Processing Unit) หน่วยความจำ (Memory) หน่วยรับเข้า หน่วยส่งออก และหน่วยเก็บรอง (Secondary storage) โดยมีการทำงานดังภาพที่ 1.1



ภาพที่ 1.1 ประเภทของอุปกรณ์ฮาร์ดแวร์
ที่มา: ปัญญาพล หอระตะ. 2545: 2

ภาพที่ 1.1 แสดงให้เห็นการทำงานของอุปกรณ์ฮาร์ดแวร์ เริ่มจากหน่วยรับเข้า (Input Unit) เช่น แผงแป้นอักขระ (Keyboard) เมาส์ (Mouse) เครื่องอ่านพิกัด (Tablet) ก้านควบคุม รับข้อมูลมาจากหน่วยรับเข้าส่งให้หน่วยประมวลผลกลาง (Central Processing Unit) ถ้าเป็นคำสั่งการคำนวณ หน่วยคำนวณและตรรกะ (Arithmetic-Logic Unit: ALU) ทำการประมวลผลระหว่างการประมวลจะเก็บ ผลลัพธ์การคำนวณไว้ในรีจีสเตอร์ (Register) เมื่อจบการประมวลผลจะส่งผลลัพธ์ไปยังหน่วยควบคุม (Control Unit) จากนั้นหน่วยควบคุมจะส่งผลลัพธ์ไปยังหน่วยความจำ (Memory) แล้วนำไปแสดงผลที่หน่วยส่งออก (Output Unit) เช่น จอภาพ (Monitor) เครื่องพิมพ์ (Printer) ถ้าผู้ใช้ต้องการบันทึกข้อมูลสามารถบันทึกลงหน่วยความจำสำรอง (Secondary storage) เช่น จานบันทึก (Disk) ได้

อุปกรณ์ต่างๆ ในภาพที่ 1.1 นั้นสามารถที่จะแสดงตัวอย่างของอุปกรณ์ในแต่ละส่วนได้ดังภาพที่ 1.2



ภาพที่ 1.2 ตัวอย่างของอุปกรณ์จำแนกตามประเภทการทำงาน
ที่มา: CQUniversity. 2012

ภาพที่ 1.2 แสดงให้เห็นตัวอย่างของอุปกรณ์ฮาร์ดแวร์ เริ่มจากหน่วยรับเข้า (Input Unit) เช่น แผงแป้นอักขระ (Keyboard) เมาส์ (Mouse) เครื่องอ่านพิกัด (Tablet) ก้านควบคุม (Joystick) และปากกาแสง (Light pen) เป็นต้น หน่วยประมวลผลกลาง (Central Processing Unit) ทำงานประสานกับซอฟต์แวร์ (Software) บันทึกข้อมูลสามารถบันทึกลงหน่วยความจำสำรอง (Secondary storage) เช่น จานบันทึก (Disk) แผ่นบันทึก (Floppy disk) ในหน่วยขับจาน (Disk drives) แสดงผลที่หน่วยส่งออก (Output Unit) เช่น จอภาพ (Monitor) เครื่องพิมพ์ (Printer) หรือ พล็อตเตอร์ (Plotter) เป็นต้น อธิบายตามหน้าที่การทำงานได้ดังนี้

1. หน่วยประมวลผลกลาง

หน่วยประมวลผลกลาง ทำหน้าที่ประมวลผลข้อมูลที่รับเข้ามาตาม วิธีการที่ได้กำหนดเอาไว้ล่วงหน้า หน้าที่หลักของหน่วยประมวลผลกลาง คือ หน่วยควบคุม (Control Unit) และหน่วยคำนวณและตรรกะ (Arithmetic-Logic Unit: ALU) หน่วยควบคุมทำหน้าที่ควบคุมการไหลของข้อมูลจากส่วนหนึ่งไปยังอีกส่วนหนึ่ง แปลคำสั่งที่เก็บไว้ในหน่วยความจำ ประสานงานกับหน่วยคำนวณและตรรกะ ส่วนหน่วยคำนวณและตรรกะ ทำหน้าที่ประมวลผลการดำเนินการทางเลขคณิต เช่นการบวก การลบ การคูณ การหาร ประมวลผลทางตรรกะ ในหน่วยนี้มีหน่วยความจำเล็ก ๆ เรียกว่า รีจิสเตอร์ (Register) ทำหน้าที่เก็บผลลัพธ์ระหว่างการประมวลผล เมื่อได้ผลลัพธ์สุดท้ายจะส่งไปยังหน่วยควบคุม (ปัญญาพล หอระตะ. 2545: 2)

2. หน่วยความจำ

หน่วยความจำ เป็นหน่วยเก็บหลัก (Primary storage) ที่คอยเก็บข้อมูลที่รับมาโดยอุปกรณ์นำเข้ามูล ไว้ชั่วขณะหนึ่งเพื่อรอการประมวลผล ซึ่งข้อมูลที่เก็บไว้ในหน่วยความจำหลักนี้จะหายไปเมื่อปิดเครื่องคอมพิวเตอร์ ตัวอย่างของหน่วยความจำหลัก ได้แก่ หน่วยความจำเข้าถึงแบบสุ่ม (Random Access Memory: RAM)

3. หน่วยรับเข้า

หน่วยรับเข้า เป็นหน่วยที่ทำหน้าที่นำข้อมูลเข้าสู่การประมวลผล ข้อมูลเหล่านี้อาจเป็นตัวอักษร ตัวเลข หรือสัญลักษณ์ต่างๆ ซึ่งการนำข้อมูลเข้านี้จะต้องได้รับการแปลงสัญญาณให้อยู่ในรูปของสัญญาณอิเล็กทรอนิกส์เสียก่อน โดยสัญญาณอิเล็กทรอนิกส์นั้นจะต้องเหมาะสมกับการประมวลผลของแต่ละลักษณะของอุปกรณ์นำข้อมูลเข้าด้วย ตัวอย่างของอุปกรณ์นำเข้าข้อมูลเช่น แผงแป้นอักขระ (Keyboard) เมาส์ (Mouse) เครื่องอ่านพิกัด (Tablet) ก้านควบคุม (Joystick) และปากกาแสง (Light pen) เป็นต้น

4. หน่วยส่งออก

หน่วยส่งออก เป็นหน่วยที่ทำหน้าที่แสดงผลลัพธ์ที่ได้จากการประมวลผล ไปสู่ผู้ใช้งานระบบคอมพิวเตอร์ ตัวอย่างของหน่วยส่งออก ได้แก่ ตัวจอภาพ (Monitor) เครื่องพิมพ์ (Printer) และพล็อตเตอร์ (Plotter)

5. หน่วยเก็บรอง

หน่วยเก็บรอง เป็นหน่วยที่ทำหน้าที่เก็บข้อมูล และผลลัพธ์จากการประมวลผลไว้ใช้เป็นข้อมูลในการประมวลผลถัดไป หรือแก้ไขข้อมูลให้ถูกต้อง ตัวอย่างของหน่วยเก็บรอง ได้แก่ จานบันทึก (Disk) แผ่นบันทึก (Floppy disk) เป็นต้น หน่วยเก็บรองนี้จะเก็บข้อมูลได้ ไม่สูญหายแม้ว่าจะปิดเครื่องคอมพิวเตอร์

ซอฟต์แวร์

ซอฟต์แวร์ (Software) เป็นชุดคำสั่งหรือโปรแกรมที่ใช้สั่งงานให้อุปกรณ์คอมพิวเตอร์สามารถทำงานได้ สามารถแบ่งได้ 2 ประเภทคือ

1. ซอฟต์แวร์ระบบ

ซอฟต์แวร์ระบบ (System Software) เป็นโปรแกรมที่ออกแบบมาเพื่ออำนวยความสะดวกแก่ผู้ใช้คอมพิวเตอร์ในการติดต่อกับระบบคอมพิวเตอร์ โดยหน้าที่หลักของซอฟต์แวร์ระบบคือ การจัดการการทำงานร่วมกันระหว่างฮาร์ดแวร์ และผู้ใช้คอมพิวเตอร์ ไม่ว่าจะเป็นการจัดการระบบแฟ้ม การแปลภาษาคอมพิวเตอร์ การเรียงลำดับก่อนหลังโดยซอฟต์แวร์ระบบ แบ่งเป็นประเภทย่อยๆ ได้ดังนี้

1.1 ระบบปฏิบัติการ (Operating System) เป็นซอฟต์แวร์ระบบที่สำคัญที่สุดของระบบคอมพิวเตอร์ซึ่งจำเป็นต้องมีทุกเครื่อง ทำหน้าที่ในการควบคุมการทำงานของหน่วยรับเข้าและหน่วยส่งออกของคอมพิวเตอร์ และทำการเรียงลำดับก่อนหลังในการใช้ทรัพยากรของระบบคอมพิวเตอร์ ตัวอย่างระบบปฏิบัติการ เช่น ไมโครซอฟต์วินโดวส์ (Microsoft Windows) ลินุกซ์ (Linux) และ ยูนิกซ์ (Unix) เป็นต้น

1.2 ตัวแปลโปรแกรม (Compiler) เป็นซอฟต์แวร์ที่ทำหน้าที่ในการแปลลำดับการสั่งงานของมนุษย์ให้อยู่ในรูปแบบที่คอมพิวเตอร์เข้าใจ และสามารถนำไปปฏิบัติตามคำสั่งได้อย่างถูกต้อง ตัวอย่างของโปรแกรมแปลโปรแกรม เช่น เทอร์โบซี (Turbo C) บอร์แลนด์ปาสคาล (Borland Pascal) และวิชวลเบสิก (Visual Basic) เป็นต้น

2. ซอฟต์แวร์สำเร็จ

ซอฟต์แวร์สำเร็จ (Software Package) เป็นซอฟต์แวร์ที่พัฒนาขึ้นตามความต้องการเฉพาะของผู้ใช้งาน ซึ่งอาจมีขายโดยทั่วไปหรือพัฒนาขึ้นใช้งานเฉพาะอย่าง เช่น โปรแกรมบัญชีสำเร็จรูปเอ็กซ์เพรสใช้ในงานด้านบัญชี โปรแกรมสำหรับอุตสาหกรรมก่อสร้าง BOQ FORMULA BOQ และ FORMA BOQ ใช้ในงานธุรกิจก่อสร้าง โปรแกรม Crystal Online ใช้บริหารการเชื่อมต่อหลายสาขา บริหารแฟรนไชส์ โปรแกรมบริหารความสัมพันธ์ลูกค้า (CRM) เป็นโปรแกรมที่รองรับแผนการตลาดพื่อให้ดำเนินการตามกลยุทธ์ทางธุรกิจ การรักษาความสัมพันธ์ที่ดีกับลูกค้า โปรแกรมที่ใช้ในสำนักงาน เช่น ไมโครซอฟต์ออฟฟิต (Microsoft Office) เช่น ไมโครซอฟต์เวิร์ด (Microsoft Word) ไมโครซอฟต์เอ็กเซล (Microsoft Excel) และไมโครซอฟต์เพาเวอร์พอยต์ (Microsoft Powerpoint) เป็นต้น

ส่วนบุคลากร

ส่วนบุคลากร เป็นบุคลากรที่ทำงานเกี่ยวข้องกับระบบงานคอมพิวเตอร์ในด้านต่าง ๆ มีดังต่อไปนี้

1. ผู้จัดการโครงการ (Project Manager) เป็นบุคลากรที่ควบคุมดูแล วางแผนงาน มอบหมายงานให้กับบุคลากรในโครงการให้ดำเนินการตามแผนโครงการ ทั้งยังเป็นผู้ดูแลโครงงานให้เสร็จตามเวลา เป็นผู้ประสานงานระหว่างผู้บริหารและบุคลากรที่เกี่ยวข้อง

2. ผู้บริหารฐานข้อมูล (Database Administrator) เป็นบุคลากรซึ่งมีหน้าที่ควบคุมและบริหารทรัพยากรข้อมูล (วราภรณ์ โกวิทวรางกูร. 2544: 25) เป็นผู้บริหารฐานข้อมูล วางแผนการใช้ข้อมูล กำหนดขอบเขตสิทธิการเข้าถึงข้อมูล

3. นักวิเคราะห์ระบบ (Systems Analyst) เป็นบุคลากรที่ทำหน้าที่วิเคราะห์ระบบงานและออกแบบฐานข้อมูล รายละเอียดของข้อมูล ขอบเขตงานของงานมอบหมายให้โปรแกรมเมอร์ทำในแต่ละส่วนงาน

4. นักเขียนโปรแกรม (Programmer) เป็นบุคลากรที่ทำหน้าที่เขียนโปรแกรมตามที่นักวิเคราะห์ระบบออกแบบให้ แบ่งเป็นนักเขียนโปรแกรมระบบ (System Programmer) และนักเขียนโปรแกรมประยุกต์ (Application Programmer) นักเขียนโปรแกรมระบบเป็นบุคลากรที่ทำหน้าที่เขียนและพัฒนาโปรแกรมควบคุมระบบคอมพิวเตอร์ และนักเขียนโปรแกรมประยุกต์เป็นผู้เขียนและพัฒนาโปรแกรมประยุกต์ในงานด้านต่าง ๆ และบางระบบงานก็มีชื่อเฉพาะ เช่น เว็บมาสเตอร์ (Web master) เป็นผู้ดูแลและพัฒนาเว็บไซต์ เว็บดีไซน์เนอร์ (Web designer) เป็นผู้ออกแบบหน้าเว็บเพจ

5. วิศวกรระบบ (System Engineer) เป็นบุคลากรที่ดูแลด้านการสื่อสารและเครือข่ายคอมพิวเตอร์ (โอภาส เอี่ยมสิริวงศ์. 2546: 25) กำหนดรายละเอียดของเครื่องคอมพิวเตอร์

รายละเอียดด้านการสื่อสารและระบบเครือข่ายที่รองรับการทำงาน บางหน่วยงานเรียกว่า ผู้บริหารเครือข่าย (Network Administrator)

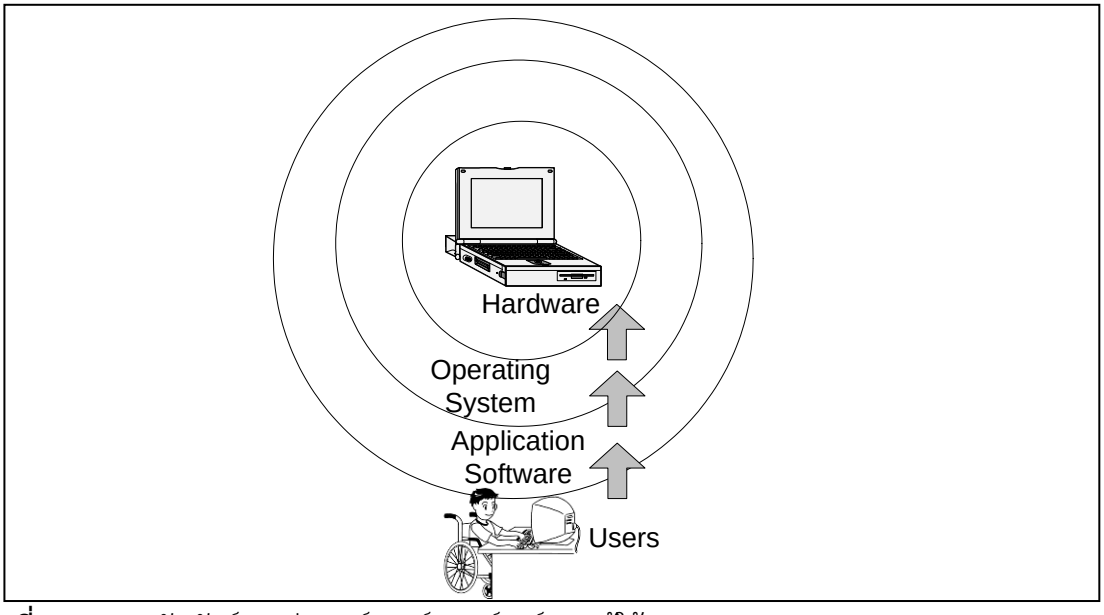
6. เจ้าหน้าที่ฝ่ายเทคนิค (Technical Support) เป็นผู้ทำงานทางด้านเทคนิคต่าง ๆ เจ้าหน้าที่ฝ่ายเทคนิคในบางบริษัททำหน้าที่ช่วยเหลือผู้ใช้ ให้คำแนะนำต่าง ๆ เกี่ยวกับการใช้งานระบบจะเรียกว่า Help desk จะคอยช่วยแก้ปัญหาเกี่ยวกับอุปกรณ์ต่าง ๆ แนะนำการใช้งานโปรแกรม รวมถึงปัญหาการเชื่อมต่อระบบเครือข่าย เป็นต้น

7. ผู้ใช้ (User) เป็นบุคลากรที่ใช้งานระบบ เช่น ระบบลงทะเบียน นักศึกษา อาจารย์เป็นผู้ใช้ในงานระบบลงทะเบียน นักศึกษาลงทะเบียนวิชาต่าง ๆ อาจารย์ต้องส่งผลการเรียนผ่านระบบงานลงทะเบียน เป็นต้น

ความสัมพันธ์ของฮาร์ดแวร์ซอฟต์แวร์ และผู้ใช้

ความสัมพันธ์ของฮาร์ดแวร์ซอฟต์แวร์ และผู้ใช้ (สานนท์ เจริญฉาย. 2552) ดังแสดงในภาพที่ 1.3 นั้น ผู้ใช้จะใช้งานคอมพิวเตอร์ใช้งานผ่านซอฟต์แวร์ ทั้งซอฟต์แวร์สำเร็จ และระบบปฏิบัติการ ซึ่งในบางขณะผู้ใช้จะทำงานผ่านซอฟต์แวร์สำเร็จซึ่งอยู่เบื้องหน้า เช่นการจัดทำเอกสารด้วยโปรแกรมประมวลผลคำ จะมีโปรแกรมไมโครซอฟต์เวิร์ด ทำงานอยู่เบื้องหน้า ในขณะเดียวกันนั้น ระบบปฏิบัติการก็จะอยู่เบื้องหลัง (Background) คือคอยควบคุมการแสดงผลและการรับข้อมูล

บางกรณีผู้ใช้อาจต้องการเข้าถึงการจัดการแฟ้มโดยตรง เช่น สำเนาข้อมูล ลบข้อมูล เป็นต้น ดังนั้นผู้ใช้จะต้องติดต่อผ่านทางระบบปฏิบัติการโดยตรงในบางขณะโดยไม่ผ่านซอฟต์แวร์สำเร็จ



ภาพที่ 1.3 ความสัมพันธ์ระหว่างฮาร์ดแวร์ซอฟต์แวร์ และผู้ใช้
ที่มา: สานนท์ เจริญฉาย. 2552: 13

ภาษาการโปรแกรมคอมพิวเตอร์

ภาษาการโปรแกรมคอมพิวเตอร์ (Computer programming language) มีหลากหลายภาษาขึ้นอยู่กับการใช้งานในแต่ละรูปแบบ สำหรับผู้ที่ต้องการศึกษาการเขียนโปรแกรมภาษาคอมพิวเตอร์นั้น จะต้องใช้โปรแกรมพิเศษอีกประเภทหนึ่งคือโปรแกรมประเภทภาษาคอมพิวเตอร์ ซึ่งเป็นซอฟต์แวร์เฉพาะที่ได้รับการพัฒนาให้ทำหน้าที่สื่อความหมายกับฮาร์ดแวร์และซอฟต์แวร์ ที่มีอยู่ในเครื่องคอมพิวเตอร์ โดยเฉพาะระบบปฏิบัติการ เนื่องจากฮาร์ดแวร์นั้นมีการทำงานด้วยวงจรอิเล็กทรอนิกส์ ดังนั้นจึงรับรู้การทำงานโดยใช้สัญญาณไฟฟ้าเท่านั้น ซึ่งมีเพียง 2 สถานะคือ “เปิด” และ “ปิด” เท่านั้น ซึ่งจะแทนด้วย 1 และ 0 ตามลำดับ ซึ่งภาษาในรูปแบบนี้เรียกว่า “ภาษาเครื่อง (Machine language)” แต่อย่างไรก็ตามภาษาเครื่องเป็นภาษาที่ยากต่อการพัฒนา ดังนั้นจึงต้องใช้ภาษาที่เป็นภาษากลางระหว่างคอมพิวเตอร์กับมนุษย์ เรียกว่า “ภาษาคอมพิวเตอร์” ซึ่งแบ่งได้เป็น 4 ระดับคือ (สานนท์ เจริญฉาย. 2552: 13)

1. ภาษาระดับต่ำ

ภาษาระดับต่ำ (Low level language) เป็นภาษาเครื่อง ใช้สัญลักษณ์ 1 และ 0 ในการแทนคำสั่ง เป็นภาษาที่เครื่องคอมพิวเตอร์ สามารถทำงานได้ทันที โดยไม่ต้องผ่านตัวแปลภาษาทำงานได้อย่างรวดเร็ว แต่เป็นภาษาที่เขียนได้ยุ่งยากมากจึงไม่เป็นที่นิยม

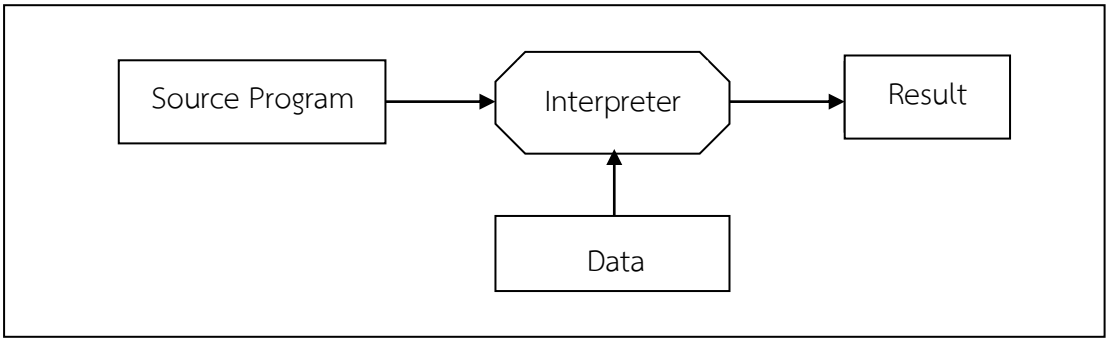
2. ภาษาสัญลักษณ์

ภาษาสัญลักษณ์ (Symbolic language) เป็นการกำหนดตัวอักษรมาใช้แทนสัญลักษณ์ 1 และ 0 เพื่อให้ง่ายต่อมนุษย์ในการจดจำ และสื่อความหมายมากยิ่งขึ้น แต่ทำให้คอมพิวเตอร์ไม่เข้าใจ จึงต้องมีการพัฒนาตัวแปลภาษา เช่น แอสแซมเบลอ (Assembler) เป็นต้น

3. ภาษาระดับสูง

ภาษาระดับสูง (High level language) พัฒนามาจากภาษาสัญลักษณ์ให้ยืดหยุ่น และใกล้เคียงกับภาษาอังกฤษมากขึ้น และสร้างไวยกรณ์ของภาษาขึ้นมา แต่ภาษาระดับสูงจะทำให้การทำการแปลเป็นภาษาเครื่องนั้นซับซ้อนมากยิ่งขึ้น ตัวแปลภาษามี 2 ประเภทคือ อินเทอพรีเตอร์ (Interpreter) เป็นการแปลภาษาโดยการแปลครั้งละ 1 คำสั่ง และคอมไพเลอร์ (Compiler) เป็นการแปลภาษาทั้งโปรแกรม

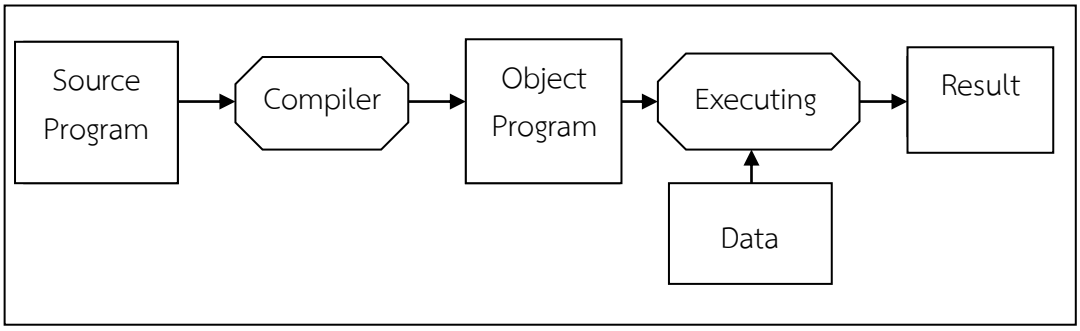
อินเทอพรีเตอร์ เป็นการแปลภาษาโดยการแปลครั้งละ 1 คำสั่ง ซึ่งทำให้การตรวจสอบโปรแกรมง่ายขึ้น สามารถแก้ไขโปรแกรมขณะทำงานอยู่ได้ทันทีที่พบข้อผิดพลาด แต่ต้องแปลทุกครั้งที่เรียกใช้งาน ไม่สามารถเก็บไว้ใช้งานภายหลังได้ ตัวอย่างภาษาที่ใช้ตัวแปลภาษาลักษณะนี้เช่น ภาษาเบสิก (BASIC) ภาษาเพิร์ล (perl) ภาษาพีเอชพี (PHP) ภาษาเอเอชพี (ASP) และภาษาจาวาสคริปต์ (JavaScript) เป็นต้น



ภาพที่ 1.4 ขั้นตอนการแปลภาษาแบบอินเทอพรีเตอร์

จากภาพที่ 1.4 แสดงขั้นตอนการแปลภาษาแบบอินเทอพรีเตอร์ เมื่อมีการประมวลผล อินเทอพรีเตอร์จะอ่านโปรแกรมต้นฉบับ (Source program) รับข้อมูล (Data) ที่เกี่ยวข้องนำมาประมวลผล แล้วแสดงผลลัพธ์ (Result) ที่ได้จากการประมวลผล

คอมไพเลอร์ จะทำการแปลภาษาทั้งโปรแกรมและบันทึกผลที่ได้ไว้ใช้งานในภายหลังได้ในรูปแบบของรหัสจุดหมาย (Object Code) แต่มีข้อเสียคือ การแปลภาษาแบบคอมไพเลอร์ จะใช้เวลานานกว่าการแปลภาษาแบบอินเทอพรีเตอร์ ตัวอย่างของภาษาที่ใช้ตัวแปลภาษาที่เป็นคอมไพเลอร์ คือ ภาษาซี ภาษาปาสคาล เป็นต้น



ภาพที่ 1.5 ขั้นตอนการแปลภาษาแบบคอมไพเลอร์
ที่มา: อรพิน ประวัติบริสุทธิ์. 2554: 14

จากภาพที่ 1.5 แสดงขั้นตอนการแปลภาษาแบบคอมไพเลอร์ เริ่มจากคอมไพเลอร์จะอ่านโปรแกรมต้นฉบับ (Source program) แล้วแปลเป็นโปรแกรมจุดหมาย (Object Program) เมื่อมีการกระทำการ (Executing) จะรับข้อมูล (Data) ที่เกี่ยวข้องนำมาประมวลผล แล้วแสดงผลลัพธ์ (Result) ที่ได้จากการประมวลผล

4. ภาษาระดับสูงมาก

ภาษาระดับสูงมาก (Very high level language) เป็นภาษายุคที่ 4 ภาษาระดับนี้ไม่ได้ระบุรายละเอียดขั้นตอนการทำงานไว้ เพียงแต่บอกว่าต้องการข้อมูลอะไรเท่านั้น เช่น ภาษาสอบถามเชิงโครงสร้าง (Structured Query Language: SQL) ซึ่งใช้ในระบบการจัดการฐานข้อมูลในการสอบถามข้อมูลเป็นต้น

สารบบแฟ้ม

สารบบแฟ้มในระบบปฏิบัติการไมโครซอฟต์นั้น มีการจัดการเก็บข้อมูลเป็น 3 ระดับคือ

1. หน่วยขับจาน

หน่วยขับจาน (Disk drives) ถูกระบุโดยใช้ อักษรภาษาอังกฤษ A-Z เรียกว่า “Drive Letter” โดยอักษร A, B หมายถึง หน่วยขับจานบันทึกตัวที่ 1 และ 2 ตามลำดับ ในระบบปฏิบัติการไมโครซอฟต์วินโดวส์นั้น หน่วยขับจาน A หมายถึง หน่วยขับจานสำหรับแผ่นบันทึก (Floppy disk) ส่วนอักษรประจำหน่วยขับจานสำหรับจานบันทึกแบบแข็ง (Hard disk) นั้นจะเริ่มจาก C เป็นต้นไป ตามจำนวนของอุปกรณ์ที่มีอยู่จริง เช่น ถ้ามีจานบันทึกแบบแข็ง 2 จานบันทึก หรือมีการแบ่งส่วนในจานบันทึกแบบแข็ง อักษรประจำจะเป็น C และ D ตามลำดับ คอมพิวเตอร์เครื่องใดมีซีดีรอม (CD-ROM) จะใช้อักษรถัดไปจากจานบันทึกแบบแข็ง สำหรับบางระบบปฏิบัติการ จะกำหนดอักษรประจำอุปกรณ์แตกต่างกันออกไป ขึ้นอยู่กับระบบปฏิบัติการนั้น ๆ

2. สารบบ

เมื่อมีอุปกรณ์เก็บข้อมูลแล้ว ระบบปฏิบัติการจัดการพื้นที่การเก็บข้อมูลออกเป็นส่วนย่อยๆ เรียกว่า สารบบ (Directory) ซึ่งแต่ละสารบบจะมีชื่อเรียก และอาจมีสารบบย่อย ซึ่งสารบบเหล่านี้ใช้เป็นที่รวมข้อมูลให้อยู่เป็นหมวดหมู่

3. แฟ้ม

แฟ้ม (File) เป็นที่เก็บข้อมูลไว้ในรูปแบบใดรูปแบบหนึ่งขึ้นอยู่กับซอฟต์แวร์ที่ได้รับการพัฒนามาโดยทั้ง 3 ระดับคือ ระดับหน่วยขับจาน ระดับสารบบ และระดับแฟ้ม ในการจัดเก็บข้อมูลนั้นจะแบ่งแยกด้วยเครื่องหมายต่าง ๆ ดังนี้

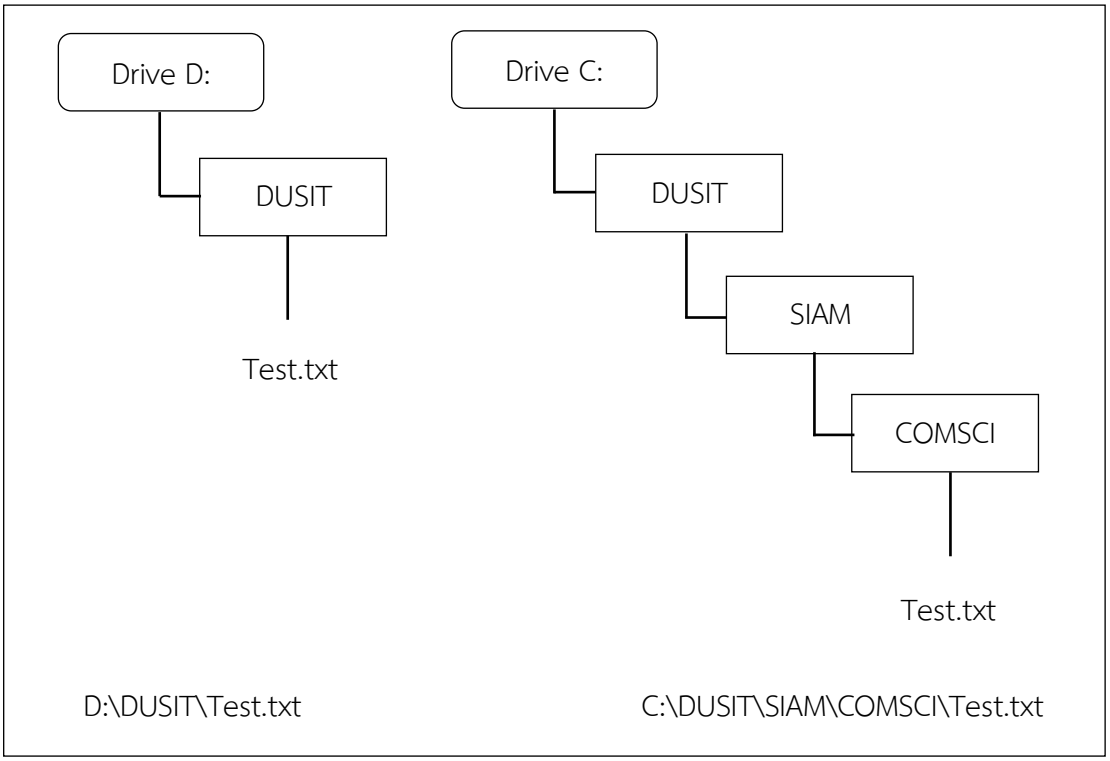
ระดับหน่วยขับจาน จบด้วยเครื่องหมายทวิภาคหรือจุดคู่ (:) หลังอักษรประจำอุปกรณ์

ระดับสารบบ จบด้วยเครื่องหมายแบ็กสแลช (\) หลังสิ้นสุดชื่อสารบบแต่ละสารบบ

ระดับแฟ้ม จะอยู่หลังจากระดับสารบบและอยู่หลังสุดเสมอ เช่น

D:\DUSIT\Test.txt หมายถึง หน่วยขับจาน D สารบบ ชื่อ DUSIT และแฟ้มชื่อ Test.txt

C:\DUSIT\SIAM\COMSCI\Test.txt หมายถึง หน่วยขับจาน (C) สารบบชื่อ DUSIT มีสารบบย่อยชื่อ SIAM และมีสารบบย่อยอีกชื่อ COMSCI โดยมีแฟ้มชื่อ Test.txt



ภาพที่ 1.6 โครงสร้างการจัดเก็บข้อมูล

หลักการตั้งชื่อแฟ้ม

ระบบปฏิบัติการไมโครซอฟต์ มีหลักการตั้งชื่อแฟ้มดังนี้ (Microsoft Corporation. 2014)

1. ในระบบปฏิบัติการ MS-DOS กำหนดให้ความยาวของชื่อไฟล์ได้เพียง 8 ตัวอักขระ ส่วนขยายอีก 3 อักขระ รวมเป็นไม่เกิน 12 อักขระรวมจุดด้วยเรียกว่ารุ่น 8.3 แต่ในปัจจุบันในระบบปฏิบัติการวินโดว์ สามารถตั้งชื่อได้โดยไม่จำกัดจำนวนอักขระ (Microsoft Corporation. 2014) เพื่อสะดวกในการค้นหาความยาวของชื่อแฟ้มไม่ควรยาวมากเกินไป

2. สามารถตั้งชื่อด้วยตัวอักขระทั้งหมดทั้งภาษาไทย ภาษาอังกฤษพิมพ์เล็ก a-z พิมพ์ใหญ่ A-Z ตัวเลข 0-9 ยกเว้นเครื่องหมายตัวหนอน (~) เครื่องหมายสี่เหลี่ยม (#) เครื่องหมายเปอร์เซ็นต์ (%) เครื่องหมายแอมเพอร์แซนด์ (&) เครื่องหมายดอกจัน (*) เครื่องหมายวงเล็บปีกกา ({ }) เครื่องหมายแบ็กสแลช (\) เครื่องหมายทวิภาคหรือจุดคู่ (:) เครื่องหมายวงเล็บสามเหลี่ยม (< >) เครื่องหมายปรัศนี (?) เครื่องหมายสแลช (/) เครื่องหมายบวก (+) เครื่องหมายขีดตั้ง (|) และเครื่องหมายอัญประกาศคู่ (") เพื่อสะดวกในการค้นหา และการนำไปใช้ควรตั้งชื่อเป็นภาษาอังกฤษและสื่อความหมายกับข้อมูลที่เก็บ

3. ชื่อไฟล์ไม่ตรงกับคำสงวนของระบบปฏิบัติการดังต่อไปนี้ CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9 เช่น NUL.CPP

สรุป

ระบบคอมพิวเตอร์ ประกอบด้วย ฮาร์ดแวร์ ซอฟต์แวร์ และบุคลากร โดยฮาร์ดแวร์เป็นอุปกรณ์อิเล็กทรอนิกส์ที่มีองค์ประกอบสำคัญ 5 ส่วนคือ หน่วยรับเข้า หน่วยส่งออก หน่วยประมวลผลกลาง หน่วยความจำ และหน่วยเก็บรอง

ซอฟต์แวร์ คือชุดคำสั่งหรือโปรแกรมโดยแบ่งเป็น 2 ประเภทคือ ซอฟต์แวร์ระบบ กับซอฟต์แวร์สำเร็จ โดยซอฟต์แวร์ระบบที่สำคัญ คือระบบปฏิบัติการ และตัวแปลภาษาที่จะได้เรียนรู้ต่อไป ซอฟต์แวร์สำเร็จ เป็นซอฟต์แวร์ที่พัฒนาตามความต้องการเพื่อใช้งานในด้านต่าง ๆ

บุคลากรที่ทำงานเกี่ยวกับระบบงานคอมพิวเตอร์ บุคลากรที่ทุกคนเกี่ยวข้องคือ ผู้ใช้ ทุกคนที่เกี่ยวข้องกับระบบงานเรียกว่าผู้ใช้ เช่น ลูกค้าธนาคารเบิกถอนเงินจากเครื่องผู้ให้บริการทางการเงินอัตโนมัติ หรือตู้เอทีเอ็มที่รู้จักกันและใช้บริการเป็นประจำ ผู้ที่ให้คำแนะนำช่วยเหลือผู้ใช้ระบบคอยแก้ไขปัญหาการใช้งานระบบ เรียกว่าเฮลป์เดสก์ (Help desk) หรือบางหน่วยงานเรียกว่าเจ้าหน้าที่ฝ่ายเทคนิค (Technical Support) ในหน่วยงานขนาดใหญ่อาจแบ่งเจ้าที่ผ่ายเทคนิคดูแลซอฟต์แวร์และเจ้าหน้าที่ฝ่ายเทคนิคดูแลฮาร์ดแวร์ เพื่อให้บริการคำแนะนำผู้ใช้ได้อย่างมีประสิทธิภาพ ส่วนผู้ที่ดูแลด้านการสื่อสารและเครือข่ายคอมพิวเตอร์เรียกว่า วิศวกรระบบ บุคลากรที่เขียนโปรแกรม เรียกว่า โปรแกรมเมอร์ ซึ่งจะเขียนโปรแกรมตามที่ นักวิเคราะห์ระบบ ออกแบบให้ รวมทั้งผู้บริหารฐานข้อมูล ที่ควบคุมดูแลบริหารทรัพยากรข้อมูล และบุคคลากรที่เป็นผู้บริหารงานระบบนั้น ๆ เรียกว่า ผู้จัดการโครงการ เป็นบุคลากรที่ควบคุมดูแลวางแผนงานทั้งหมด

ความสัมพันธ์ของระบบคอมพิวเตอร์ ซึ่งประกอบไปด้วยอุปกรณ์ต่าง ๆ ที่มีหน้าที่เพื่อรับข้อมูล แล้วนำมาประมวลผล เมื่อประมวลผลเสร็จก็จะแสดงผลลัพธ์ทางหน่วยส่งออก ซึ่งอุปกรณ์แต่ละตัวไม่สามารถทำงานด้วยกันได้ หากไม่มีระบบปฏิบัติการที่ทำหน้าที่ประสานงานอุปกรณ์ต่าง ๆ ให้สามารถสื่อสาร ส่งข้อมูล ทำงานต่าง ๆ ด้วยกันได้ ที่สำคัญที่สุดก็คือ มนุษย์ผู้มีความต้องการต่าง ๆ ที่จะสั่งให้คอมพิวเตอร์ทำงานให้บรรลุผลตามความต้องการต่างกันไป โดยผ่านซอฟต์แวร์สำเร็จต่าง ๆ

ภาษาคอมพิวเตอร์ได้เกิดขึ้นมากมายตามลักษณะการใช้งานและความต้องการของมนุษย์ ซึ่งมีตั้งแต่ภาษาระดับต่ำซึ่งเขียนยากแต่สามารถทำงานได้เร็วเพราะติดต่อกับฮาร์ดแวร์ได้โดยตรง เมื่อเป็นภาษาที่เขียนยากจึงมีการพัฒนาภาษาให้เขียนง่ายขึ้นซึ่งใกล้เคียงกับภาษามนุษย์มากขึ้นแต่เพื่อให้เป็นภาษาสากล ภาษาที่ใช้จึงเป็นภาษาอังกฤษ ซึ่งต่อไปก็จะพัฒนาให้ใกล้เคียงกับภาษาพูดของมนุษย์มากขึ้น ซึ่งต่อไปในอนาคตก็ไม่จำเป็นต้องเขียนโปรแกรมแล้วเพียงแต่พูดสั่งเครื่องคอมพิวเตอร์เครื่องคอมพิวเตอร์จะทำงานได้ตามที่ต้องการ

สารบบแฟ้มในระบบปฏิบัติการไมโครซอฟต์มีการจัดเก็บข้อมูล 3 ระดับคือ หน่วยขับจาน สารบบ และแฟ้ม ระดับหน่วยขับจาน จบด้วยเครื่องหมายทวิภาคหรือจุดคู่ (:) หลังอักษรประจำอุปกรณ์ ระดับสารบบ จบด้วยเครื่องหมายแบ็กสแลช (\) หลังสิ้นสุดชื่อสารบบแต่ละสารบบ และระดับแฟ้ม จะอยู่หลังจากระดับสารบบและอยู่หลังสุดเสมอ เช่น D:\DUSIT\Test.txt เป็นต้น

แบบฝึกหัด

1. จงบอกอุปกรณ์ที่ทำหน้าที่เป็นหน่วยนำเข้า มีอะไรบ้าง
2. จงอธิบายว่าหน่วยประมวลผลกลาง มีหน้าที่ทำอะไร
3. จงบอกอุปกรณ์ที่ทำหน้าที่เป็นหน่วยส่งออก มีอะไรบ้าง
4. จงอธิบายซอฟต์แวร์แต่ละประเภท มีอะไรบ้าง
5. ภาษาคอมพิวเตอร์ มีกี่ระดับอะไรบ้าง
6. โครงสร้างของการจัดเก็บข้อมูล มีกี่ระดับ
7. จงอธิบายหลักการตั้งชื่อแฟ้ม มีหลักการอย่างไร
8. 123.C สามารถตั้งชื่อแฟ้ม ได้หรือไม่
9. การเก็บแฟ้มในสารบบมีประโยชน์อย่างไร
10. จงอธิบายถึงที่เก็บแฟ้ม C:\dusit\a.c ว่าเป็นแฟ้มอยู่ที่ใด

เอกสารอ้างอิง

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

วราภรณ์ โกวิทวรางกูร. (2544). *ระบบฐานข้อมูลและการออกแบบ.* กรุงเทพมหานคร: พิทักษ์อักษร.

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* ขอนแก่น: คลังนานาวิทยา.

โอภาส เอี่ยมสิริวงศ์. (2546). *การวิเคราะห์และออกแบบระบบ.* กรุงเทพมหานคร: ซีเอ็ดยูเคชั่น.

อรพิน ประวัติบริสุทธิ์. (2554). *คู่มือเรียนภาษาซี ฉบับปรับปรุงใหม่.* กรุงเทพมหานคร: โปรวิชั่น.

CQUniversity. (2012). *Study Material*. Retrieved July 15, 2005, from http://webclass.cqu.au/Units/81120_FOCT_Hardware/Study_Material/Study_Guidechap1/chapter1.html.

Microsoft Corporation. (2014). *Naming Files, Paths and Namespaces*. Retrieved Jan 14, 2014, from https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx.

University of Rhode Island. (2014). *The CPU and Memory*. Retrieved April 5, 2014, from http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm.

บทที่ 2
อัลกอริทึมและการวิเคราะห์ปัญหา

ในชีวิตประจำวัน จะพบกับปัญหามากมาย ว่าจะเป็นปัญหาของตนเอง ปัญหาของเพื่อน ปัญหาของผู้อื่น บางปัญหาแก้ได้ บางปัญหาก็แก้ไม่ได้ ปัญหาต่าง ๆ หากพิจารณาแล้ว จะพบว่ามีต้นสายปลายเหตุอย่างไร การแก้ปัญหาจึงควรแก้ที่ต้นเหตุเพื่อตัดปัญหาไม่ให้เกิดขึ้นอีก ในปัญหาในการเขียนโปรแกรมก็เช่นกันมีขั้นตอนในการแก้ปัญหา ในบทนี้อธิบายปัญหาและวิธีการแก้ปัญหา หลักการวิเคราะห์ปัญหา ผังงาน และหลักการเขียนผังงาน

ปัญหาและวิธีการแก้ปัญหา

ปัญหาที่พบในชีวิตประจำวันนั้นมีอยู่ 2 ประเภทด้วยกันคือ (สานนท์ เจริญฉาย. 2552: 41) ปัญหาที่มีขั้นตอนที่แน่นอนในการแก้ปัญหา เช่น ปัญหาทางวิทยาศาสตร์ ซึ่งมีสูตร และกฎเกณฑ์เงื่อนไขที่แน่นอนที่จะแก้ปัญหา ปัญหาอีกประเภทหนึ่งคือปัญหาที่ไม่มีขั้นตอนแน่นอนมากนักในการแก้ปัญหา ซึ่งต้องแก้ปัญหาโดยการใช้ดุลยพินิจ เช่นน้ำ 1 ลิตรมากเกินไปหรือไม่นั้นไม่สามารถที่จะตอบได้ชัดเจนว่ามากเกินไปหรือไม่ ขึ้นอยู่กับการใช้ประโยชน์ของน้ำจำนวนนั้น โดยหากต้องการนำเอาน้ำแก้วนั้นไปใช้ชงกาแฟอาจจะมากเกินไปสำหรับ 1 คนรับประทาน แต่ถ้าต้องการนำน้ำแก้วนั้นไปใช้ในการซักผ้าก็ตอบได้ว่ามันน้อยเกินไปด้วยซ้ำ

ด้วยเหตุนี้ในการเขียนโปรแกรมคอมพิวเตอร์จึงมักใช้กับปัญหาที่มีกฎ หรือสูตร หรือแนวคิดที่แน่นอนในการแก้ปัญหา ซึ่งเรียกปัญหาประเภทนี้ว่า “ปัญหาที่เป็นขั้นตอนวิธี (Algorithmic Problem)” และเรียกวิธีการในการแก้ปัญหาว่า “ขั้นตอนวิธี (Algorithm)” ซึ่งในวิชานี้จะกล่าวถึงหลักการเขียนโปรแกรมเพื่อแก้ปัญหาประเภทนี้เท่านั้น

ส่วนปัญหาที่ไม่มีแนวคิดแน่นอนนั้นจะต้องใช้วิธีการสร้างกฎ เพื่อให้คอมพิวเตอร์สามารถตัดสินใจได้ตามเงื่อนไขแต่ละประการของดุลยพินิจนั้นๆ แนวการแก้ปัญหาปัญหานี้จะใช้วิธีการที่เรียกว่า “ปัญญาประดิษฐ์ (Artificial Intelligence)” ซึ่งมีด้วยกันหลายวิธีด้วยกัน แต่จะไม่ได้กล่าวถึงรายละเอียดในวิชานี้

หลักการวิเคราะห์ปัญหา

การที่จะวิเคราะห์ปัญหาหาแนวทางในการแก้ปัญหานั้น มีแนวทางในการวิเคราะห์ปัญหา 3 ประการ ได้แก่ (สานนท์ เจริญฉาย. 2552: 42)

1. ขั้นวิเคราะห์ผลลัพธ์

ขั้นตอนนี้จะพิจารณาว่าโจทย์ต้องการคำตอบอะไร มีกี่คำตอบ คำตอบอยู่ในรูปแบบข้อมูลใด เป็นตัวเลข ตัวอักษร หรือรูปแบบอื่นๆ และต้องการนำไปทำอะไรต่อไป ผลลัพธ์ที่ได้ต้องไปคำนวณต่อหรือไม่ เป็นต้น

2. ขั้นการวิเคราะห์ที่มาข้อมูล

ขั้นตอนนี้จะพิจารณาว่ามีข้อมูลอะไรบ้างที่เกี่ยวข้องในการจะคำนวณหาผลลัพธ์ โดยที่มาของข้อมูลแบ่งได้เป็น 2 แหล่งคือ

1) ผู้ใช้ป้อนเข้ามูลเข้าทางแผงแป้นอักขระ โจทย์จะบอกโดยตรงว่าให้ป้อนข้อมูลเช่น รับตัวเลข 5 จำนวน เป็นต้น

2) ได้จากการคำนวณโดยใช้โปรแกรมคอมพิวเตอร์ ขึ้นอยู่กับข้อมูลที่ป้อนเข้ามาทางแผงแป้นอักขระ

3. ขั้นการวิเคราะห์ความสัมพันธ์ของข้อมูล

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญที่สุดในการเขียนโปรแกรมเพื่อแก้ปัญหาโดยใช้คอมพิวเตอร์ เพราะต้องแสดงความสัมพันธ์ระหว่างข้อมูลที่ได้จากข้อ 2.1 และข้อ 2.2 ให้อยู่ในรูปแบบการคำนวณทางคณิตศาสตร์ หรือเงื่อนไขที่คอมพิวเตอร์รู้จักเท่านั้น

ตัวอย่างที่ 2.1 ให้เขียนโปรแกรมหาคะแนนรวมจากคะแนนกลางภาคและปลายภาค

วิธีคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลรวมของคะแนน ซึ่งเป็นข้อมูลชนิดตัวเลขซึ่งอาจเป็นตัวเลขที่มีหรือไม่มีทศนิยม ในเบื้องต้นถือว่าไม่มีทศนิยม

2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกชัดเจนว่าคะแนนรวมของคะแนนกลางภาค และปลายภาค เพราะฉะนั้นจึงมีที่มาของข้อมูลอยู่ 2 จำนวน คือ คะแนนกลางภาค และคะแนนปลายภาค จากนั้นต้องดูว่าทั้งสองตัวเป็นข้อมูลชนิดอะไร ควรเป็นชนิดตัวเลข เพราะว่าต้องเอามาบวกกันเช่น 10+20 เป็นตัวเลขทั้งคู่

3. วิเคราะห์ความสัมพันธ์ของข้อมูล โจทย์บอกชัดเจนว่าต้องนำ คะแนนกลางภาค บวกกับคะแนนปลายภาค จึงจะได้คะแนนรวม ดังนี้

คะแนนรวม = คะแนนกลางภาค + คะแนนปลายภาค

ตัวอย่างที่ 2.2 ให้นักศึกษาเขียนโปรแกรมคำนวณหาพื้นที่ของสามเหลี่ยม

วิธีคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือพื้นที่ของสามเหลี่ยม ซึ่งเป็นตัวเลข

2. วิเคราะห์ที่มาของข้อมูล โจทย์ไม่ได้บอกชัดเจนว่าทำอย่างไร แต่ข้อนี้เป็นสูตรคณิตศาสตร์พื้นฐานในการหาพื้นที่สามเหลี่ยมคือต้องรู้สูตรเสียก่อนดังนี้

$$Area = \frac{1}{2} \times Base \times Height$$

ดังนั้น เห็นได้ว่าจำเป็นต้องรู้ค่าความยาวฐาน (Base) และความสูง (Height) ของรูปสามเหลี่ยม ข้อมูลมาจากผู้ใช้ป้อนเข้าทางแผงแป้นอักขระ

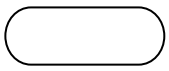
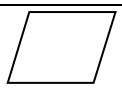
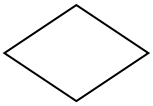
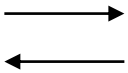
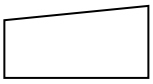
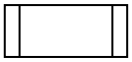
3. วิเคราะห์ความสัมพันธ์ของข้อมูล ข้อนี้มีสูตรชัดเจน ดังนั้นจึงต้องคำนวณตามสูตร

เมื่อสามารถวิเคราะห์ปัญหาของโจทย์ ได้แล้วต่อไปก็จะต้องทำการออกแบบโปรแกรมว่าในโปรแกรมจะต้องมีการทำงานอะไรบ้าง ซึ่งเทคนิคที่นิยมในการใช้ออกแบบโปรแกรมก็คือการเขียนผังงาน (Flowchart)

ผังงาน

ผังงานเป็นการแสดงขั้นตอนการทำงานของโปรแกรม หรือลำดับการคิดการกระทำ และมีคำบรรยายสั้นๆ เพื่อประกอบการทำงานที่จะเกิดขึ้นในโปรแกรม สำหรับสัญลักษณ์ที่ใช้ในการเขียนผังงานจะแตกต่างกันออกไปตามผู้พัฒนาโปรแกรมแต่ละราย สำหรับในรายวิชานี้จะใช้ สัญลักษณ์ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 สัญลักษณ์ที่ใช้ในการเขียนผังงาน

สัญลักษณ์	ความหมาย
	Terminal แสดงจุดเริ่มต้นและจุดสิ้นสุดของการประมวลผล ในการเขียนมักกำกับคำว่า START หรือ STOP ไว้ภายในสัญลักษณ์ด้วย
	Input /Output เป็นสัญลักษณ์แสดงการรับหรือแสดงผลข้อมูล
	Process แสดงคำสั่งการประมวลผลในโปรแกรม เช่น การคำนวณ การเพิ่ม – ลดค่า เป็นต้น
	Decision เป็นสัญลักษณ์แสดงตรรกะในการตัดสินใจการประมวลผล ซึ่งจะได้ผลลัพธ์ออกมา 2 กรณี คือ จริง กับ เท็จ นิยมเขียนเครื่องหมายคำถาม (?) ไว้ภายในสัญลักษณ์ด้วย
	Flow direction lines แสดงทิศทางของการประมวลผล (ระบุด้วยทิศทางหัวลูกศร)
	Manual Input การนำเข้าข้อมูลด้วยตัวเอง ที่ใช้อุปกรณ์นำเข้าง่าย ๆ เช่น แป้นพิมพ์
	Display ใช้เป็นสัญลักษณ์การแสดงผลทางจอภาพ
	Document output การแสดงผลเอกสารออกทางเครื่องพิมพ์
	Magnetic disk เป็นสัญลักษณ์แสดงอุปกรณ์จานแม่หล็ก
	Subroutine เป็นส่วนของโปรแกรมที่นิยามการทำงานไว้ล่วงหน้าแล้ว
	Connector จุดเชื่อมต่อ ใช้เชื่อมต่อกันกรณีที่มีผังงานหลายส่วนในหน้าเดียกัน
	Off-page connector ใช้เชื่อมต่อผังงานที่อยู่ต่างหน้ากระดาษกัน

ที่มา: สานนท์ เจริญฉาย. 2552

หลักในการเขียนผังงาน

โดยทั่วไปมักนิยมเขียนผังงานโดยมีหลักเกณฑ์ดังนี้

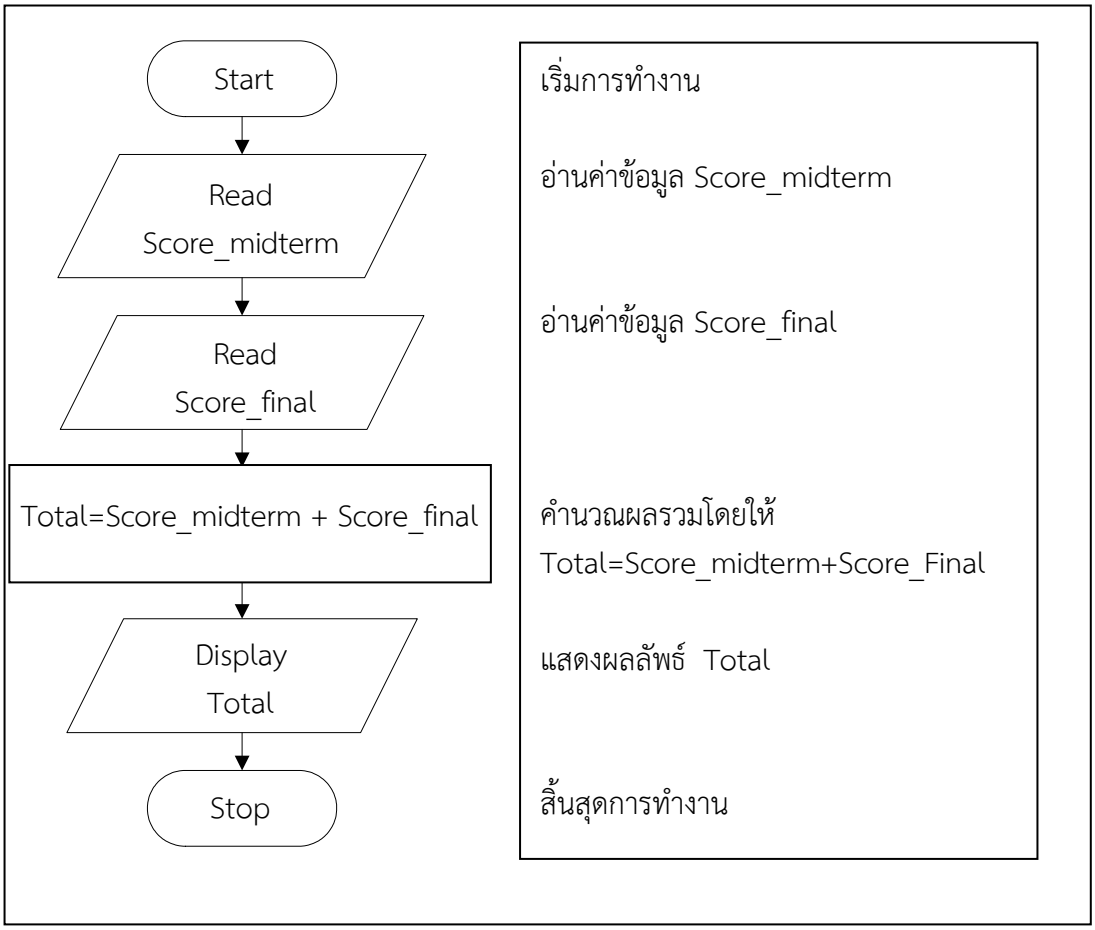
1. ระบุจุดเริ่มต้นและสิ้นสุดโดยใช้ สัญลักษณ์เครื่องปลายทาง (Terminal) โดยระบุคำว่า เริ่ม (START) และหยุด (STOP) แทนการเริ่มต้นการทำงานและสิ้นสุดโปรแกรมตามลำดับ

2. มีการเขียนสัญลักษณ์รับเข้า/ส่งออก (Input/Output) ในจุดที่มีการนำเข้าและการส่งออกข้อมูลตามลำดับ

3. เมื่อมีการคำนวณหรือการประมวลผลใดๆ ที่กระทำโดยไม่มีเงื่อนไขจะใช้สัญลักษณ์ประมวลผล (Process) ระบุถึงการประมวลผล

ตัวอย่างที่ 2.3 ให้นักศึกษาลองเขียนผังงานในหาคะแนนรวมจากคะแนนกลางภาคและปลายภาค

วิธีคิด เริ่มวิเคราะห์โจทย์แล้วนำมาเขียนผังงานได้ดังนี้



ภาพที่ 2.1 ผังงานของการหาคะแนนรวม

จากภาพที่ 2.1 จะเห็นได้ว่าหากทำการวิเคราะห์ปัญหาได้ถูกต้องแล้วสามารถนำข้อมูลที่ได้จากการวิเคราะห์มาประกอบกันจนเป็นผังงานได้โดยง่าย ซึ่งมีขั้นตอนง่ายๆ ดังนี้

1. ข้อมูลใดที่ต้องมีการรับจากผู้ใช้ให้ใช้สัญลักษณ์ Input/Output โดยระบุภายในสัญลักษณ์ว่า Read หมายถึงการรับข้อมูลเข้าสู่ระบบ หรือการอ่านข้อมูล

2. ข้อมูลใดที่ต้องการแสดงผลสู่ผู้ใช้ให้ใช้สัญลักษณ์ Input/Output โดยระบุภายในสัญลักษณ์ว่าต้อง Display หรือ Print (คือการแสดงข้อมูล)

3. เมื่อใดที่ต้องมีการคำนวณให้ใส่สัญลักษณ์ Process พร้อมทั้งระบุความสัมพันธ์ที่ต้องการในรูปแบบของสูตร ทางคณิตศาสตร์ เช่น บวก, ลบ, คูณ, หาร เป็นต้น

ตัวอย่างที่ 2.4 ให้เขียนผังงานในการรับตัวเลข 2 จำนวนแล้วแสดงว่าเลขจำนวนใดมีค่ามากกว่ากัน

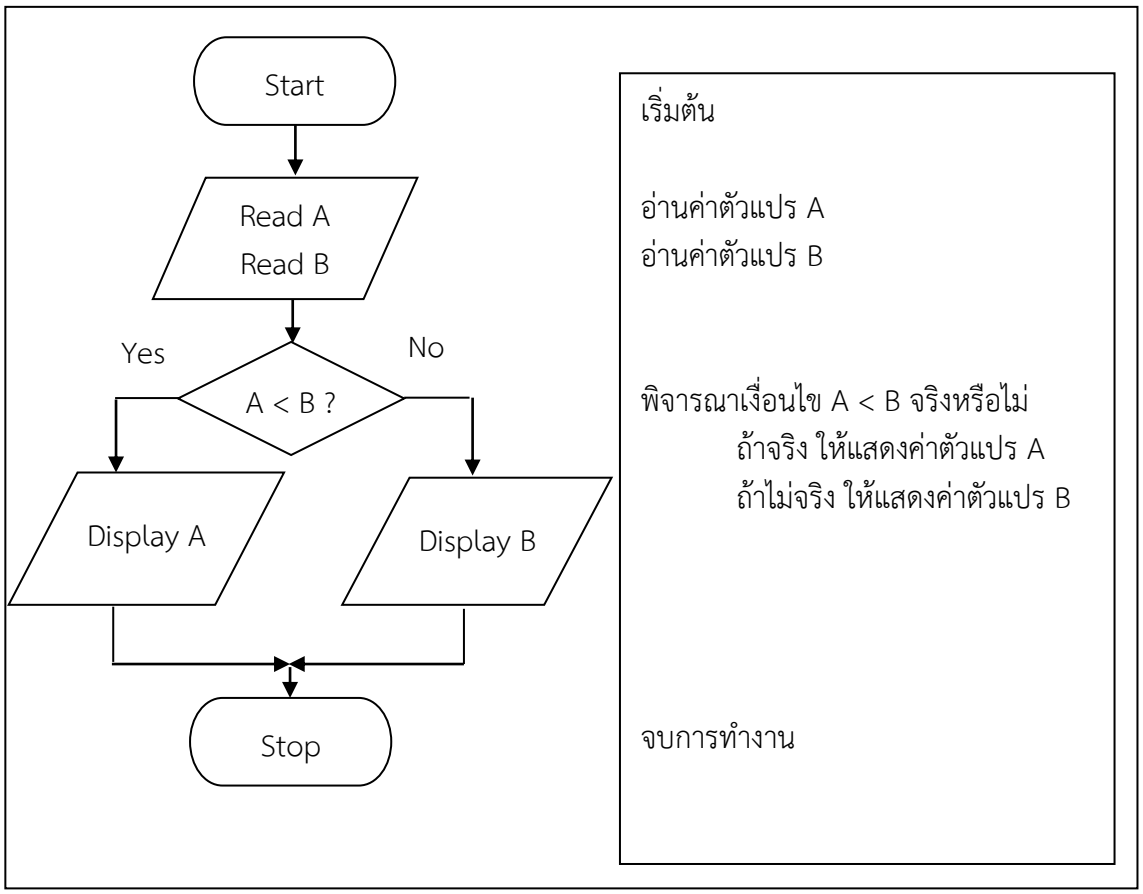
วิธีคิด ต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือ เลขจำนวนที่น้อยกว่าอีกจำนวน
2. วิเคราะห์ที่มาของข้อมูล โจทย์ต้องการให้รับตัวเลข 2 จำนวนสมมติว่าชื่อ a และ b
3. วิเคราะห์ความสัมพันธ์ของข้อมูล มีเงื่อนไขที่เป็นไปได้อยู่ 2 กรณีคือ

ถ้า a น้อยกว่า b จริง ให้แสดงค่าของตัวแปร a

ถ้า a น้อยกว่า b ไม่จริง ให้แสดงค่าของตัวแปร b

ซึ่งเขียนผังงานได้ดังภาพที่ 2.2



ภาพที่ 2.2 ผังงานของการสร้างเงื่อนไข

จากตัวอย่างที่ 2.1 ถึง ตัวอย่างที่ 2.4 เป็นตัวอย่างในการเขียนผังงานอย่างง่ายเท่านั้น ไม่ได้มีการทำงานซับซ้อนแต่อย่างใด แต่ในความเป็นจริงแล้วการเขียนโปรแกรมคอมพิวเตอร์จะใช้ในการแก้ปัญหาที่มีการทำงานซับซ้อน และมีเป็นจำนวนมาก นั่นหมายความว่าจะต้องมีการออกแบบโปรแกรมที่ยากขึ้น

ตัวอย่างที่ 2.5 ให้นักศึกษาเขียนผังงานในการหาบวกของตัวเลขตั้งแต่ 1 จนถึง 10

วิธีคิด วิเคราะห์ปัญหาดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลรวมของตัวเลข ซึ่งแน่นอนว่าต้องเป็นตัวเลข

2. วิเคราะห์ที่มาของข้อมูล โจทย์ไม่ได้บอกชัดเจนว่าทำอย่างไร แต่บอกว่าเป็นการหาผลบวกตั้งเลขตัวเลข 1 จนถึง 10 นั่นคือข้อมูล 1 ถึง 10 นี้ต้องได้โดยตัวของโปรแกรมเอง

3. วิเคราะห์ความสัมพันธ์ของข้อมูล ซึ่งข้อนี้มีความซับซ้อนมากขึ้น หลักของการหาผลรวมตั้งแต่เลข 1-10 เป็นดังนี้

เขียนตัวเลขจาก 1 ถึง 10 คือ 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 จากนั้นหาผลบวกมาเก็บไว้ในตัวแปรอีกตัวหนึ่งกำหนด ให้ชื่อ Sum จะได้ดังนี้

ตารางที่ 2.2 การทำงานของผลบวกในแต่ละรอบ

รอบที่ (I)	**เริ่ม**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
เลขที่บวก	0	1	2	3	4	5	6	7	8	9	10
Sum	0	1	3	6	10	15	21	28	36	45	55

จะเห็นได้ว่าในแต่ละรอบที่บวกไปค่าของตัวแปร sum จะมีค่าเพิ่มขึ้นเท่ากับรอบที่บวก (I) โดยการทำงานนี้ต้องวนซ้ำๆ กันตลอดเวลาที่ค่าของ I ยังคงน้อยกว่าหรือเท่ากับ 10 ซึ่งเกิดเป็นเงื่อนไขขึ้นมาดังนี้

ถ้าค่าของตัวแปร I น้อยกว่าหรือเท่ากับ 10 เป็นจริง ให้ทำงานดังต่อไปนี้

1) ค่าของตัวแปรเพิ่มไปอีก I (Sum = Sum + I)

2) เพิ่มค่าของตัวแปร I อีก 1 (I = I+1)

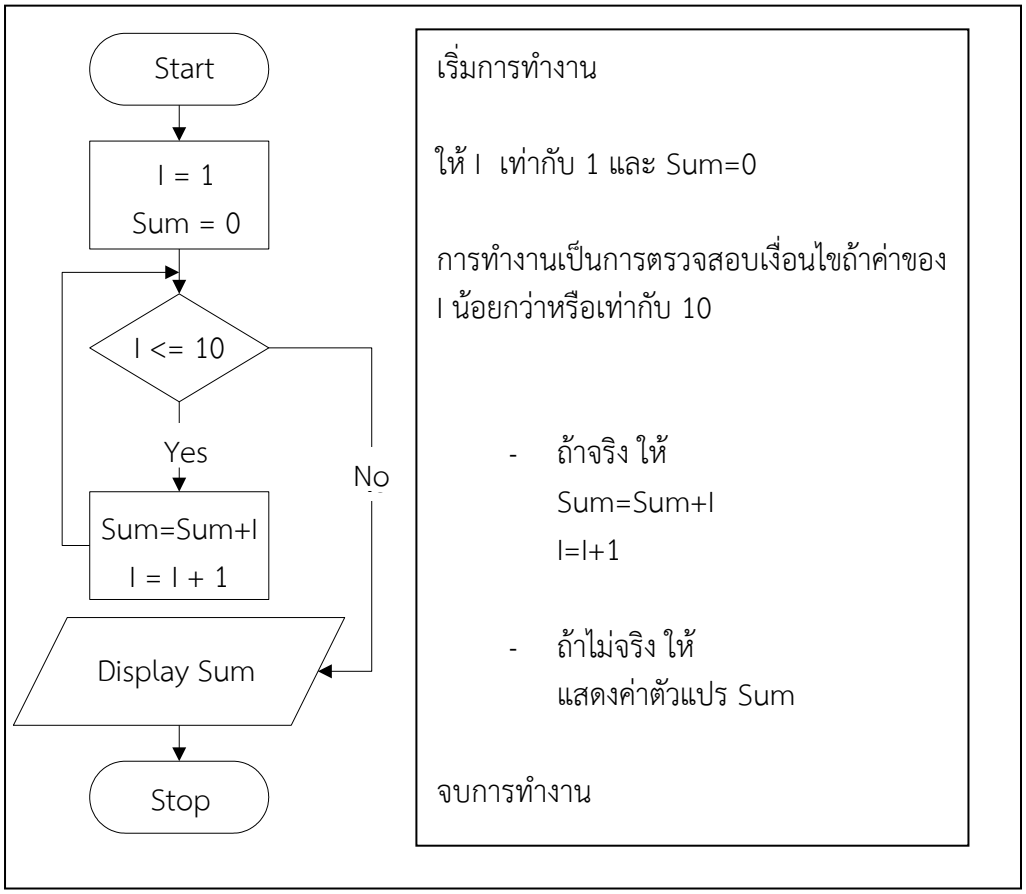
3) ย้อนกลับไปตรวจสอบเงื่อนไข อีกครั้ง

ถ้าค่าของตัวแปร I น้อยกว่าหรือเท่ากับ 10 เป็นเท็จ ให้ทำงานดังต่อไปนี้

1) ให้แสดงค่าของตัวแปร Sum

2) จบการทำงาน

เขียนผังงานได้ดังภาพที่ 2.3



ภาพที่ 2.3 ผังงานการหาบวกของตัวเลขตั้งแต่ 1 จนถึง 10

ตัวอย่างที่ 2.6 ให้นักศึกษาเขียนผังงานให้ห้างสรรพสินค้าแห่งหนึ่ง มีการจัดวางสินค้าดังนี้

ชั้น 1 สินค้าทั่วไป (Department Store)

ชั้น 2 เครื่องแต่งกายบุรุษ (Men's Wear)

ชั้น 3 เครื่องแต่งกายสตรี (Women's Wear)

ชั้น 4 เครื่องตกแต่งบ้าน (Home Decoration)

ห้างแห่งนี้ต้องการเขียนโปรแกรมอำนวยความสะดวกให้ลูกค้าไปซื้อสินค้าได้ง่าย ควรจะมีผังงานอย่างไร

วิธีคิด วิเคราะห์ปัญหาดังนี้

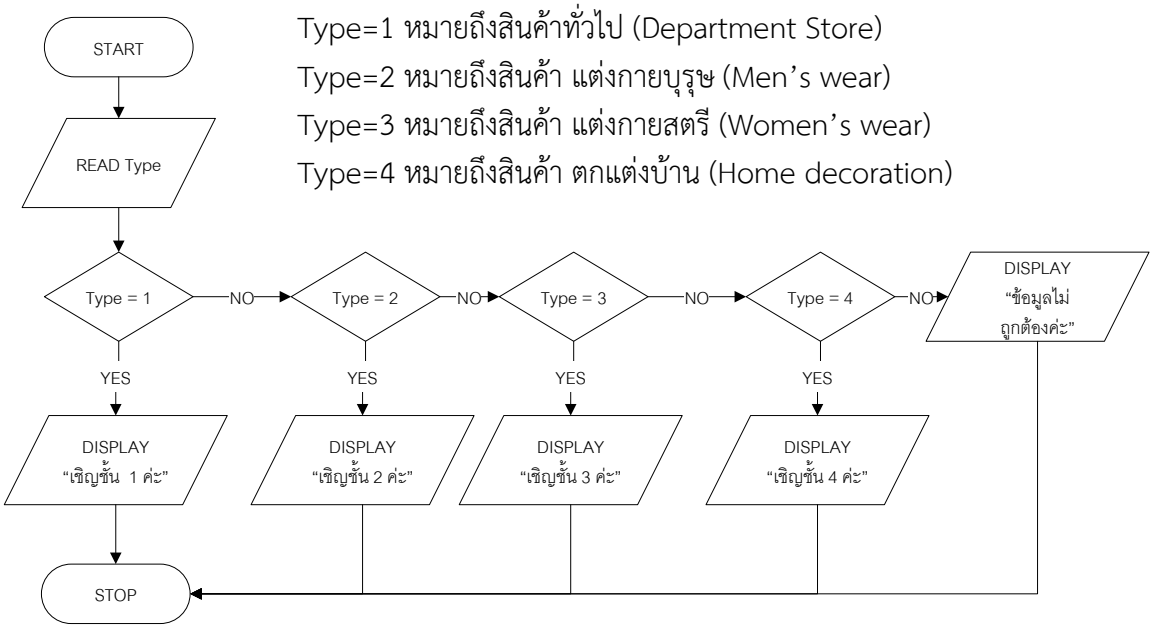
1. วิเคราะห์ผลลัพธ์

ผลลัพธ์ที่ต้องการคือสินค้าที่ต้องการอยู่ชั้นไหน ซึ่งต้องเป็นตัวเลข

2. วิเคราะห์ที่มาของข้อมูล

โจทย์ไม่ได้บอกชัดเจนว่าทำอย่างไร แต่บอกว่าเป็นสินค้ามี 4 ประเภท เพราะฉะนั้นข้อมูลเข้าควรจะเป็นสินค้าที่ต้องการจะซื้อ ควรจะแสดงให้ผู้ใช้ก่อนว่ามีสินค้าอะไร โดยเลือกหมายเลขที่ต้องการ

3. วิเคราะห์ความสัมพันธ์ของข้อมูล
ข้อนี้จะเป็นเงื่อนไข ซึ่งเป็นเงื่อนไขตามที่โจทย์ให้มา คือ
ชั้น 1 สินค้าทั่วไป (Department Store)
ชั้น 2 เครื่องแต่งกายบุรุษ (Men's Wear)
ชั้น 3 เครื่องแต่งกายสตรี (Women's Wear)
ชั้น 4 เครื่องตกแต่งบ้าน (Home Decoration)
4. เขียนผังงานได้ดังภาพที่ 2.4



ภาพที่ 2.4 ผังงานการจัดวางสินค้าของห้างสรรพสินค้าแห่งหนึ่ง

การทำงานของผังงานนี้ จะอ่านค่า Type แทน สินค้าที่ต้องการจะซื้อ แล้วตรวจสอบว่าตรงกับเงื่อนไขใด ก็จะแสดงว่าสินค้านั้นอยู่ชั้นไหน เช่น ถ้าต้องการซื้อเสื้อสตรี ก็เลือก Type เป็น "3" ก็จะแสดงว่าข้อความว่า "เชิญชั้น 3 ค่ะ" หากต้องการซื้อของตกแต่งบ้านก็เลือก Type เป็น "4" ก็จะแสดงข้อความว่า "เชิญชั้น 4 ค่ะ" หากเลือก Type ผิด คือไม่ใช่ 1 – 4 ก็จะแสดงข้อความว่า "ข้อมูลไม่ถูกต้อง"

ตัวอย่างที่ 2.7 พนักงานตามเงื่อนไขดังนี้
พนักงานเงินเดือน น้อยกว่า 10,000 ปรับเงินเดือนให้ 10 %
พนักงานเงินเดือน 10,001 - 50,000 ปรับเงินเดือนให้ 5 %
พนักงานเงินเดือน 50,001 - 100,000 ปรับเงินเดือนให้ 3 %
ให้นักศึกษาเขียนผังงานของโปรแกรมในการปรับเงินเดือนขึ้นของบริษัทแห่งนี้

วิธีคิด ก่อนอื่นจะต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์

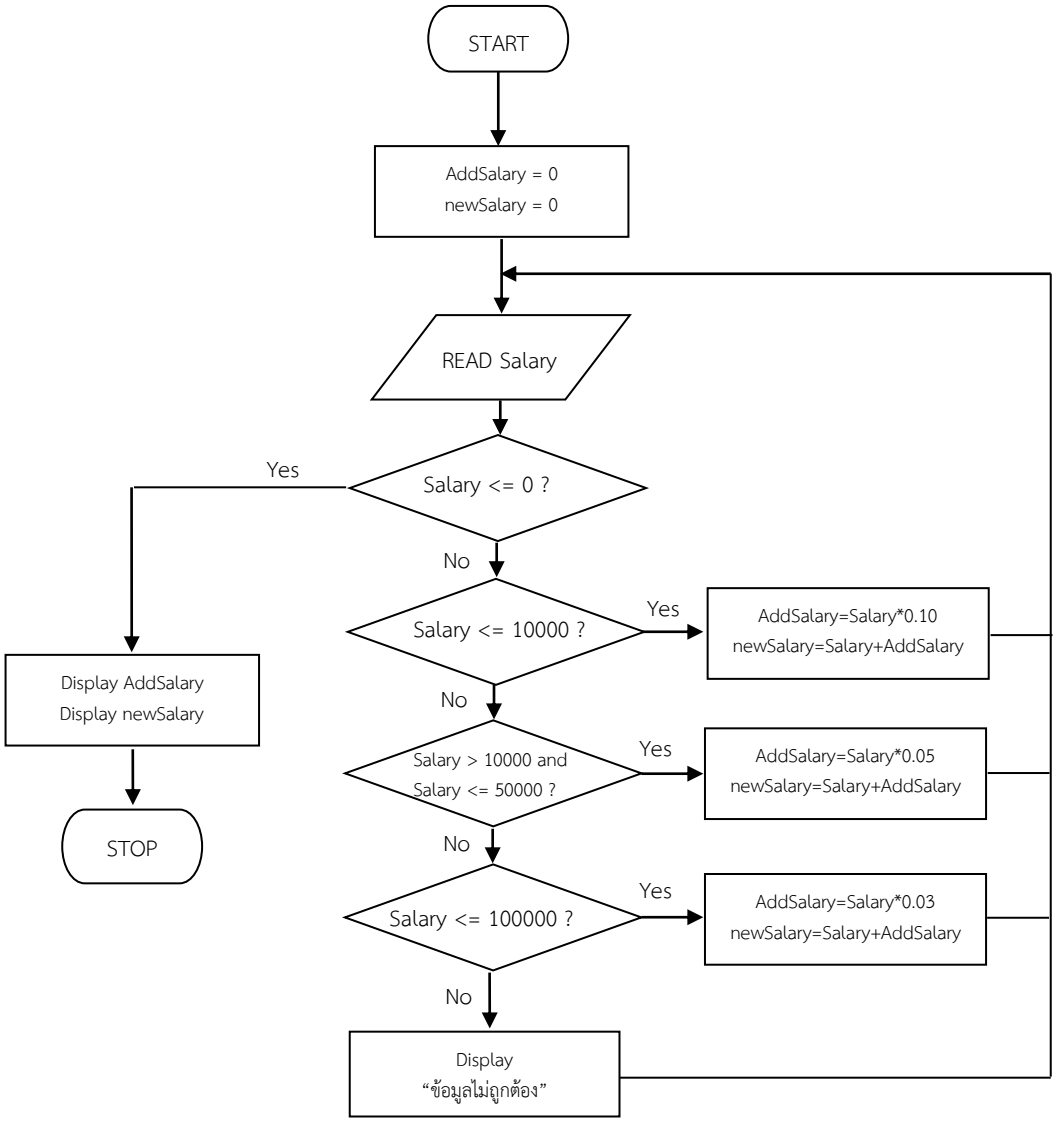
ผลลัพธ์ที่ต้องการคือจะต้องปรับเงินเดือนให้พนักงาน ตามขั้นเงินเดือน

2. วิเคราะห์ที่มาของข้อมูล

ข้อมูลเข้าที่ต้องการคือ เงินเดือนของพนักงาน ซึ่งได้มาจากการกรอกข้อมูล หรือรับค่าจากแป้นพิมพ์

3. วิเคราะห์ความสัมพันธ์ของข้อมูล ซึ่งข้อนี้จะเป็นเงื่อนไขซึ่งเป็นเงื่อนไขตามที่โจทย์ให้มา และสร้างเงื่อนไขการวนซ้ำรับเงินเดือนใหม่ โดยกำหนดที่เงินเดือนมากกว่า 0 ในวนรับค่าเงินเดือนของพนักงานคนต่อไป ถ้าเงินเดือนน้อยกว่า หรือ เท่ากับ 0 คือจบการทำงาน

4. เขียนผังงานได้ดังภาพที่ 2.5



ภาพที่ 2.5 ผังงานของการปรับเงินเดือน

จากภาพที่ 2.5 แสดงให้เห็นการทำงานของโปรแกรมเริ่มจากกำหนดค่าเริ่มต้นให้กับตัวแปร AddSalary มีค่าเท่ากับ 0 และตัวแปร newSalary มีค่าเท่ากับ 0

รับค่าเงินเดือน (Salary) แล้วตรวจสอบเงื่อนไขว่าเงินเดือนมีค่ามากกว่า 0 หรือไม่ถ้ามีค่าน้อยกว่า 0 หรือเท่ากับ 0 ให้โปรแกรมแสดงค่าจากตัวแปร AddSalary และตัวแปร newSalary ถ้าค่าเงินเดือนมีค่ามากกว่า 0 ตรวจสอบต่อว่าเงินเดือนน้อยกว่า 10,000 เงื่อนไข (Salary <= 10000) เป็นจริง โปรแกรมคำนวณเพิ่มเงินเดือน 10 % (AddSalary=Salary*0.10) และคำนวณหาเงินเดือนใหม่ โดยบวกเงินเดือนที่ได้เพิ่มขึ้นกับเงินเดือนเดิม (newSalary=Salary+AddSalary) ถ้าเงินเดือนอยู่ในช่วงที่ 2 คือมากกว่า 10,000 แต่ไม่เกิน 50,000 เงื่อนไข (Salary > 10000 and Salary <= 50000) เป็นจริง จะได้เพิ่มเงินเดือน 5 % (AddSalary=Salary*0.05) และหากเงินเดือนไม่เกิน 100,000 เงื่อนไข (Salary <= 100000) เป็นจริง ได้เพิ่มเงินเดือน 3 % (AddSalary=Salary*0.03) แต่หากเงินเดือนมากกว่า 100,000 จะแสดงข้อความว่า “ข้อมูลไม่ถูกต้อง”

ตัวอย่างที่ 2.8 จากตัวอย่างที่ 2.7 ให้แสดงจำนวนผู้ได้รับเงินเดือนขึ้นในแต่ละเงื่อนไข และบริษัทต้องจ่ายเงินเดือนเพิ่มขึ้น เดือนละกี่บาท

วิธีคิด วิเคราะห์ปัญหา

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือจะต้องปรับเงินเดือนให้พนักงาน ตามขั้นเงินเดือน จำนวนของผู้ได้รับเงินเดือนขึ้นในแต่ละเงื่อนไข และเงินเดือนทั้งหมด ที่ต้องจ่าย

2. วิเคราะห์ที่มาของข้อมูล ข้อมูลเข้าที่ต้องการคือ เงินเดือนของพนักงาน ซึ่งได้มาจากการกรอกข้อมูล หรือรับค่าจากแผงแป้นอักขระ

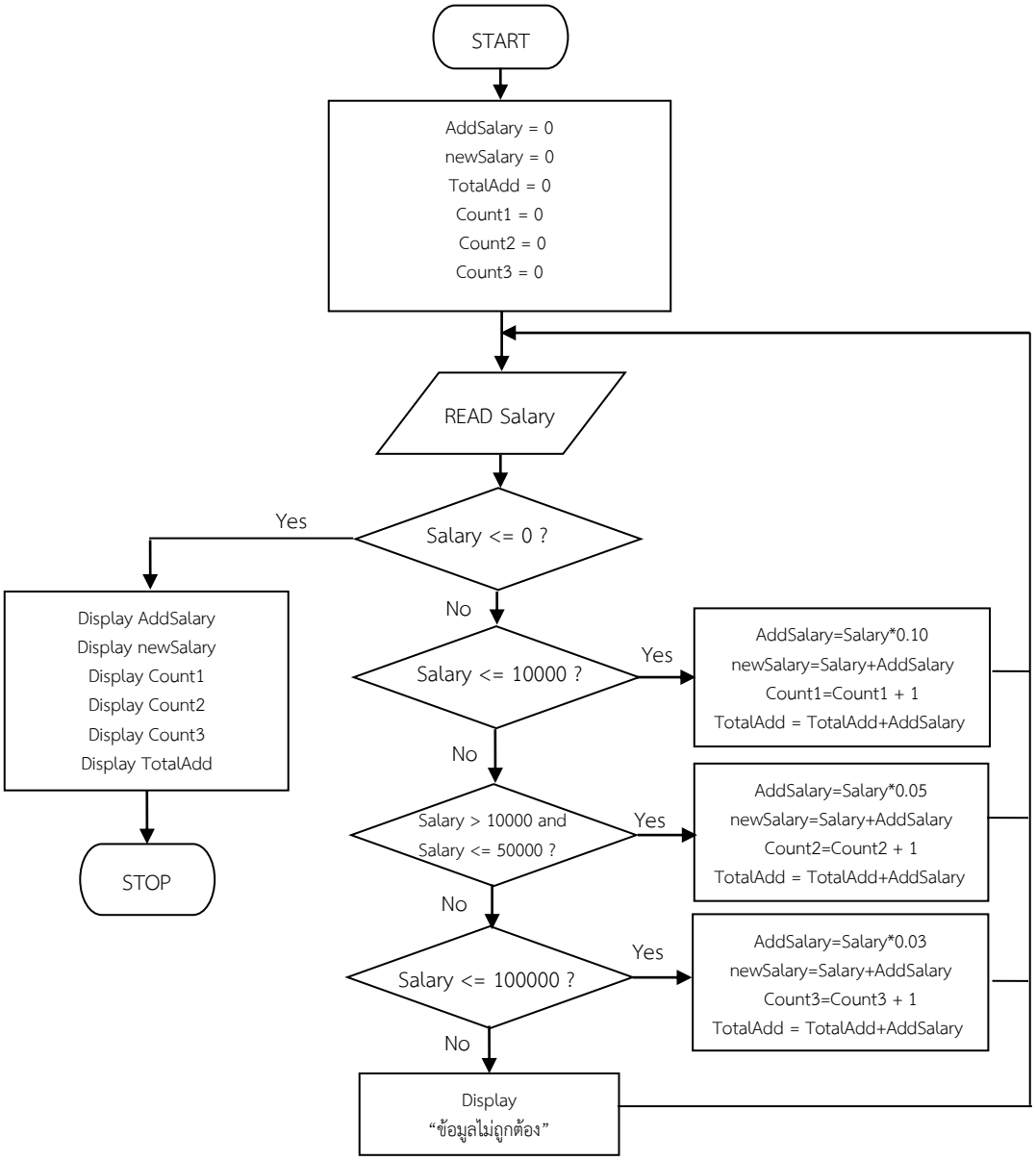
3. วิเคราะห์ความสัมพันธ์ของข้อมูล ซึ่งข้อนี้จะเป็นเงื่อนไข ซึ่งเป็นเงื่อนไขตามที่โจทย์ให้มา และต้องนับจำนวนพนักงานในแต่เงื่อนไขโดยบอกค่าครั้งละ 1 ในแต่ละเงื่อนไข และหาผลบวกของเงินเดือนที่เพิ่มขึ้นทั้งหมดโดยทำการบวกสะสม

4. เขียนผังงานได้ดังภาพที่ 2.5

การทำงานของผังงานข้อนี้ คล้ายกับตัวอย่างที่ 2.7 คือ เริ่มจากกำหนดค่าเริ่มต้นให้กับตัวแปร AddSalary ตัวแปร newSalary ตัวแปร TotalAdd ตัวแปร Count1 ตัวแปร Count2 และตัวแปร Count3 มีค่าเท่ากับ 0

รับค่าเงินเดือน (Salary) แล้วตรวจสอบเงื่อนไขว่าเงินเดือนมีค่ามากกว่า 0 หรือไม่ถ้ามีค่าน้อยกว่า 0 หรือเท่ากับ 0 ให้โปรแกรมแสดงค่าจากตัวแปร AddSalary และตัวแปร newSalary ถ้าค่าเงินเดือนมีค่ามากกว่า 0 ตรวจสอบเงื่อนไข ว่าเข้าเงื่อนไขใด ถ้าเงินเดือนน้อยกว่า 10,000 ก็จะเข้าเงื่อนไขแรก คือได้เพิ่มเงินเดือน 10 % โดยคูณเงินเดือนด้วย 0.10 และคำนวณหาเงินเดือนใหม่ โดยบวกเงินที่ได้เพิ่มขึ้น กับเงินเดือนเดิม และบวก 1 เพิ่มที่ Count1 เพื่อนับจำนวนพนักงานที่ได้เงินเพิ่ม 10 % และหาผลบวกสะสมของเงินเดือนที่เพิ่มขึ้นทั้งหมด ถ้าเงินเดือนอยู่ในช่วงที่ 2 คือ มากกว่า 10,000 แต่ไม่เกิน 50,000 ก็จะได้เพิ่มเงินเดือน 5 % และบวก 1 เพิ่มที่ Count2 เพื่อนับจำนวนพนักงานที่ได้เงินเพิ่ม 5 % และหาผลบวกสะสมของเงินเดือนที่เพิ่มขึ้นทั้งหมด และหากเงินเดือนมากกว่า 50,000 แต่ไม่เกิน 100,000 ก็จะได้เพิ่ม 3 % และบวก 1 เพิ่มที่ Count3 เพื่อนับจำนวนพนักงานที่ได้เงินเพิ่ม 3 % และหาผลบวกสะสมของเงินเดือนที่เพิ่มขึ้นทั้งหมดแต่หาก เงินเดือน

มากกว่า 100,000 ซึ่งโจทย์ไม่ได้บอกว่าจะเพิ่มให้กี่เปอร์เซ็นต์ ก็จะแสดงข้อความว่า “ข้อมูลไม่ถูกต้อง”



ภาพที่ 2.6 ผังงานของการจ่ายเงินเดือน

จากภาพที่ 2.6 แสดงให้เห็นการทำงานของโปรแกรมเริ่มจากรับค่าเงินเดือน (Salary) แล้วตรวจสอบเงื่อนไข ถ้าเงินเดือนน้อยกว่า 10,000 เงื่อนไข (Salary <= 10000) เป็นจริง โปรแกรมคำนวณเพิ่มเงินเดือน 10 % (AddSalary=Salary*0.10) และคำนวณเงินเดือนใหม่ โดยบวกเงินเดือนที่ได้เพิ่มขึ้นกับเงินเดือนเดิม (newSalary=Salary+AddSalary) นับจำนวนพนักงานในขั้นนี้ด้วยการใช้ Count1 (Count1=Count1 + 1) และคำนวณหาผลรวมของเงินเดือนที่เพิ่มทั้งหมด (TotalAdd = TotalAdd+AddSalary) ถ้าเงินเดือนอยู่ในช่วงที่ 2 คือมากกว่า 10,000 แต่ไม่เกิน 50,000 เงื่อนไข (Salary > 10000 and Salary <= 50000) เป็นจริง จะได้เพิ่มเงินเดือน 5 % (AddSalary=Salary*0.05) และคำนวณหาเงินเดือนใหม่ โดยบวกเงินเดือนที่ได้เพิ่มขึ้นกับ

เงินเดือนเดิม (newSalary=Salary+AddSalary) นับจำนวนพนักงานในขั้นนี้ด้วยการใช้ Count2 (Count2=Count2 + 1) และคำนวณหาผลรวมของเงินเดือนที่เพิ่มทั้งหมด (TotalAdd = TotalAdd+AddSalary) และหากเงินเดือนไม่เกิน 100,000 เงื่อนไข (Salary <= 100000) เป็นจริง ได้เพิ่มเงินเดือน 3 % (AddSalary=Salary*0.03) และคำนวณหาเงินเดือนใหม่ โดยบวกเงินเดือนที่ได้เพิ่มขึ้นกับเงินเดือนเดิม (newSalary=Salary+AddSalary) นับจำนวนพนักงานในขั้นนี้ด้วยการใช้ Count3 (Count3=Count3 + 1) และคำนวณหาผลรวมของเงินเดือนที่เพิ่มทั้งหมด (TotalAdd = TotalAdd+AddSalary) แต่หากเงินเดือนมากกว่า 100,000 จะแสดงข้อความว่า “ข้อมูลไม่ถูกต้อง”

สรุป

ปัญหาที่เกิดขึ้นแบ่งเป็น 2 ประเภท คือ ปัญหาที่มีวิธีการแก้ไขได้โดย กฎ สูตร หรืออัลกอริทึมได้ และอีกประเภทหนึ่งคือปัญหาที่ไม่มีวิธีการแก้ไขแน่นอน ซึ่งต้องเขียนโปรแกรมเพื่อแก้ไขปัญหา หรือต้องใช้ทฤษฎีทางปัญญาประดิษฐ์ในการแก้ปัญหา เช่นปัญหาในการหาเส้นทางที่สั้นที่สุดของการเดินทางไปเมืองต่าง ๆ เป็นต้น

หลักการวิเคราะห์ปัญหา ต้องเริ่มจากการวิเคราะห์ผลลัพธ์ แล้วจึงวิเคราะห์ที่มาของข้อมูล แล้วจึงมาวิเคราะห์หาความสัมพันธ์ของของข้อมูล หรือวิธีการแก้ไขปัญหา เพื่อให้ได้ผลลัพธ์ตามต้องการ

ผังงานมีสัญลักษณ์ที่สำคัญคือ สัญลักษณ์ Input/Output ใช้ในการรับและแสดงผลข้อมูล สัญลักษณ์ Process ใช้ในการคำนวณ และมีลูกศรบอกทิศทางทางไหลของการทำงาน

แบบฝึกหัด

1. อัลกอริทึม คืออะไร
2. จงอธิบายหลักการวิเคราะห์ปัญหา
3. จงเขียนสัญลักษณ์ที่ใช้แทน การประมวลผล การตัดสินใจ การรับและการแสดงผล
4. จงเขียนผังงานเพื่อคำนวณหาพื้นที่วงกลม
5. จงเขียนผังงานเพื่อแปลงค่าจากเซนติเมตร เป็นนิ้ว
6. จงเขียนผังงานเพื่อคำนวณหาประกันสังคม 5 % จากเงินเดือนไม่เกิน 750 บาท
7. จงเขียนผังงานเพื่อรับข้อมูลมา 10 จำนวน
8. จงเขียนผังงานเพื่อหาผลบวกของเลข 11 ถึง 100
9. จงเขียนผังงานเพื่อแสดงเกรด A - F
10. จงเขียนผังงานเพื่อหาผลบวกของเลขคี่และเลขคู่ของเลข 1 - 10
11. จงเขียนผังงานเพื่อหาค่าต่ำสุดของตัวเลข 10 จำนวน
12. จงเขียนผังงานเพื่อสลับค่า A กับ B

เอกสารอ้างอิง

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

บทที่ 3
ตัวแปลภาษา

ในบทนี้อธิบายถึงประวัติความเป็นมาของภาษาซี ข้อเด่นของภาษาซี การใช้งานตัวแปลภาษา ขั้นตอนการทำงานของการเขียนโปรแกรมเบื้องต้น การพิมพ์รหัสต้นฉบับ (Source Code) การบันทึก การแปลรหัสต้นฉบับเป็นรหัสจุดหมาย (Object Code) การประมวลผล และการแสดงผลลัพธ์การประมวลผล

การเขียนโปรแกรมด้วยภาษาซีนั้น เครื่องคอมพิวเตอร์ไม่สามารถที่จะทำงานได้ตามรหัสต้นฉบับของภาษาซี เพราะเครื่องคอมพิวเตอร์คืออุปกรณ์อิเล็กทรอนิกส์ ไม่สามารถเข้าใจรหัสต้นฉบับของภาษาซี ซึ่งเขียนด้วยภาษาอังกฤษ จึงต้องมีตัวแปลภาษาเพื่อทำการแปลงรหัสต้นฉบับภาษาซี ให้เป็นภาษาเครื่อง ตัวแปลภาษาของภาษาซีเป็นแบบคอมไพเลอร์ มีบริษัทผลิตซอฟต์แวร์ผลิตคอมไพเลอร์เพื่อเป็นตัวแปลภาษาซีหลายบริษัทหลายรุ่น สำหรับซอฟต์แวร์ที่ใช้อ้างอิงบทนี้ คือ Turbo C++ Version 1.00 Copyright © 1990 by Borland International, Inc. ซึ่งเหมาะสำหรับผู้เริ่มต้นเรียนรู้การเขียนโปรแกรมเบื้องต้น เช่นเดียวกับภาษาซี เป็นภาษาที่เหมาะสำหรับผู้เริ่มต้นศึกษาการเขียนโปรแกรม ภาษาซีเป็นภาษาที่โครงสร้างภาษาที่ชัดเจนเป็นระเบียบ มีกฎเกณฑ์รูปแบบไวยากรณ์ มีชนิดของตัวแปร มีการประกาศตัวแปร มีตัวดำเนินการทางคณิตศาสตร์ มีหลักการเขียนโปรแกรม มีคำสั่งรองรับการเขียนโปรแกรมแบบมีเงื่อนไข การวนซ้ำ มีฟังก์ชันมาตรฐานรองรับการทำงานทั้งด้านการคำนวณทางคณิตศาสตร์ ผู้เขียนสามารถสร้างฟังก์ชันเพื่อใช้งาน เหมาะสำหรับนักวิทยาศาสตร์สามารถนำไปประยุกต์ใช้ในการคำนวณต่าง ๆ ได้ดี

ประวัติความเป็นมาของภาษาซี

ภาษาซี เป็นภาษาคอมพิวเตอร์ที่ได้พัฒนาขึ้นในปี ค.ศ. 1972 (พ.ศ. 2515) โดย เดนนิส ริทชี่ (Dennis Ritchie) ณ ห้องปฏิบัติการเบล บริษัทเบลล์เทเลโฟนแลบอลาทอรีส์ (Bell Telephone Laboratories, Inc.) ปัจจุบันคือ บริษัทเอทีแอนด์ทีเบลล์แลบอลาทอรีส์ (AT&T Bell Laboratories, Inc.) ซึ่งได้พัฒนามาจากภาษาบี (ไกรศร ตั้งโอภากุล. 2554: 1) ในเบื้องต้นภาษาซีได้ถูกนำมาใช้ในการเขียนโปรแกรมบนระบบปฏิบัติยูนิกซ์ (UNIX) ซึ่งเป็นระบบปฏิบัติการที่แพร่หลายมากในขณะนั้นจนถึงปัจจุบัน จนกระทั่งได้มีการพัฒนาเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) และได้รับความแพร่หลายมากขึ้น จึงได้มีการพัฒนาตัวแปลภาษาซี ขึ้นเพื่อการพาณิชย์ ทำให้ภาษาซีเป็นภาษาที่ได้รับความนิยมแพร่หลายมากขึ้น เพราะใช้งานได้ในหลายๆ ระบบปฏิบัติการ

เนื่องจากภาษาซี เป็นภาษาที่มีในทุกๆ ระบบปฏิบัติการและได้รับการพัฒนาตัวแปลภาษาจากหลายๆ บริษัท องค์กรมาตรฐานแห่งชาติสหรัฐอเมริกา (American Nation Standards Institute: ANSI) ได้ทำการสร้างข้อตกลงที่เป็นมาตรฐานเดียวกันขึ้นของภาษาซี ในปี ค.ศ.1983 เรียกภาษาซีในมาตรฐานนี้ว่า ANSI C

ภาษาซีเป็นภาษาที่มีการพัฒนาตัวแปลภาษาในหลายระบบปฏิบัติการ ตั้งแต่ไมโครคอมพิวเตอร์ขนาด 8 บิต ไปจนถึงเครื่องขนาดใหญ่ เช่น มินิคอมพิวเตอร์ เมนเฟรมคอมพิวเตอร์ และซุปเปอร์คอมพิวเตอร์

ข้อเด่นของภาษาซี

ภาษาซี เป็นภาษาที่ใช้ในการเขียนโปรแกรมโดยมีวัตถุประสงค์ทั่วไป สามารถเขียนได้ง่าย มีการควบคุมการเขียนโปรแกรม มีโครงสร้างข้อมูล และมีตัวดำเนินการมากมาย ภาษาซีเป็นภาษาที่มีประสิทธิภาพมากในงานด้านต่าง ๆ ภาษาซีเริ่มออกแบบและพัฒนาบนระบบปฏิบัติการ UNIX บนเครื่อง DEC PDP-11 โดย เดนนิส ริชชี่ (Dennis Ritchie) ภาษาซีสามารถเขียนได้ง่าย สามารถเขียนและประมวลผลได้โดยไม่ต้องเปลี่ยนแปลงรหัสต้นฉบบับ (Brian W. Kernigham, Dennis M. Ritchie. 1988: 8)

ภาษาซี มีอยู่ในเกือบทุกระบบปฏิบัติการดังได้กล่าวนั้น จึงทำให้การที่จะนำเอารหัสต้นฉบับจากระบบหนึ่งไปใช้ในอีกระบบหนึ่งจึงเป็นเรื่องง่าย เพียงแค่แปลจะต้องทำการแปลโปรแกรมทุกครั้งเท่านั้น ซึ่งสามารถสรุปข้อดีของภาษาซี ได้ดังนี้

1. มีการพัฒนาขึ้นใช้งานเพื่อให้เป็นภาษามาตรฐาน ไม่ขึ้นกับระบบปฏิบัติการ
2. ใช้หลักการเขียนโปรแกรมในลักษณะที่เป็นโปรแกรมโครงสร้าง มีหลักการทำงานที่ไม่ซับซ้อนและเข้าใจง่าย (ไกรศร ตั้งโอภากุล. 2554: 1) เหมาะสำหรับผู้เริ่มต้นเขียนโปรแกรม
3. เป็นภาษาที่มีตัวแปลภาษาชนิดคอมไพเลอร์ ที่มีประสิทธิภาพสูง สามารถแปลงเป็นรหัสภาษาเครื่อง ที่สั้นกะทัดรัด ทำงานได้เร็ว
4. มีความยืดหยุ่น สามารถค้นหาที่ผิด และแก้ไขรหัสต้นฉบับได้ง่ายเมื่อพบข้อผิดพลาดเกิดขึ้น
5. สามารถประยุกต์เข้ากับงานต่างๆ ได้เป็นอย่างดี
6. ปัจจุบันได้รับการพัฒนาให้สามารถรองรับการรหัสคำสั่งในรูปแบบของการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP) ได้ ซึ่งปัจจุบันมีการใช้อย่างแพร่หลาย
7. มีฟังก์ชันมาตรฐานเกี่ยวกับการคำนวณทางคณิตศาสตร์ในคลังโปรแกรมรองรับการทำงานด้านวิทยาศาสตร์ เหมาะสำหรับนักวิทยาศาสตร์
8. มีชนิดของตัวแปรรองรับทศนิยมจำนวนมาก เหมาะสำหรับการคำนวณงานด้านวิทยาศาสตร์

ตัวแปลภาษา

ภาษาซี (C Language) จัดเป็นภาษาระดับสูงในยุคที่ 3 ต้องมีการใช้ตัวแปลภาษา โดยตัวแปลภาษาที่ใช้ในภาษาซี นั้นจะเป็นชนิดคอมไพเลอร์ (Complier) กล่าวคือจะทำการอ่านรหัสต้นฉบับครั้งเดียวทั้งหมด ซึ่งขั้นตอนในการเปลี่ยนจากรหัสต้นฉบับไปเป็นภาษาเครื่องนั้น มี 2 ขั้นตอนย่อย คือ แปลโปรแกรม (compile) และเชื่อมโยง (link)

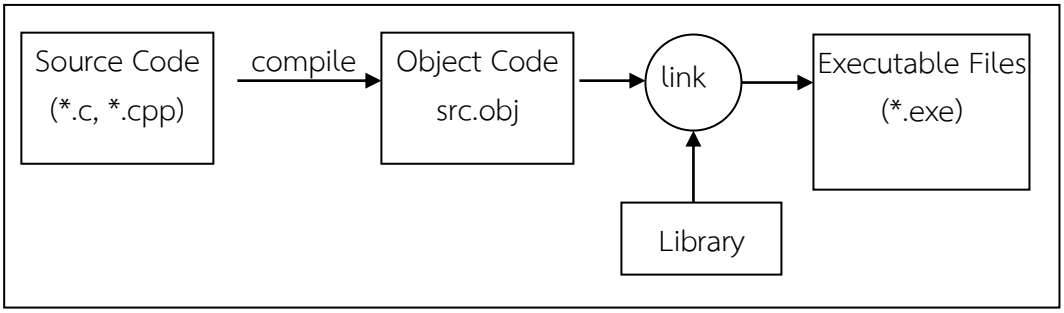
1. แปลโปรแกรม

แปลโปรแกรม เป็นขั้นตอนการแปลรหัสต้นฉบับให้กลายเป็นรหัสจุดหมาย ซึ่งประกอบไปด้วยรายละเอียดของการใช้งานหน่วยความจำ เช่น การประกาศตัวแปร การประกาศฟังก์ชันย่อย

เป็นต้น ในขั้นนี้จะมีการตรวจสอบความถูกต้องของไวยกรณ์ด้วย แต่ไม่ได้หมายความว่าโปรแกรมที่ผ่านการแปลแล้วจะต้องเป็นโปรแกรมที่สามารถทำงานได้

2. เชื่อมโยง

เชื่อมโยง เป็นการนำรหัสจุดหมาย ที่ได้มาทำการรวมกับชุดคำสั่งในคลัง (Library) ซึ่งได้รับการพัฒนามาล่วงหน้า พร้อมกับตัวแปลภาษาซี เพื่อให้ได้เป็นโปรแกรมที่สามารถใช้งานได้จริง เรียกขั้นตอนที่ 2 นี้ว่าเชื่อมโยง



ภาพที่ 3.1 ขั้นตอนการพัฒนาโปรแกรมด้วยภาษาซี
ที่มา: อรพิน ประวัติบริสุทธ์. 2554: 14

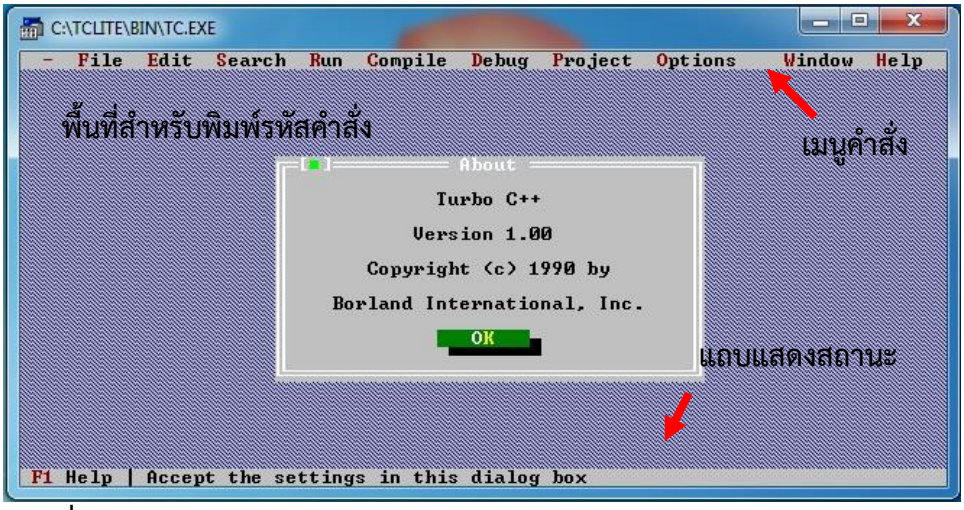
การใช้งานตัวแปลภาษาซี

ตัวแปลภาษาซี มีหลากหลายบริษัทที่พัฒนาตัวแปลภาษาซี เพื่อความสะดวกและง่ายกับผู้เริ่มต้นเขียนโปรแกรม แนะนำให้ใช้ Turbo C++ พัฒนาโดยบริษัท Borland International, Inc. ตั้งแต่เดือน พฤษภาคม พ.ศ. 2533 (1990) จนถึงเดือนกันยายน 2549 (2006) สามารถทำงานบนระบบปฏิบัติการไมโครซอฟต์ได้ เพื่อให้ง่ายและสะดวกในบทนี้อ้างอิง Turbo C++ Version 1.00 สามารถดาวน์โหลดได้ที่ http://www.napatsarun.com/c/TCLITE.rar สำหรับเวอร์ชันนี้สามารถคัดลอกโฟลเดอร์ TCLITE วางไว้ที่ C:/ ประมวลผลที่แฟ้ม C:\TCLITE\BIN\TC.EXE จะได้ผลลัพธ์ดังภาพที่ 3.2 ซึ่งจะกล่าวในรายละเอียดต่อไป

การใช้งานตัวแปลภาษาซี ควรรู้ส่วนประกอบของโปรแกรม หลักการแปลภาษาเบื้องต้น การสร้างแฟ้มข้อมูลใหม่ การบันทึกแฟ้มข้อมูล การเปิดแฟ้มข้อมูลเก่า การแปลรหัสต้นฉบับ การประมวลผล

1. หน้าต่างของตัวแปลภาษาซี

เมื่อเรียกใช้งานของหน้าต่างของตัวแปลภาษาซี แล้วจะพบหน้าจอดังภาพที่ 3.2



ภาพที่ 3.2 หน้าต่างของคอมไพเลอร์
ที่มา: Borland International, Inc. 1990.

จาก ภาพที่ 3.2 จะเห็นว่าประกอบด้วย 3 ส่วนหลัก ๆ คือ

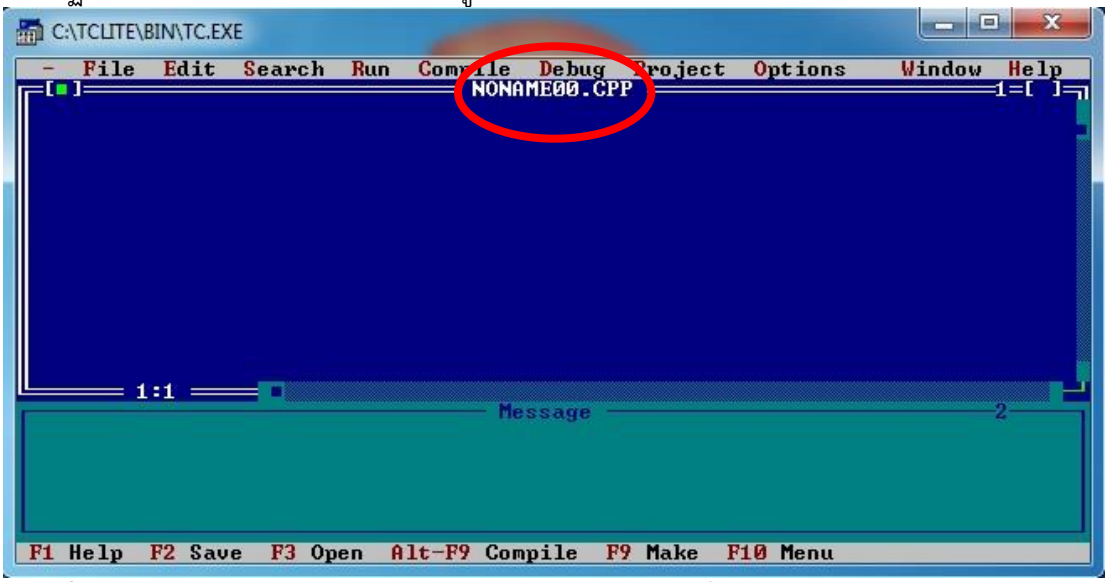
1.1 เมนูคำสั่ง เป็นส่วนที่อยู่บนสุดของโปรแกรม ทำหน้าที่เรียกใช้คำสั่งต่างๆ เช่น คำสั่งเกี่ยวกับแฟ้ม (File) การแก้ไข (Edit) การค้นหา (Search) เป็นต้น เมนูคำสั่งใดที่ถูกเลือกจะแสดงด้วยแถบสีเขียว

1.2 ส่วนสำหรับพิมพ์รหัสคำสั่งของโปรแกรมภาษาซี จากรูปจะเห็นเป็นแถบสีฟ้าซึ่งไม่สามารถพิมพ์ได้ในขณะนี้

1.3 ส่วนแสดงสถานะใช้บอกรายละเอียดของคำสั่งปัจจุบัน ซึ่งในบางกรณีสามารถใช้ในการบอกคำสั่งย่อยของคำสั่งหลักนั้นๆ

2. การพิมพ์รหัสคำสั่งของโปรแกรม

การพิมพ์รหัสคำสั่งนั้นทำได้โดยการใช้เมาส์ ไปคลิกที่เมนูไฟล์ (File) แล้วคลิกที่ New ตามลำดับซึ่งจะปรากฏหน้าจอดังภาพที่ 3.3 โดยส่วนที่เป็นพื้นที่สีน้ำเงินภายใต้กรอบสีขาว จะปรากฏตัวชี้ตำแหน่ง (Cursor) กระพริบอยู่ ซึ่งแสดงให้เห็นว่าสามารถพิมพ์รหัสคำสั่งได้แล้ว



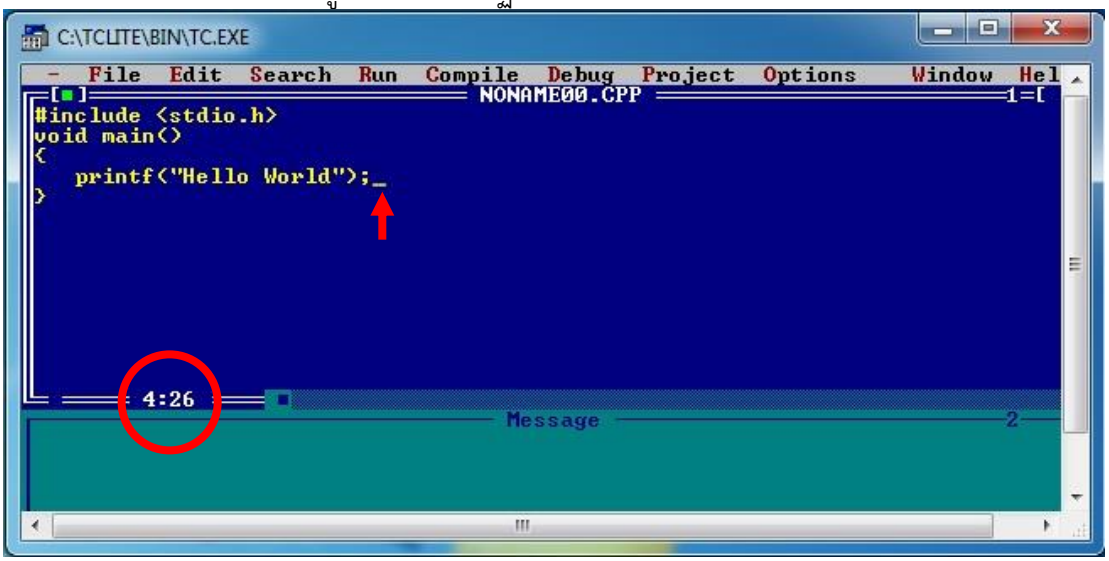
ภาพที่ 3.3 หน้าต่างของตัวแปรภาษาซี สำหรับการพิมพ์รหัสคำสั่ง

ตอนบนจะเห็นข้อความ NONAME00.C ในพื้นที่ส่วนที่วงกลม ซึ่งแสดงว่ารหัสคำสั่งที่กำลังพิมพ์อยู่ยังไม่ได้รับการบันทึก ซึ่งหากบันทึกแล้วข้อความนี้จะเปลี่ยนเป็นชื่อแฟ้มที่บันทึก

ให้ทดลองพิมพ์รหัสคำสั่งดังต่อไปนี้ ลงไปในพื้นที่สำหรับการพิมพ์รหัสคำสั่ง ให้ระวังตัวพิมพ์เล็กใหญ่ด้วย

```
#include <stdio.h>
void main()
{
    printf(“Hello World”);
}
```

ซึ่งหากนักศึกษาพิมพ์ได้อย่างถูกต้องจะปรากฏหน้าจอดังภาพที่ 3.4

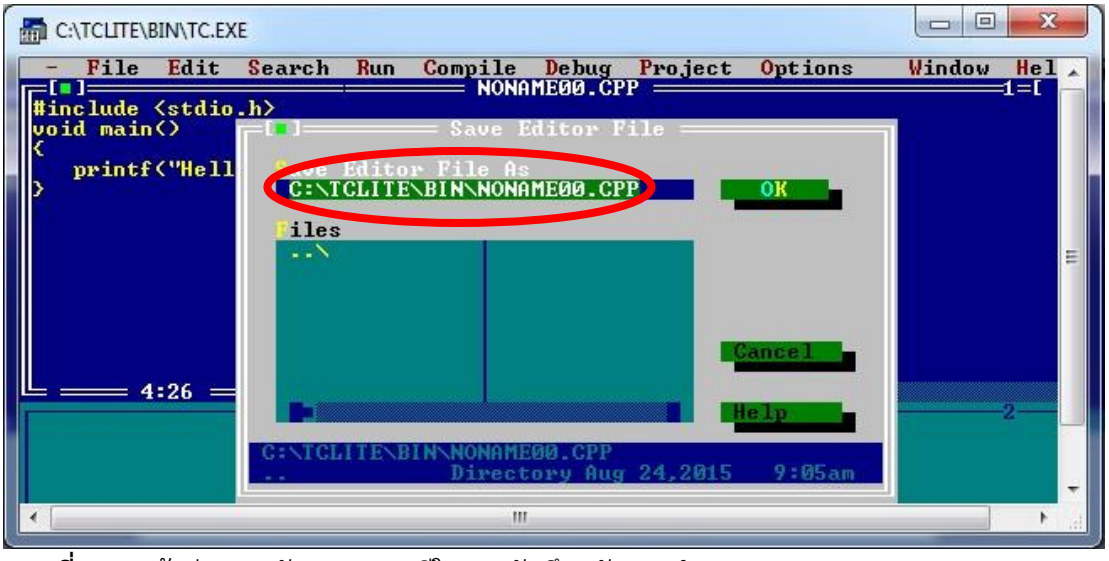


ภาพที่ 3.4 จอภาพของตัวแปลภาษาซี หลังจากพิมพ์รหัสคำสั่ง

ให้นักศึกษาสังเกตที่บริเวณด้านล่างของจอภาพ (ในวงกลม) จะพบว่ามีตัวเลข 4:26 ซึ่งบอกถึงตัวชี้ตำแหน่งในปัจจุบัน ว่ากำลังอยู่ที่บรรทัดที่ 4 ในตัวอักษรที่ 26 ซึ่งหมายเลขบรรทัดจะช่วยให้การตรวจสอบข้อผิดพลาดของโปรแกรมง่ายขึ้น

3. การบันทึกรหัสโปรแกรม

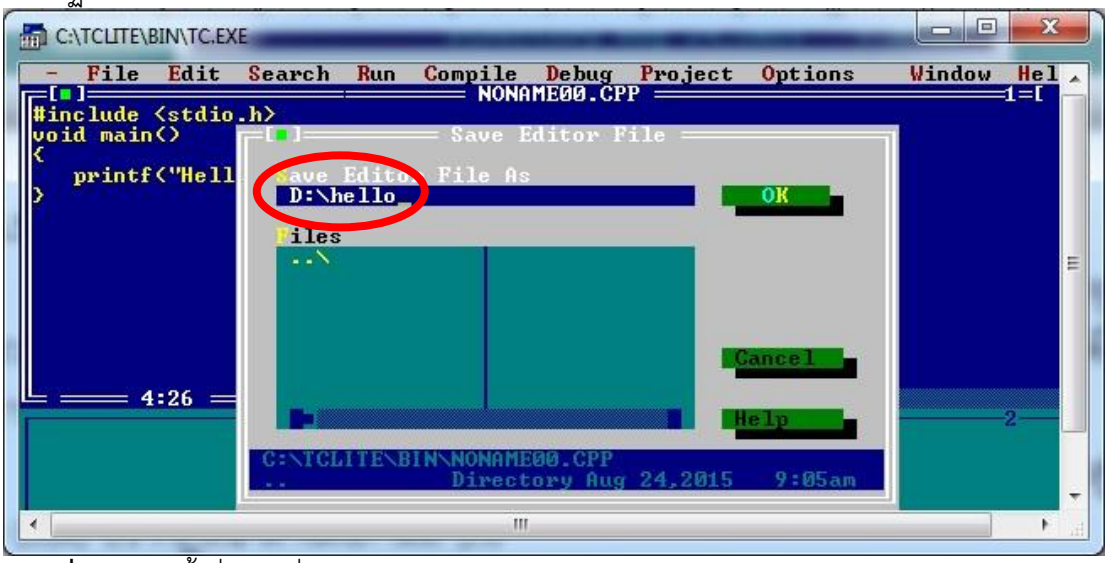
เมื่อพิมพ์รหัสคำสั่งเสร็จแล้ว ต้องทำการบันทึกไว้ทุกครั้งที่มีการแก้ไข วิธีการบันทึกรหัสคำสั่งทำได้โดยการใช้เมาส์คลิกที่เมนูไฟล์ (File) และบันทึก (Save) ตามลำดับ หรืออาจกดปุ่ม F2 บนแผงแป้นอักขระ ซึ่งจะปรากฏหน้าต่างดังภาพที่ 3.4



ภาพที่ 3.5 หน้าต่างของตัวแปลภาษาซีในการบันทึกรหัสการทำงาน

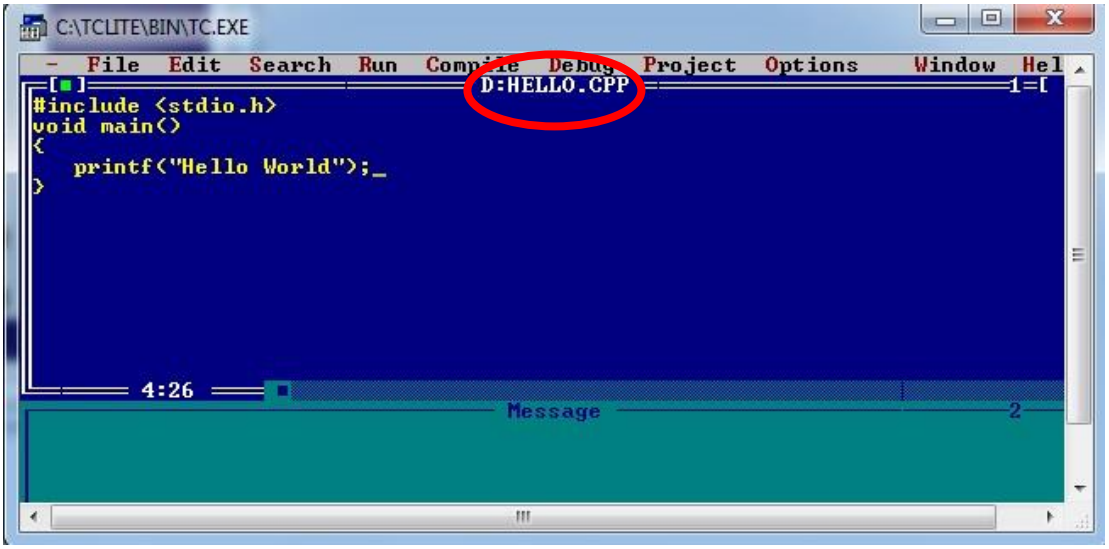
การบันทึกนี้ตัวแปรภาษาซีจะบันทึกใน Folder เดียวกับที่แสดงอยู่ในปัจจุบันด้วยแถบสีเขียวซึ่งวงกลมไว้ โดยหากต้องการบันทึกใน Folder นี้ก็ให้พิมพ์ชื่อของรหัสโปรแกรมซึ่งต้องเป็นไปตามกฎการตั้งชื่อแฟ้มในบทที่ 1 หัวข้อ 5

ในตัวอย่างนี้ให้นักศึกษาตั้งชื่อ File ว่า “D:\hello” เพื่อบันทึกข้อมูลลงสู่ Drive D ปรากฏหน้าต่างดังภาพที่ 3.6



ภาพที่ 3.6 การตั้งชื่อไฟล์ซึ่งบรรจุรหัสโปรแกรม

คลิกที่ปุ่ม OK หน้าต่างของการพิมพ์รหัสคำสั่งจะเปลี่ยนไปดังภาพที่ 3.7



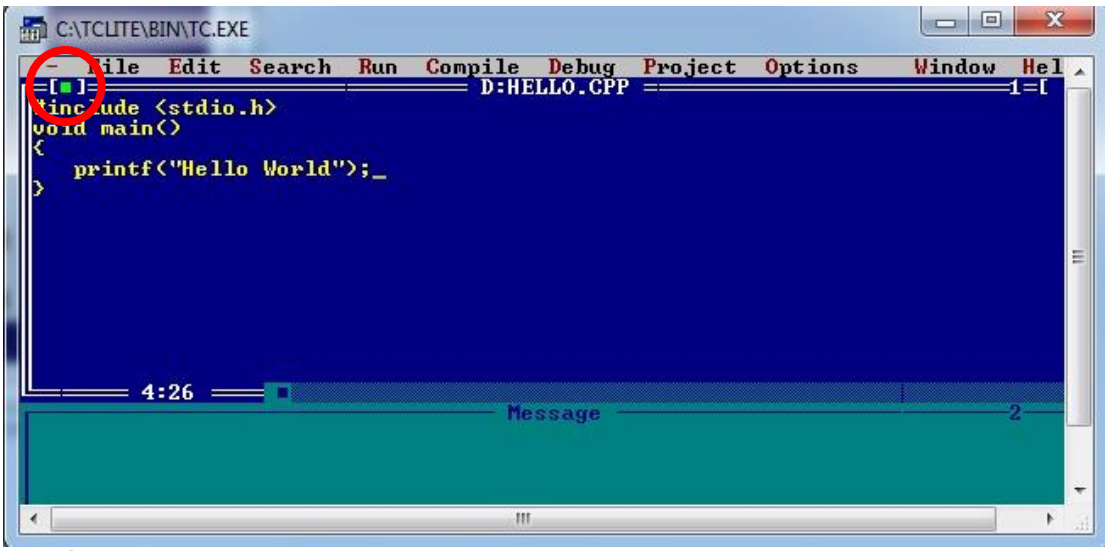
ภาพที่ 3.7 หน้าต่างของตัวแปลภาษาซีหลังจากมีการบันทึกรหัสคำสั่ง

จากภาพที่ 3.7 จะเห็นได้ว่าส่วนที่บอกชื่อของแฟ้มจะเปลี่ยนจาก NONAME00.CPP ในภาพที่ 3.3 เป็น D:\HELLO.CPP สำหรับชื่อแฟ้มจะได้รับการต่อส่วนขยาย (File Extension) เป็น. CPP โดยอัตโนมัติ

แต่หากนักศึกษาบันทึกโดยไม่ระบุโฟลเดอร์ ที่ต้องการจะถือว่าเป็นการบันทึกในโฟลเดอร์ปัจจุบันแสดงไว้ในภาพที่ 3.5 คือ C:\TCLITE\BIN\ ดังนั้นจึงควรระวังด้วยว่าบันทึกไว้ที่โฟลเดอร์ใด

4. การเปิดรหัสการทำงานที่พัฒนาไว้แล้ว

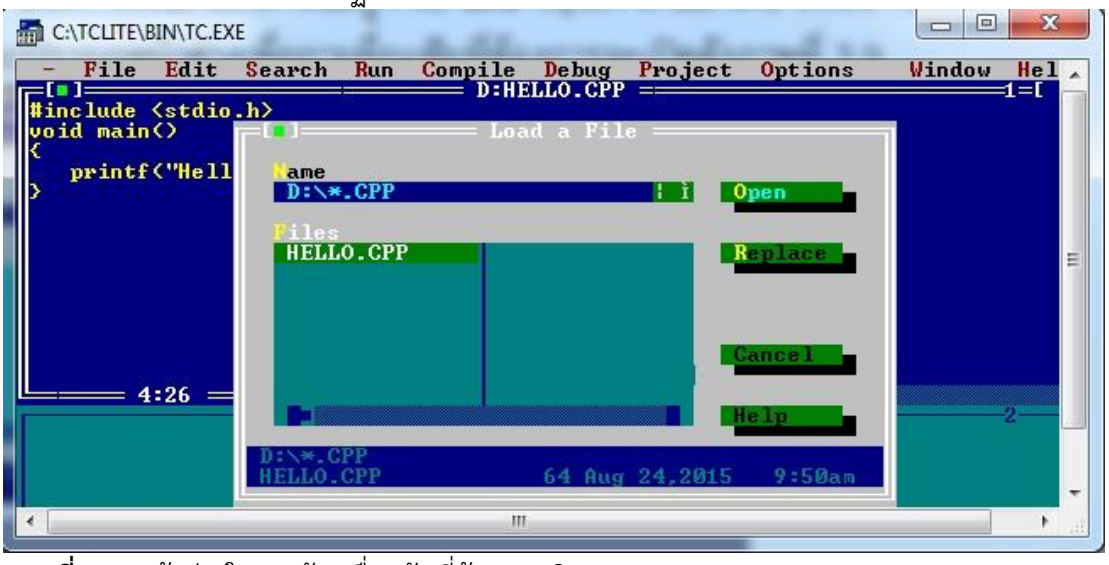
การปิดไฟล์ที่เปิดอยู่ สามารถทำการปิดแฟ้ม โดยคลิกที่ปุ่มสีเขียวมุมบนซ้ายมือของหน้าต่าง บริเวณที่วงกลม ในภาพที่ 3.8



ภาพที่ 3.8 ปุ่มปิดหน้าต่างปัจจุบัน

เมื่อปิดแล้วจะปรากฏหน้าต่างดังภาพที่ 3.2 อีกครั้งหนึ่ง แต่หากมีหน้าต่างอื่นๆ ซ้อนอยู่จะต้องปิดให้หมดเสียก่อนจึงจะปรากฏหน้าที่ในภาพที่ 3.2

ในกรณีที่ต้องการเปิดรหัสโปรแกรมบันทึกไว้แล้วมาแก้ไข หรือดำเนินการอย่างไรต่อไป จะต้องทำการเปิดแฟ้ม โดยการใช้เมาส์คลิกที่เมนู File แล้วคลิกที่ Open ตามลำดับ หรือกดปุ่ม F3 บนแผงแป้นอักขระ ซึ่งจะปรากฏหน้าต่างเพื่อถามชื่อแฟ้มที่ต้องการจะเปิดดังภาพที่ 3.9

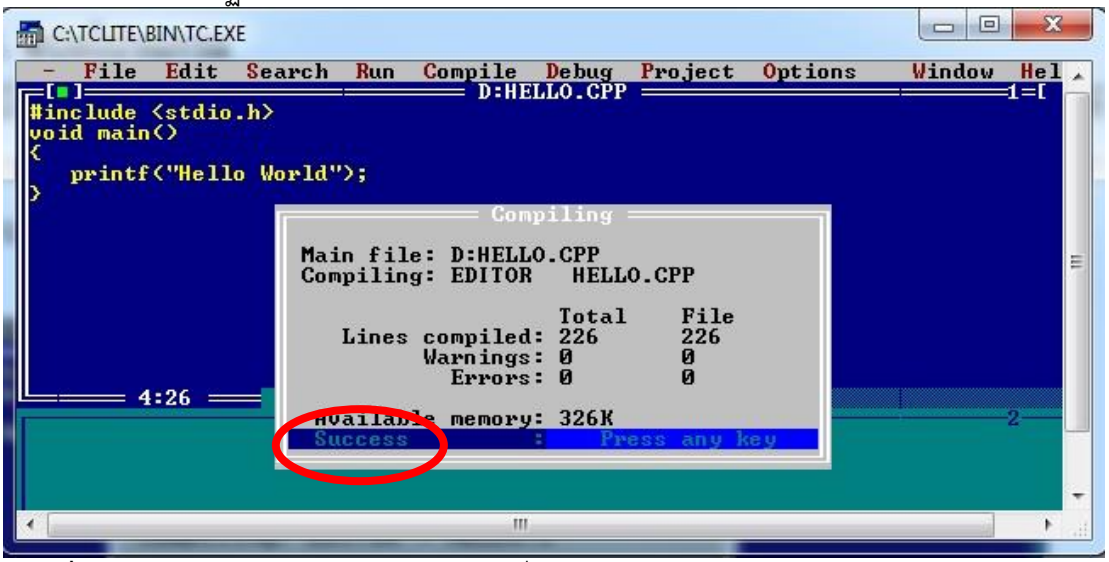


ภาพที่ 3.9 หน้าต่างในการป้อนชื่อแฟ้มที่ต้องการเปิด

ให้สังเกตว่าหน้าต่างนี้จะคล้ายกับหน้าต่างในภาพที่ 3.6 ซึ่งเป็นหน้าต่างที่ใช้ในการบันทึก แต่ถ้าเป็นการเปิดข้อมูลที่มีอยู่แล้วจะใช้ชื่อเป็น “Load a File” และในส่วนของ Name จะเป็น *.CPP ส่วนในช่อง Files จะแสดงชื่อแฟ้มทั้งหมดให้เลือกซึ่งสามารถเลือกได้โดย Double Click ที่ชื่อแฟ้มที่ต้องการ ต่อไปลองพิมพ์ชื่อแฟ้มเป็น D:\HELLO.CPP เพื่อเปิดแฟ้มที่บันทึกไว้มาใช้

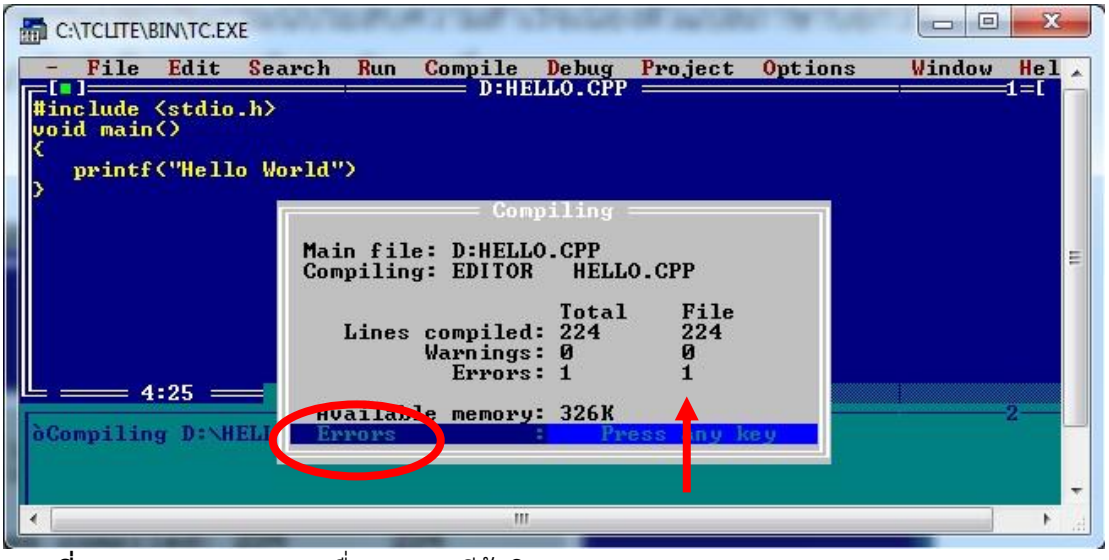
5. แปลโปรแกรม

เมื่อเปิดแฟ้มแล้วจะพบรหัสคำสั่งที่พิมพ์ไว้ หากต้องการที่จะให้ตัวแปลภาษาซี ทำการแปลรหัสต้นฉบับให้กดปุ่ม Alt ค้างไว้แล้วกดปุ่ม F9 (Alt + F9) เพื่อให้ตัวแปลภาษา ทำการแปลรหัสต้นฉบับ ซึ่งจะปรากฏหน้าต่างดังภาพที่ 3.10



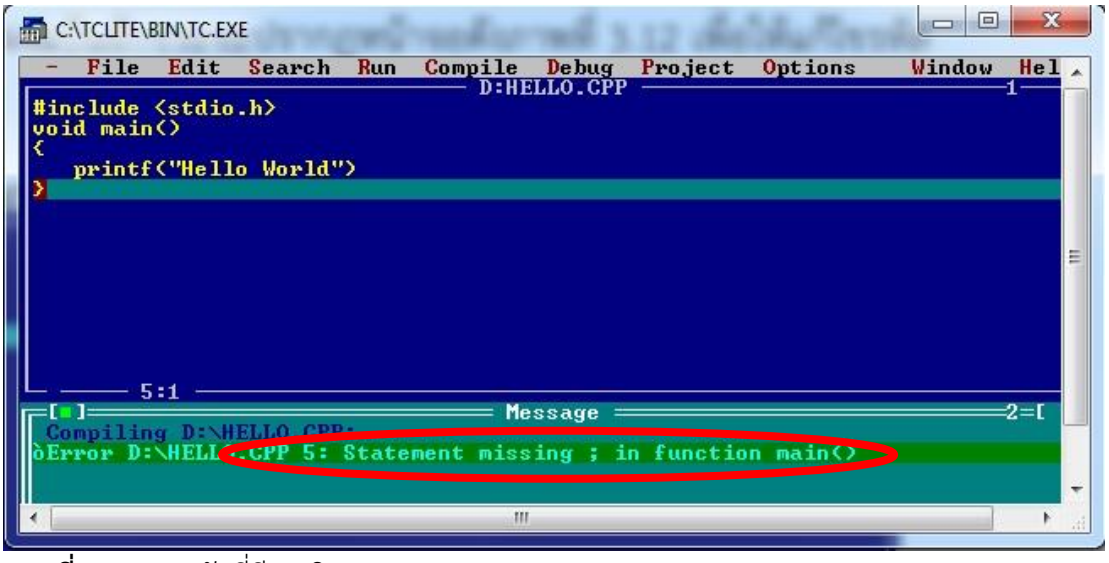
ภาพที่ 3.10 ผลจากการแปลรหัสต้นฉบับกรณีที่ไม่มีข้อผิดพลาด

จากภาพที่ 3.10 พบว่าการแปลรหัสต้นฉบับนี้ประสบความสำเร็จเนื่องตัวแปลภาษาบอกว่า “Success” แต่ถ้ามีข้อผิดพลาดในรหัสคำสั่งจะพบหน้าต่างดังภาพที่ 3.11



ภาพที่ 3.11 ผลจากการแปลเมื่อการแปลมีข้อผิดพลาด

หากมีข้อผิดพลาดจะแสดงข้อความว่ามีข้อผิดพลาด “Errors” โดยจะระบุจำนวนข้อผิดพลาดไว้ว่ามีจำนวน 1 ตำแหน่ง สังเกตได้จากจำนวนในตำแหน่งที่ลูกศรชี้อยู่ ซึ่งในกรณีที่มีข้อผิดพลาดหากกดปุ่มใดๆ บนแผงแป้นอักขระจะปรากฏหน้าจอดังภาพที่ 3.12 เพื่อให้แก้ไขรหัสคำสั่งในตำแหน่งที่ผิดพลาด

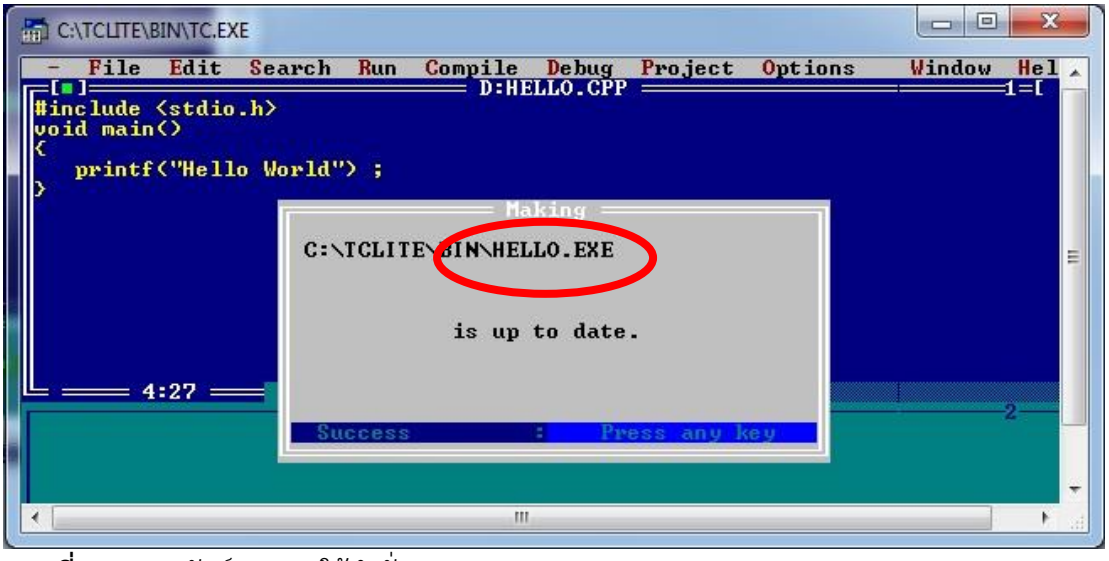


ภาพที่ 3.12 บรรทัดที่มีการผิดพลาด

จากภาพที่ 3.12 ตัวแปลภาษาซีรายงานว่ามีการผิดพลาด ไม่มีเครื่องหมายอัฒภาค ; ในฟังก์ชันหลัก main() ให้แก้ไขให้ถูกต้องแล้วจึงแปลรหัสต้นฉบับใหม่

6. เชื่อมโยง (link)

เมื่อทำการแปลรหัสต้นฉบับ รหัสการทำงานสำเร็จแล้ว จะพบว่ายังไม่สามารถที่จะใช้งานรหัสการทำงานได้เพราะยังไม่เกิดแฟ้ม HELLO.EXE ต้องทำการ make ซึ่งเป็นการเชื่อมโยง แฟ้มที่เป็น รหัสจุดหมายเข้ากับรหัสคลังเสียก่อน ซึ่งทำได้โดยการกดปุ่ม F9 ซึ่งจะพบหน้าต่างดังภาพที่ 3.13



ภาพที่ 3.13 ผลลัพธ์จากการใช้คำสั่ง

สังเกตว่าตัวแปลภาษา จะทำการสร้างแฟ้มมีชื่อว่า HELLO.EXE ในส่วนที่วงกลม ซึ่งจะมีชื่อเป็นชื่อเดียวกับแฟ้มรหัสการทำงานเสมอ

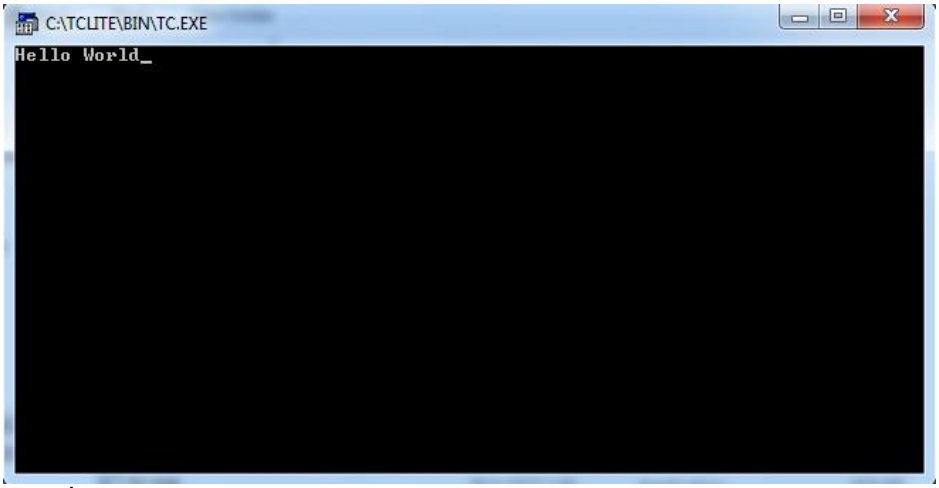
ถ้ามีข้อผิดพลาดขึ้นก็จะแสดงข้อผิดพลาดเช่นเดียวกับที่กล่าวมาแล้ว

7. การทำงาน

การทำงานทำได้โดยการใช้เมาส์คลิกที่เมนู Run แล้วคลิกที่ Run หรือกดปุ่ม Ctrl ค้างไว้แล้วกดปุ่ม F9 ซึ่งตัวแปลภาษา จะทำการแปลรหัสต้นฉบับและเชื่มโยงให้โดยอัตโนมัติ จากนั้นตัวแปรภาษาจะเรียกใช้แฟ้มรหัสการทำงานให้ด้วยซึ่งเมื่อทดสอบการทำงาน จะมองเห็นจอภาพเกิดการกระพริบเล็กน้อย จากนั้นจึงกลับสู่ตัวแปลภาษาทันที ซึ่งแสดงว่าโปรแกรมทำงานเสร็จแล้ว เนื่องจากทำการแสดงข้อความ “Hello World” บนหน้าจอเสร็จแล้วก็กลับเข้าสู่หน้าต่างของตัวแปลภาษา

8. การแสดงหน้าจอผลลัพธ์

ในกรณีที่ผลลัพธ์เร็วมากจนสังเกตไม่ทันอาจดูผลลัพธ์จากการทำงานของโปรแกรมสุดท้ายได้โดยการใช้กดปุ่ม Alt + F5 ซึ่งจะปรากฏดังภาพที่ 3.14



ภาพที่ 3.14 ผลลัพธ์จากการทำงานของโปรแกรมโดยการกด Alt + F5

หากไม่ต้องการกด Alt + F5 สามารถทำได้โดยการใส่คำสั่ง getch(); ไว้ภายหลังจากโปรแกรมทั้งหมด โดยต้องเพิ่ม #include <conio.h> ด้วย เพราะฟังก์ชัน getch() เป็นฟังก์ชันรับอักขระอยู่ในคลังโปรแกรม conio.h ซึ่งจะอธิบายในรายละเอียดในบทต่อไป

```
#include <stdio.h>
#include <conio.h>
void main()
{
    printf("Hello World");
    getch();
}
```

เมื่อโปรแกรมแสดงข้อความ Hello World บนหน้าจอแล้ว โปรแกรมจะรอรับอักขระอีก 1 อักขระจากฟังก์ชัน getch() ผู้ใช้จะเห็นข้อความที่แสดงบนหน้าจอ

สรุป

ภาษาซีเป็นภาษาคอมพิวเตอร์ที่พัฒนาขึ้นในปี ค.ศ. 1972 โดยเดนนิส ริทชี่ โดยพัฒนามาจากภาษาบี ในเบื้องต้นภาษาซีได้นำมาใช้ในการเขียนโปรแกรมบนระบบปฏิบัติการยูนิกซ์ ต่อมาได้พัฒนาจนสามารถใช้ได้ในทุกระบบปฏิบัติการ จากนั้นองค์กรมาตรฐานแห่งชาติสหรัฐอเมริกาได้ทำข้อตกลงที่เป็นมาตรฐานเดียวกันขึ้นของภาษาซีในปี ค.ศ. 1983 เรียกว่า มาตรฐาน ANSI C

ภาษาซีสามารถพัฒนาได้ในทุกบนระบบปฏิบัติ สามารถพัฒนาได้ตั้งแต่เครื่องคอมพิวเตอร์ส่วนบุคคล มินิคอมพิวเตอร์ เมนเฟรมและซุปเปอร์คอมพิวเตอร์ ภาษาซีมีหลักการเขียนโปรแกรมในลักษณะโครงสร้างเหมาะสำหรับผู้เริ่มต้นเขียนโปรแกรม มีฟังก์ชันมาตรฐานรองรับการคำนวณด้านวิทยาศาสตร์ มีตัวแปรที่รองรับการคำนวณสำหรับงานด้านคณิตศาสตร์

ภาษาซีเป็นภาษาที่มีการพัฒนาเป็นมาตรฐาน มีประสิทธิภาพสูง มีความยืดหยุ่น ภาษาซีจัดเป็นภาษาระดับสูงจำเป็นต้องใช้ตัวแปลภาษาแบบคอมไพเลอร์เพื่อแปลจากรหัสต้นฉบับเป็นรหัสจุดหมายเพื่อนำมารวมกับรหัสคลังใช้ในการประมวลผล

การใช้งานตัวแปลภาษาซี มีขั้นตอนในการใช้งานโดยเริ่มจากการเปิดไฟล์ เขียนโปรแกรม บันทึกไฟล์ แปลภาษา และประมวลผลโปรแกรม แต่หากมีข้อผิดพลาดต้องแก้ข้อผิดพลาดก่อนจึงจะรันได้ โดยมีคีย์ลัดที่สำคัญดังนี้

F3	เปิดโปรแกรมที่มีอยู่แล้ว
F2	บันทึกไฟล์
Alt – F9	แปลภาษา
Ctrl – F9	ประมวลผลโปรแกรม
Alt – F5	ดูผลลัพธ์การประมวลผลโปรแกรม

แบบฝึกหัด

1. จงอธิบายประวัติความเป็นมาของภาษาซี
2. จงบอกลักษณะเด่นของภาษาซี
3. จงบอกขั้นตอนในการบันทึกไฟล์ของภาษาซี
4. จงบอกขั้นตอนในการแปลภาษา
5. จงบอกขั้นตอนในการประมวลผลโปรแกรม
6. จงบอกคีย์ลัดในการบันทึกไฟล์
7. จงบอกคีย์ลัดในการแปลภาษา
8. จงบอกคีย์ลัดในการประมวลผลโปรแกรม
9. จงบอกคีย์ลัดในการดูผลลัพธ์ของการรัน
10. จงบอกหลักการตั้งชื่อของไฟล์ภาษาซี

เอกสารอ้างอิง

ไกรศร ตั้งโอภากุล. (2554). *คู่มือเรียนเขียนโปรแกรมภาษา C.* นนทบุรี: ไอดีซีพรีเมียร์.

ประภาพร ช่างไม้. (2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงทพมหานคร: อินโฟเพรส.

อรพิน ประวัติบริสุทธ์. (2554). *คู่มือเรียนภาษา C ฉบับปรับปรุงใหม่.* กรุงทพมหานคร: โปรวิชั่น.

Borland International, Inc. (1990). *Turbo C++ Version 1.00.*

Brian W. Kernighan, Dennis M. Ritchie. (1988). *The C Programming Language*. 2nd Edition. London: Prentice Hall.

Stephen G. Kochan. (1994). *Programming in ANSI C revised Edition.* USA: SAMS Publishing.

บทที่ 4
การเขียนโปรแกรมด้วยภาษาซี

การเขียนโปรแกรมภาษาคอมพิวเตอร์ จำเป็นต้องศึกษาโครงสร้างของโปรแกรมภาษานั้นๆ ศึกษารูปแบบของคำสั่งต่าง ๆ ศึกษาชนิดของตัวแปรต่าง ๆ เพื่อเลือกใช้ให้ตรงกับลักษณะงานได้ถูกต้อง รวมถึงตัวดำเนินการประเภทต่าง ๆ เพื่อการคำนวณค่าที่ถูกต้อง

ในบทนี้อธิบายถึงโครงสร้างของโปรแกรมที่เขียนด้วยภาษาซี การเขียนหมายเหตุ การตั้งชื่อตัวแปรในการเขียนโปรแกรมด้วยภาษาซี ชนิดของข้อมูล การแปลงชนิดของข้อมูลที่เกิดขึ้นอัตโนมัติ การแปลงชนิดของข้อมูล และการใช้งานตัวแปรชนิดต่าง ๆ

โครงสร้างของโปรแกรมที่เขียนด้วยภาษาซี

ภาษาซีเป็นภาษาที่ประกอบด้วยฟังก์ชัน คำสั่งต่างๆ ที่ใช้ในภาษาซี ล้วนอยู่ในรูปของฟังก์ชันทั้งสิ้น โดยแต่ละฟังก์ชันจะทำหน้าที่เฉพาะอย่างเช่นฟังก์ชัน printf(.....) จะทำหน้าที่แสดงผล ฟังก์ชัน scanf(.....) จะทำหน้าที่ในการรับค่า ส่วนรายละเอียดของการทำงานจะต้องระบุในรูปแบบขององค์ประกอบ (Parameters) ภายในเครื่องหมาย (.....) ซึ่งรายละเอียดได้กล่าวต่อไป โดยในโปรแกรมหนึ่งๆ สำหรับภาษาซี ที่จะได้ศึกษากันนั้น จะต้องมีอย่างน้อย 1 ฟังก์ชัน นั่นคือ ฟังก์ชัน main() ซึ่งจะเป็นฟังก์ชันหลักในการทำงาน เมื่อโปรแกรมเริ่มงานจะต้องเริ่มทำงานที่ฟังก์ชันนี้ก่อนเสมอ

โครงสร้างการเขียนโปรแกรมด้วยภาษาซี สามารถแสดงได้ดังภาพที่ 4.1

```
//  This is my first program        ← 1. ส่วน comment
#include  <stdio.h>
#include <conio.h>                  } 2. ส่วน Preprocessor
int i;
void main()              ← 3. ส่วนฟังก์ชันหลักในการเริ่มต้นการทำงานของโปรแกรม
{
        .....                       ← 4. ส่วน ชุดคำสั่ง
}
```

ภาพที่ 4.1 โครงสร้างของภาษาซี

โครงสร้างของภาษาซีนั้นแบ่งออกได้เป็น 4 ส่วน คือ ส่วนพรีโปรเซสเซอร์ไดเร็คทีฟ (Preprocessor Directive) ส่วนระบุรูปแบบของชนิดข้อมูลมาตรฐานของภาษาซี ชุดคำสั่งที่จะให้คอมพิวเตอร์ปฏิบัติตาม และหมายเหตุ (Comment)

1. ส่วนพรีโปรเซสเซอร์ไดเร็คทีฟ

ส่วนพรีโปรเซสเซอร์ไดเร็คทีฟ เป็นส่วนที่ใช้กำหนดค่าในการแปลภาษา (Compile) ว่าต้องการให้นำเอาฟังก์ชัน

#include	#define	#error	#if	#endif
#elif	#else	#ifdef	#ifndef	#undef
#line	#pragma			

ที่มา: Stephen G. Kochan. 1994

หลักที่มีอยู่แล้วของภาษา ซึ่งมีให้พร้อมกับภาษาซีฟังก์ชันใดมาใช้ เช่น รูปแบบของการเขียนส่วนพรีโปรเซสเซอร์ไดเร็คทีฟ จะต้องขึ้นต้นด้วยเครื่องหมายสี่เหลี่ยม # แต่ไม่ต้องลงท้ายด้วยเครื่องหมายอัฒภาค ;

ส่วนเบื้องต้นที่จำเป็นต้องใช้คือ #include โดยกำหนดในรูปของชื่อไฟล์ที่จะให้นำมารวมเข้าด้วยกันเรียกว่าไฟล์นี้ว่า คลังโปรแกรมซึ่งจะมีส่วนขยายของไฟล์เป็น.h เช่น stdio.h, conio.h เป็นต้น โดยการใช้คำสั่ง #include โดยคลังโปรแกรมนั้นจะมีตัวอย่างดังตารางที่ 4.1

ตารางที่ 4.1 ตัวอย่างของไฟล์ส่วนหัวในภาษาซี

ชื่อไฟล์ส่วนหัว	ลักษณะการทำงานของฟังก์ชัน
alloc.h	การจัดการหน่วยความจำ การจองพื้นที่ การคืนพื้นที่ในหน่วยความจำ
bios.h	การเรียกใช้ฟังก์ชันของ IBM-PC ROM BIOS.
complex.h	การคำนวณทางคณิตศาสตร์เกี่ยวกับจำนวนเชิงซ้อน
conio.h	การเรียกใช้ฟังก์ชันของ console และ MSDOS.
ctype.h	การจัดการเกี่ยวกับตัวอักษร
dir.h	การจัดการเกี่ยวกับการเข้าถึง File และ Directory Structure
dDos.h	การกำหนดค่าที่จำเป็นในการเรียกใช้ใน DOS
ilo.h	การกำหนดฟังก์ชันในการ Input และ Output ระดับต่ำ
math.h	การกำหนดฟังก์ชันในการคำนวณทางคณิตศาสตร์ทั่วๆ ไป
stdio.h	การกำหนดฟังก์ชันในการ Input และ Output โดยทั่วๆ ไป
stdlib.h	การกำหนดฟังก์ชันในการใช้งานทั่วไป เช่นการเรียงลำดับข้อมูล
string.h	การกำหนดฟังก์ชันในการใช้งานชุดของอักขระ (String)
time.h	การกำหนดฟังก์ชันในการจัดการเกี่ยวกับเวลา เช่นการเปลี่ยนรูปแบบของเวลา

2. ส่วนระบุรูปแบบของชนิดข้อมูลมาตรฐานของภาษาซี

ในภาษาซี นั้นมีการกำหนดชนิดของข้อมูลเอาไว้ให้เลือกใช้อยู่หลายชนิด ด้วยกันดังตารางที่ 4.2

ตารางที่ 4.2 ตัวอย่างชนิดของข้อมูลที่ถูกกำหนดไว้แล้วในภาษาซี

ชนิดตัวแปร	ความหมาย	ค่าที่เป็นไปได้	ตัวอย่าง
integer	เลขจำนวนเต็ม (16 bits)	-32768 ถึง 32767	int i;
unsigned int	จำนวนเต็มไม่คิดเครื่องหมาย (16 บิต)	0 ถึง 65,535	unsigned int ui;
long	จำนวนเต็มยาว (32 bits)	-2,147,483,648 ถึง 2,147,483,647	long l;
unsigned long	จำนวนเต็มยาวไม่คิดเครื่องหมาย (32 bits)	0 ถึง 4,294,967,295	unsigned long ul;
float	เลขทศนิยม (32 bits)	1.2×10^{-38} ถึง $3.4 \times 10^{+38}$	float f;
double	เลขทศนิยมยาว 2 เท่า (64 bits)	2.3×10^{-308} ถึง $1.7 \times 10^{+308}$	double d;
long double	เลขทศนิยมยาวมาก (80 bits)	3.4×10^{-4932} ถึง $1.1 \times 10^{+4932}$	long double ld;
char	อักขระ (ASCII) 1 ตัว (8 bits)	-128 ถึง 127	char ch;
unsigned char	อักขระไม่คิดเครื่องหมาย	0 ถึง 255	unsigned char uch;

ที่มา: อรพิน ประวัติบริสุทธิ์. 2554: 58-59

หมายเหตุ ขนาดของข้อมูลแต่ละประเภทอาจเปลี่ยนแปลงได้ขึ้นอยู่กับตัวแปลภาษาซีที่ใช้ และระบบปฏิบัติการ

3. ชุดคำสั่งที่จะให้คอมพิวเตอร์ปฏิบัติตาม

ชุดคำสั่งที่จะให้คอมพิวเตอร์ปฏิบัติตามส่วนนี้จะถูกเขียนไว้ในระหว่างกลางของวงเล็บปีกกาเปิด และวงเล็บปีกกาปิด ซึ่งอยู่ภายหลังชื่อฟังก์ชัน โดยในภาษาซี นั้นได้ถูกกำหนดเฉพาะว่าให้ทุกๆ โปรแกรมที่เขียนขึ้นเริ่มต้นการทำงานจากฟังก์ชันหลักคือ main() เสมอ ดังนั้นอย่างน้อยที่สุดในการเขียนโปรแกรมด้วยภาษาซีจะต้องมีการกำหนดฟังก์ชันจำนวน 1 ฟังก์ชัน คือ main()

4. หมายเหตุ

นอกจากชุดคำสั่งที่เป็นคำสั่งในการสั่งให้คอมพิวเตอร์ทำงานแล้ว ในภาษาซี อนุญาตให้ผู้ใช้ทำการเขียนหมายเหตุไว้เพื่อกันลืมได้อีกด้วย โดยเขียนหมายเหตุนั้นแบ่งได้เป็น 2 ลักษณะคือ

4.1 การเขียนหมายเหตุแบบบรรทัดเดียว ใช้เครื่องหมาย // เขียนไว้หน้าข้อความ ซึ่งทำให้ข้อความหลังเครื่องหมาย // เป็นข้อความหมายเหตุทันที แต่ข้อความหน้าเครื่องหมาย // จะยังคงเป็นคำสั่งตามปกติ เช่น

```
int   i;   // Declare Variable “i” to an integer type data
```

ซึ่งคำสั่งดังกล่าวทำหน้าเกิดความหมายดังนี้

int i; เป็นการประกาศตัวแปร i

Declare Variable “i” to an integer type data เป็นหมายเหตุ

4.2 การเขียนหมายเหตุหลายบรรทัด ใช้เครื่องหมาย /* เริ่มต้นในการเขียนหมายเหตุ และ */ ในการจบการเขียนหมายเหตุ ซึ่งข้อความระหว่างเครื่องหมาย /* และ */ ดังกล่าวยาวกี่บรรทัดก็ตาม ตัวแปลภาษาซีจะไม่แปลความหมายข้อความดังกล่าว เช่น

```
/*     This is my First  C Program:
       My name is  Micheal                */
```

ข้อควรระวังในการเขียนโปรแกรมด้วยภาษาซี

1. คำสั่งแต่ละคำสั่งจะต้องจบด้วยเครื่องหมายอัฒภาค ; (semicolon) ตัวแปลของภาษาซี ถือว่าจบคำสั่งเมื่อเจอเครื่องหมายอัฒภาค ; เท่านั้น ยกเว้นในส่วนของ Preprocessor ซึ่งถือว่าการขึ้นบรรทัดใหม่เป็นการสิ้นสุดคำสั่ง

2. ภาษาซี จะแยกความแตกต่างระหว่างอักษรพิมพ์ตัวใหญ่ (A-Z) และอักษรตัวพิมพ์เล็ก (a-z) ดังนั้นอักษรทั้งสองตัวจะแตกต่างกัน เช่น MyType กับ mytype แล้ว Compiler ของภาษาซี จะแยกว่าเป็นตัวแปรคนละตัวกัน ซึ่งอาจทำให้โปรแกรมผิดพลาดได้

3. ตัวแปรต้องมีการประกาศตัวแปรก่อนใช้งานทุกครั้ง ในการเขียนโปรแกรมภาษาซีนั้นจะต้องมีการประกาศตัวแปรไว้ก่อนทุกครั้ง เพื่อให้ตัวแปลภาษาทำการจองพื้นที่ในหน่วยความจำ ให้เพียงพอตามที่ต้องการกับความต้องการในแต่ละฟังก์ชัน ในโปรแกรมนั้นๆ

การตั้งชื่อตัวแปรในภาษาซี

การตั้งชื่อตัวแปรในการเขียนโปรแกรมด้วยภาษาซี นั้น มีกฎเกณฑ์ดังนี้

1. ชื่อต้องขึ้นต้นด้วยอักษรภาษาอังกฤษ (A-Z, a-z) หรือเครื่องหมาย _ (ขีดเส้นใต้)

2. ห้ามขึ้นต้นชื่อด้วยตัวเลข

3. ภายในชื่อห้ามเว้นวรรค แต่อาจใช้เครื่องหมาย _ (underscore) ช่วยให้ดูง่ายได้ แต่ห้ามใช้เครื่องหมาย – (dash)

4. ตัวอักษรถัดจากตัวแรกของชื่ออาจเป็นตัวอักษร, ตัวเลข, หรือเครื่องหมาย _ ยกเว้นเครื่องหมายพิเศษในการดำเนินการทางคณิตศาสตร์ (เช่น $,#,+,-,*,/,%, !, | , ^, @, =, \ )

5. การเปรียบเทียบความแตกต่างของชื่อจะใช้หลักเกณฑ์ 2 ข้อ คือ

   1) เปรียบเทียบเฉพาะอักขระ 32 ตัวอักษรแรกของชื่อตัวแปรหรือชื่อฟังก์ชัน

   2) อักษรตัวพิมพ์เล็กและพิมพ์ใหญ่ในภาษาซีมีความแตกต่างกัน

6. ชื่อต้องไม่ตรงกับคำสงวนในภาษาซี ไม่ว่าจะเป็นชนิดของตัวแปร คำสั่ง ฟังก์ชัน ดังต่อไปนี้

Auto	break	case	char	const
continue	default	do	Double	Else
enum	extern	float	for	goto
if	int	long	Register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

ที่มา: Stephen G. Kochan. 1994: 35

ชนิดของตัวแปรในภาษาซี

ภาษาซีมีตัวแปรให้เลือกใช้มากมายหลากหลายชนิด เช่น ตัวแปรชนิดเลขจำนวนเต็ม, เลขทศนิยม หรืออื่นๆ โดยตัวแปรที่นิยมใช้กันมีดังแสดงอยู่ดังตัวอย่างในตารางที่ 4.2 โดยในการเลือกใช้ตัวแปรนั้นต้องเลือกใช้ให้เหมาะสมกับการใช้งานด้วย

การประกาศตัวแปรในภาษาซี นั้นสามารถทำได้โดยการประกาศในรูปแบบ

รูปแบบ : ชนิดตัวแปร ชื่อตัวแปร;

โดย ชนิดตัวแปรเป็นชนิดข้อมูลที่จะเก็บในตัวแปรนั้นๆ ซึ่งดูได้จากตารางที่ 4.2 ในคอลัมน์แรกหากต้องการประกาศตัวแปรจำนวนหลายๆ ตัวให้มีชนิดเดียวกัน สามารถทำได้โดยการใส่เครื่องหมาย , (comma) ลงไปหลังชื่อตัวแปรตัวแรก ตามด้วยชื่อตัวแปรตัวถัดไปได้ทันที

เช่น int a;
float f1, f2, f3;

โดยในบรรทัดแรกเป็นการกำหนดให้ a เป็นตัวแปรชนิดเลขจำนวนเต็ม และในบรรทัดที่ 2 เป็นการกำหนดให้ ตัวแปร f1, f2, f3 เป็นเลขทศนิยม เป็นต้น

หากต้องการให้ตัวแปรที่ประกาศมีความยาวเพิ่มเติมมากขึ้นกว่าที่ภาษาซีได้กำหนดไว้ให้ นั่นคือ ตัวแปรชนิด int จะมีความยาวเท่ากับ 16 บิต, ตัวแปร float จะมีความยาว 32 บิต ก็สามารถทำได้โดยการใช้ Type modifier เช่น long ซึ่งจะช่วยให้ตัวแปรชนิดที่กำหนดมีความยาวเพิ่มขึ้นอีก (เป็น 2 เท่า เช่น long int จะมีความยาวเท่ากับ 2*16 = 32 บิต) เป็นต้น

หรือหากต้องการตัวแปรชนิดที่ไม่คิดเครื่องหมายก็ใช้ Type modifier เป็น unsigned ก็จะเป็นการกำหนดให้ภาษาซี ทำการคิดคำนวณเป็นแบบไม่คิดเครื่องหมาย แต่โดยปกติแล้วการคิดคำนวณของภาษาซีนั้นจะเป็นการคำนวณแบบคิดเครื่องหมาย

การประกาศตัวแปรในภาษาซี นั้นสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรได้โดยการประกาศในรูปแบบ

รูปแบบ : ชนิดตัวแปร ชื่อตัวแปร = ค่าเริ่มต้น ;

เช่น

```
int     a = 10 ;
float   f1 = 2.50 ;
char    grade = 'A';
char    name[10] = "dusit";
```

ตัวดำเนินการในภาษาซี

ในการสั่งให้คอมพิวเตอร์ดำเนินการกับข้อมูลที่รับเข้ามานั้นจะต้องใช้คำสั่งที่เรียกว่า "ตัวดำเนินการ (operators)" ซึ่งแบ่งได้เป็น 4 ประเภทดังนี้

1. ตัวกำหนดค่า

ตัวกำหนดค่า (Assignment Operator) ใช้ในการกำหนดค่าให้กับตัวแปร สัญลักษณ์ที่ใช้คือเครื่องหมาย เท่ากับ (=) ซึ่งจะกำหนดค่าด้านขวามือให้กับตัวแปรที่อยู่ด้านซ้ายมือเท่านั้น ซึ่งการกำหนดค่านี้จะทำให้ค่าของตัวแปรเท่ากับค่าที่กำหนดเพียงครั้งเดียวเท่านั้น ไม่ใช่เท่ากันตลอดทั้งโปรแกรม เช่น a=8; กำหนดให้ตัวแปร a มีค่าเท่ากับ 8

2. ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการทางคณิตศาสตร์ ใช้ในการคำนวณทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เป็นต้น ซึ่งตัวดำเนินการที่สำคัญดังนี้

ตารางที่ 4.3 การดำเนินการทางคณิตศาสตร์ระหว่างตัวแปร

ตัวแปรที่ 1	ตัวแปรที่ 2	การดำเนินการ	ผลลัพธ์
integer	integer	+	integer
integer	integer	-	integer
integer	integer	*	integer
integer	integer	/	integer
integer	integer	%	integer
integer	float	+	float
integer	float	-	float
integer	float	*	float
integer	float	/	float
float	Integer	/	float
float	float	+	float
float	float	-	float
float	float	*	float
float	float	/	float
float	float	%	None

ที่มา: Stephen G. Kochan. 1994:41

ตารางที่ 4.4 ตัวอย่างการดำเนินการทางคณิตศาสตร์

สัญลักษณ์	ตัวอย่าง	ความหมาย
+	5+6	หาผลลัพธ์ของ 5+6 เท่ากับ 11
-	5-6	หาผลลัพธ์ของ 5-6 เท่ากับ –1
*	5*6	หาผลลัพธ์ของ 5 * 6 เท่ากับ 30
/	5/6	หาผลลัพธ์ของ 5/6 เท่ากับ 0 (ไม่มีทศนิยมเพราะเป็นเลขจำนวนเต็ม
	5.0 / 6.0	หาผลลัพธ์ของ 5/6 เท่ากับ 0.83 (เป็นทศนิยมเพราะหารเลขทศนิยม
%	5 % 6	หาเศษของการหาร ระหว่าง 5 และ 6 (เท่ากับ 5) (Mod)

3. ตัวดำเนินการเพิ่มค่าและลดค่า

ตัวดำเนินการเพิ่มค่าและลดค่า ในการเพิ่มค่าและลดลงครั้งละ 1 นั้นภาษาซี ได้กำหนดไว้ให้โดยใช้สัญลักษณ์ดังนี้ ในตัวอย่างนี้สมมติว่าในตอนเริ่มต้นนั้นตัวแปร m มีค่า 4

ตารางที่ 4.5 ตัวดำเนินการเพิ่มค่าและลดค่า ตัวอย่างการใช้งานตามลำดับความสำคัญ

สัญลักษณ์	ตัวอย่าง	ค่าสุดท้าย	ความหมาย
++	X = m++;	X =4 m=5	กำหนดค่าก่อนแล้วจึงเพิ่มค่า
	X = ++m;	X =5 m=5	เพิ่มค่าก่อนแล้วจึงกำหนดค่า
--	X = m--;	X =4 m=3	กำหนดค่าก่อนแล้วจึงลดค่า
	X = --m;	X =3 m=3	ลดค่าก่อนแล้วจึงกำหนดค่า

ดังจะเห็นได้จากตารางที่ 4.5 ว่าการวางตัวดำเนินไว้ในตำแหน่งที่แตกต่างกันนั้นจะส่งผลให้ผลลัพธ์จากการดำเนินการแตกต่างกันไปด้วย ดังนั้นจึงไม่แนะนำให้เขียนคู่กับอย่างนี้ให้แยกเป็นสองคำสั่ง คือ m++ ; และ x = m ;

4. ตัวดำเนินการทางตรรกะ

ตัวดำเนินการทางตรรกะ ใช้ประโยชน์ในการสร้างเงื่อนไขต่างๆให้คอมพิวเตอร์ โดยตัวดำเนินการประเภทนี้จะให้ผลลัพธ์จากการดำเนินการเป็นค่าจริง (TRUE) หรือเท็จ (FALSE) เท่านั้น ในตัวอย่างนี้

สมมติว่าในตอนเริ่มต้นนั้นตัวแปร A มีค่าเป็น 4 และ B มีค่า 10

ตารางที่ 4.6 ตัวดำเนินการทางตรรกะ และตัวอย่างการใช้งาน

สัญลักษณ์	ตัวอย่าง	ผลลัพธ์	อธิบาย
<	A < 4	FALSE	A น้อยกว่า 4 หรือไม่ (เท็จ) เพราะ A มีค่าเท่ากับ 4
>	B > 4	TRUE	B มากกว่า 4 หรือไม่ (เท็จ) เพราะค่า B มีค่าเท่ากับ 10
<=	A <= 4	TRUE	A น้อยกว่า/เท่ากับ 4 (จริง) เพราะ A มีค่าเท่ากับ 4
>=	B >= 4	TRUE	B มากกว่า/เท่ากับ 4 (จริง) เพราะค่า B มีค่าเท่ากับ 10
==	A == 4	TRUE	A เท่ากับ 4 (จริง) เพราะ A มีค่าเท่ากับ 4
!=	A != 4	FALSE	A ไม่เท่ากับ 4 (ไม่จริง) เพราะ A มีค่าเท่ากับ 4
&&	(A<4) && (B>=4)	F && T = FALSE	ใช้เชื่อมเงื่อนไข และ (AND) เพราะ A มีค่าเท่ากับ 4 และค่า B มีค่าเท่ากับ 10
||	(A<4) || (B>=4)	F || T = TRUE	ใช้เชื่อมเงื่อนไข หรือ (OR) เพราะ A มีค่าเท่ากับ 4 และค่า B มีค่าเท่ากับ 10
!	!(A<4)	!F = TRUE	ใช้เชื่อมปฏิเสธ (ไม่) (NOT) เพราะ A มีค่าเท่ากับ 4

นอกจากนั้นภาษาซี ยังอนุญาตให้เขียนตัวดำเนินการหลายๆ ตัวไว้ในคำสั่งเดียวกันได้อีกด้วย เช่น A + B * 8 + 2 / 20 ซึ่งหากเขียนเช่นนี้จะเกิดปัญหาว่าตัวดำเนินการใดจะเกิดขึ้นก่อน ดังนั้นภาษาซี จึงกำหนดให้มีลำดับความสำคัญของตัวดำเนินการดังตารางที่ 4.7 เรียงตามลำดับความสำคัญสูงสุด

ตารางที่ 4.7 ลำดับความสำคัญของตัวดำเนินการในภาษาซี

ลำดับ	ตัวดำเนินการ	ความหมาย
1	( ), []	วงเล็บ
2	-	เครื่องหมายลบสำหรับจำนวนที่น้อยกว่า 0 (ค่าลบ)
3	++, --, !	เพิ่มค่า ลดค่า และ Logical NOT (!)
4	* / และ %	คูณ หารและ โมดูลัส (Mod)
5	+ และ -	บวก และลบ
6	<, <=, >, >=	น้อยกว่า น้อยกว่าหรือเท่ากับ มากกว่า และมากกว่าหรือเท่ากับ
7	==, !=	เท่ากับ และไม่เท่ากับ
8	&&	Logical AND
9	||	Logical OR
10	=, += , *=, %=	เท่ากับ ลดรูปบวก ลดรูปคูณ และลดรูปโมดูลัส

ที่มา: Kris Jamsa. 2002: 40

หากมีตัวดำเนินการที่อยู่ลำดับเดียวกัน จะทำงานจากซ้ายไปขวา คือจะดำเนินการตัวดำเนินการที่อยู่ข้างซ้ายก่อนตัวดำเนินการที่อยู่ข้างขวาตามลำดับ ยกเว้นตัวดำเนินการทางคณิตศาสตร์แบบลดรูปมีรายละเอียดดังตารางที่ 4.8

ตารางที่ 4.8 ลำดับในการดำเนินการ

ตัวดำเนินการ	ลำดับในการดำเนินการ	ประเภท
++ -- + - !	ขวาไปซ้าย	ตัวดำเนินการการเพิ่ม/ลดค่า เครื่องหมายลบ
* / %	ซ้ายไปขวา	ตัวดำเนินการการคูณ
+ -	ซ้ายไปขวา	ตัวดำเนินการการบวก
< <= > >=	ซ้ายไปขวา	ตัวดำเนินการเปรียบเทียบ
&&	ซ้ายไปขวา	ตัวดำเนินการตรรกะ AND
||	ซ้ายไปขวา	ตัวดำเนินการตรรกะ OR
== !=	ซ้ายไปขวา	ตัวดำเนินการการเท่ากัน
?:	ขวาไปซ้าย	เงื่อนไข
= += -= *= /= %=	ขวาไปซ้าย	ตัวดำเนินการแบบลดรูป
,	ซ้ายไปขวา	เครื่องหมาย

ที่มา: Harvey M. Deitel, Paul J. Deitel. 2004: 123

กรณีที่มีตัวดำเนินการในระดับเดียวกันอยู่ในคำสั่งเดียวกัน ถ้าลำดับในการดำเนินการเรียงลำดับจากซ้ายไปขวา หมายความว่า ถ้ามีตัวดำเนินการที่อยู่ลำดับเดียวกัน จะทำงานจากซ้ายไปขวา คือจะดำเนินการตัวดำเนินการที่อยู่ข้างซ้ายก่อนตัวดำเนินการที่อยู่ข้างขวาตามลำดับ ถ้าลำดับในการดำเนินการเรียงลำดับจากขวาไปซ้าย หมายความว่า ตัวดำเนินการที่อยู่ลำดับเดียวกัน จะทำงานจากขวาไปซ้าย คือจะดำเนินการตัวดำเนินการที่อยู่ข้างขวาก่อนตัวดำเนินการที่อยู่ข้างซ้ายตามลำดับ

5. ตัวดำเนินการทางคณิตศาสตร์แบบลดรูป

ตัวดำเนินการทางคณิตศาสตร์ ในภาษาซี สามารถเขียนลดรูปได้ดังนี้

ตารางที่ 4.9 ตัวดำเนินการทางคณิตศาสตร์แบบลดรูปในภาษาซี

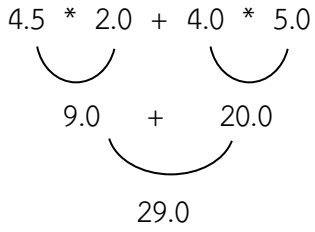
เครื่องหมาย	การใช้งาน	เท่ากับ	ความหมาย
+=	Y += X	Y=Y+X	บวกค่า Y ด้วย X แล้วเก็บผลลัพธ์ไว้ที่ Y
-=	Y -= X	Y=Y-X	ลบค่า Y ด้วย X แล้วเก็บผลลัพธ์ไว้ที่ Y
*=	Y *= X	Y=Y*X	คูณค่า Y ด้วย X แล้วเก็บผลลัพธ์ไว้ที่ Y
/=	Y /= X	Y=Y/X	หารค่า Y ด้วย X แล้วเก็บผลลัพธ์ไว้ที่ Y
%=	Y %= X	Y=Y%X	หาเศษจากการหารค่า Y ด้วย X แล้วเก็บผลลัพธ์ไว้ที่ Y

ตัวอย่างที่ 4.1 ให้นักศึกษาเขียนลำดับการประมวลผลก่อนหลังของนิพจน์ต่อไปนี้

ก. 4.5 * 2.0 + 4.0 * 5.0

ข. 5/2 * 4 + 3 * 12 / 6

วิธีคิด ข้อ ก. ต้องหาตัวดำเนินการที่มีความสำคัญสูงที่สุดก่อน คือ * จะได้ว่ามีเครื่องหมาย * (คูณ) สองตัวคือ (4.5*2.0) และ (4.0*5.0) คำนวณ 4.5*2.0 ก่อนเนื่องจากอยู่ด้ายซ้ายได้ค่าเท่ากับ 9.0 แล้วคำนวณ 4.0*5.0 มีค่าเท่ากับ 20.0 จากนั้นนำค่าที่ได้มาบวกกันคือ 9.0 + 20.0 มีค่าเท่ากับ 29.0 ดังภาพที่ 4.2



ภาพที่ 4.2 ลำดับการดำเนินการของการคำนวณตัวอย่างที่ 4.1 ข้อ ก.

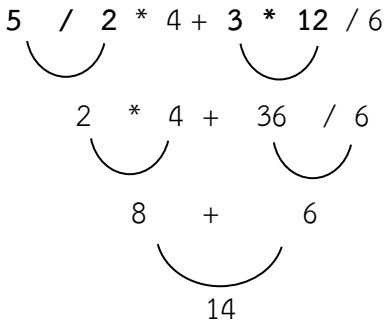
ข้อ ก. 4.5 * 2.0 + 4.0 * 5.0 เขียนลำดับการดำเนินการได้เป็น (4.5*2.0) + (4.0 * 5.0) เหตุที่ต้องเขียนทศนิยมด้วยเพราะว่าภาษาซี จะดำเนินการคูณในลักษณะของเลขทศนิยม ถ้าไม่ใส่จุดทศนิยมภาษาซี จะคำนวณโดยคิดว่าเป็นเลขจำนวนเต็ม อาจทำให้ค่าที่คำนวณคลาดเคลื่อนได้ดังตัวอย่างในข้อ ข.

ข้อ ข. 5/2 * 4 + 3 * 12 / 6 ใช้หลักคิดแบบเดียวกัน

เขียนลำดับการดำเนินการได้เป็น ((5/2) * 4) + ((3*12) / 6)

ได้ค่าเป็น (2*4) + (36/6) = 8 + 6 ได้ผลลัพธ์เท่ากับ 14

ข้อ ข. 5/2 * 4 + 3 * 12 / 6 ต้องหาตัวดำเนินการที่มีความสำคัญสูงที่สุดก่อนคือ * (คูณ) และ / (หาร) ซึ่งมีความสำคัญในระดับเดียวกัน ให้ดำเนินการจากซ้ายไปขวา ดังนั้นจะคำนวณ 5/2 ก่อน ในภาษาซี 5/2 เท่ากับ 2 เนื่องจาก 5 เป็นข้อมูลชนิดตัวเลข 2 ข้อมูลชนิดตัวเลข หารกันผลลัพธ์จะได้ข้อมูลชนิดตัวเลขด้วย จากนั้นนำผลลัพธ์ที่ได้ คือ 2 คูณด้วย 4 ได้เป็น 8 จากนั้นคำนวณที่ 3 * 12 ได้เท่ากับ 36 / 6 ได้เท่ากับ 6 นำ 8 + 6 ได้ผลลัพธ์เท่ากับ 14 ดังภาพที่ 4.3



ภาพที่ 4.3 ลำดับการดำเนินการของการคำนวณตัวอย่างที่ 4.1 ข้อ ข.

การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ

การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ (Implicit Casting) ใช้ในกรณีที่มีการกระทำทางคณิตศาสตร์ กรณีที่ตัวแปร 2 ตัวแปร ไม่เป็นชนิดเดียวกัน ภาษาซีจะทำการเปลี่ยนชนิดของตัวแปรที่มีขนาดเล็กให้เท่ากับขนาดของชนิดของตัวแปรที่ใหญ่กว่าให้อัตโนมัติก่อนที่จะกระทำการทางคณิตศาสตร์ ซึ่งจะเป็นไปตามตารางที่ 4.10 การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ

ตารางที่ 4.10 การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ

ชนิดของตัวแปรที่ 1	ชนิดของตัวแปรที่ 2	ผลการแปลงชนิดข้อมูล
char	int	char แปลงเป็น int
int	long	int แปลงเป็น long
int	unsigned int	int แปลงเป็น unsigned int
int	float	int แปลงเป็น float
int	double	int แปลงเป็น double
float	double	Float แปลงเป็น double
long	double	long แปลงเป็น double
double	long double	double แปลงเป็น long double

ที่มา: Kris Jamsa. 2002: 45

ตัวอย่างที่ 4.2 การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ

```
1  #include <stdio.h>
2  #include <conio.h>
3  void  main()
4  {
5        int  x = 4000 ;
6        float  a, b = 50000.00 ;
7        a = b + x ;
8        printf (“a = %.2f \n”, a) ;
9  }
```

จากตัวอย่างที่ 4.2 ตัวแปร x เป็นตัวแปรชนิดตัวเลขจำนวนเต็ม (int) ขนาด 16 บิต ส่วนตัวแปร b เป็นตัวแปรชนิดตัวเลขทศนิยม (float) ขนาด 32 บิต เมื่อต้องคำนวณ a = b + x แต่ตัวแปร b เป็น float ตัวแปร x เป็น int ซึ่งเป็นคนละชนิดกัน ขนาดไม่เท่ากัน ดังนั้นโปรแกรมจะแปลงชนิดของตัวแปรขนาดเล็กไปเป็นชนิดเดียวกับชนิดของตัวแปรที่มีขนาดใหญ่ ในกรณีนี้ แปลงชนิดของตัวแปร x จาก int เป็น float ก่อนแล้วคำนวณหาผลลัพธ์ ผลลัพธ์จะต้องเป็นชนิดของตัวแปรขนาดใหญ่ ผลลัพธ์จากการคำนวณ a = 50000.00 + 4000.00 ได้ผลลัพธ์ a = 54000.00

ผลลัพธ์

a = 54000.00

การแปลงชนิดของข้อมูล

การแปลงชนิดของข้อมูล (Explicit Casting) ใช้ในกรณีที่มีต้องการเปลี่ยนชนิดของข้อมูล ขนาดที่ใหญ่กว่าเป็นขนาดที่เล็กกว่าสามารถทำได้โดยใช้วิธีการแปลงชนิดของข้อมูลได้ โดยมีรูปแบบดังนี้

รูปแบบ : ชื่อตัวแปรใหม่ = (ชนิดของตัวแปรใหม่) ชื่อตัวแปรเก่า ;

ตัวอย่างที่ 4.3 การแปลงชนิดข้อมูล

```
1   #include <stdio.h>
2   #include <conio.h>
3   void  main()
4   {
5     int x = -32;
6     float y = 1.25e03;
7     char z = ‘C’;
8     int  num1;
9     float  num2;
10    num1 = (int) y;
11    num1 = (int) z;
12    num1 = (int) -4.246;
13    num1 = (int) ‘A’;
14    num2 = (float) x;
15    num2 = (float) –125;
16    num2 = (float) z;
17  }
```

จากตัวอย่างที่ 4.3 สามารถเปลี่ยนชนิดของตัวแปร จาก float เป็น integer ได้โดยใส่ (int) ไว้ข้างหน้าตัวแปรที่เป็น float เช่น y เป็น float ต้องการเปลี่ยน y เป็น integer ใช้คำสั่ง (int)y เป็นต้น ในภาษาซี สามารถเปลี่ยนชนิดของตัวแปร float และ char เป็นชนิดของตัวแปร integer ได้ และสามารถเปลี่ยน integer และ char เป็นชนิดของตัวแปร float ได้ตามตัวอย่าง

สรุป

โครงสร้างของภาษาซีประกอบด้วยส่วนหัวและส่วนของตัวโปรแกรม ซึ่งในส่วนหัวสามารถกำหนดส่วนที่เป็นพรีโพรเซสเซอร์ สามารถประกาศตัวแปรที่เป็นโกลบอลคือสามารถใช้ได้ทั้งโปรแกรมได้ในทุกฟังก์ชัน และสามารถใส่หมายเหตุของโปรแกรมเพื่ออธิบายการทำงานของโปรแกรมได้ โดยใช้เครื่องหมาย /* เป็นจุดเริ่มต้นของหมายเหตุ และเครื่องหมาย */ เป็นจุดจบของหมายเหตุ

ชนิดของตัวแปรในภาษาซีมีทั้งชนิดตัวเลขและตัวอักษร เช่น เลขจำนวนเต็ม เลขทศนิยม อักขระ สายอักขระ หรือตัวแปรแบบแถวลำดับ เป็นต้น การตั้งชื่อตัวแปรต้องขึ้นต้นด้วยอักษรภาษาอังกฤษ ห้ามเว้นวรรค ชื่อตัวแปรไม่ตรงกับคำสงวนในภาษาซี

ตัวดำเนินการในภาษาซี มีตัวกำหนดค่า ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการเพิ่มค่าและลดค่า ตัวดำเนินการทางตรรกะ ตัวดำเนินการทางคณิตศาสตร์แบบลดรูป การคำนวณต่าง ๆ ต้องมีการใช้ตัวดำเนินการเพื่อคำนวณค่าต่าง ๆ จึงควรเข้าใจถึงลำดับการทำงานของตัวดำเนินการที่นำมาใช้อย่างถูกต้องเพื่อการคำนวณที่ถูกต้องไม่มีความผิดพลาดในภายหลัง

การเปลี่ยนชนิดของตัวแปรในกรณีที่มีการคำนวณทางคณิตศาสตร์ ตัวแปร 2 ตัวแปร ไม่เป็นชนิดเดียวกัน ภาษาซีจะทำการเปลี่ยนชนิดของตัวแปนขนาดเล็ก ให้เป็นขนาดที่เท่ากันอัตโนมัติแล้วจึงทำการคำนวณ ซึ่งการเปลี่ยนชนิดของตัวแปรอัตโนมัติเรียกว่า การแปลงชนิดข้อมูลที่เกิดขึ้นอัตโนมัติ (Implicit Casting) ส่วนการเปลี่ยนชนิดของตัวแปร โดยใช้คำสั่งเปลี่ยนชนิดตัวแปรเรียกว่า การแปลงชนิดของข้อมูล (Explicit Casting) ซึ่งสามารถเปลี่ยนชนิดของตัวแปร ที่มีขนาดใหญ่ เป็นตัวแปร ชนิดที่มีขนาดเล็กกว่าได้

แบบฝึกหัด

1. โครงสร้างของโปรแกรมภาษาซีประกอบด้วยกี่ส่วน อธิบายแต่ละส่วน
2. จงอธิบายว่าชนิดของตัวแปรมีกี่ชนิด อะไรบ้าง มีขนาดเท่าใด
3. จงอธิบายลักษณะการใช้งานของตัวแปรแต่ละชนิด
4. จงประกาศตัวแปรชื่อ Num เพื่อเก็บจำนวนเต็ม
5. จงตัวแปรชื่อ Total เพื่อเก็บจำนวนจริงมีทศนิยม โดยกำหนดค่าเริ่มต้นเป็น 137.897
6. จงตัวแปรชื่อ Name เพื่อเก็บข้อความ โดยกำหนดค่าเริ่มต้นเป็นชื่อตัวเอง
7. จงตัวแปรชื่อ Letter เพื่อเก็บอักขระ ‘S’
8. ตัวดำเนินการมีกี่ประเภท ลักษณะการใช้งานแตกต่างกันอย่างไร
9. จงเรียงลำดับการดำเนินการ
    - 9.1 a * b + 10 % c
    - 9.2 (a-b) * 10 / c && d + 5
    - 9.3 a / b + c * 10 % (d - 3)
    - 9.4 a * (++b - c) / d + 5 % e
    - 9.5 a – b + 15 && c * d / 5 || e = = 3
    - 9.6 (a + b) * 2 = = c % d * (e + 7)
    - 9.7 (a + b) / 2 && c * d
    - 9.8 x / y + 4 | | 3 * z
    - 9.9 (a + b) * c && d + 7 % d
    - 9.10 +x / y * z && a + 3 * b
    - 9.11 a * 7 + b % c | | x + y * 5 && 4 / z
10. จงหาผลลัพธ์ต่อไปนี้

    int x = 4;

    float y = -1.2;

    char z = ‘A’;

    10.1 (x<y) | | (z >=y)

    10.2 (y * 100) && (x>=z)

    10.3 (!z) | | (x < y) && (y <= x)

    10.4 (x * y) && (z) || (y< x)

    10.5 (y == z) | | (x > y)

เอกสารอ้างอิง

ประภาพร ช่างไม้. (2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส.

อรพิน ประวัติบริสุทธิ์. (2554). *คู่มือเรียนภาษาซี ฉบับปรับปรุงใหม่.* กรุงเทพมหานคร: โปรวิชั่น.

Borland International, Inc. (1990). *Turbo C++ Version 1.00.*

Harvey M. Deitel, Paul J. Deitel. (2004). *C How to program*. 4th Edition. London: Prentice Hall.

Stephen G. Kochan. (1994). *Programming in ANSI C revised Edition.* USA: SAMS Publishing.

Kris Jamsa. (2002). *Jamsa's C/C++/C# programmer's bible: The ultimate guide to C/C++/C# programming.* 2nd Edition. USA: Onword Press.

บทที่ 5
การรับและการแสดงผลข้อมูล

การรับข้อมูลมีฟังก์ชันหลายฟังก์ชันในภาษาซีที่ใช้ในการรับข้อมูล ในบทนี้จะกล่าวถึง ฟังก์ชัน scanf() ฟังก์ชัน getch() ฟังก์ชัน getche() และ ฟังก์ชัน getchar() และในการแสดงผลข้อมูล ฟังก์ชัน printf() ฟังก์ชัน putchar ซึ่งการใช้งานแต่ละฟังก์ชันมีการใช้งานที่แตกต่างกันออกไป

การแสดงผลข้อมูล

การแสดงผลข้อมูลโดยใช้ฟังก์ชัน printf() ในการแสดงผลข้อมูล และการรับข้อมูลโดยใช้ฟังก์ชัน scanf() ในภาษาซีบรรจุในคลังโปรแกรม stdio.h (Standard Input and Output) โดยต้องนำเข้ามาด้วย โดยเขียนในส่วนของ Preprocessor ใช้คำสั่ง #include <stdio.h> ซึ่งฟังก์ชันที่จะได้กล่าวถึงต่อไปนี้คือฟังก์ชัน printf() ซึ่งมีรูปแบบการใช้งานดังนี้

รูปแบบ	:	int printf(char *format , arg1, arg2, ...)
Header file	:	stdio.h

ที่มา: Brian W. Kernigham, Dennis M. Ritchie. 1988: 153

ฟังก์ชัน printf() เป็นฟังก์ชันเพื่อการแสดงผลมีพารามิเตอร์ที่สำคัญ 2 อย่างคือ รูปแบบการแสดงผล (char *format) และรายการอาร์กิวเมนต์ (list of argument)

1. รูปแบบการแสดงผลมี 3 รูปแบบด้วยกัน คือ ข้อความที่ต้องการแสดง อักขระควบคุมการแสดงผล และอักษรกำหนดรูปแบบ

1) ข้อความที่ต้องการแสดง คือข้อความที่ต้องการแสดงสามารถแสดงข้อความที่ต้องการสื่อสารกับผู้ใช้ให้เข้าใจ ว่าข้อมูลที่ป้อนคือข้อมูลใด ข้อมูลที่จะแสดงต่อไปคือค่าอะไร เช่น

ข้อความแสดงคำทักทาย “Hello” , “Welcome...”

ข้อความแสดงการตอบรับ “Thank you”

ข้อความแสดงการทำงาน “Data is inserted...” , “Processing, Please wait....”

ข้อความนำเพื่อการรับข้อมูล “Enter your name : ” , “Press any key to exit”

ข้อความแสดงผลลัพธ์ “Your name is .... ” , “Value of sum is .... ”

2) อักขระควบคุมการแสดงผล ใช้ควบคุมการแสดงผลลัพธ์เพื่อสื่อสารกับผู้ใช้ได้เข้าใจถึงผลลัพธ์การแสดงผลเช่น “\n” ขึ้นบรรทัดใหม่ และอักขระควบคุมการแสดงผลอื่นแสดงไว้ในตารางที่ 5.1

3) อักษรกำหนดรูปแบบ จะใช้กำกับการแสดงผลของตัวแปร เช่น %d, %f เป็นต้น หากไม่มีอักษรกำหนดรูปแบบ ตัวแปลภาษาซีจะแสดงข้อความนั้นโดยตรง รูปแบบการแสดงผลที่ภาษาซี เข้าใจพร้อมการใช้งานได้แสดงไว้ในตารางที่ 5.2

2. รายการอาร์กิวเมนต์ สามารถเป็นรายการของตัวแปร ค่าคงที่ หรือเป็นนิพจน์ (expression) ที่ต้องการแสดงผล เช่น

ตัวแปร เช่น x , y , salary เป็นต้น

ค่าคงที่ เช่น 365 , 123.456 , “Napatsarun” เป็นต้น

นิพจน์ เช่น x+5 , y*5.0 เป็นต้น

ฟังก์ชัน เช่น pow(2,3) , sqrt(25) เป็นต้น

โดยรายการอาร์กิวเมนต์ที่ต้องการจะแสดงผล จะต้องกำกับด้วยอักษรกำหนดรูปแบบ ดังตารางที่ 5.2 เช่น

ตัวแปร x เป็นตัวแปรชนิดตัวเลข (int) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%d”

ตัวแปร y เป็นตัวแปรชนิดตัวเลขทศนิยม (float) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%f”

ค่าคงที่ 123.456 เทียบได้กับตัวเลขทศนิยม (float) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%f”

ข้อความ “Napatsarun” เป็นข้อความเทียบได้กับสายอักขระ (sting) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%s”

นิพจน์ y*5.0 พิจารณาที่ผลลัพธ์ ผลลัพธ์เทียบได้กับตัวเลขทศนิยม (float) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%f”

ฟังก์ชัน pow(2,3) ผลของฟังก์ชันมีค่าเป็นตัวเลขทศนิยม (double) ต้องกำกับด้วยอักษรกำหนดรูปแบบ “%f”

ตารางที่ 5.1 อักษรกำหนดการแสดงผล

สัญลักษณ์	ความหมาย
\b	backspace
\f	formfeed (เลื่อนกระดาษ ขึ้นหน้าใหม่)
\n	newline (ขึ้นบรรทัดใหม่)
\r	cariage return (Enter)
\t	horizontal Tab (เว้นวรรค 8 อักขระ)
\\	backslash
\?	question mark
\”	double quote (เครื่องหมายฟันหนู)
\’	single quote (เครื่องหมายฝนทอง)
\0	NULL (จบข้อความ)
\v	Vertical Tab (Tab แนวตั้ง)
\a, \007	Bell (เสียงบี๊ปของลำโพง:Speaker)
\ooo	octal number (ตัวเลขฐาน 8)
\xhh	hexadecimal number (ตัวเลขฐาน 16)

ที่มา: Delores M. Etter. 2013: 66

ตารางที่ 5.2 อักษรกำหนดรูปแบบ

อักษรกำหนดรูปแบบ	ชนิดของข้อมูล Input	ชนิดของข้อมูล Output
%d	integer	เลขจำนวนเต็มฐานสิบคิดเครื่องหมาย
%i	integer	เลขจำนวนเต็มฐานสิบคิดเครื่องหมาย
%c	integer	แสดงเป็นอักขระ
%o	integer	เลขจำนวนเต็มฐานแปดไม่คิดเครื่องหมาย (ไม่มีเลข 0 นำหน้า)
%u	integer	เลขจำนวนเต็มฐานสิบไม่คิดเครื่องหมาย
%x	integer	เลขจำนวนเต็มฐาน 16 ไม่คิดเครื่องหมาย โดย a, b, c, d, e, f แสดงเป็นตัวอักษรตัวพิมพ์เล็ก (ไม่มีเลข 0x นำหน้า)
%X	integer	เลขจำนวนเต็มฐาน 16 ไม่คิดเครื่องหมาย โดย A, B, C, D, E, F แสดงเป็นตัวอักษรตัวพิมพ์ใหญ่ (ไม่มีเลข 0X นำหน้า)
%s	char *	แสดงเป็นอักขระ จนถึง อักขระ ‘\0’
%f	floating point	เลขทศนิยมคิดเครื่องหมาย ในรูแปบบ[-]m.dddddd แสดงทศนิยม 6 ตำแหน่ง เช่น -8.123456
%e	floating point	เลขทศนิยมคิดเครื่องหมายในรูปแบบ วิทยาศาสตร์ [-]m.dddddde+/-xx เช่น -8.123456e+02
%E	floating point	เหมือน %e; แต่ใช้ e เป็นตัวพิมพ์ใหญ่ (E)
%g, %G	double	เหมือน %e, %E, %f แสดงทศนิยมไม่เกิน 4 ตำแหน่ง
%p	pointer	ตัวแปรตัวชี้ (pointer)
%%	none	แสดง ‘%’

ที่มา: Brian W. Kernigham, Dennis M. Ritchie. 1988: 154

การใช้งานฟังก์ชัน printf()

สำหรับการใช้งานฟังก์ชัน printf() นั้นสามารถอธิบายในตามลักษณะการใช้งาน 4 ลักษณะดังนี้

1. การแสดงผลข้อความอย่างเดียว

การแสดงผลข้อความอย่างเดียว ข้อความที่ต้องการจะแสดงผลจะต้องเขียนไว้ในเครื่องหมาย “...” ภายใต้เครื่องหมาย () อีกชั้นหนึ่ง โดยข้อความที่แสดงผลนี้จะไม่ได้รับการสนใจจาก Compiler ภาษาซี แต่จะทำการแสดงผลข้อความนั้นโดยตรง ยกเว้นข้อความที่เป็นเครื่องหมาย % จะถูกตัวแปลภาษาซี เข้าใจว่าเป็นการจัดรูปแบบการแสดงผลตัวแปร เช่น

printf(“Hello World”); เพื่อแสดงข้อความ “Hello World”
printf(“ Thank you... ”); เพื่อแสดงข้อความ “ Thank you... ”
printf(“Data is inserted...”); เพื่อแสดงข้อความ “Data is inserted...”
printf(“Please wait...”); เพื่อแสดงข้อความ “Please wait...”
printf(“Data is inserted...”); เพื่อแสดงข้อความ “Data is inserted...”
printf(“Enter your name : ”); เพื่อแสดงข้อความ “Enter your name : ”
printf(“Press any key to exit”); เพื่อแสดงข้อความ “Press any key to exit”

2. การแสดงผลตัวแปร

การแสดงผลตัวแปร ต้องมีการกำหนดรูปแบบการแสดงผลตัวแปรนั้นๆ ว่าจะให้แสดงผลในรูปแบบใด เช่น ตัวแปรชนิดตัวเลข (integer) สามารถแสดงได้ในรูปแบบเลขฐาน 10 รูปแบบเลขฐาน 8 หรือรูปแบบเลขฐาน 16 ตัวแปรประเภทตัวเลขโดยการกำหนดรูปแบบการแสดงผลนั้นจะใช้อักษรภาษาอังกฤษนำหน้าด้วยเครื่องหมาย % เรียกว่า “อักษรกำหนดรูปแบบ” ซึ่งตัวอย่างของอักษรกำหนดรูปแบบเป็นดังแสดงในตารางที่ 5.2

ตัวอย่างที่ 5.1 การแสดงผลตัวแปร

```
int  sum=20;
printf(“%d”,sum);
printf(“%x”,sum);
```

int sum; ประกาศตัวแปร sum เป็นตัวแปรชนิดตัวเลข (integer)

printf(“%d”,sum); เพื่อแสดงค่าที่เก็บอยู่ในตัวแปร sum ในรูปแบบเลขฐาน 10 ผลลัพธ์แสดง 20

printf(“%x”,sum); เพื่อแสดงค่าที่เก็บอยู่ในตัวแปร sum ในรูปแบบเลขฐาน 16 ผลลัพธ์แสดง 14

จะเห็นได้จากตัวอย่าง ตัวแปร sum เพียงตัวเดียวสามารถที่จะถูกกำหนดให้แสดงในรูปแบบ %d คือเลขฐาน 10 หรือ %x คือเลขฐาน 16 ได้ ดังนั้นจึงเป็นสิ่งสำคัญที่ต้องระบุรูปแบบแสดงผลของแต่ละตัวแปรด้วย ซึ่งถ้าไม่ระบุตัวแปรภาษาซี จะไม่แสดงผล

สำหรับอักษรกำหนดรูปแบบการแสดงผลที่ใช้ได้กับตัวแปรแต่ละประเภทเป็นดังตารางที่ 5.2 สำหรับรูปแบบการแสดงผลที่ใช้ได้กับตัวแปรชนิดตัวเลข สามารถกำหนดรายละเอียดได้อีกเช่น

%d ใช้ในการแสดงค่าตัวแปรชนิดตัวเลขในเลขฐาน 10

%6d ใช้ในการแสดงค่าตัวแปรชนิดตัวเลขในเลขฐาน 10 โดยมีความกว้างอย่างน้อย 6 อักขระ โดยจัดการแสดงผลชิดขวา

%-6d ใช้ในการแสดงค่าตัวแปรชนิดตัวเลขในเลขฐาน 10 โดยมีความกว้างอย่างน้อย 6 อักขระ โดยจัดการแสดงผลชิดซ้าย

สำหรับเลขทศนิยม สามารถกำหนดความยาวของเลขทศนิยมได้ด้วยโดยการเพิ่มอักษรในการจัดรูปแบบ เช่น

%f แสดงผลจำนวนทศนิยมในรูปแบบปกติ แสดงทศนิยม 6 ตำแหน่ง

%.2f แสดงผลจำนวนทศนิยมโดยแสดงในเพียงทศนิยม 2 ตำแหน่ง โดยหากตำแหน่งถัดไปมีค่ามากกว่า 5 จะปัดขึ้น แต่หากน้อยกว่า 5 จะปัดทิ้ง

%0.2f แสดงผล จำนวนทศนิยม 2 ตำแหน่งโดยหากมีทศนิยมไม่ถึง 2 ตำแหน่งให้ใส่ 0 เพิ่มจนครบ จำนวนตำแหน่งที่ต้องการ ดังตัวอย่างการใช้งาน (0.5 จะแสดงเป็น 0.50)

%5.2f กำหนดการแสดงผลให้เป็น 5 ตัวอักษร และมีทศนิยม 2 ตำแหน่งโดยจัดการแสดงผลชิดขวา เช่น 8.25 จะแสดงเป็น _8.25 โดย _ เป็นตัวอักษรว่างที่แสดงเติมให้ครบ 5 ตัวอักษร (รวมทศนิยม)

%-5.2f กำหนดการแสดงผลให้เป็น 5 ตัวอักษร และมีทศนิยม 2 ตำแหน่งโดยจัดการแสดงผลชิดซ้าย เช่น 8.25 จะแสดงเป็น 8.25_ โดย _ เป็นตัวอักษรว่างที่แสดงเติมให้ครบ 5 ตัวอักษร (รวมทศนิยม)

ตัวอย่างที่ 5.2 การจัดรูปแบบการแสดงผลทศนิยมโดยใช้ฟังก์ชัน printf()

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       float f, f2;
6       clrscr();
7       f=20.12828;
8       f2=1.1;
9       printf("f= %f\n",f);
10      printf("f= %.2f\n",f);
11      printf("f= %0.2f\n",f);
12      printf("f2= %f\n",f2);
13      printf("f2= %.2f\n",f2);
14      printf("f2= %0.2f\n",f2);
15      printf("f2= %6.2f\n",f2);
16      getch();
17  }
```

โปรแกรมที่ 5.1 โปรแกรมการจัดรูปแบบการแสดงผลทศนิยมโดยใช้ฟังก์ชัน printf()

ซึ่งเมื่อทำการประมวลผลโปรแกรมจะได้ผลลัพธ์ดังผลลัพธ์

```
f= 20.128281
f= 20.13
f= 20.13
f2= 1.100000
f2= 1.10
f2= 1.10
f2=  1.10
```

3. การแสดงผลตัวแปรประกอบกับค่าคงที่ซึ่งเป็นอักขระ

การแสดงผลตัวแปรประกอบกับค่าคงที่ซึ่งเป็นอักขระ สามารถทำได้โดยการนำเอาอักษรกำหนดรูปแบบการแสดงผลสำหรับการแสดงผลตัวแปร ไปแทรกไว้ระหว่างค่าคงที่ซึ่งเป็นอักขระ จากนั้นเพิ่มรายการของตัวแปรลงไปหลังจากเครื่องหมายอัญประกาศ หรือ เครื่องหมายคำพูดปิดของค่าคงที่อักขระ จนกว่าจะครบทุกตัวแปรที่ต้องการแสดงผล ดังตัวอย่างที่ 5.3

ตัวอย่างที่ 5.3 การแสดงผลตัวแปรประกอบกับค่าคงที่ซึ่งเป็นอักขระ

```
int sum = 20;
printf("Value of sum is %d",sum);
printf("Value of sum is %x",sum);
```

การแสดงผลโดยทั่วไป ต้องมีการอธิบายว่า ค่าที่จะแสดงคือค่าอะไร ได้มาจากไหน มีที่มาอย่างไร เช่น ต้องการแสดงค่าตัวแปร sum ควรจะบอกให้ผู้ใช้ทราบว่า คือค่าของตัวแปร sum

printf("Value of sum is %d",sum);

ผลลัพธ์ที่ได้ คือ Value of sum is 20

Value of sum is คือ ข้อความใด ๆ

20 คือ ค่าของตัวแปร sum แสดงในรูปแบบของเลขฐาน 10

printf("Value of sum is %x",sum);

ผลลัพธ์ที่ได้ คือ Value of sum is 14

ตัวแปร sum มีค่าเท่ากับ 20 ในเลขฐาน 10 และมีค่าเท่ากับ 14 ในเลขฐาน 16

ในกรณีที่ต้องการแสดงค่าตัวแปรมากกว่า 1 ตัวก็สามารถทำได้ ดังตัวอย่างที่ 5.4

ตัวอย่างที่ 5.4 การแสดงผลตัวแปรหลายตัว

```
int  sum = 20 ;
char  a = 'T';
printf("Value of sum = %d Value of a = %c ", sum, a);
```

จากตัวอย่างนี้ตัวแปร sum จะได้รับการแสดงผลในรูปแบบเลขฐาน 10 เพราะคู่กับ %d ในขณะที่ตัวแปร a นั้นคู่กับ %c จึงแสดงผลในรูปตัวอักษร

โปรแกรมต้องการแสดงค่าตัวแปร 2 ตัวแปร คือ ตัวแปร sum และตัวแปร a

printf("Value of sum = %d Value of a = %c ", sum, a);

ผลลัพธ์

Value of sum = 20 Value of a = T

4. การแสดงผลอักขระพิเศษ

การแสดงผลอักขระพิเศษ ในภาษาซีสามารถที่จะทำการแสดงผลอักขระพิเศษ ที่ไม่สามารถพิมพ์ลงสู่รหัสต้นฉบับได้โดยตรงเช่น รหัสขึ้นบรรทัดใหม่ ได้โดยการใช้เครื่องหมาย \ นำหน้ารหัสสำหรับอักขระพิเศษตัวนั้นๆ โดยตัวอย่างของอักษรสำหรับอักขระพิเศษมีดังตารางที่ 5.3

ตัวอย่างที่ 5.5 โปรแกรมการแสดงผลโดยใช้ รูปแบบในการพิมพ์

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main()
4   {
5           char ch = 'A';
6           char str[20] = "ComputerScience";
7           float a = 88.86666, b = 88.8 ;
8           printf("ch = %3c.\n",ch);
9           printf("a = %.0f, b = %.0f\n",a,b);
10          printf("a = %5.2f#\n,a);
11          printf("a = %8.2f#\n”,a);
12          printf("a = %-8.2f#\n",a);
13          printf("str: %15.8s.\n",str);
14          printf("str: %-15.8s.",str);
15         }
```

โปรแกรมที่ 5.2 โปรแกรมการแสดงผลโดยใช้ รูปแบบในการพิมพ์

การแสดงผลตัวแปรโดยใช้รูปแบบต่าง ๆ โดยใช้ฟังก์ชัน printf() ซึ่งเมื่อทำการประมวลผลโปรแกรมจะได้ผลลัพธ์ดังนี้

```
ch  =    A.
a = 89, b = 89
a = 88.87#
a =       88.87#
a = 88.87      #
str:  _________Computer.
str:  Computer_________.
```

จากผลการประมวลผลโปรแกรมจะเห็นว่าการพิมพ์ตัวอักษรโดยใช้ %3c หมายความว่าจองพื้นที่ในการพิมพ์ 3 ตำแหน่งข้อมูลจะพิมพ์ชิดขวา ส่วนการพิมพ์ข้อมูลทศนิยมโดยใช้ %8.2f หมายความว่า จองพื้นที่ในการพิมพ์ 8 ตำแหน่ง รวมจุดทศนิยม และมีทศนิยม 2 ตำแหน่ง และข้อมูลชิดขวา หากมีเครื่องหมายลบ (-) เช่น %-8.2f ข้อมูลจะชิดซ้าย ส่วนการพิมพ์ข้อความหากใช้ %15.8s หมายถึง การจองพื้นที่ในการพิมพ์ 15 ตำแหน่ง แต่พิมพ์ข้อมูลเพียง 8 ตัวอักษร และข้อมูลชิดขวา หากมีเครื่องหมายลบหมายถึงการจัดชิดซ้าย

การรับข้อมูล

ฟังก์ชัน scanf() เป็นฟังก์ชันที่ใช้ในการรับข้อมูล ซึ่งมีรูปแบบการใช้งานคล้ายกับฟังก์ชัน printf นั่นคือเวลาใช้งานต้องทำการเรียกใช้ไฟล์ส่วนหัว stdio.h คล้ายกับฟังก์ชัน printf แต่ฟังก์ชัน scanf ใช้ในการรับข้อมูลได้อย่างเดียว ไม่สามารถแสดงข้อความใด ๆ ได้ ซึ่งรูปแบบการใช้งานของฟังก์ชัน scanf เป็นดังนี้

รูปแบบ	: scanf(char *format , arg1 , arg2 , ...)
Header File	: stdio.h
หมายเหตุ	: ต้องมีเครื่องหมาย & นำหน้าชื่อตัวแปรเสมอยกเว้นตัวแปรแบบตัวชี้ (pointer)

จะเห็นได้การอ่านค่าของฟังก์ชัน scanf() จะคล้ายกับฟังก์ชัน printf() คือต้องมีการระบุรูปแบบการอ่านข้อมูลซึ่งเป็นอักษรกำหนดรูปแบบเช่นเดียวกับที่ใช้ในฟังก์ชัน printf() ซึ่งดูได้จากตารางที่ 5.2

ตัวอย่างที่ 5.6 การรับค่ารหัสนักศึกษาซึ่งเป็นข้อมูลชนิด int

```
1   #include <stdio.h>
2   #include <conio.h>
3
4   void main()
5   {
6           int SID;
7           clrscr();
8           scanf("%d",&SID);
9           printf("Your Student ID is %d",SID);
10          getch();
11  }
```

โปรแกรมที่ 5.3 โปรแกรมการรับค่ารหัสนักศึกษาซึ่งเป็นข้อมูลชนิด int

ผลลัพธ์ การอ่านค่าจากแป้นพิมพ์ (Keyboard) โดยการใช้ฟังก์ชัน scanf()

```
100
Your Student ID is 100
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

โปรแกรมต้องการรับค่า SID ซึ่งกำหนดเป็นตัวแปรชนิดตัวเลข (int SID;) การรับค่าของฟังก์ชัน scanf ต้องกำหนดอักษรกำหนดรูปแบบเป็น “%d” และที่ตัวแปร SID ต้องมีเครื่องหมายแอนด์ & นำหน้าด้วยเป็น &SID จึงเขียนได้ดังนี้

scanf("%d",**&SID**);

เมื่อโปรแกรมประมวลผลฟังก์ชันนี้ โปรแกรมจะรอรับค่าจากผู้ใช้ เพื่อบันทึกไว้ที่อยู่ของตัวแปร SID สมมติผู้ใช้ป้อมค่า 100 และเมื่อประมวลผลต่อไป ฟังก์ชัน printf("Your Student ID is %d",SID); โปรแกรมจะแสดงผลลัพธ์ดังนี้

Your Student ID is 100

โปรแกรมจะแสดงค่า SID มีค่าเท่ากับ 100 ตามที่ผู้ใช้ป้อนค่าเข้ามา

จากผลลัพธ์ จะเห็นได้ว่าหากผู้พัฒนาโปรแกรมไม่ได้เป็นผู้ใช้โปรแกรม จะทำให้ผู้ใช้โปรแกรมไม่เข้าใจว่าเมื่อตัวชี้ตำแหน่ง (Cursor) ปรากฏแล้วผู้ใช้จะต้องป้อนข้อมูลอะไร ดังนั้นในการรับค่าจำเป็นต้องแสดงข้อความ เพื่อให้ผู้ใช้โปรแกรมคอมพิวเตอร์สามารถทราบได้ว่าโปรแกรมต้องการรับค่าอะไร เพื่อจะได้ป้อนค่าให้ถูกต้อง ซึ่งจำเป็นต้องใช้ฟังก์ชัน printf(); เข้าช่วยด้วยดังตัวอย่างที่ 5.7

ตัวอย่างที่ 5.7 การอ่านค่าจากแป้นพิมพ์ (Keyboard) โดยการใช้ฟังก์ชัน scanf()

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       int SID;
6       clrscr();
7       printf("What is your Student ID ? ");
8       scanf("%d",&SID);
9       printf("Your Student ID is %d" , SID);
10      getch();
11  }
```

โปรแกรมที่ 5.4 โปรแกรมการอ่านค่าจากแป้นพิมพ์ (Keyboard) โดยการใช้ฟังก์ชัน scanf()

ผลลัพธ์

```
What is your Student ID ? 100
Your Student ID is 100
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

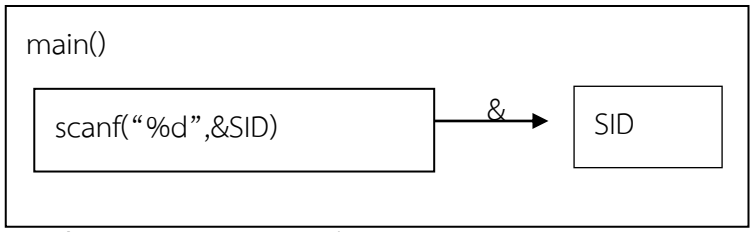
จากตัวอย่างที่ 5.6 เวลาประมวลผลโปรแกรมจะรอรับการป้อนข้อมูลจากผู้ใช้เลย ซึ่งผู้ใช้จะไม่ทราบว่าสิ่งที่ควรป้อนคืออะไร ตัวอย่างที่ 5.7 จึงเพิ่มให้มีข้อความนำก่อนการรับข้อมูล คือ ข้อความ What is your Student ID ? เพื่อให้ผู้ใช้ทราบว่า สิ่งที่ต้องการป้อนคือ รหัสนักศึกษา โดยใช้ printf("What is your Student ID ? "); เมื่อประมวลผลโปรแกรมจะมีข้อความนำก่อน ดังนี้

What is your Student ID ? ผู้ใช้ป้อนตัวเลข เช่น 100

โปรแกรมจะแสดงผผลดังนี้

Your Student ID is 100

ให้สังเกตว่าในการแสดงค่าโดย printf() นั้นตัวแปรซึ่งอยู่หลังเครื่องหมายฟันหนูไม่ต้องมีเครื่องหมายใดๆ นำหน้า นั่นหมายถึงค่าของตัวแปร แต่สำหรับการรับค่าด้วยฟังก์ชัน scanf() นั้นจะต้องกำหนดว่าเมื่อรับค่าแล้วให้นำค่าที่รับได้ไปไว้ที่ใด ซึ่งในการเขียนโปรแกรมด้วยภาษาซี นั้นจะใช้เครื่องหมาย & นำหน้าชื่อตัวแปรเพื่อเป็นการระบุถึงตำแหน่งของหน่วยความจำที่แทนตัวแปรนั้น ในที่นี้ต้องการให้รับค่าไปไว้ในตัวแปรชื่อ SID ดังนั้นจึงต้องใส่เครื่องหมาย & นำหน้าชื่อตัวแปรด้วย เป็น &SID ดังแสดงในภาพที่ 5.1



ภาพที่ 5.1 ความหมายของเครื่องหมาย & ในฟังก์ชัน scanf()

หากต้องการให้ฟังก์ชัน scanf() รับค่าครั้งเดียวมากกว่า 1 ตัวแปรก็ทำได้ดังตัวอย่างที่ 5.8

ตัวอย่างที่ 5.8 การอ่านค่าจากแป้นพิมพ์ (Keyboard) จำนวนหลายๆ ค่าโดยการใช้ฟังก์ชัน scanf()

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int a,b,c;
6           clrscr();
7           printf("Enter a b c ?");
8           scanf("%d %d %d",&a,&b,&c);
9           printf("a is %d\n",a);
10          printf("b is %d\n",b);
11          printf("c is %d\n",c);
12          getch();
13  }
```

โปรแกรมที่ 5.5 โปรแกรมการอ่านค่าจากแป้นพิมพ์ (Keyboard) จำนวนหลายๆ ค่า

ในตัวอย่างที่ 5.8 การรับค่าของฟังก์ชัน scanf() ใช้รูปแบบการแสดงผลเหมือนกับฟังก์ชัน printf() โดยเพิ่มการใส่เครื่องหมายแอนด์ (&) หน้าตัวแปรรับค่า เพื่อกำหนดตำแหน่งของหน่วยความจำของตัวแปรที่ต้องการนำค่าที่รับไปเก็บ โดยในการรับค่านั้นผู้ใช้ต้องคีย์ค่าแต่ละตัวแปรโดยใช้เว้นวรรคในการแยกค่าของตัวแปรในแต่ละตัว เมื่อประมวลผลโปรแกรมนี้จะได้ผลลัพธ์ดังนี้

ผลลัพธ์

```
Enter a b c ?1 2 3
a is 1
b is 2
c is 3
```

ในตัวอย่างที่ 5.8 แสดงให้เห็นการรับค่าจากตัวแปร 3 ค่าหากต้องการให้ฟังก์ชัน scanf() รับค่ากำหนดให้ตัวแปรแต่ละตัวแปร เพื่อใช้ในการคำนวณสามารถทำได้ดังตัวอย่างที่ 5.9

ตัวอย่างที่ 5.9 การอ่านค่าจากแป้นพิมพ์ (Keyboard) และทำการคำนวณแล้วแสดงผล

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       Int  r;
6       float  area;
7       printf("Enter your radius: ");
8       scanf("%d",&r);
9       area = 3.1414 * r* r;
10      printf("Area of circle is %8.2f", area);
11      getch();
12  }
```

โปรแกรมที่ 5.6 โปรแกรมการอ่านค่าจากแป้นพิมพ์ (Keyboard) และทำการคำนวณแล้วแสดงผล

ในตัวอย่างที่ 5.9 รับค่ารัศมี r แล้วคำนวณ หาพื้นที่วงกลม Area แล้วแสดงผลโดยใช้ฟังก์ชัน printf() ซึ่งเมื่อประมวลผลโปรแกรมนี้จะได้ผลลัพธ์ดังผลลัพธ์ดังนี้

```
Enter your radius:  10
Area of  circle is __314.14
```

ตัวอย่างที่ 5.10 จงเขียนโปรแกรมเพื่อคำนวณหาความเร็วของคันหนึ่ง โดยรับระยะทางที่รถเคลื่อนที่และเวลาที่ใช้ในการเคลื่อนที่ จากสูตรการคำนวณ ความเร็ว = ระยะทาง / เวลา
ซึ่งเมื่อนำมาเขียนโปรแกรมจะได้โปรแกรมดังนี้

```
1   #include <stdio.h>
2   void main()
3   {
4      float v, s, t ;
5      clrscr();
6      printf("Enter distance (meter) : ");    scanf("%f" , &s);
7      printf("Enter time (second) : ");       scanf("%f" , &t);
8      v = s/t ;
9      printf(" Velocity is = %.2f meter/second", v);
10  }
```

โปรแกรมที่ 5.7 โปรแกรมเพื่อคำนวณหาความเร็วของคันหนึ่ง

ผลลัพธ์

Enter distance (meter) : ***200***
Enter time (second) : ***40***
Velocity is = 50.00 meter/second

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้นใต้***เป็นข้อมูลที่ป้อนเข้าไป

การรับและแสดงผลข้อมูล 1 อักขระ

นอกจากฟังก์ชัน printf() และ scanf() แล้วภาษาซี ยังมีฟังก์ชันที่ใช้ในการรับ และแสดงผลข้อมูลอื่นๆ อีกได้แก่

1. การแสดงข้อมูลอักขระ

การแสดงข้อมูลอักขระโดยฟังก์ชัน putchar() เป็นฟังก์ชันที่ใช้ในการแสดงผลอักขระออกทางจอภาพครั้งละ 1 อักขระเท่านั้น เปรียบเสมือนการใช้ printf(“%c”,...) รูปแบบการใช้งานของฟังก์ชัน putchar() เป็นดังนี้

รูปแบบ	: int putchar (int c);
Header File	: stdio.h

ที่มา: Brian W. Kernigham, Dennis M. Ritchie. 1988: 153

ฟังก์ชัน putchar() เป็นฟังก์ชันที่ใช้ในการแสดงผลอักขระโดยพารามิเตอร์ที่ต้องการคืออักขระที่ต้องการแสดง ซึ่งสามารถกำหนดเป็นอักขระ หรือรหัสแอสกี ซึ่งรหัสแอสกี คือรหัสมาตรฐานของสหรัฐอเมริกาเพื่อการสับเปลี่ยนสารสารสนเทศ (American Standard Code for Information Interchange: ASCII) เช่น อักขระ B มีค่ารหัสแอสกีเท่ากับ 66

ตัวอย่าง

```
char  ch='B';    ให้นำเอาอักษร 'B' ไปเก็บไว้ในตัวชื่อ ch
putchar('A');    ให้แสดงอักษร 'A' ออกทางจอภาพ
putchar(65);     ให้แสดงอักษร 'A' ออกทางจอภาพ
putchar(ch);     ให้แสดงอักษรที่เก็บอยู่ในตัวแปร ch  ก็คือตัวอักษร 'B' ออกทางจอภาพ
putchar(66);     ให้แสดงตัวอักษร 'B' ออกทางจอภาพ
```

จากตัวอย่างข้างต้นจะเห็นว่าถ้าหมายความถึงอักษรเพียงตัวเดียวนั้นจะใช้เครื่องหมายฝนทองคือ (‘’) คร่อมระหว่างตัวอักษร เช่น ‘B’ เป็นต้น

2. การรับข้อมูลอักขระโดยฟังก์ชัน getchar(), getch(), getche()

การรับข้อมูลอักขระโดยฟังก์ชัน getchar(), getch(), getche() เป็นฟังก์ชันที่ใช้ในการรับค่าอักขระ เปรียบเหมือนการใช้ scanf(“%c”,&VarName) สำหรับฟังก์ชันทั้ง 3 นี้มีการใช้งานในรูปแบบที่เหมือนกันคือ

```
รูปแบบ
        int   getchar (void)  ;
        int   getche (void)  ;
        int   getch (void)  ;
การเรียกใช้ฟังก์ชัน
        ตัวแปร = getchar ();
        ตัวแปร = getche();
        ตัวแปร = getch();
Header File    :  stdio.h  สำหรับ getchar()
               :  conio.h  สำหรับ getche(); และ getch();
```

การรับข้อมูลอักขระโดยฟังก์ชัน getchar(), getch(), getche() เป็นฟังก์ชันที่ไม่มีพารามิเตอร์ และผลของฟังก์ชันเป็นตัวแปรประเภทตัวเลข คือ ค่ารหัสแอสกีของอักขระที่รับเข้ามา เช่น รับค่าอักขระ ‘C’ เข้ามาผลของฟังก์ชันจะได้เท่ากับ 67 นอกจากนี้มีข้อแตกต่างของทั้ง 3 ฟังก์ชันพอจะสรุปได้ดังตารางที่ 5.5

ตารางที่ 5.3 ข้อแตกต่างของฟังก์ชัน getchar(), getch(), getche()

การใช้งาน	getchar()	getch()	getche()
กด Enter หลังป้อนค่า	✓	-	-
ค่าที่ป้อนปรากฏบนจอภาพ	✓	-	✓
รับค่า 1 ตัวอักษร	✓	✓	✓
Header File	stdio.h	conio.h	conio.h

การใช้งานของฟังก์ชัน putchar(), getchar(), getch(), getche() เป็นดังตัวอย่างที่ 5.11

ตัวอย่างที่ 5.11 การใช้งานฟังก์ชัน putchar(), getchar(), getch(), getche()

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       char  DemoChar;
6       DemoChar = 'a';
7       printf("DemoChar is ");
8       putchar(DemoChar);
9       printf("\n1.User Input (getch) : ");
10      DemoChar=getch();
11      putchar(DemoChar);
12      printf("\n2.User Input (getche) : ");
13      DemoChar=getche();
14      putchar(DemoChar);
15      printf("\n3.User Input (getchar): ");
16      DemoChar=getchar();
17      putchar(DemoChar);
18  }
```

โปรแกรมที่ 5.8 โปรแกรมการใช้งานฟังก์ชัน putchar(), getchar(), getch(), getche()

ผลลัพธ์

DemoChar is a
1.User Input (getch) : P
2.User Input (getche) : ***P***P
3.User Input (getchar) : ***P***
P

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้นใต้***เป็นข้อมูลที่ป้อนเข้าไป

อธิบายการทำงาน

DemoChar is a

ผลลัพธ์จากฟังก์ชันบรรทัดที่ 9-11 printf("DemoChar is "); putchar(DemoChar); และขึ้นบรรทัดใหม่ด้วยฟังก์ชัน printf("\n");

1.User Input (getch) : P

ผลลัพธ์จากฟังก์ชัน printf("1.User Input (getch) : "); แสดงผลที่จอภาพและรอรับอักขระด้วยคำสั่ง DemoChar=getch(); รับอักขระที่พิมพ์โดยไม่แสดง แล้วแสดงผลจากตัวแปร P ด้วยฟังก์ชัน putchar(DemoChar);

2.User Input (getche) : ***P***P

ผลลัพธ์จากฟังก์ชัน printf("2.User Input (getche) : "); แสดงผลที่จอภาพและรอรับอักขระด้วยคำสั่ง DemoChar=getche(); รับอักขระที่พิมพ์โดยแสดง ***P*** แล้วแสดงผลจากตัวแปร P ด้วยฟังก์ชัน putchar(DemoChar);

3.User Input (getchar) : ***P***

ผลลัพธ์จากฟังก์ชัน printf("3.User Input (getchar) : "); แสดงผลที่จอภาพและรอรับอักขระด้วยคำสั่ง DemoChar=getchar(); รับอักขระที่พิมพ์ ***P*** แล้วกดปุ่ม Enter แล้วแสดงผลจากตัวแปร P ในบรรทัดใหม่ ด้วยฟังก์ชัน putchar(DemoChar);

ตัวอย่างที่ 5.12 เขียนโปรแกรมเพื่อคำนวณหาระยะทาง และความเร็วของรถคันหนึ่งซึ่งเคลื่อนที่จากหยุดนิ่งด้วยอัตราเร่ง 2 เมตร/วินาที^{2} เป็นเวลา 5 วินาที

วิธีคิด คำนวณหาระยะทาง จากสูตร $s = ut + \frac{1}{2} at^2$

เมื่อ s หมายถึง ระยะทาง มีหน่วยเป็น เมตร
u หมายถึง ความเร็วเริ่มต้น มีหน่วยเป็น เมตร/วินาที
a หมายถึง อัตราเร่ง มีหน่วยเป็น เมตร/วินาที^{2}
t หมายถึง เวลา มีหน่วยเป็น วินาที

จากโจทย์ u = 0 รถเคลื่อนที่จากหยุดนิ่ง ความเร็วเริ่มต้นเป็น 0
a = 2 อัตราเร่งของรถคันนี้
t = 5 เวลาในการเคลื่อนที่
$s = ut + \frac{1}{2} at^2$

แทนค่าในสูตร $s = 0 * 5 + \frac{1}{2} * 2 * 5^2$
$s = 25$

คำนวณหาความเร็ว จากสูตร $v = u + at$

เมื่อ v หมายถึง ความเร็ว มีหน่วยเป็น เมตร/วินาที
a หมายถึง อัตราเร่ง มีหน่วยเป็น เมตร/วินาที^{2}
t หมายถึง เวลา มีหน่วยเป็น วินาที

จากโจทย์ a = 2 อัตราเร่งของรถคันนี้
t = 5 เวลาในการเคลื่อนที่

แทนค่าในสูตร $v = 0 + 2 * 5$
$v = 10$ m/s

วิธีการเขียนโปรแกรมเพื่อให้สามารถนำมาใช้กับโจทย์ข้ออื่นได้ในทำนองเดียวกัน จะกำหนดให้รับอัตราเร่ง และเวลาในการเคลื่อนที่จากผู้ใช้ แล้วคำนวณหาระยะทาง และความเร็ว

ข้อมูลนำเข้า ประกอบด้วย อัตราเร่ง (a) กำหนดให้เป็นตัวเลขทศนิยม และเวลาในการเคลื่อนที่ (t) กำหนดให้เป็นตัวเลขทศนิยม และความเร็วเริ่มต้น (u) กำหนดให้เป็นตัวเลขทศนิยม

ผลลัพธ์ ประกอบด้วย ระยะทาง (s) กำหนดให้เป็น ตัวเลขทศนิยม และความเร็ว (v) กำหนดให้เป็น ตัวเลขทศนิยม

ความสัมพันธ์ของข้อมูล คำนวณจากสูตร

คำนวณระยะทางจากสูตร $s = ut + \frac{1}{2} at^2$

คำนวณความเร็วจากสูตร $v = u + at$

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  #include <math.h>
 4  void main()
 5  {
 6        float  s , v , u , a , t ;
 7        printf("Enter value of acceleration : ");
 8        scanf("%f" , &a);
 9        printf("Enter time of moving : ");
10        scanf("%f" , &t);
11        printf("Enter start of velocity : ");
12        scanf("%f" , &u);
13
14        s = u * t + 0.5 * a * pow(t,2)   ;
15        printf("Distance of moving is %.2f\n", s);
16        v = u + a * t ;
17        printf("Velocity of moving is %.2f\n", v);
18
19        getch();
20  }
```

โปรแกรมที่ 5.9 โปรแกรมเพื่อคำนวณหาระยะทาง

ผลลัพธ์

Enter value of acceleration : ***2***

Enter time of moving : ***5***

Enter start of velocity : ***0***

Distance of moving is 25.00

Velocity of moving is 10.00

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้นใต้***เป็นข้อมูลที่ป้อนเข้าไป

อธิบายการทำงาน

การคำนวณระยะทางจากสูตร $s = ut + \frac{1}{2} at^2$

ใช้ฟังก์ชัน pow(x,y) หมายถึง การคำนวณค่า x^y เช่น pow(5,2) = 25

การใช้ฟังก์ชัน pow(x,y) ต้องรวมคลังโปรแกรมโดยใช้คำสั่ง #include <math.h>

จากนั้นใช้ในสูตรการคำนวณโดยแทนค่าตัวแปร s = u * t + 0.5 * a * pow(t,2) ;

แทนค่าในสูตร $s = 0 * 5 + 0.5 * 2 * 5^2$

$s = 25.00$

แสดงผล Distance of moving is 25.00

สรุป

การรับข้อมูลด้วยฟังก์ชัน scanf() โดยกำหนดรูปแบบให้ตรงกับข้อมูลที่จะรับ เช่น %d ใช้กับตัวแปรที่เป็นชนิดตัวเลข %f ใช้กับตัวแปรที่เป็นทศนิยม %c ใช้กับตัวแปรที่เป็นชนิดอักขระ และ %s ใช้กับตัวแปรชนิดสายอักขระ ที่สำคัญหน้าตัวแปรต้องมีเครื่องหมาย & กำกับด้วย

การแสดงผลข้อมูลใช้ฟังก์ชัน printf() เพื่อแสดงผลของข้อความ และค่าในตัวแปรโดยกำหนดรูปแบบให้ตรงกับชนิดของตัวแปรที่ใช้ในการแสดง การแสดงผลข้อมูลด้วยฟังก์ชัน printf() สามารถแสดงผลประเภทข้อความ และค่าจากตัวแปรโดยมีอักษรกำหนดรูปแบบ และอักขระควบคุมการแสดงผล

การรับข้อมูลอักขระด้วยฟังก์ชัน getchar() getch() และ getche() มีลักษระการใช้งานแตกต่างกันออกไป ฟังก์ชัน getch() ไม่แสดงอักขระที่รับ ฟังก์ชัน getche() แสดงผลอักขระที่รับ และฟังก์ชัน getchar() แสดงอักขระที่รับ และจะต้องกดปุ่ม Enter ด้วย

การแสดงข้อมูลอักขระด้วยฟังก์ชัน putchar() ใช้แสดงผลอักขระออกทางจอภาพครั้งละ 1 อักขระ

แบบฝึกหัด

1. จงเขียนรูปแบบของฟังก์ชัน printf()
2. จงเขียนรูปแบบของฟังก์ชัน scanf()
3. จงอธิบายความแตกต่างของฟังก์ชัน getch(), getche() และ getchar()
4. เขียนโปรแกรมแสดงชื่อ-นามสกุล รหัสนักศึกษา และรายละเอียดของตัวเองบนจอภาพโดยใช้ฟังก์ชัน printf()
5. เขียนโปรแกรมรับค่าของตัวแปรชนิด int, char, และ float อย่างละ 1 จำนวนและแสดงค่าข้อมูลที่รับมาออกทางจอภาพให้เหมาะสม (บรรทัดละ 1 ข้อมูล)
6. เขียนโปรแกรมรับรหัสนักศึกษา 1 หลัก, ชื่อ-นามสกุล ของนักศึกษาโดยให้รับค่าเข้าไปและแสดงออกมาใหม่ด้วย
7. จงแสดงผลลัพธ์ของโปรแกรมดังต่อไปนี้

```
#include <stdio.h>
void main()
{
        int x = 8, y = 9;
        printf(“x\n”) ;
        printf(“x = %d”,x);
        printf(“x = \n %d”,y);
}
```

8. จงแสดงผลลัพธ์ของโปรแกรมดังต่อไปนี้

```
#include <stdio.h>
void main()
{
        char message[10] = “Test”
        printf(“\nHello  %s”,message);
        printf(“Hello %c”,message[1]);
        printf(“Hello \t Message”);
}
```

9. จงแสดงผลลัพธ์ของโปรแกรมดังต่อไปนี้

```
#include <stdio.h>
void main()
{
        int x = 8,y = 9;
        printf(“%d%d”,i,j) ;
}
```

10. เขียนโปรแกรมรับรหัสเอทีเอ็มจำนวน 4 ตัว โดยแสดงเป็นเครื่องหมาย * แทนรหัสเอทีเอ็มที่คีย์เข้าไป

Enter ATM password: ***** ผู้ใช้พิมพ์ 4567

เอกสารอ้างอิง

ประภาพร ช่างไม้. (2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส.

Brian W. Kernighan, Dennis M. Ritchie. (1988). *The C Programming Language*. 2nd Edition. London: Prentice Hall.

Delores M. Etter. (2013). Engineering Problem Solving with C. 4th Edition. London: Prentice Hall.

บทที่ 6
การเขียนโปรแกรมแบบมีเงื่อนไข

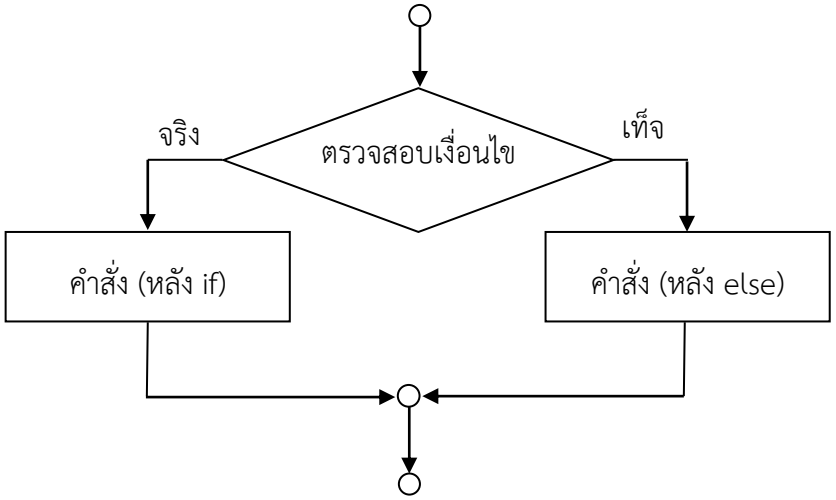
การเขียนโปรแกรมคอมพิวเตอร์จะต้องมีการพิจารณาให้ครอบคลุมในทุกๆ กรณีของการแก้ปัญหา ซึ่งบางปัญหาจำเป็นต้องแยกวิธีการในการดำเนินการให้สอดคล้องกับข้อมูลแต่ละประเภท ดังนั้นในการเขียนโปรแกรมคอมพิวเตอร์จึงมักมีการเขียนเงื่อนไขเข้ามาร่วมด้วยเสมอ ซึ่งวิธีการในการเขียนเงื่อนไขได้รับการพัฒนาเป็นคำสั่งในการควบคุมการทำงานของโปรแกรมประเภทหนึ่งนอกเหนือไปจากฟังก์ชันในการรับและแสดงผล ในบทนี้จะอธิบายการเขียนโปรแกรมแบบมีเงื่อนไขเบื้องต้น 2 คำสั่ง คือ คำสั่ง if-else และคำสั่ง switch

การเขียนโปรแกรมแบบมีเงื่อนไข

การเขียนโปรแกรมแบบมีเงื่อนไข (Condition) ต้องเกิดขึ้นเสมอในกรณี ที่มีการตัดสินใจเพื่อทำอย่างใดอย่างหนึ่ง หรือไม่กระทำอย่างหนึ่ง หรือมีเงื่อนไขเพื่อให้โปรแกรมทำตามเงื่อนไขที่ผู้ใช้ต้องการ คำสั่ง if-else สามารถสร้างเงื่อนไขการทำงาน ได้ครอบคลุมทุกกรณี มากกว่า น้อยกว่า เท่ากัน ไม่เท่ากัน ได้ครบทุกกรณี สามารถสร้างเงื่อนไขกรณีที่มีมากกว่า 1 เงื่อนไข โดยใช้ตัวดำเนินการและ (&&) ในกรณีที่ต้องการตรวจสอบเงื่อนไขทั้ง 2 เงื่อนไข ใช้ตัวดำเนินการหรือ (||) ในกรณีที่ตรวจสอบเงื่อนไขใดเงื่อนไขหนึ่ง ซึ่งสามารถแยกเป็นประเภทได้ดังนี้

1. เงื่อนไข 1 เงื่อนไขมีทางเลือก 2 ทาง

การเขียนโปรแกรมแบบมี 1 เงื่อนไข มีทางเลือก 2 ทางเลือก จะใช้คำสั่ง if-else ในการตัดสินใจเมื่อเงื่อนไขเป็นจริง (true) โปรแกรมจะเลือกทำงานคำสั่งหลัง if แต่หากเงื่อนไขเป็นเท็จ โปรแกรมจะเลือกทำงานคำสั่งหลัง else ซึ่งหลังคำสั่ง if-else จะเป็นคำสั่งเดียวหรือเป็นชุดคำสั่งก็ได้ ถ้าเป็นชุดคำสั่ง ใส่วงเล็บปีกกาครอบชุดคำสั่งไว้ หรือหลังคำสั่ง else เป็นคำสั่งเดียว หรือเป็นชุดคำสั่งก็ได้ ถ้าเป็นชุดคำสั่งใส่วงเล็บปีกกาครอบชุดคำสั่งไว้ และที่สำคัญเงื่อนไขต้องอยู่ภายในวงเล็บ



ภาพที่ 6.1 แสดงผังการไหลคำสั่ง if-else แบบ 2 ทางเลือก

จากภาพที่ 6.1 เมื่อตรวจสอบเงื่อนไข ผลการตรวจสอบเงื่อนไขเป็นจริง โปรแกรมจะทำงานหลังคำสั่ง if แต่ถ้าผลการตรวจสอบเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานหลังคำสั่ง else

คำสั่ง if-else ใช้ในการตรวจสอบเงื่อนไข เพื่อตรวจสอบเงื่อนไขของการทำงานของโปรแกรม โดยหากเงื่อนไขเป็นจริงจะมีการทำงานตามคำสั่งหลัง if และหากเงื่อนไขเป็นเท็จจะมีการทำงานหลังคำสั่ง else โดยมีรูปแบบทั่วไปของการใช้งานของคำสั่ง if-else เป็นดังนี้

```
if (expression)
        statement1;
else
        statement2;
```

ที่มา: Brian W. Kernigham, Dennis M. Ritchie. 1988: 21

คำสั่ง if สิ่งสำคัญที่สุดคือ expression คือเงื่อนไขการทำงานที่สร้างขึ้น โดยโปรแกรมจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำคำสั่ง statement1; เป็นเป็นคำสั่งหลัง if และไม่ทำคำสั่ง statement2; กรณีตรงข้ามกัน ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำคำสั่ง statement2; เป็นเป็นคำสั่งหลัง else และไม่ทำคำสั่ง statement1;

คำสั่ง if-else จะมีผลครอบคลุมเฉพาะคำสั่งเพียงคำสั่งเดียวเท่านั้นที่อยู่ถัดไปจากคำสั่ง if หรือ else เท่านั้น

หากต้องการให้โปรแกรมทำตามคำสั่งมากกว่า 1 คำสั่งให้ใส่เครื่องหมาย { และ } ครอบคำสั่งที่ต้องการให้โปรแกรมทำงานจะทำให้ตัวแปลภาษาซี จะทำคำสั่งที่อยู่ระหว่าง { และ } เป็นชุดคำสั่งเดียวกัน

```
if (expression)
{
        statement1;
        statement2;
        ...
}
else
{
        statement3;
        statement4;
        ...
}
```

ชุดคำสั่ง หมายถึง คำสั่งที่มีได้มากกว่า 1 คำสั่ง เช่น

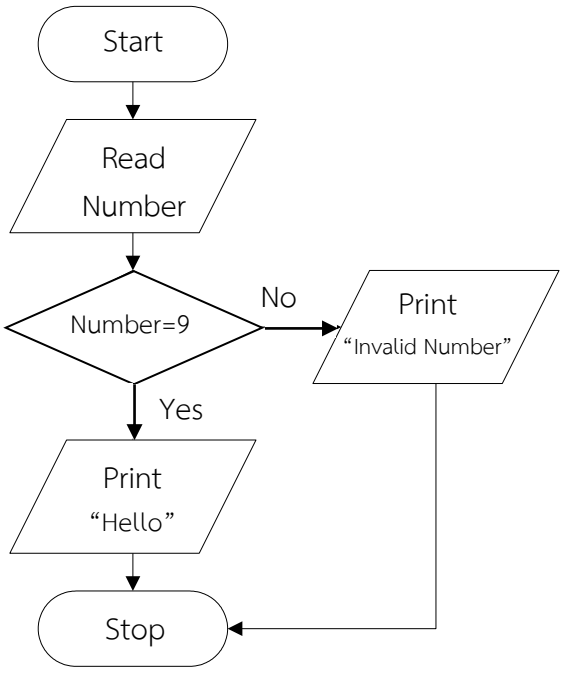
```
if (x > 5)
{
   x = x + 10 ;
   printf (“Thank you”) ;
}
```

ตัวอย่างที่ 6.1 สมมติว่าต้องการเขียนโปรแกรมเพื่อแสดงข้อความตามเงื่อนไขดังนี้

- ถ้าผู้ใช้ป้อนเลข 9 แสดงข้อความว่า “Hello”
- ถ้าผู้ใช้ป้อนเลขอื่นใด หรืออาจเป็นตัวอักษร ให้แสดงข้อความว่า “Invalid Number”

วิธีคิด จะต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลข้อความที่จะแสดงผลซึ่งมี 2 ข้อความ
2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกชัดเจนว่าให้รับตัวเลขมาจาก Keyboard แต่ต้องคิดเพิ่มเติมว่าจะประกาศตัวแปรเป็นชนิดอะไรจึงเหมาะสมที่สุด ควรประกาศเป็น char เพราะโจทย์บอกว่าอาจป้อนตัวอักษรได้ด้วย
3. วิเคราะห์ความสัมพันธ์ของข้อมูล ข้อนี้โจทย์บอกความสัมพันธ์ระหว่างที่มาและผลลัพธ์ไว้ชัดเจน
4. ทำการเขียนผังงาน Flowchart ได้ดังภาพที่ 6.1 และนำไปเขียนโปรแกรมได้ดังโปรแกรมที่ 6.1



ภาพที่ 6.2 ผังงานโปรแกรมเพื่อแสดงข้อความ

```
1   /* Demostation of if-else... */
2   #include <stdio.h>
3
4   void main()
5   {
6           char Ch;
7           printf("Enter Number: ");
8           scanf("%c" , &Ch);
9           if (Ch=='9')
10                  printf("Hello");
11          else
12                  printf("Invalid Number");
13  }
```

โปรแกรมที่ 6.1 รหัสต้นฉบับของโปรแกรมในการแก้ปัญหาตามตัวอย่างที่ 6.1

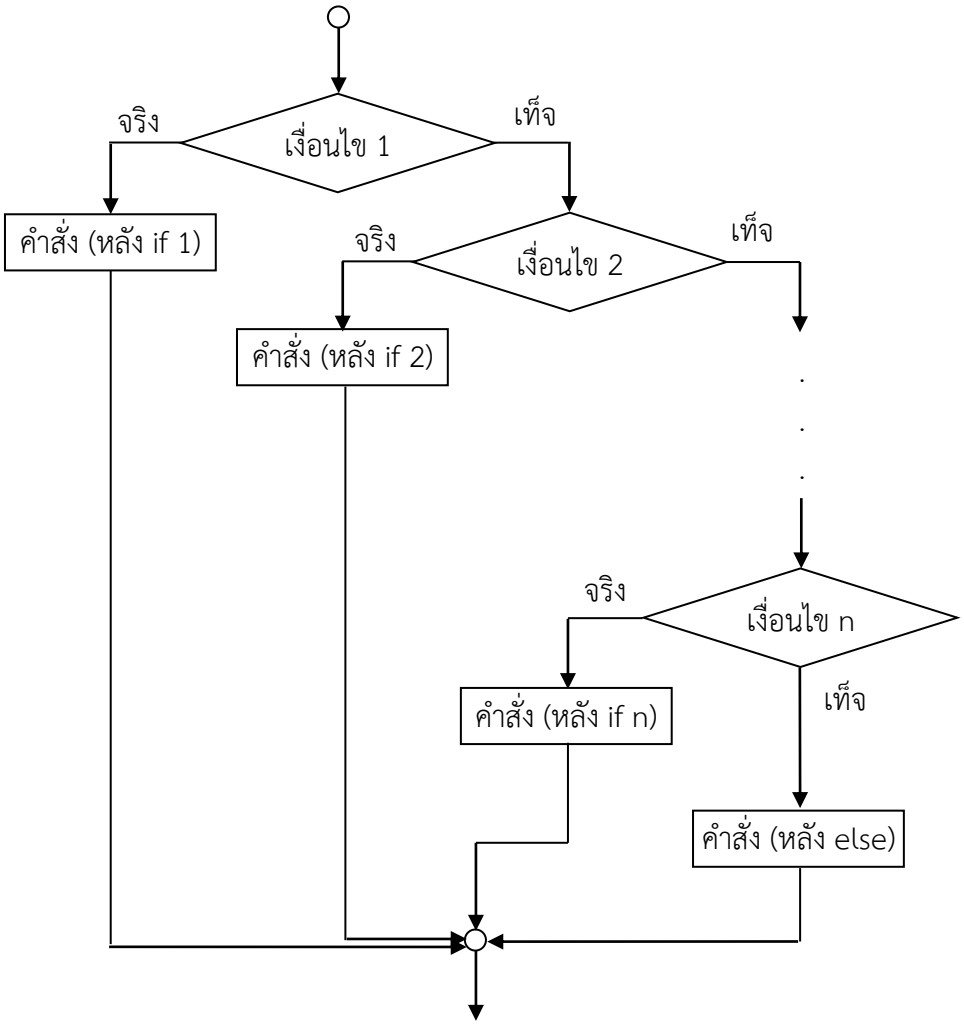
จากรหัสต้นฉบับที่ 1 ในบรรทัดที่ 9 และ 11 นั้นจะพบว่าการเปรียบเทียบระหว่างค่าของตัวแปร Ch กับตัวเลขนั้นจะต้องทำการใส่เครื่องหมายฝนทอง ครอบเลข 9 ไว้ด้วย เพื่อระบุว่าค่าที่ต้องการเปรียบเทียบนั้นเป็นอักขระ เพราะอ่านค่าด้วยฟังก์ชัน scanf() โดยใช้ %c เนื่องจากการเปรียบเทียบต้องทำระหว่างตัวแปรชนิดเดียวกันเท่านั้น

หมายเหตุ การใช้คำสั่ง if-else นั้นคำสั่ง if หรือ else ไม่จบท้ายคำสั่งด้วยเครื่องหมายอัฒภาค ; ส่วนบรรทัดอื่นๆ ยังคงต้องจบด้วยเครื่องหมายอัฒภาค ; ตามปกติ

2. เงื่อนไขมากกว่า 1 เงื่อนไข

การเขียนโปรแกรมที่มีเงื่อนไขมีมากกว่า 1 เงื่อนไข จึงต้องมีคำสั่ง else if เพื่อตัดสินใจเลือก จากทางเงื่อนไขมากกว่า 1 เงื่อนไข เช่น การตัดเกรดนักศึกษา เป็นกรณีที่เห็นได้ชัดเจนว่า ต้องมีเงื่อนไขในการกำหนดเกรด แต่ละเกรด ซึ่งไม่สามารถใช้ เงื่อนไขเดียวเพื่อตัดเกรดนักศึกษา 5 เกรดได้ เช่น เกรด A B C D และ F

การทำงานของคำสั่ง if นั้น โปรแกรมจะทำงานตรวจสอบเงื่อนไขที่ 1 ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำงานตามคำสั่งหลัง if หากเงื่อนไขที่ 1 ไม่เป็นจริง โปรแกรมจะตรวจสอบเงื่อนไขที่ 2 ต่อไป หากตรวจสอบเงื่อนไขที่ 2 เป็นจริง โปรแกรมจะทำงานตามคำสั่ง หลังเงื่อนไขที่ 2 หากเงื่อนไขที่ 2 ไม่เป็นจริง จะทำการตรวจสอบเงื่อนไขต่อไป ถ้าผลการตรวจสอบเงื่อนไขเป็นจริง โปรแกรมจะทำงานตามคำสั่ง แต่ถ้าผลการตรวจสอบเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานตามคำสั่งหลัง else ดังภาพที่ 6.2



ภาพที่ 6.3 แสดงผังการไหลคำสั่ง else if แบบหลายทางเลือก

ภาพที่ 6.2 เห็นได้ว่า เมื่อตรวจสอบเงื่อนไขที่ 1 ถ้าผลการตรวจสอบเงื่อนไขที่ 1 เป็นจริง โปรแกรมจะทำงานหลังคำสั่ง if เงื่อนไขที่ 1 แต่ถ้าผลการตรวจสอบเงื่อนไขที่ 1 เป็นเท็จ โปรแกรมจะตรวจสอบเงื่อนไขที่ 2 ต่อไป ถ้าผลการตรวจสอบเงื่อนไขที่ 2 เป็นจริง โปรแกรมจะทำงานหลังคำสั่ง if เงื่อนไขที่ 2 แต่ถ้าผลการตรวจสอบเงื่อนไขที่ 2 เป็นเท็จ โปรแกรมจะตรวจสอบเงื่อนไขไปเรื่อย จนถึงเงื่อนไขสุดท้าย เงื่อนไขที่ n ถ้าผลการตรวจสอบเงื่อนไขที่ n เป็นจริง โปรแกรมจะทำงานหลังคำสั่ง if เงื่อนไขที่ n แต่ถ้าผลการตรวจสอบเงื่อนไขที่ n เป็นเท็จ โปรแกรมจะทำงานหลังคำสั่ง else

รูปแบบคำสั่ง else-if มีดังนี้

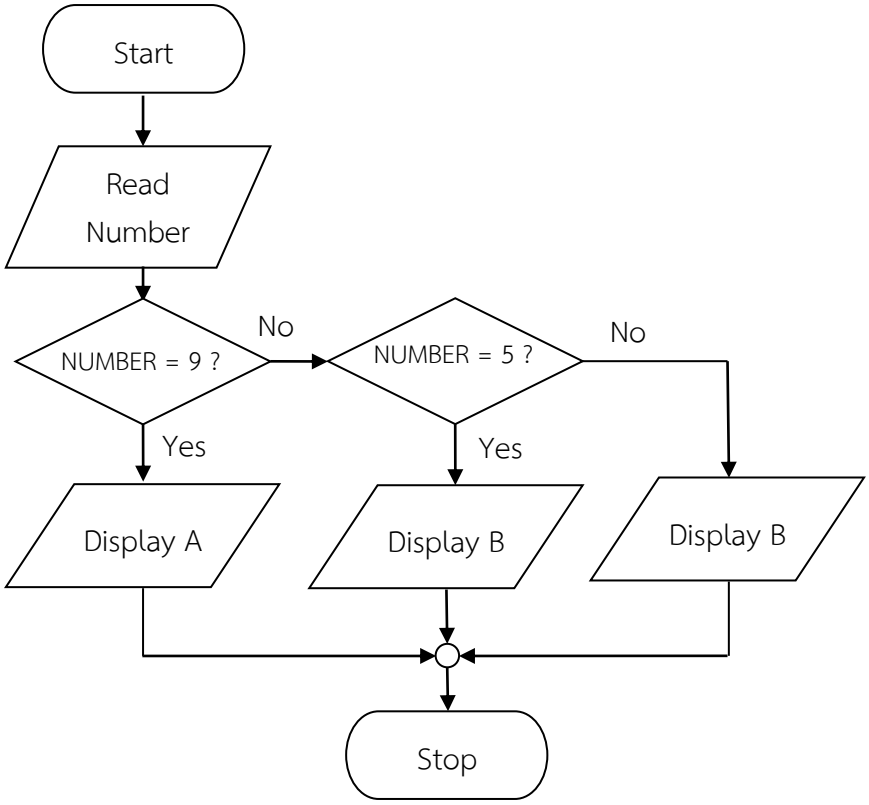
```
if (เงื่อนไขที่ 1)
{
        <ชุดคำสั่งเมื่อเงื่อนไขที่ 1 ตรวจสอบแล้วเป็นจริง>
}
else if (เงื่อนไขที่ 2)
{
        <ชุดคำสั่งเมื่อเงื่อนไขที่ 2 ตรวจสอบแล้วเป็นจริง>
}
.
.
.
else if (เงื่อนไขที่ n)
{
        <ชุดคำสั่งเมื่อเงื่อนไขที่ n ตรวจสอบแล้วเป็นจริง>
}
else
{
        <ชุดคำสั่งเมื่อไม่ตรงกับเงื่อนไขใดๆ >
}
```

กรณีที่ต้องการตรวจสอบเงื่อนไข 2 เงื่อนไขพร้อมกัน จะใช้สัญลักษณ์ || เป็นสัญลักษณ์ OR ใช้สัญลักษณ์ vertical slashes กรณีที่ต้องการตรวจสอบเงื่อนไขเป็นจริงเงื่อนไขใดเงื่อนไขหนึ่ง ถ้าต้องการตรวจสอบเป็นจริงทั้ง 2 เงื่อนไขใช้สัญลักษณ์ && เป็นสัญลักษณ์ AND เช่น i<=5 && i>=2 กรณีที่ต้องการตรวจสอบทั้ง 2 เงื่อนไขต้องเป็นจริง หรือกรณีที่ต้องการตรวจสอบเงื่อนไขเป็นจริงเงื่อนไขใดเงื่อนไขหนึ่ง x > 5 || x < 0 เป็นต้น

ตัวอย่างที่ 6.2 จากตัวอย่างที่ 6.1 ถ้าต้องการกำหนดเงื่อนไขเป็น 3 เงื่อนไขดังนี้

- ถ้าผู้ใช้ป้อนเลข 9 แสดงข้อความว่า “Hello”
- ถ้าผู้ใช้ป้อนเลข 5 แสดงข้อความว่า “Good Bye”
- ถ้าผู้ใช้ป้อนเลขอื่นใด ให้แสดงข้อความว่า “Invalid Number”

วิธีคิด แนวคิดจะใกล้เคียงกับตัวอย่างที่ 6.1 แต่เพิ่มคำสั่งในส่วนที่เป็นเงื่อนไขเป็น 3 เงื่อนไข



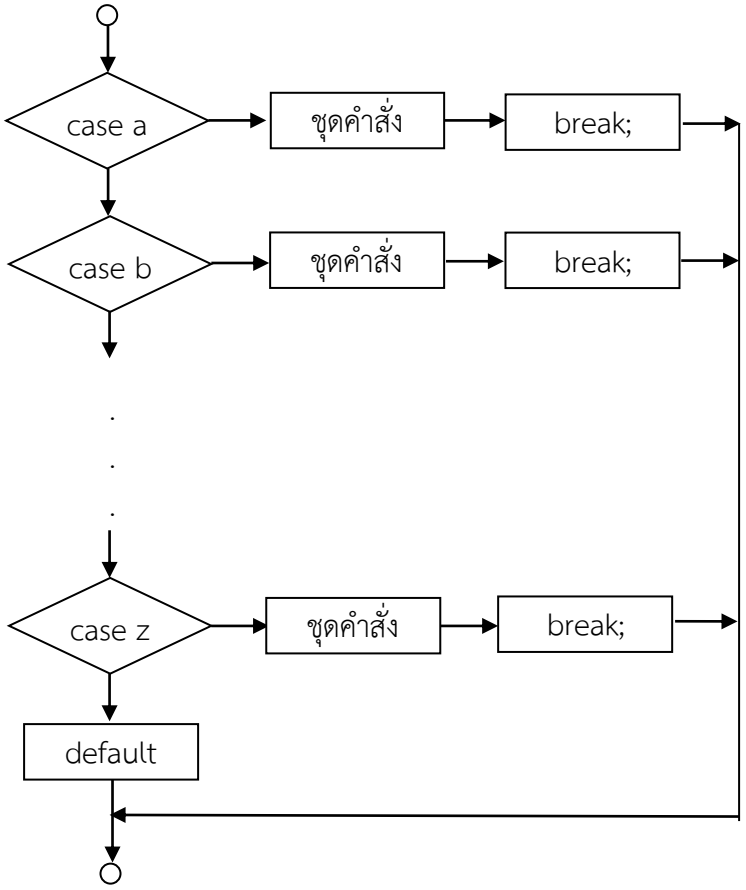
ภาพที่ 6.4 ผังงานโปรแกรมตัวอย่างคำสั่ง else if แบบหลายทางเลือก

```
1  #include <stdio.h>
2  void main()
3  {
4          char Ch;
5          printf("Enter Number: ");
6          scanf("%c",&Ch);
7          if (Ch=='9')
8                  printf("Hello");
9          else if (Ch=='5')
10                 printf("Good Bye");
11         else
12                 printf("Invalid Number");
13 }
```

โปรแกรมที่ 6.2 รหัสต้นฉบับของโปรแกรมในการแก้ปัญหาตามตัวอย่างที่ 6.2

การเขียนโปรแกรมแบบมีเงื่อนไขโดยใช้คำสั่ง switch

คำสั่ง switch ในภาษาซีสามารถสร้างเงื่อนไขโดยการเปรียบค่าคงที่ เปรียบเทียบค่าตัวแปร เปรียบเทียบค่าตัวเลข และเปรียบเทียบค่าอักขระ โดยมีการทำงานตามผังงานดังนี้



ภาพที่ 6.5 แสดงผังการไหลคำสั่ง switch
ที่มา: Harvey M. Deitel, Paul J. Deitel. 2004: 115

จากภาพที่ 6.3 เมื่อตรวจสอบเงื่อนไข ตรงกับเงื่อนไขใด โปรแกรมตามเงื่อนไขนั้น ๆ ถ้าไม่ตรงกับเงื่อนไขใดเลย โปรแกรมจะทำงานหลังคำสั่ง default โดยมีรูปแบบการใช้งานดังนี้

รูปแบบคำสั่ง switch

คำสั่ง switch นั้นเป็นคำสั่งในการตัดสินใจเช่นเดียวกับคำสั่ง if-else โดยคำสั่ง switch มีรูปแบบการเปรียบเทียบค่าตัวแปรกับค่าคงที่ในกรณีต่าง ๆ ดังรูปแบบต่อไปนี้

```
รูปแบบ          : switch (ตัวแปร)
                  {
                   case ค่าคงที่ 1 : คำสั่งที่ให้ดำเนินการเมื่อกรณีที่ 1 เป็นจริง ;
                                     break ;
                   case ค่าคงที่ 2 : คำสั่งที่ให้ดำเนินการเมื่อกรณีที่ 2 เป็นจริง ;
                                     break ;
                                      ⋮
                    default : คำสั่งที่ให้ดำเนินการเมื่อไม่มีกรณีใดเป็นจริง ;
                                     break;
                      }
หมายเหตุ:    ขอให้สังเกตว่าหลังบรรทัดคำสั่ง switch ไม่มีเครื่องหมายอัฒภาค ;
                  :    ต้องมี { } ต่อจากคำสั่ง switch() ด้วยเสมอ
                  :   switch นั้นจะใช้ตัวแปรชนิด char หรือ int เพื่อมาดูว่ามีค่าตรงกับค่าที่กำหนดไว้กรณีใดก็ทำตามเงื่อนไขในกรณีนั้น  แต่ถ้าไม่ตรงกรณีใดเลยจะทำในคำสั่ง default
                  :    default นั้นต้องอยู่สุดท้ายเสมอ
```

คำสั่ง switch นั้นจะเห็นได้ว่าโครงสร้าง switch นั้นจะต้องทำการเปรียบเทียบในลักษณะที่เงื่อนไขที่กำหนดใน switch() จะต้องเป็นไปตามค่าที่กำหนดไว้ใน case เท่านั้น ไม่สามารถใช้การเปรียบเทียบในลักษณะที่น้อยกว่า มากกว่า หรือไม่เท่ากันได้ โดยเมื่อคำสั่ง switch ไปตรวจสอบแล้วพบว่าตัวแปร (เงื่อนไข) ที่กำหนดมีค่าตรงกับ case ใดก็จะไปทำเงื่อนไขตาม case นั้น ซึ่งอาจมีมากกว่า 1 คำสั่ง ดังนั้นจึงต้องทำการบอกตัวแปรภาษาด้วยว่าจบ case ที่ต้องการให้ทำงานแล้วด้วยคำสั่ง break; ซึ่งเมื่อเจอคำสั่งนี้ตัวแปลภาษาจะทำไปทำคำสั่งถัดไปหลังจากวงเล็บปีกกาปิดของคำสั่ง switch ทันที แต่หากไม่บอกการจบคำสั่ง case หนึ่งๆ ตัวแปรภาษาจะถือว่าไม่จบ และจะไปทำคำสั่งในส่วนของ case อื่นๆ ต่อไป จนกว่าจะเจอเครื่องหมาย วงเล็บปีกกาปิดของคำสั่ง switch หรือเจอคำสั่ง break;

ส่วนคำสั่งที่อยู่หลัง default นั้น จะเกิดการทำงานก็ต่อเมื่อ เงื่อนไขที่กำหนดไม่เป็นไปตาม case ที่ระบุใดๆ เลย ซึ่งส่วน default นี้จะต้องอยู่หลังสุดของเงื่อนไขเสมอ และเมื่อทำงานเสร็จ จะกระโดดไปทำงานต่อที่บรรทัดถัดไปหลังจากเครื่องหมายวงเล็บปีกกาปิดของคำสั่ง switch โดยได้รับการยกเว้นว่าไม่ต้องมี break;

สำหรับตัวอย่างที่ 6.2 นั้นจะเห็นว่าค่าของตัวแปร Ch นั้นมีค่าที่คงที่และแยกเป็นกรณีๆ ไป คือ ค่า 9 ค่า 5 และค่าอื่นๆ ดังนั้นจึงสามารถใช้คำสั่ง switch ในการเขียนการตัดสินใจนี้ได้ ดังแสดงในโปรแกรมที่ 6.3

```
1   /* Demostation of switch */
2   #include <stdio.h>
3
4   void main()
5   {
6           char Ch;
7           printf("Enter Number: ");
8           scanf("%c",&Ch);
9           switch(Ch)  {
10              case '9': printf("Hello");
11                      break;
12              case '5': printf("Good Bye");
13                      break;
14              default: printf("Invalid Number");
15          }
16  }
```

โปรแกรมที่ 6.3 รหัสต้นฉบับของโปรแกรมในการแก้ปัญหาที่ 1 โดยการใช้คำสั่ง switch

หมายเหตุ

สังเกตว่าทุกๆ กรณีที่อยู่หลังคำสั่ง case จะต้องมีคำสั่ง break; เพื่อให้โปรแกรมข้ามไปทำงานยังบรรทัดที่อยู่ถัดจากวงเล็บปีกกาปิดของ case เสมอ หากไม่มีจะทำให้โปรแกรมกระโดดไปทำงานต่อยังกรณีถัดไป ซึ่งเป็นสิ่งที่ไม่ต้องการ

ข้อดีของคำสั่ง if-else และคำสั่ง switch

คำสั่ง switch มีความเป็นระเบียบและตรวจสอบได้ของโปรแกรมมากกว่าการใช้โครงสร้าง if-else เมื่อมีเงื่อนไขที่เป็นค่าคงที่จำนวนมาก เนื่องจากการตัดสินใจไม่มีการตัดสินใจใดๆ ที่อยู่ภายใต้การตัดสินใจก่อนหน้า

คำสั่งที่อยู่หลัง case มีได้ไม่จำกัดจำนวนของคำสั่ง โดยไม่ต้องใส่ { และ } เพื่อรวมคำสั่ง แต่ทั้งนี้จะต้องจบคำสั่งในแต่ละเงื่อนไขด้วย break; ยกเว้น case สุดท้ายของคำสั่ง switch()

ข้อจำกัดของคำสั่ง if-else และคำสั่ง switch

คำสั่ง if-else ปฏิบัติตามคำสั่งที่อยู่ภายหลังคำสั่ง if หรือ else เพียง 1 คำสั่งเท่านั้น หากต้องการให้ปฏิบัติตามคำสั่งมากกว่า 1 คำสั่ง ต้องใส่ { และ } คร่อมระหว่างคำสั่ง

คำสั่ง switch ไม่สามารถระบุค่าเป็นช่วงๆ ของแต่ละ case ได้

ตัวอย่างที่ 6.3 ให้นักศึกษาเขียนโปรแกรมเพื่อแสดงผลว่าค่าตัวเลขจำนวนเต็มที่ผู้ใช้ป้อนมีค่าเป็นเลขคู่ หรือเลขคี่ โดยหากเป็นเลขคู่ให้เลขข้อความว่า “Even Number” แต่หากเป็นเลขคี่ให้แสดงข้อความว่า “Odd Number”

แนวคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลข้อความที่จะแสดงผลซึ่งมี 2 ข้อความคือ Odd Number และ Even Number

2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกชัดเจนว่าให้รับตัวเลขมาจาก Keyboard แต่ต้องคิดเพิ่มเติมว่าจะประกาศตัวแปรเป็นชนิดอะไรจึงเหมาะสมที่สุด

3. วิเคราะห์ความสัมพันธ์ของข้อมูล ไม่ได้บอกต้องคิดเอาเองว่าจะรู้ได้อย่างไรว่าเลขคู่หรือเลขคี่

กระบวนการทางคณิตศาสตร์ที่สามารถทำให้คอมพิวเตอร์แยกได้ว่าตัวเลขตัวใดเป็นเลขคู่และตัวใดเป็นเลขคี่ (เนื่องจากไม่สามารถกำหนดตายตัวได้ว่าถ้าเป็นเลข 1 ให้เป็นเลขคี่, เลข 2 เป็นเลขคู่, เลข 3 เป็นเลขคี่เป็นต้น)

เลขดูว่าเลขคู่มีอะไรบ้าง 0 2 4 6 8 10 12..... (จะหาร 2 ลงตัว)

เลขคี่มีอะไรบ้าง 1 3 5 7 9 11 13..... (จะหาร 2 ไม่ลงตัว)

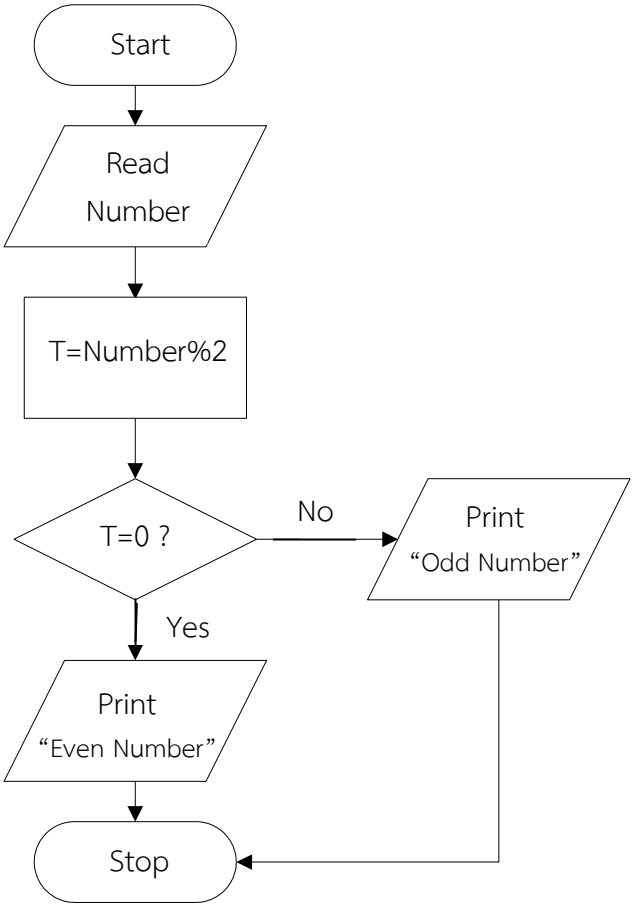
แล้วจะรู้ได้ยังไงว่าหารลงตัวหรือไม่ลงตัว ใช้ตัวดำเนินการอะไรดี (คำตอบก็คือ % ถ้าหาร 2 ลงตัว ก็ต้อง %2 == 0 เพื่อความไม่สับสนจะเขียนเป็น

T = Number %2; (เมื่อ Number เป็นตัวเลขที่ได้จากการรับค่าทาง Keyboard และ T เป็นเศษจากการหาร)

If (T==0) ต้องดูว่า T เป็น 0 หรือเปล่า 0 -> เลขคู่ 1-> เลขคี่

กรณีนี้จะใช้ switch ตรวจสอบ ว่า T มีค่าเท่ากับ 0 หรือไม่ก็ได้ดังโปรแกรมที่ 6.5

เขียนผังงานได้ดังตัวอย่างที่ 6.3 และนำไปเขียนโปรแกรมได้ดังโปรแกรมที่ 6.4 เขียนโดยใช้คำสั่ง if และโปรแกรมที่ 6.5 เขียนโดยใช้คำสั่ง switch



ภาพที่ 6.6 ผังงานโปรแกรมเลขคู่เลขคี่

เขียนโปรแกรมโดยใช้คำสั่ง if

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int Number;
6           int T;
7           clrscr();
8           printf("Enter Your Number: ");
9           scanf("%d",&Number);
10          T = Number % 2;
11          if (T==0)  printf("%d is Even Number ",Number);
12          else       printf("%d is Odd Number ",Number);
13
14  }
```

โปรแกรมที่ 6.4 รหัสต้นฉบับของโปรแกรมในการแก้ปัญหาโดยการใช้คำสั่ง if-else

เขียนโปรแกรมโดยใช้คำสั่ง switch

```
1  #include <stdio.h>
2  #include <conio.h>
3  void main()
4  {
5          int Number;
6          int T;
7          clrscr();
8          printf("Enter Your Number: ");
9          scanf("%d",&Number);
10         T = Number % 2;
11         switch (T)  {
12             case 0 :  printf("%d is Even Number ",Number);
13                       break;
14             default :  printf("%d is Odd Number ",Number);
15 }
```

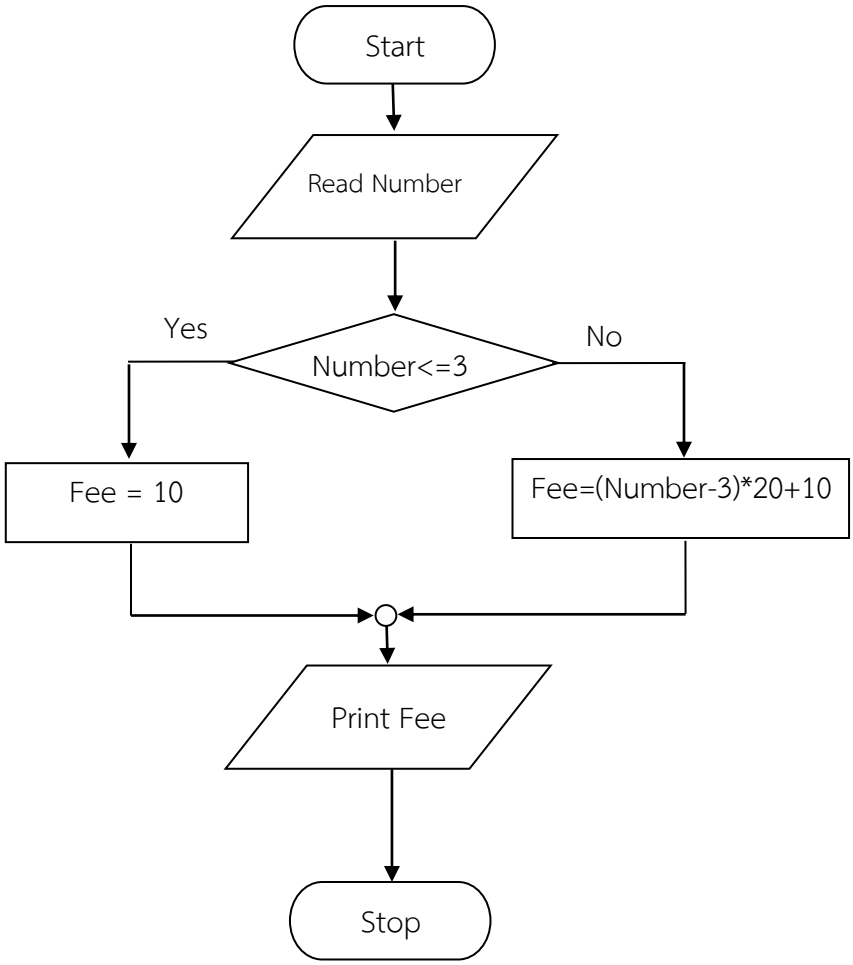
โปรแกรมที่ 6.5 รหัสต้นฉบับของโปรแกรมในการแก้ปัญหาโดยการใช้คำสั่ง switch case

ตัวอย่างที่ 6.4 เขียนโปรแกรมเพื่อคำนวณค่าจอดรถในห้างสรรพสินค้าแห่งหนึ่ง โดยคำนวณจากจำนวนชั่วโมงที่ลูกค้าเข้ามา ลูกค้าจอด 3 ชั่วโมงแรกคิดค่าจอดรถ 10 บาท เกินจาก 3 ชั่วโมงคิดชั่วโมงละ 20 บาท

แนวคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือค่าจอดรถเป็นตัวเลข
2. วิเคราะห์ที่มาของข้อมูล รับค่าจำนวนชั่วโมงที่ลูกค้าจอดรถ เป็นตัวเลข
3. วิเคราะห์ความสัมพันธ์ของข้อมูล คำนวณค่าจอดรถโดยแยกกรณีคือ
   กรณีที่ลูกค้าจอดรถไม่เกิน 3 ชั่วโมง ค่าจอดรถ = 10 บาท
   กรณีที่ลูกค้าจอดรถเกิน 3 ชั่วโมง คำนวณค่าจอดรถ
   เช่น จอดรถ 5 ชั่วโมง 3 ชั่วโมงแรก ค่าจอดรถ 10 บาทเกิน 2 ชั่วโมง คิดชั่วโมงละ 20 บาท เป็นเงิน 40 บาทรวมเป็น 50 บาท
   ค่าจอดรถ = (5 - 3) * 20 + 10 = 50

เขียนผังงานได้ดังนี้



ภาพที่ 6.7 ผังงานโปรแกรมคำนวณค่าจอดรถ

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int Number ;
6           int Fee ;
7           clrscr();
8           printf("Enter number of hour : ");
9           scanf("%d",&Number);
10
11          if (Number <= 3)
12                Fee = 10 ;
13          else
14                Fee =  (Number - 3) * 20 + 10 ;
15
16          printf("Fee  = %d " , Fee);
17  }
```

โปรแกรมที่ 6.6 รหัสต้นฉบับของโปรแกรมในการปัญหาตัวอย่างที่ 6.4

สรุป

การตรวจสอบเงื่อนไข ของโปรแกรมภาษาซีมี 2 คำสั่งด้วยกันคือ คำสั่ง if-else และคำสั่ง switch ซึ่งลักษณะการใช้งานตกต่างกันออกไป คำสั่ง if สามารถตรวจสอบเงื่อนไขเปรียบเทียบค่า มากกว่า น้อยกว่า เท่ากัน ได้ ส่วนคำสั่ง switch จะตรวจสอบเงื่อนไขได้เฉพาะค่าเท่ากัน และตัวแปรที่ใช้กับคำสั่ง switch ใช้กับตัวแปร จำนวนเต็มและตัวอักษร ไม่สามารถใช้กับตัวแปรเลขทศนิยมได้ แต่คำสั่ง if-else สามารถใช้เปรียบเทียบค่าได้ในทุกตัวแปร

รูปแบบคำสั่ง if-else สิ่งที่สำคัญที่สุดคือ การกำหนดเงื่อนไข ต้องพิจารณาจากโจทย์ว่าสิ่งใดเป็นเงื่อนไข ต่อมาพิจารณาตัวดำเนินการว่าควรเป็นตัวดำเนินการตัวใด มากกว่า น้อยกว่า เท่ากัน หรือกรณีที่มีเงื่อนไขมากกว่าหนึ่งเงื่อนไข ควรสร้างเงื่อนไขให้รองรับการเกิดขึ้นของข้อมูลในทุกกรณี ถ้าเงื่อนไขเป็นจริง คำสั่งที่ให้ทำงานมีกี่คำสั่งอะไรบ้างเขียนไว้หลังคำสั่ง if ถ้าเงื่อนไขเป็นเท็จ คำสั่งที่ให้ทำงานมีกี่คำสั่งอะไรบ้างเขียนไว้หลังคำสั่ง else ควรใช้เป็นคำสั่งเดียว หรือชุดคำสั่ง

แบบฝึกหัด

1. ให้เขียนโปรแกรมในการรับตัวเลข 2 จำนวนและตอบว่าเลขจำนวนใดมีค่ามากกว่ากัน

2. ให้นักศึกษาเขียนโปรแกรมในการคิดระดับคะแนน (เกรด) ของนักศึกษา เมื่อทำการป้อน คะแนนกลางภาค และคะแนนสอบปลายภาค โดยมีหลักเกณฑ์การให้ระดับคะแนนดังนี้

ระดับคะแนน A ได้คะแนนรวม 80 คะแนนขึ้นไป
ระดับคะแนน B ได้คะแนนรวม 70 คะแนนขึ้นไป
ระดับคะแนน C ได้คะแนนรวม 60 คะแนนขึ้นไป
ระดับคะแนน D ได้คะแนนรวม 50 คะแนนขึ้นไป
ระดับคะแนน E ได้คะแนนรวม ไม่ถึง 50 คะแนน

3. ให้เขียนโปรแกรมแก้ปัญหาต่อไปนี้ โดยใช้คำสั่ง switch... case

(ก) แก้ปัญหาการคิดระดับคะแนน
(ข) ปัญหาการคิดเลขคู่ / เลขคี่

4. ให้เขียนโปรแกรมในการคิดส่วนลดแก่ลูกค้าแห่งหนึ่งเมื่อมีมูลค่าการสั่งซื้อและส่วนลดต่าง ๆ ดังนี้

(ก) มูลค่าการซื้อสินค้าน้อยกว่า 2,000 บาท ส่วนลด 5 %
(ข) มูลค่าการซื้อสินค้า 2,000 – 10,000 บาท ส่วนลด 15 %
(ค) มูลค่าการซื้อสินค้า 10,001 บาทขึ้นไป ส่วนลด 30 %

โดยโปรแกรมจะต้องแสดงการทำงานดังตัวอย่าง

Enter sale money: ***10000***
Discount Rate = 30.00 %

Discount money = 3000.00 bat
Total money pay = 7000.00

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

5. เขียนโปรแกรมรับชื่อพนักงาน และ รับเงินเดือน จากนั้นให้คำนวณ หากเงินเดือนมากกว่า 10000 ให้ขึ้นเงินเดือนให้ 10 % หากน้อยกว่า หรือเท่ากับ 10000 ให้ขึ้นเงินเดือน 20 % แล้วแสดงผลลัพธ์ ชื่อพนักงาน เงินเดือนเก่า ได้ขึ้นเงินเดือนเท่าไร และรวมแล้วเงินเดือนใหม่เท่าไร

6. เขียนโปรแกรม รับตัวเลขมา 1 จำนวน แล้วแสดงผลลัพธ์หากหารด้วย 7 ลงตัว และแสดง เศษหากการด้วย 7 ไม่ลงตัว

7. เขียนโปรแกรม รับชื่อ และ อายุ แล้วแสดงว่า

หาก อายุ น้อยกว่า 20 แสดงว่า ยังเป็นเด็ก
หาก อายุ 20 – 30 แสดงว่า คุณเป็นผู้ใหญ่แล้ว
หาก อายุ มากกว่า 30 แสดงว่า คุณแก่แล้ว

8. เขียนโปรแกรมรับค่าตัวเลข 1 – 7 แล้ว แปลงเป็นวันจันทร์ – อาทิตย์

9. จงเขียนโปรแกรมเพื่อ

9.1 รับค่ารหัสสินค้า ชื่อสินค้า จำนวนสินค้าที่ขาย ราคาต่อหน่วย จากแป้นพิมพ์ แล้วคำนวณหาราคารวม

9.2 คำนวณหาส่วนลด โดยหากซื้อสินค้าราคาตั้งแต่ 50,000 บาท ขึ้นไป ลดให้ 30 % หากซื้อไม่ถึง 50,000 บาท ลดให้เพียง 20 %

9.3 คำนวณหาภาษี 7 % และแสดงผลเป็น รหัสสินค้า ชื่อสินค้า จำนวนสินค้าที่ขาย และราคาต่อหน่วย ราคาสินค้าที่ขายได้ ส่วนลด และราคาหลังหักส่วนลด,ภาษี และ ราคาสุทธิ

โดยโปรแกรมจะต้องแสดงการทำงานดังตัวอย่าง

Enter Your Product Code: ***PRN001***

Enter Your Product Name: ***Printer***

Enter Unit Price: *5500*

Enter Amount of Printer: ***10***

Code	Name	Unit Price	Amount	Total
PRN001	Printer	5500	10	55000
			Discount 30 %	16500
			Price after Discount Vat 7 %	38500
			Price after discount Vat 7 %	2695
			Net Price	41195

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

10. คำนวณหาเงินสมทบกองทุนประกันสังคม โดยมีเงื่อนไข คิดคำนวณ 5 % ของเงินเดือนพนักงาน โดยเงินสมทบกองทุนประกันสังคม ไม่เกิน 750 บาท ตัวอย่างเช่น

เงินเดือน 10,000 บาท ต้องจ่ายเงินสมทบกองทุนประกันสังคม เท่ากับ 500 บาท

เงินเดือน 20,000 บาท ต้องจ่ายเงินสมทบกองทุนประกันสังคม เท่ากับ 750 บาท

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร:ดวงกมลสมัย.

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึ่ม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

ประภาพร ช่างไม้.(2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส.

อนรรฆนงค์ คุณมณี. (2551). *Basic of PHP*. พิมพ์ครั้งที่ 2. กรุงเทพฯ: ไอดีซี พรีเมียร์.

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for engineering and technology.* London: Prentice Hall.

Harvey M. Deitel, Paul J. Deitel. (2004). *C How to program*. 4th Edition. London: Prentice Hall.

Kris Jamsa. (2002). *Jamsa's C/C++/C# programmer's bible: The ultimate guide to C/C++/C# programming.* 2nd Edition. USA: Onword Press.

บทที่ 7
การวนซ้ำ

การทำงานจริงนั้นมีงานในหลายลักษณะที่ไม่สามารถที่จะทำเสร็จได้ภายในครั้งเดียว ต้องมีการทำงานซ้ำๆ กันหลายครั้ง ดังนั้นการแก้ปัญหาโดยคอมพิวเตอร์จำเป็นจะต้องมีการทำงานซ้ำๆ กันเป็นจำนวนมาก ด้วยเหตุนี้ตัวแปลภาษาซีจึงมีการจัดทำชุดคำสั่งในการวนซ้ำไว้ให้ด้วย ในบทนี้อธิบายการวนซ้ำด้วยการใช้คำสั่ง for การวนซ้ำด้วยการใช้คำสั่ง while และการวนซ้ำด้วยการใช้คำสั่ง do wile อธิบายผังงาน รูปแบบคำสั่ง ตัวอย่างการเขียนโปรแกรมเพื่อการใช้งาน

การวนซ้ำด้วยการใช้คำสั่ง for

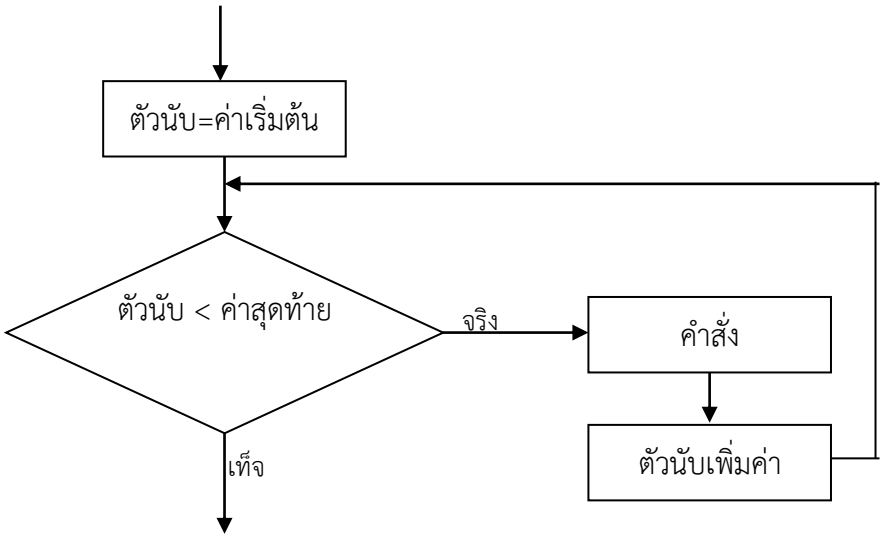
คำสั่ง for เป็นคำสั่งการวนซ้ำ ที่สามารถกำหนดจำนวนรอบของการวนซ้ำได้แน่นอนโดยคำสั่ง for นั้นจะต้องมีการระบุองค์ประกอบทั้งสิ้น 3 องค์ประกอบได้แก่

1. ค่าเริ่มต้นของตัวแปรที่ใช้ในการนับ ซึ่งต้องเป็นตัวแปรชนิดที่คำนวณได้
2. ค่าสุดท้าย (เงื่อนไขของการจบการทำงานของการวนซ้ำ)
3. รูปแบบการเพิ่มค่าของตัวแปรที่ใช้ในการนับ

โดยการระบุองค์ประกอบต่างๆ ของคำสั่ง for จะมีรูปแบบดังรูปแบบดังนี้

```
รูปแบบ:
for (expression1; expression2; expression3;)
                    statement ;
for (ค่าเริ่มต้นของการนับ;  ค่าสุดท้ายในการจบการทำงาน; รูปแบบของการเพิ่มค่าของการนับ;)
{
            ชุดคำสั่งที่ต้องการให้วนซ้ำ;
}
หมายเหตุ: ขอให้สังเกตว่าหลังบรรทัดคำสั่ง for ไม่มีเครื่องหมายอัฒภาค ;
แต่ ; จะอยู่ภายในส่วนของคำสั่ง for
หากไม่ใส่ {} คร่อมคำสั่งที่ให้ทำงานจะมีผลเพียงคำสั่งเดียวที่อยู่ถัดไปจาก for
```

การระบุค่าแรกและค่าสุดท้ายที่แน่นอน ทำให้คำสั่ง for เหมาะกับการเขียนการวนซ้ำที่มีการระบุจำนวนรอบที่ชัดเจนเช่น 10 รอบ, 20 รอบ เป็นต้น สามารถที่จะเขียนผังงานของการทำงานของคำสั่ง for ได้ดังภาพที่ 7.1



ภาพที่ 7.1 ผังงานแสดงการทำงานของคำสั่ง for

จากผังงานในภาพที่ 7.1 จะเห็นได้ว่าคำสั่ง for เริ่มต้นจากการกำหนดค่าเริ่มต้นให้กับตัวนับ ลำดับต่อไปจะเช็คเงื่อนไขการจบการทำงาน โดยส่วนใหญ่จะสร้างเงื่อนไขตัวนับ มีค่าน้อยกว่าค่าสุดท้ายที่กำหนดไว้ เช่น I < 5 ถ้าเงื่อนไขเป็นจริง จำดำเนินการตามคำสั่งที่เขียนไว้ เมื่อทำคำสั่งทั้งหมดเสร็จ ตัวนับจะเพิ่มค่า การเพิ่มค่าขึ้นอยู่กับการกำหนดการเพิ่มค่าว่าเพิ่มครั้งละเท่าไร ปกติจะเพิ่มครั้งละ 1 แต่ผู้เขียนโปรแกรมสามารถประยุกต์ใช้งานได้ เมื่อตัวนับเพิ่มค่าแล้ว ก็ไปขั้นตอนของการตรวจสอบเงื่อนไขอีกครั้ง ถ้าเงื่อนไขเป็นจริง ก็ดำเนินการตามคำสั่ง ทำคำสั่งเสร็จก็เพิ่มค่า แล้วกลับไปตรวจสอบเงื่อนไขแบบนี้จนกว่าเงื่อนไขจะเป็นเท็จ ถือว่าจบคำสั่ง for

ตัวอย่างที่ 7.1 การใช้ for

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int i;
6           clrscr();
7           for ( i=0 ; i <= 5 ; i++)
8           {
9                 printf(“%d” , i)  ;
10          }
11  }
```

การทำงานของโปรแกรม

โปรแกรมทำงานเริ่มจากกำหนดค่าเริ่มต้นให้กับค่า i มีค่าเท่ากับ 0

รอบที่ 1 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 0 ทางจอภาพ เพิ่มค่า i เป็น 1

รอบที่ 2 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 1 ทางจอภาพ เพิ่มค่า i เป็น 2

รอบที่ 3 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 2 ทางจอภาพ เพิ่มค่า i เป็น 3

รอบที่ 4 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 3 ทางจอภาพ เพิ่มค่า i เป็น 4

รอบที่ 5 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 4 ทางจอภาพ เพิ่มค่า i เป็น 5

รอบที่ 6 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 5 ทางจอภาพ เพิ่มค่า i เป็น 6

รอบที่ 7 ตรวจสอบเงื่อนไข i<=5 เป็นเท็จ เพราะ i มีค่าเป็น 6 ออกจากการวนซ้ำ

ผลลัพธ์

012345

ตัวอย่างที่ 7.2 การใช้ for โดยเพิ่มค่า ครั้งละ 2

```
1  #include <stdio.h>
2  #include <conio.h>
3  void main()
4  {
5          int i;
6          clrscr();
7          for ( i=0 ; i <= 5 ; i+=2)
8          {
9              printf(“%d” , i)  ;
10         }
11 }
```

การทำงานของโปรแกรม

โปรแกรมทำงานเริ่มจากกำหนดค่าเริ่มต้นให้กับค่า i มีค่าเท่ากับ 0

รอบที่ 1 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 0 เพิ่มค่า i เป็น 2

รอบที่ 2 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 2 เพิ่มค่า i เป็น 4

รอบที่ 3 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามฟังก์ชัน printf(“%d” , i); แสดงค่า i เท่ากับ 4 เพิ่มค่า i เป็น 6

รอบที่ 4 ตรวจสอบเงื่อนไข i<=5 เป็นเท็จ เพราะ i มีค่าเป็น 6 ออกจากการวนซ้ำ

ผลลัพธ์

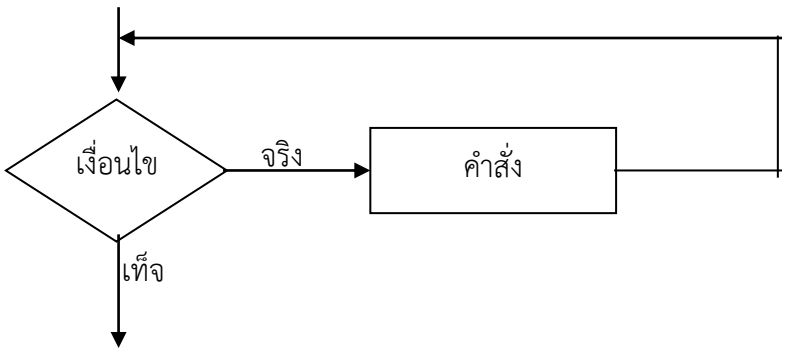
024

การวนซ้ำด้วยการใช้คำสั่ง while

สำหรับคำสั่ง while นั้นจะเป็นคำสั่งที่มีลักษณะการทำงานแบบตรวจสอบแล้วจึงทำตามคำสั่ง ซึ่งมีรูปแบบการใช้งานดังนี้

```
รูปแบบ:  while (เงื่อนไข)
              {
               คำสั่งที่ต้องการให้วนซ้ำ;
              }
หมายเหตุ:   ขอให้สังเกตว่าหลังบรรทัดคำสั่ง while  ไม่มีเครื่องหมายอัฒภาค ;
        :   หากไม่ใส่ {} คร่อมคำสั่งที่ให้ทำงานจะมีผลเพียงคำสั่งเดียวที่อยู่ถัดไปจาก while เท่านั้น
```

นั้นเป็นคำสั่งที่มีลักษณะการทำงานดังแสดงในผังงานในภาพที่ 7.2



ภาพที่ 7.2 ผังงานการทำงานของคำสั่ง while

จากผังงานในภาพที่ 7.2 จะเห็นได้ว่าคำสั่ง while มีลักษณะใกล้เคียงกันกับคำสั่ง for() แต่คำสั่ง while จะไม่มีการกำหนดค่าเริ่มต้น และการเปลี่ยนค่าของตัวนับให้ ซึ่งการทำงานทั้งสองนี้ผู้พัฒนาโปรแกรมจะต้องทำการเพิ่มเข้าไปเอง

ตัวอย่างที่ 7.3 การใช้ while

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int i = 0 ;
6           clrscr();
7           while ( i <= 5 )
8           {
9                 printf("%d" , i++)  ;
10          }
11  }
```

การทำงานของโปรแกรม

โปรแกรมทำงานเริ่มจากกำหนดค่าเริ่มต้นให้กับค่าตัวแปร i มีค่าเท่ากับ 0

รอบที่ 1 ค่าตัวแปร i = 0 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 0 ก่อนแล้วจึงเพิ่มค่า i เป็น 1 จากคำสั่ง i++

รอบที่ 2 ค่าตัวแปร i = 1 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 1 ก่อนแล้วจึงเพิ่มค่า i เป็น 2

รอบที่ 3 ค่าตัวแปร i = 2 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 2 ก่อนแล้วจึงเพิ่มค่า i เป็น 3

รอบที่ 4 ค่าตัวแปร i = 3 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 3 ก่อนแล้วจึงเพิ่มค่า i เป็น 4

รอบที่ 5 ค่าตัวแปร i = 4 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 4 ก่อนแล้วจึงเพิ่มค่า i เป็น 5

รอบที่ 6 ค่าตัวแปร i = 5 ตรวจสอบเงื่อนไข i<=5 เป็นจริง ทำงานตามคำสั่ง แสดงค่าตัวแปร i เท่ากับ 5 ก่อนแล้วจึงเพิ่มค่า i เป็น 6

รอบที่ 7 ค่าตัวแปร i = 6 ตรวจสอบเงื่อนไข i<=5 เป็นเท็จ เพราะค่าตัวแปร i มีค่าเป็น 6 ออกจากการวนซ้ำ

ผลลัพธ์

012345

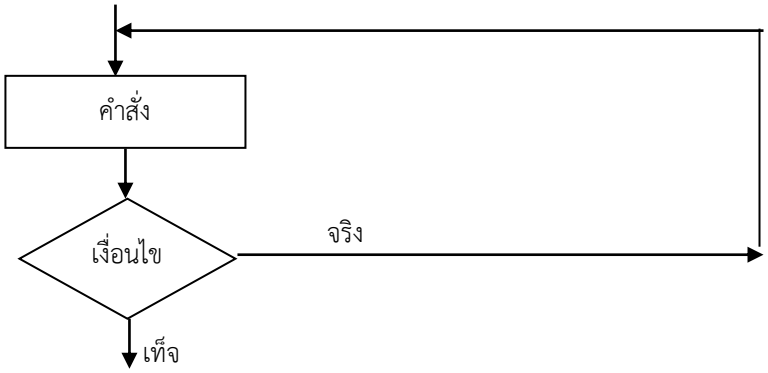
การวนซ้ำด้วยการใช้คำสั่ง do-while

คำสั่ง do-while นั้นจะเป็นคำสั่งที่มีลักษณะการทำงานแบบทำตามคำสั่งก่อนแล้วจึงตรวจสอบเงื่อนไข ดังนั้นการวนรอบโดยคำสั่ง do-while จะต้องมีการทำตามคำสั่งระหว่าง do กับคำสั่ง while อย่างน้อย 1 รอบเสมอ ซึ่งมีรูปแบบการใช้งานดังรูปแบบต่อไปนี้

```
รูปแบบ
do
                statement;
while (expression);
```

หมายเหตุ: ขอให้สังเกตว่าหลังบรรทัดคำสั่ง do ไม่มีเครื่องหมายอัฒภาค ; แต่จะมีหลัง while หากไม่ใส่ {} คร่อมคำสั่งที่ให้ทำงานจะทำให้คำสั่งระหว่าง do.. while มีได้เพียง 1 คำสั่งเท่านั้น

การทำงานของคำสั่ง do...while นั้นจะมีลักษณะใกล้เคียงกันกับ while มาก โดยคำสั่ง do..while มีลำดับการทำงานดังภาพที่ 7.3



ภาพที่ 7.3 ผังงานการทำงานของคำสั่ง do-while

ตัวอย่างที่ 7.4 การใช้ do-while

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int i = 0 ;
6           clrscr();
7           do
8           {
9               printf(“%d” , i++) ;
10          } while ( i < 0 )
11  }
```

การทำงานของโปรแกรม

โปรแกรมทำงานเริ่มจากกำหนดค่าเริ่มต้นให้กับค่า i มีค่าเท่ากับ 0

รอบที่ 1 ค่า i = 0 แสดงค่า i เท่ากับ 0 ก่อนแล้วจึงเพิ่มค่า i เป็น 1 จากคำสั่ง i++ ตรวจสอบเงื่อนไข I < 0 เป็นเท็จ เพราะ i มีค่าเป็น 1 ออกจากการวนซ้ำ

ผลลัพธ์

0

หมายเหตุ

- คำสั่งในการวนซ้ำทั้งหมดไม่ว่าจะเป็น for(), while(), do... while() จะสามารถมีเพียงคำสั่ง (Statement) เดียวเท่านั้นที่ถูกวนซ้ำ นั่นคือ Statement ที่อยู่ติดอยู่กับตัวคำสั่งในการวนซ้ำนั้นๆ
- หากต้องการให้การวนซ้ำมีการทำงานมากกว่า 1 คำสั่ง ให้ใส่เครื่องหมาย { และ } คร่อมระหว่างคำสั่งที่ต้องการให้วนซ้ำนั้นๆ

ตัวอย่างที่ 7.5 เขียนโปรแกรมเพื่อทำการคำนวณผลบวกของเลขทุกๆ จำนวนระหว่าง 1 จนถึง n เมื่อ n เป็นเลขจำนวนเต็มบวกใดๆ ที่ป้อนเข้าทางแผงแป้นอักขระโดยผู้ใช้ และ มีค่ามากกว่า 1 และให้แสดงผลบวกเมื่อสิ้นสุดการทำงานของโปรแกรม ให้ตรวจสอบการความถูกต้องของการป้อนค่าของผู้ใช้ด้วย

หลักการคิด เนื่องจากไม่สามารถที่จะกำหนดคำตอบตายตัวได้ว่าหากผู้ใช้ป้อนเลข 1 ให้ได้คำตอบเป็น 1, ป้อนเลข 2 ได้คำตอบเป็น 1+2=3, ป้อนเลข 3 ได้คำตอบเป็น 1+2+3=6 เป็นต้น เนื่องจากผู้ใช้อาจป้อนเลข 1000 ซึ่งไม่สามารถคิดล่วงหน้าได้ ดังนั้นวิธีการที่ง่ายที่สุดคือการวนซ้ำและบวกเลขทีละตัวตั้งแต่ 1,2,3 จนถึง n ซึ่งอาจทำได้โดยการใช้คำสั่ง for เพราะรู้จำนวนแน่นอนว่าต้องมีการวนซ้ำจำนวน n ครั้ง

ในการนี้จำเป็นจะต้องมีตัวแปรจำนวน 3 ตัวคือ ตัวแปร Counter เพื่อใช้นับว่าปัจจุบันนี้บวกไปได้ถึงไหนแล้ว ตัวแปร Sum เพื่อเก็บค่าผลบวกปัจจุบัน ตัวแปร n เป็นจำนวนที่ผู้ใช้ป้อนเข้ามา ซึ่งหลังจากประกาศตัวแปร และทำการรับค่า ตัวแปร n แล้วจะต้องมีการกำหนดค่าเริ่มต้นของตัวแปร Sum ให้เป็น 0 เสียก่อนในบรรทัดที่ 11 จากนั้นให้โปรแกรมทำการวนซ้ำโดยค่าของตัวแปร Sum จะเพิ่มขึ้นเท่ากับค่าของตัวแปร Counter ในทุกๆ รอบ (ในบรรทัดที่ 13) ในโปรแกรมที่ 7.1

```
1   /* Demostation of Looping to Sum the number from 1 to n
       with for syntax */
2   #include <stdio.h>
3
4   void main()
5   {
6           int counter;
7           int sum;
8           int n;
9           printf("Enter Number (n): ");
10          scanf("%d",&n);
11          sum=0;
12          for(counter=1;counter<=n;counter++)
13                  sum=sum+counter;
14          printf("Sum of 1 to %d is %d",n,sum);
15  }
```

โปรแกรมที่ 7.1 รหัสต้นฉบับในการแก้ปัญหาที่ 7.5 โดยการใช้คำสั่ง for()

หมายเหตุ - อย่าลืมว่าในการเขียนโปรแกรมโดยใช้คำสั่งต่างๆ ในการตรวจสอบเงื่อนไข หรือการวนซ้ำนั้นหากต้องการให้คำสั่งนั้นมีผลกับ Statement มากกว่า 1 Statement ให้ใส่เครื่องหมาย { และ } ครอบ Statement ที่ต้องการให้ทำงานไว้

- คำสั่งในการวนรอบคือ for(), while() และ do.. while() ไม่ต้องมีเครื่องหมายอัฒภาค ; หลังคำสั่ง

สำหรับปัญหานี้สามารถแก้ปัญหาได้อีกวิธีการหนึ่ง โดยใช้หลักการของการคำนวณทางคณิตศาสตร์แทน เนื่องจากปัญหานี้อาจแก้ปัญหาได้โดยใช้วิธีการคำนวณ โดยใช้สมการ

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

นั่นคืออาจใช้วิธีการคำนวณตามสมการข้างต้น เพื่อให้ได้คำตอบในครั้งเดียว ดังแสดงในโปรแกรมที่ 7.2

```
1   /* Demostation of Calculate Sum of the number from 1..n */
2   #include <stdio.h>
3   void main()
4   {
5           int counter;
6           int sum;
7           int n;
8           printf("Enter Number (n): ");
9           scanf("%d",&n);
10          sum = (n*(n+1))/2;
11          printf("Sum of 1 to %d is %d",n,sum);
12  }
```

โปรแกรมที่ 7.2 รหัสต้นฉบับในการแก้ปัญหาตามตัวอย่างที่ 7.5

ตัวอย่างที่ 7.6 ต้องการเขียนโปรแกรมให้รับค่าตัวอักษรครั้งละ 1 ตัวจนกว่าผู้ใช้จะกดปุ่ม '@' เพื่อจบการทำงาน ซึ่งเมื่อจบการทำงานให้แสดงตัวอักษรที่ทุกตัวออกทางจอภาพ ดังตัวอย่าง

Enter Message: ***Hello World@***

You enter message: Hello World@

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

แนวคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลข้อความที่จะแสดงผลซึ่งมี 1 ข้อความที่ป้อนเข้ามา แต่ปรากฏว่าไม่รู้ว่าเป็นข้อความอะไร ยาวแค่ไหนไม่รู้ ดังนั้นจึงต้องคิดว่าจะเก็บข้อมูลนี้อย่างไร ขึ้นอยู่กับการรับข้อมูล

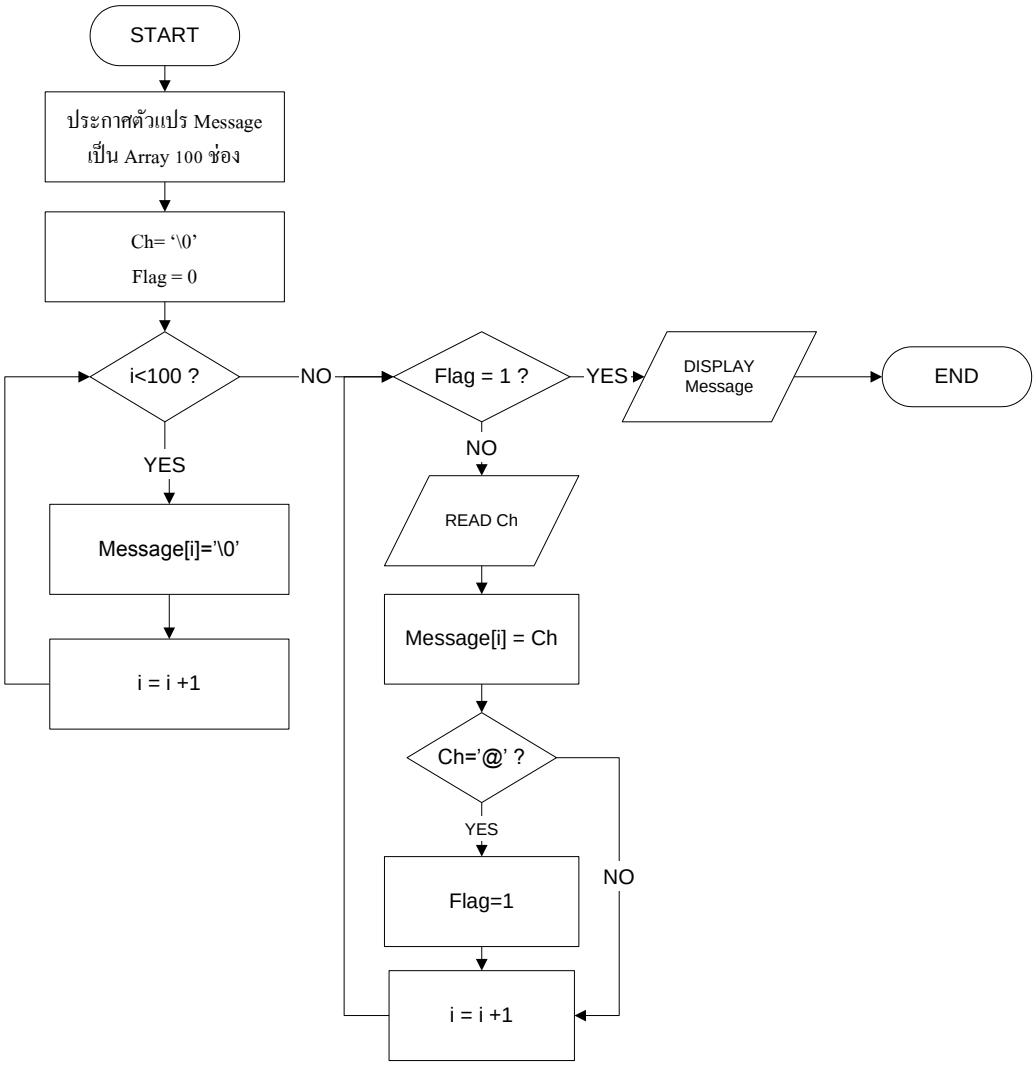
2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกชัดเจนว่าให้รับตัวอักษรมาจาก Keyboard ทีละตัว แต่โจทย์ไม่ได้บอกว่ากี่ตัว เก็บยังไง ดังนั้นเพื่อให้ง่ายจึงต้องกำหนดเอาเองว่าข้อมูลยาวไม่เกินเท่าไร (สมมติว่า 100 ตัวอักษร) เป็นชนิด char และต้องเป็นตัวแปรแถวลำดับด้วย ดังนี้

2.1 ลบค่าทั้งหมดในตัวแปรแถวลำดับ

2.2 รับค่าโดยใช้ getch()

3. วิเคราะห์ความสัมพันธ์ของข้อมูล เวลารับข้อมูลมา 1 ตัวอักษรก็เอาไปเก็บไว้ในตัวแปรแถวลำดับก่อน แล้วพอจะแสดงผลก็ต้องใช้ฟังก์ชัน puts() แสดงผลลัพธ์

4. ทำการเขียนผังงาน Flowchart ได้ดังภาพที่ 7.3 และนำไปเขียนโปรแกรมได้ดังโปรแกรมที่ 7.3



ภาพที่ 7.4 ผังงานโปรแกรมรับค่าตัวอักษรครั้งละ 1 ตัว

ซึ่งสามารถเขียนโปรแกรมได้ตามโปรแกรมที่ 7.3

```
1  #include <stdio.h>
2  void main()
3  {
4      char Message[100];
5      char Ch;
6      int flag=0;
7      int i;
8      clrscr();
9      for (i=0;i<100;i++)
10         Message[i]='\0';
11     Ch='\0';
12     i=0;
13     printf("Enter Message: ");
14     while(flag!=1)
15     {
16         Ch=getche();
17         Message[i]=Ch;
18         if (Ch=='@')
19             flag=1;
20         i++;
21     }
22     printf("\nYou enter message: "); puts(Message);
23     getch();
24 }
```

โปรแกรมที่ 7.3 รหัสต้นฉบับในการแก้ปัญหาตามตัวอย่างที่ 7.6

ตัวอย่างที่ 7.7 เขียนโปรแกรมรับชื่อนักศึกษา แสดงข้อความต้อนรับนักศึกษาคนนั้น และรับอายุ โดยอายุที่ป้อนเข้ามาต้องไม่เกิน 100 หากเกิน 100 ให้รับอายุจนกว่าอายุจะเป็นค่าที่ถูกต้อง ดังตัวอย่าง

What is your name : ***Napatsarun***
Hello Napatsarun
How old are you : ***180***
Invalid Data
How old are you : ***25***
Thank you

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

แนวคิด

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือ ข้อความต้อนรับ และการรับค่าอายุที่ถูกต้อง
2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกชัดเจนว่าให้รับชื่อนักศึกษา และ อายุ มาจาก Keyboard โดยมีเงื่อนไขว่าอายุ ห้ามเกิน 100
3. วิเคราะห์ความสัมพันธ์ของข้อมูล เงื่อนไขของอายุ ต้องวนรับค่าจนกว่าจะถูกต้อง การวนรับค่าต้องใช้ ลูป ซึ่งลูปมี 3 คำสั่ง คือ for ซึ่งต้องรู้จำนวนรอบการทำงานที่แน่นอน ซึ่งกรณีนี้ ไม่ทราบว่า ผู้ใช้จะคีย์ข้อมูลถูกต้องในรอบที่เท่าไร จึงไม่ควรใช้ พิจารณา while ต้องตรวจสอบเงื่อนไขก่อน ซึ่งข้อมูลที่เป็นเงื่อนไข คือ อายุ ซึ่งต้องรับค่ามาก่อน จึงควรใช้ do while แล้วเช็คเงื่อนไข อายุมากกว่า 100

ซึ่งสามารถเขียนโปรแกรมได้ตามโปรแกรมที่ 7.4

```
1   #include <stdio.h>
2   void main()
3   {
4       char name[100];
5       int age;
6
7       clrscr();
8       printf("What is Your Name ? : ");
9       scanf(“%s”,name) ;
10      printf(" Hello %s  !!",name);
11
12      do {
13            printf("How old are you ?: ");
14            scanf(“%d”, &age) ;
15            if  (age > 100)
16                  printf(" Invalid Data  \n") ;
17      }      while( age > 100 )
18
19     printf("\n Thank you \n ");
20      getch();
21  }
```

โปรแกรมที่ 7.4 รหัสต้นฉบับในการแก้ปัญหาตามตัวอย่างที่ 7.7

สรุป

การเขียนโปรแกรมเพื่อการวนซ้ำ ในภาษาซีสามารถเลือกใช้คำสั่ง for คำสั่ง while และคำสั่ง do while ได้ตามลักษณะการทำงานของโปรแกรม โดยคำสั่ง for สามารถกำหนดรอบการทำงานได้ชัดเจน คำสั่ง do มีการตรวจสอบเงื่อนไขก่อนคำสั่ง และคำสั่ง do while จะทำคำสั่งก่อน 1 รอบการทำงานแล้วจึงตรวจสอบเงื่อนไขในการวนซ้ำ

การเขียนโปรแกรมเพื่อการวนซ้ำ มีความสำคัญมากเพราะโปรแกรมส่วนใหญ่ต้องมีการวนซ้ำ ซึ่งเป็นลักษณะสำคัญของโปรแกรมคอมพิวเตอร์ เพราะฉะนั้นการเลือกคำสั่งเพื่อการวนซ้ำควรเลือกให้เหมาะสมกับลักษณะการทำงานในแต่ละงาน และที่สำคัญคือเงื่อนไขในการทำงาน และเงื่อนไขการจบการทำงาน จะตรวจสอบให้ดีว่าครบทุกเงื่อนไข ไม่มีเงื่อนไขใดที่หลุดทำให้โปรแกรมวนซ้ำจนไม่มีการจบการทำงานได้

แบบฝึกหัด

1. ให้เขียนโปรแกรมในการรับตัวเลข 20 จำนวนและตอบว่าเลขจำนวนใดมีค่ามากที่สุด

2. ให้เขียนโปรแกรมในการคิดระดับคะแนน (เกรด) ของนักศึกษา เมื่อทำการป้อนคะแนนกลางภาค และคะแนนสอบปลายภาค ของนักศึกษาจำนวน 10 คน โดยมีหลักเกณฑ์การให้ระดับคะแนนดังนี้

ระดับคะแนน A ได้คะแนนรวม 80 คะแนนขึ้นไป
ระดับคะแนน B ได้คะแนนรวม 70 คะแนนขึ้นไป
ระดับคะแนน C ได้คะแนนรวม 60 คะแนนขึ้นไป
ระดับคะแนน D ได้คะแนนรวม 50 คะแนนขึ้นไป
ระดับคะแนน E ได้คะแนนรวม ไม่ถึง 50 คะแนน

3. เขียนโปรแกรมเพื่อทำการคำนวณผลบวกของเลขทุกๆ จำนวนระหว่าง 1 จนถึง n เมื่อ n เป็นเลขจำนวนเต็มบวกใดๆ ที่ป้อนเข้าทางแป้นพิมพ์โดยผู้ใช้ และ n เป็นค่าที่มีค่ามากกว่า 1 และให้แสดงผลบวกเมื่อสิ้นสุดการทำงานของโปรแกรม (ให้ตรวจสอบการความถูกต้องของการป้อนค่าของ ผู้ใช้ด้วย)

(ก) while..

(ข) do... while

4. ให้เขียนโปรแกรมเล่นเกมจารชน โดยผู้ใช้โปรแกรมคนที่ 1 กำหนดค่าตัวเลขมรณะขึ้นมาและให้ผู้ใช้คนที่ 2 ทายตัวเลข ให้เครื่องคอมพิวเตอร์บอกว่าเลขที่ผู้ใช้คนที่ 2 ทายนั้นมากกว่า/น้อยกว่าเลขมรณะจนกว่าจะทายถูก

5. ให้นักศึกษาเขียนโปรแกรมในการคิดส่วนลดจากการป้อนมูลค่าการสั่งซื้อแก่ลูกค้าแห่งหนึ่งเมื่อมีมูลค่าการสั่งซื้อและส่วนลดต่าง ๆ ดังนี้

(ก) มูลค่าการซื้อสินค้าน้อยกว่า 2,000 บาท ส่วนลด 5 %
(ข) มูลค่าการซื้อสินค้า 2,000 – 10,000 บาท ส่วนลด 15 %
(ค) มูลค่าการซื้อสินค้า 10,001 บาทขึ้นไป ส่วนลด 30 %

โดยให้ทำการวนรับค่าจนกว่าจะป้อนมูลค่าการสั่งซื้อเป็น 0 ให้ออกจากโปรแกรมให้บอกจำนวนเงินที่ขายได้ทั้งหมดก่อนหักส่วนลด, จำนวนเงินหลังหักส่วนลดทั้งหมด โดยโปรแกรมจะต้องแสดงการทำงานดังตัวอย่าง

```
Enter sale money: 10000
   Discount Rate = 30.00 %
   Discount money = 3000.00 bath
   Net money pay =7000.00
Enter sale money: 3000
   Discount Rate = 15.00 %
   Discount money = 450.00 bath
   Net money pay =2550.00
Enter sale money: 1000
   Discount Rate = 5.00 %
   Discount money = 50.00 bath
   Net money pay =950.00
Enter sale money: 0
   Discount Rate = 5.00 %
   Discount money = 0.00 bath
   Net money pay =0.00
Total sale BEFORE Discount = 14000.00 bat
Total money pay =10500.00
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

6. ให้นักศึกษาลองตอบคำถามต่อไปนี้

ก) จากตัวอย่าง 7.2 ทำไมถึงต้องกำหนดให้ Message[i] ในทุกช่องเป็น ‘\0’ เป็นตัวอื่นได้หรือไม่ เพราะเหตุใด

ข) ถ้าต้องการไม่ให้แสดงเครื่องหมาย ‘@’ ในตอนจบต้องแก้ไขโปรแกรม และผังงาน (Flowchart) เป็นอย่างไร เพราะเหตุใด

```
Enter Message: Hello World@
You enter message: Hello World   (ไม่ให้มี @ ข้างหลัง)
```

7. ให้เขียนโปรแกรมทำการตรวจสอบรหัสผ่าน ATM ของธนาคารแห่งหนึ่งซึ่งมีรหัส ATM จำนวนไม่เกิน 10 อักษร โดยเบื้องต้นมีการกำหนดรหัสผ่านตายตัวเป็น “Micheal” ดังตัวอย่างการทำงาน

```
Enter password: Micheal
Your ATM Password is correct
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

8. ให้นักศึกษาแก้ไขโปรแกรมให้ทำการแสดงเครื่องหมาย * ออกมาทุกครั้งที่ผู้ใช้ป้อนรหัสผ่าน และให้ทำการรับรหัสผ่านไปตรวจสอบ ดังตัวอย่าง

Enter password: ******* ผู้ใช้พิมพ์ Micheal และกด Enter

Your ATM Password is correct

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้นใต้***เป็นข้อมูลที่ป้อนเข้าไป

9. ให้เขียนโปรแกรมทำการแสดงเลขฐานสอง ของเลขฐานสิบที่ผู้ใช้ป้อนผ่านทางแป้นพิมพ์ ซึ่งเป็นเลขจำนวนเต็มฐานสิบมีค่าระหว่าง 0-255 ให้แสดงย้อนกลับ เช่น $12_{10} = 1100_2$ ให้แสดงเป็น 0011_2

10. จงอธิบายความแตกต่างลักษณะการใช้งานของคำสั่ง for, while และ do while

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร:ดวงกมลสมัย.

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึ่ม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

ประภาพร ช่างไม้.(2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส.

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for engineering and technology.* London: Prentice Hall.

Kris Jamsa. (2002). *Jamsa's C/C++/C# programmer's bible: The ultimate guide to C/C++/C# programming.* 2nd Edition. USA: Onword Press.

บทที่ 8
ตัวแปรแบบแถวลำดับ

ในบทนี้จะอธิบายถึงตัวแปรแบบแถวลำดับ (Array Variable) การประกาศตัวแปรแบบแถวลำดับ การรับค่าและการแสดงตัวแปรแบบแถวลำดับ ตัวอย่างการนำไปใช้งาน ข้อควรระวังในการใช้ตัวแปรแบบแถวลำดับ และการใช้งานตัวแปรแบบแถวลำดับแบบ 2 มิติ

ในบทที่ 5 ได้อธิบายถึงการรับค่าและการแสดงผลข้อมูลชนิดอักขระ (char) ซึ่งสามารถใช้กับตัวอักษรเพียงหนึ่งตัวเท่านั้น หากนักศึกษาต้องการรับข้อมูลที่มีขนาดมากกว่า 1 อักขระ เช่น ชื่อนักศึกษา ซึ่งประกอบด้วยตัวอักขระ มากกว่า 1 อักขระจะไม่สามารถทำได้ ซึ่งนักศึกษาอาจแก้ปัญหาโดยการประกาศตัวแปรขึ้นมาจำนวนหนึ่ง สมมติว่า 10 ตัวเพื่อเก็บชื่อซึ่งยาว 10 อักขระ เช่น char a,b,c,d,e,f,g,h,i,j; a='T'; b='h'; ..... เป็นต้น ซึ่งจะเห็นว่าไม่สะดวกในการใช้งานเนื่องจากข้อมูลเดียวกับต้องแยกกันอยู่หลายตัวแปร ดังนั้นตัวแปลภาษาซี จึงได้มีการอนุญาตให้มีการสร้างตัวแปรแบบพิเศษเรียกว่าตัวแปรแบบแถวลำดับ (array variable) ได้โดยตัวแปรมากกว่า 1 ตัวสามารถใช้ชื่อเดียวกันได้ แต่ต้องแยกระหว่างข้อมูลแต่ละข้อมูลโดยใช้หมายเลขประจำตัวเรียกว่า "ดรรชนี (index)" ดังภาพที่ 8.1

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

ภาพที่ 8.1 ตัวแปรแบบแถวลำดับ

จากภาพที่ 8.1 ในแต่ละช่องจะเป็น A[0] จนถึง A[9] รวมทั้งสิ้น 10 ช่องโดย A เป็นชื่อของตัวแปรแบบแถวลำดับ (array variable) ชุดนี้มีชื่อว่า A ดังนั้นทั้ง 10 ช่องล้วนมีชื่อว่า A ทั้งหมด แต่หากจะหมายถึงข้อมูลใดในตัวแปรแบบแถวลำดับ (array variable) จะต้องบ่งบอกถึงหมายเลขประจำของแต่ละช่อง (ดรรชนี:Index) ของช่องนั้นๆ ด้วย เช่น A[3] หมายถึงข้อมูลในช่องที่ 3 ของตัวแปรแบบแถวลำดับ (array variable) ชุดที่มีชื่อว่า A

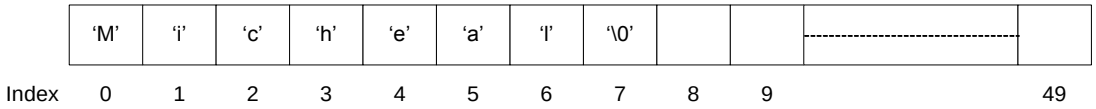
แถวลำดับ (Array) เป็นกลุ่มของข้อมูลชนิดเดียวกันวางเรียงติดต่อกันในหน่วยความจำ สมาชิกแต่ละตัวต้องมีขนาดเท่ากัน ตัวแปรที่ใช้เก็บที่อยู่เริ่มต้นของสมาชิกตัวแรก เรียกว่า ตัวแปรแบบแถวลำดับ (Array Variable) (ปัญญาพล. 2545: 162)

การประกาศตัวแปรแบบแถวลำดับ

การประกาศตัวแปรแบบแถวลำดับ ขึ้นใช้งานในภาษาซีนั้นมีรูปแบบดังนี้

รูปแบบ	:	ชนิดตัวแปร ชื่อตัวแปร[จำนวนของดรรชนี];
เช่น	:	char Name[50];
ความหมาย	:	ให้ตัวแปร Name เก็บข้อมูลชนิดอักขระ (char) จำนวน 50 ช่อง (จะมีดรรชนีตั้งแต่ 0-49)

สำหรับการประกาศตัวแปรที่แสดงให้เห็นนั้นเป็นการประกาศตัวแปรชนิดอักขระ (char) ซึ่งเอาไว้ใช้ในการเก็บชุดอักขระ เช่น ประกาศไว้เพื่อเก็บชื่อนักศึกษา ซึ่งมีความยาวไม่เท่ากันสำหรับนักศึกษาแต่ละคน ดังนั้นจึงต้องมีการบอกตัวแปลภาษาซี ด้วยว่าข้อมูลที่เก็บจริงมีเท่าไร เช่นถ้าเก็บชื่อว่า “Micheal” ก็จะมีความยาวทั้งหมด 7 ตัวอักษร จาก 50 ตัวอักษร ดังนั้นหลังจากที่เก็บข้อมูลตัวสุดท้ายแล้วก็จะเก็บอักขระพิเศษเพื่อบ่งบอกถึงการจบข้อมูลดังภาพที่ 8.2



ภาพที่ 8.2 การใช้ ‘\0’ ในการจบข้อมูล

หมายเหตุ ‘\0’ นั้นต้องกำหนดไว้สำหรับตัวแปรแบบแถวลำดับ (Array Variable) ของตัวอักษรเท่านั้น ชนิดอื่นไม่มี เช่น int IntArray[10]; ใช้งานได้ตั้งแต่ IntArray[0] ถึง IntArray[9]
float FloatArray[10]; ใช้งานได้ตั้งแต่ FloatArray [0] ถึง FloatArray [9]
แต่ถ้าเป็น char: char Name[10]; ใช้งานได้ตั้งแต่ Name [0] ถึง Name [8] ต้องเผื่อ Name[9] ไว้ใส่ ‘\0’

เมื่อประกาศตัวแปรแบบแถวลำดับ (Array Variable) แล้วและต้องการใส่ค่าตัวแปรลงไปให้กับตัวแปรในแต่ละตำแหน่งของดรรชนีโดยการรับค่าผ่านทางคีย์บอร์ด ดังตัวอย่างที่ 8.1

ตัวอย่างที่ 8.1 การรับและแสดงค่าตัวเลขแบบตัวแปรแบบแถวลำดับได้ดังโปรแกรมที่ 8.1

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       int  MyArray[2];
6        printf(“Enter number 0 : ”);  scanf(“%d”,&MyArray[0]);
7        printf(“Enter number 1 : ”);  scanf(“%d”,&MyArray[1]);
8
9        printf(“Number 0  is:  %d\n”,MyArray[0]);
10       printf(“Number 1  is:  %d\n”,MyArray[1]);
11      getch();
12  }
```

โปรแกรมที่ 8.1 โปรแกรมในตัวอย่างที่ 8.1

เมื่อประมวลผลโปรแกรมที่ 8.1 จะได้ผลลัพธ์ดังนี้

```
Enter number  0: 10
Enter number  1: 200
Number 0  is:  10
Number 1  is:  200
```

จะเห็นได้ว่าตัวแปรแถวลำดับ นั้นสามารถที่จะใช้งานได้เช่นเดียวกับตัวแปรปกติ เพียงแต่จะต้องระบุหมายเลขประจำของตัวของตัวแปรแต่ละช่องเพิ่มขึ้นไปเท่านั้นเอง ไม่ว่าจะเป็น printf() หรือ scanf() แต่อย่าลืมว่าในส่วนของชื่อตัวแปรในการที่จะรับค่าโดย scanf() นั้นก็ยังคงต้องมีเครื่องหมาย & อยู่ด้วย

การรับและแสดงผลตัวแปรแบบแถวลำดับของตัวอักขระ

สำหรับตัวแปรแบบแถวลำดับของตัวอักขระนั้นภาษาซี ได้มีการกำหนดฟังก์ชันมาตรฐานมาให้ 2 ฟังก์ชันก็คือ gets() ใช้สำหรับการอ่านข้อมูลที่เป็นอาร์เรย์ของตัวอักขระ และ ฟังก์ชัน puts() ใช้ในการแสดงข้อมูลทางจอภาพซึ่งมีรูปแบบการใช้งานดังนี้

1. ฟังก์ชัน gets

ฟังก์ชัน gets เป็นฟังก์ชันที่ใช้ในการรับข้อมูลที่เป็นข้อความโดยมีรูปแบบดังนี้

```
รูปแบบ        :  gets(ชื่อตัวแปรแบบแถวลำดับ)
Header File   :  stdio.h
เช่น          :  char  Name[50];
              :  gets(Name);
ความหมาย      :  ให้อ่านค่าตัวแปรจากแผงแป้นอักขระมาไว้ในตัวแปรแบบแถวลำดับชื่อ Name
จำนวน 50 ช่อง
```

2. ฟังก์ชัน puts

ฟังก์ชัน puts เป็นฟังก์ชันที่ใช้ในการแสดงข้อมูลที่เป็นข้อความ โดยมีรูปแบบดังนี้

```
รูปแบบ        : puts(ชื่อตัวแปรแบบแถวลำดับ)
Header File   : stdio.h
เช่น          : char  Name[50];
              : puts(Name);
ความหมาย      : ให้แสดงค่าของตัวแปรแบบแถวลำดับ  Name เก็บข้อมูลชนิดอักขระ(char)
จำนวน 50 ช่อง ออกทางจอภาพ
```

ตัวอย่างที่ 8.2 การรับค่าโดยใช้ฟังก์ชัน gets() และแสดงผลโดยใช้ puts()

ให้สังเกตว่าทั้งสองคำสั่งสามารถใช้ชื่อของตัวแปรแบบแถวลำดับได้เลยโดยไม่ต้องระบุดรรชนี นั่นคือให้อ่านหรือแสดงได้สูงสุดเท่าที่ประกาศไว้ ในที่นี้คือ 50 ตัวอักษร รวม ‘\0’ ซึ่งจำเป็นต้องมี ดังนั้น เป็นข้อมูลจริงๆ ใส่ได้ไม่เกิน 49 ตัวอักขระ ซึ่งแสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.2

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5       char MyName[50];
6       printf(“Enter your name: “);
7       gets(MyName);
8       printf(“Your name is: “);
9       puts(MyName);
10      getch();
11  }
```

โปรแกรมที่ 8.2 การรับค่าสู่ตัวแปรแบบแถวลำดับของตัวอักษรโดยใช้ฟังก์ชัน gets

เมื่อประมวลผลโปรแกรมที่ 8.2 จะได้ผลลัพธ์ดังนี้

```
Enter your name: Micheal
Your name is:  Micheal
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

ซึ่งจะเห็นว่าไม่ต้องใช้วิธีการรับค่าเรียงลำดับตั้งแต่ Name[0] จนถึง Name[9] นอกจากนี้ฟังก์ชัน gets() ยังช่วยทำการเติม ‘\0’ เพื่อบ่งบอกจุดสิ้นสุดของข้อความให้โดยอัตโนมัติอีกด้วย

ข้อควรระวังในการใช้ตัวแปรแบบแถวลำดับ

1. ดรรชนีของตัวแปรแบบแถวลำดับทุกประเภทในภาษาซี เริ่มจาก 0
2. จำนวนข้อมูลในตัวแปรแบบแถวลำดับทุกประเภท จะมีขนาดคงที่เท่ากับที่ประกาศไว้ตอนต้น ไม่สามารถเปลี่ยนแปลงได้
3. ไม่สามารถใส่ข้อมูลเกินจำนวนช่องของดรรชนีที่ประกาศไว้ในตอนต้นได้

นอกจากฟังก์ชัน gets() และฟังก์ชัน puts() แล้วยังมีฟังก์ชันที่เกี่ยวข้องกับสายอักขระอีกดังแสดงในตารางที่ 8.1

ตารางที่ 8.1 ฟังก์ชันที่เกี่ยวข้องกับสายอักขระในคลังโปรแกรม string.h

ฟังก์ชัน	การใช้งาน

strcpy()	char *stpcpy(char *dest, const char *src);
	ฟังก์ชันคัดลอกค่าจาก dest ไปยัง src
	ผลลัพธ์ของฟังก์ชัน คือ ความยาวอักขระที่คัดลอกได้
	ตัวอย่าง
	char s1[10]=”Hello”;
	char s2[10];
	strcpy(s2,s1);

strcmp()	int strcmp(const char *s1, const char *s2); ฟังก์ชันเปรียบเทียบค่าใน s1 และ s2 ว่ามีอักขระเหมือนกัน**ทุกตัว**หรือไม่ ผลลัพธ์ของฟังก์ชัน < 0 เมื่อ s1 มีค่าน้อยกว่า s2 == 0 เมื่อ s1 มีค่าเท่ากับ s2 (เหมือนกันทุกตัว) > 0 เมื่อ s1 มีค่ามากกว่า s2 ตัวอย่าง char s1[10]=”Hello”; char s2[10]=”Hello”; strcmp(s1,s2); ค่าที่ได้ 0
strlen()	size_t strlen(const char *s); ฟังก์ชันหาจำนวนตัวอักษรที่แท้จริงที่เก็บอยู่ใน s ผลลัพธ์ของฟังก์ชัน คือ จำนวนตัวอักษร ไม่รวม ‘\0’ (NULL String) ตัวอย่าง char s[10]=”Hello”; i=strlen(s); จะได้ i=5

ตัวอย่างที่ 8.3 ต้องการรับค่า E-mail และ ตรวจสอบว่าเป็น E-mail ที่ถูกต้องหรือไม่ หากเป็น Email ที่ไม่ถูกต้อง ให้รับจนกว่าจะได้ค่าที่ถูกต้อง ซึ่งแสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.3

```
1   #include<stdio.h>
2   #include<conio.h>
3   #include<string.h>
4   void main()
5   {
6      char email[20];
7      int len , i = 0, found = 0 ;
8      do {
9            found = 0 ;
10            printf("Enter Your E-mail: ");
11            scanf("%s",&email);
12            len = strlen(email);
13            while (i < len && found == 0)
14              {
15                 if (email[i++] == '@')
16                          found = 1 ;
17              }
18              if (found==0)
19                printf("\nE-mail Incorrect \n");
20     } while (found == 0) ;
21     printf("\n Thank you  \n");
22  }
```

โปรแกรมที่ 8.3 การตรวจสอบ E-mail

เมื่อประมวลผลโปรแกรมที่ 8.3 จะได้ผลลัพธ์ดังนี้

```
Enter your E-mail : Micheal
E-mail  Incorrect
Enter your E-mail : Micheal@hotmail.com
Thank you.
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

การตรวจสอบ E-mail สามารถตรวจสอบได้จากเครื่องหมาย @ ซึ่งจะเห็นว่าในโปรแกรมมีการกำหนดตัวแปร found เป็นตัวแปรกำหนดว่าพบ @ หรือยัง ถ้าพบแล้ว สามารถออกจากลูป while ได้เลย ไม่จำเป็นต้องวนไปจนกว่าจะครบข้อมูลทั้งหมด

ตัวอย่างที่ 8.4 ต้องการรับค่า username และ password โดยกำหนดว่าให้ username คือ “admin” และ password คือ “1234” และกำหนดให้ใส่ข้อมูลได้เพียง 3 ครั้งหากใส่ข้อมูลไม่ถูกต้องเกิน 3 ครั้งก็ต้องออกจากโปรแกรม

วิธีคิด การรับค่า username สามารถใช้ฟังก์ชัน scanf เข้ามาได้ แต่กรณีของการรับค่า password ต้องใช้เขียนโปรแกรมเพื่อซ้อนค่าที่ป้อนเข้ามาด้วย จึงได้นำฟังก์ชัน getch มาใช้ เพราะไม่แสดงอักขระที่คีย์ โดยใช้ร่วมกับ putchar ในการแสดง * และตรวจสอบการกดปุ่ม enter ด้วยค่า 13 ซึ่งแสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.4

```
1  main()
2  {
3      char password[10],user[10],ch ;
4      int  i = 0     , cnt = 0,pass =0;
5      do {
6         printf("\n Enter  your User Name: ");
7         scanf("%s",user);
8         i = 0 ;
9         printf("\n Enter  your Password: ");
10        cnt++;
11        do {
12           ch = getch();
13           password[i++] = ch ;
14           putchar('*');
15        } while (ch!=13) ;
16        password[--i] = '\0' ;
17        printf("\n Your user name: %s",user);
18        printf("\n Your Password: %s",password);
19        if (strcmp(password,"1234")==0  && strcmp(user,"admin")==0)
20              {
21                    printf("\nWelcome to system\n") ;
22                    pass = 1 ;
23              } else {
24                    pass = 0 ;
25                    printf("\nPassword Incorrect\n") ;
26        }
27     } while (pass==0 && cnt < 3 );
28 }
```

โปรแกรมที่ 8.4 การรับ User และ Password

เมื่อประมวลผลโปรแกรมที่ 8.4 จะได้ผลลัพธ์ดังนี้

```
Enter  your username   :  nok
Enter  your password   :  ****
Your  username: nok
Your  password : 4567
Password Incorrect

Enter  your username   :  admin
Enter  your password   :  ****
Your  username: admin
Your  password : 1234
Welcome to system
```

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

การรับค่า username จะรับจนกว่าจะรับค่า username เท่ากับ “admin” และ password เท่ากับ “1234” หากไม่ตรงกับค่าดังกล่าว จะทำการรับข้อมูลใหม่ และถ้าไม่ถูกต้อง ทั้ง 3 ครั้งก็ออกจากโปรแกรม หากใส่ username และ password ถูกต้อง ก็จะแสดง ข้อความว่า “Welcome to system”

การประกาศตัวแปรแบบแถวลำดับ 2 มิติ

การประกาศตัวแปรแบบแถวลำดับ แบบ 2 มิติ ในภาษาซี นั้นมีรูปแบบดังนี้

```
รูปแบบ        :  ชนิดตัวแปร   ชื่อตัวแปร [จำนวนแถว] [จำนวนคอลัมน์];
เช่น           :  int   matrix [2] [3];
ความหมาย      :  ให้ตัวแปร matrix เก็บข้อมูลชนิด int  จำนวนแถว 2 แถว 3 คอลัมน์
        (จะมีดรรชนีตั้งแต่ 0, 0 – 1, 2 )
```

สำหรับการประกาศตัวแปรแบบแถวลำดับ แบบ 2 มิติ จะมีดรรชนี 2 ดรรชนี ดรรชนีแรก เป็นดรรชนีของแถว และดรรชนีที่สอง เป็นดรรชนีของคอลัมน์ เช่น matrix[0][1] หมายถึงข้อมูลในแถวแรก คอลัมน์ที่ 2 เพื่อให้เข้าใจการเก็บข้อมูล แสดงให้เห็นดังนี้

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]

ภาพที่ 8.3 แสดงดรรชนีของตัวแปร

การประกาศตัวแปรและกำหนดค่าให้กับตัวแปรแบบแถวลำดับแบบ 2 มิติ ในภาษาซีนั้นมีรูปแบบดังนี้

รูปแบบ	:	ชนิดตัวแปร ชื่อตัวแปร [จำนวนแถว] [จำนวนคอลัมน์] = {.., .., ...};
เช่น	:	int matrix [2] [3] = { 1, 2, 3, 4, 5, 6 } ;
ความหมาย	:	ให้ตัวแปร matrix เก็บข้อมูลชนิด int จำนวนแถว 2 แถว 3 คอลัมน์ (โดยกำหนดค่าให้เป็น 1 2 3 4 5 และ 6 ตามลำดับ)

สำหรับการกำหนดค่าจะเรียงไปตามลำดับ เพื่อให้เข้าใจการเก็บข้อมูล แสดงให้เห็นดังนี้

1	2	3
4	5	6

ภาพที่ 8.4 การกำหนดค่าของตัวแปรแบบแถวลำดับ 2 มิติ

ตัวอย่างที่ 8.5 ต้องการรับค่า ให้กับ Matrix 3 x 2 แล้ว แสดงค่าดังกล่าว

วิธีคิด รับค่าใช้ฟังก์ชัน scanf ให้กับ Matrix 3 x 2 แล้ว กำหนดเป็นตัวแปรแบบแถวลำดับ 2 มิติ และ ตัวแปร 2 มิติต้องใช้คำสั่งวนซ้ำ for ในการกำหนดดรรชนี เพื่อความชัดเจนซึ่งแสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.5

```
1   main()
2   {
3           int matrix[3][2], i, j;
4           // input data to matrix
5           for (i = 0 ; i < 3 ; i++)
6              for (j=0; j < 2 ; j++)
7              {
8           printf ("Enter value at row %d column %d: ", i+1,j+1) ;
9                   scanf("%d",&matrix[i][j]);
10             }    // display data
11          for (i = 0 ; i < 3 ; i++)
12          {
13                  for (j=0; j < 2 ; j++)
14                          printf ("%d ",matrix[i][j]) ;
15                  printf("\n");
16          }
17  }
```

โปรแกรมที่ 8.5 การรับและแสดงค่าตัวแปรแบบแถวลำดับ 2 มิติ

เมื่อประมวลผลโปรแกรมที่ 8.5 จะได้ผลลัพธ์ดังนี้

```
Enter value at row 1 column 1  : 1
Enter value at row 1 column 2  : 2
Enter value at row 2 column 1  : 3
Enter value at row 2 column 2  : 4
Enter value at row 3 column 1  : 5
Enter value at row 3 column 2  : 6
1       2
3       4
5       6
```

การรับค่าและการแสดงผล ของตัวแปรแบบแถวลำดับ 2 มิติ ต้องคำนึงถึง Index เป็นสำคัญ การใช้ ลูป for กำกับดรรชนี (index) ทำให้เข้าใจ ง่ายกว่า ลูป while คำสั่งที่ใช้ในการรับค่าแสดงแถวและ คอลัมน์ ด้วยตัวแปร i คือ แถว j คือ คอลัมน์ และใช้ เป็น i+1 และ j+1 เพราะให้เป็นไปตามธรรมชาติ เพราะ Index ภาษาซีเริ่มตั้งแต่ 0 อาจทำให้งงได้ ส่วนการพิมพ์ ในแต่ละแถวแล้วขึ้นบรรทัดใหม่ ต้องกำหนดที่ลูป i เมื่อเปลี่ยน i ก็ขึ้นบรรทัดใหม่

ตัวอย่างที่ 8.6 กำหนดค่าให้ กับ Matrix 3 x 3 แล้ว คำนวณหาค่าผลบวกของเส้นทแยงมุมของ matrix ดังกล่าว

วิธีคิด กำหนดค่า ให้กับ Matrix 3 x 3 แล้ว คำนวณหาค่าผลบวกของเส้นทแยงมุม ซึ่งสามารถคำนวณได้โดยหาผลบวกของค่าในแนวทแยงมุมหลัก คือที่ตำแหน่ง 1,1 2,2 3,3 เพื่อความชัดเจน สามารถแสดงได้ดังภาพที่ 8.5

1	2	3
4	5	6
7	8	9

ภาพที่ 8.5 แสดงการคำนวณหาค่าผลบวกของเส้นทแยงมุมของ Matrix

แสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.6

```
1   #include<stdio.h>
2   void main()
3   {
4           int matrix[3][3], i, j , Sum ;
5           // input data to matrix
6           for (i = 0 ; i < 3 ; i++)
7              for (j=0; j < 3 ; j++)
8              {
9           printf ("Enter value at row %d column %d: ", i+1,j+1) ;
10                  scanf("%d",&matrix[i][j]);
11             }
12          // cal  sum det
13          for (i = 0 ; i < 3 ; i++)
14              sum = Sum  + matrix [i][i] ;
15          printf (" Sum of Matrix is:   %d: ", Sum ) ;
16  }
```

โปรแกรมที่ 8.6 การคำนวณค่าผลบวกของเส้นทแยงมุมของ Matrix 3 x 3

เมื่อประมวลผลโปรแกรมที่ 8.6 จะได้ผลลัพธ์ดังนี้

```
Enter value at row 1 column 1  : 1
Enter value at row 1 column 2  : 2
Enter value at row 1 column 3  : 3

Enter value at row 2 column 1  : 4
Enter value at row 2 column 2  : 5
Enter value at row 2 column 3  : 6

Enter value at row 3 column 1  : 7
Enter value at row 3 column 2  : 8
Enter value at row 3 column 3  : 9

Det of  matrix   is :  15
```

การคำนวณหาผลบวกของเส้นทแยงมุมที่สำคัญ คือ ตำแหน่ง i = j เพราะฉะนั้น ไม่จำเป็นต้องใช้การวนซ้ำซ้อน 2 ชั้น สามารถใช้วนซ้ำ i ชั้นเดียวพอ สามารถหาผลบวกของ Matrix ได้

ตัวอย่างที่ 8.7 คำนวณหา ผลคูณของ Matrix 2 x 2 โดยกำหนดให้ Matrix A มีค่า เท่ากับ 1 2 3 และ 4 ตามลำดับ และ Matrix B มีค่า 5 6 7 และ 8

วิธีคิด กำหนดค่า ให้กับ Matrix A และ B แล้ว คำนวณหาค่าผลบวกของ Matrix โดยใช้หลักการ แถว คูณ หลัก เช่นที่ตำแหน่งที่ 1,1 ได้มาจาก A[1,1] * B[1,1] + A[1,2]*B[2,1] = 1*5 + 2*7 = 19 เป็นต้น เพื่อความชัดเจน สามารถแสดงได้ดังภาพที่ 8.6

A

1	2
3	4

B

5	6
7	8

C

19	22
43	50

ภาพที่ 8.6 แสดงการคำนวณหาผลบวกของ Matrix

ซึ่งแสดงการเขียนโปรแกรมได้ดังโปรแกรมที่ 8.7

```
1   #include <stdio.h>
2   void main()
3   {
4           int i, j, m,c[2][2] = { 0,0,0,0 } ;
5           int a[2][2] = { 1, 2, 3, 4 } ;
6           int b[2][2] = { 5, 6, 7, 8 } ;
7           for (i = 0 ; i < 2 ; i++)
8                   for (j = 0; j < 2 ; j++)
9                           for (m = 0 ; m < 2 ; m++)
10                                  c[i][j] = c[i][j] + a[i][m] * b[m][j];
11          for (i = 0 ; i < 2 ; i++)
12          {
13                  for (j = 0; j < 2 ; j++)
14                          printf (" %d ",c[i][j]);
15                  printf("\n") ;
16          }
17  }
```

โปรแกรมที่ 8.7 การคำนวณผลคูณของ Matrix

เมื่อประมวลผลโปรแกรมที่ 8.7 จะได้ผลลัพธ์ดังนี้

```
19   22
43   50
```

การทำงานของโปรแกรม

ลูป i	ลูป j	ลูป m
i=0	j=0	m=0 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[0][0] = c[0][0] + a[0][0] * b[0][0] c[0][0] = 0 + 1 * 5 = 5
i=0	j=0	m=1 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[0][0] = c[0][0] + a[0][1] * b[1][0] c[0][0] = 5 + 2 * 7 = **19**
i=0	j=1	m=0 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[0][1] = c[0][1] + a[0][0] * b[0][1] c[0][1] = 0 + 1 * 6 = 6
i=0	j=1	m=1 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[0][1] = c[0][1] + a[0][1] * b[1][1] c[0][1] = 6 + 2 * 8 = **22**
i=1	j=0	m=0 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[1][0] = c[1][0] + a[1][0] * b[0][0] c[1][0] = 0 + 3 * 5 = 15
i=1	j=0	m=1 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[1][0] = c[1][0] + a[1][1] * b[1][0] c[1][0] = 15 + 4 * 7 = **15+28=43**
i=1	j=1	m=0 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[1][1] = c[1][1] + a[1][0] * b[0][1] c[1][1] = 0 + 3 * 6 = 18
i=1	j=1	m=1 c[i][j] = c[i][j] + a[i][m] * b[m][j]; c[1][1] = c[1][1] + a[0][1] * b[1][1] c[1][1] = 18 + 4 * 8 = **18+32=50**

สรุป

ตัวแปรแบบแถวลำดับเป็นข้อมูลชนิดหนึ่งที่สามารถเก็บข้อมูลชนิดเดียวกันได้เป็นกลุ่มเป็นชุดในตัวแปรตัวกัน โดยสามารถกำหนดจำนวนของข้อมูลได้ในตอนประกาศตัวแปรว่าต้องการข้อมูลกี่จำนวนก็ประกาศเท่านั้น และในการอ้างอิงข้อมูลแต่ละตัวจะใช้ดรรชนีหรือซับสคริปต์ในการอ้างถึงโดยดรรชนีของโปรแกรมภาษาซีจะเริ่มตั้งแต่ 0 เพราะฉะนั้นหากประกาศตัวแปร int i[10]; หมายความว่าตัวแปร i เก็บข้อมูลได้ 10 จำนวน โดยดรรชนี เริ่มตั้งแต่ 0 – 9 คือ i[0] ถึง i[9] การรับและการแสดงข้อมูลสามารถใช้ฟังก์ชันได้เหมือนกับตัวแปรปกติ และการทำงานต่าง ๆ จะนิยมใช้คำสั่งการวนซ้ำมาช่วยในการเขียนโปรแกรม ซึ่งตัวแปรแบบแถวลำดับนิยมนำมาใช้ในการเขียนโปรแกรมระบบงานเป็นส่วนใหญ่

แบบฝึกหัด

1. จงประกาศตัวแปรแบบแถวลำดับเพื่อเก็บข้อมูลจำนวนทศนิยมจำนวน 5 ตัว
2. จงประกาศตัวแปรแบบแถวลำดับเพื่อเก็บข้อมูลจำนวนเต็มจำนวน 9 ตัว โดยกำหนดค่าเริ่มต้นเป็น 0
3. ให้นักศึกษาเขียนโปรแกรมเพื่อรับรหัสนักศึกษาจำนวน 11 หลัก ชื่อ-นามสกุล ของนักศึกษาแล้วแสดงข้อมูลดังกล่าวทางจอภาพ
4. ให้นักศึกษาเขียนโปรแกรมเพื่อรับตัวเลขจำนวนเต็ม 10 จำนวน เก็บไว้ในตัวแปรแบบแถวลำดับ แล้วแสดงออกมาทางจอภาพ
5. ให้นักศึกษาเขียนโปรแกรมเพื่อรับตัวเลขจำนวนเต็ม 10 จำนวน เก็บไว้ในตัวแปรแบบแถวลำดับ แล้วหาผลรวมของตัวเลขทั้ง 10 จำนวนแล้วแสดงจำนวนที่รับเข้ามาและผลรวมทางจอภาพ
6. ให้นักศึกษาเขียนโปรแกรมเพื่อรับคะแนนนักศึกษาเพื่อคำนวณหาคะแนนเฉลี่ยแสดงคะแนนทั้งหมดที่รับเข้ามาและค่าเฉลี่ยที่คำนวณได้
7. ให้นักศึกษาเขียนโปรแกรมเพื่อรับคะแนนนักศึกษาเพื่อคำนวณหาค่าฐานนิยม และแสดงคะแนนทั้งหมดที่รับเข้ามาและค่าฐานนิยมที่คำนวณได้
8. ให้นักศึกษาเขียนโปรแกรมเพื่อรับคะแนนนักศึกษาเพื่อคำนวณหาค่าเกรดของนักศึกษาแต่ละคน
9. ให้นักศึกษาเขียนโปรแกรมเพื่อรับข้อมูลตัวเลขจำนวน 10 จำนวนใส่ในตัวแปรแบบแถวลำดับแล้วหาค่ามากที่สุดและน้อยที่สุด
10. ให้นักศึกษาเขียนโปรแกรมเพื่อรับข้อมูลตัวเลขจำนวน 10 จำนวนแล้วทำการเรียงข้อมูลจากน้อยไปมากเพื่อแสดงผลทางจอภาพ

11. ให้นักศึกษาเขียนโปรแกรมรับค่าตัวเลขจำนวน 2 ชุด แต่ละชุดมีตัวเลข 3 จำนวน จากนั้นให้หาผลต่างของตัวเลขจำนวนที่ 1 ของชุดที่ 1 กับเลขจำนวนที่ 1 ของชุดที่ 2,.....ตัวเลขจำนวนที่ 3 ของชุดที่ 1 กับเลขจำนวนที่ 3 ของชุดที่ 2 และแสดงผลต่างเหล่านั้น โดยเก็บผลต่างที่ได้นั้นไว้เพื่อใช้งานต่อไป ดังตัวอย่าง *(ตัวเอียงเป็นข้อมูลที่ผู้ใช้ป้อน)*

หมายเหตุ ในการรับค่าตัวเลขชุดที่ 1 และ 2 นั้นให้เก็บค่าเอาไว้ใช้ต่อไปด้วย

```
Enter Matrix 1
    m1[0] = 1
    m1[1] = 2
    m1[2] = 3
Enter Matrix 2
    m2[0] = 4
    m2[1] = 5
    m2[2] = 6
Result Matrix 3
    m3[0] = -3
    m3[1] = -3
    m3[2] = -3
```

12. การส่งข้อมูลผ่านเครือข่าย อินเทอร์เน็ต โดยเฉพาะ Web Site ในวิธีการ GET จะต้องทำการรวมข้อมูลต่างๆ จำนวนหลายๆ ข้อมูล ในวิธีการที่เรียกว่า URL Encoding ซึ่งวิธีการนี้จะทำการแบ่งข้อมูลเป็น 2 ส่วน ดังนี้

(ก) คีย์ (Key) เป็นส่วนที่บ่งบอกว่าเป็นข้อมูลอะไร

(ข) ค่าของข้อมูล (Value) เป็นส่วนที่บ่งบอกว่าข้อมูลมีค่าอะไร

Key และ Value ของข้อมูลแต่ละชุดจะถูกคั่นด้วยเครื่องหมายเท่ากับ (=) และข้อมูลแต่ละชุดจะถูกคั่นด้วยเครื่องหมาย & อีกครั้งหนึ่ง ดังตัวอย่างของ URL Encoding ต่อไปนี้

Name=Thanyapon&Surname=Sananakin&Age=20

ข้อมูลข้างต้นประกอบด้วย 3 ข้อมูลดังนี้

(ก) ข้อมูลที่ 1 มีชื่อว่า Name มีค่า เป็น Thanyapon

(ข) ข้อมูลที่ 2 มีชื่อว่า Surname มีค่า เป็น Sananakin

(ค) ข้อมูลที่ 3 มีชื่อว่า Age มีค่า เป็น 20

โดยสมมติว่าผู้ใช้ไม่มีการป้อนข้อมูลที่เป็นอักขระพิเศษ เช่น @, # เป็นต้น

โจทย์ ให้นักศึกษาเขียนโปรแกรมเพื่อทำรับข้อมูล Key และ Value ของข้อมูลต่างๆ จนกว่าจะสิ้นสุดข้อมูลโดยการป้อนค่า Key เป็นค่าว่าง (มีความยาวเป็น 0) และทำการรวมข้อมูลในวิธีการ URL Encoding โดยมีตัวอย่างการทำงานดังนี้

ตัวอย่างการทำงานของโปรแกรม

Enter key : ***Name***

Enter value: ***Thanyapon***

Enter key : ***Surname***

Enter value: ***Sananakin***

Enter key : ***Age***

Enter value: ***20***

Enter key : __ (ค่าว่างเพื่อสิ้นสุดข้อมูล)

URL Encoding String is : Name=Thanyapon&Surname=Sananakin&Age=20

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: ดวงกมลสมัย

บทที่ 9
ฟังก์ชัน

โปรแกรมภาษาซี มีโครงสร้างการทำงานเป็นฟังก์ชัน ตั้งแต่การเริ่มต้นการทำงานของฟังก์ชัน main() และฟังก์ชันในการรับและแสดงข้อมูล scanf() และ printf() มีการทำงานเป็นฟังก์ชันทั้งสิ้น แต่ถึงแม้ว่าโปรแกรมภาษาซีได้มีฟังก์ชันมากมายให้ใช้งานแต่ในบางครั้งการทำงานของระบบงานก็ไม่ตรงกับฟังก์ชันที่มีในโปรแกรมภาษาซี โปรแกรมเมอร์จึงต้องสร้างฟังก์ชันเพื่อการทำงานเอง ซึ่งโปรแกรมภาษาซีก็มีรูปแบบในการสร้างฟังก์ชันไว้

การออกแบบฟังก์ชัน

การเขียนโปรแกรมคอมพิวเตอร์ทุกภาษานั้น จำเป็นต้องแบ่งงานที่ทำออกเป็นงานย่อยๆ หลายๆ งานแล้วจึงนำมารวมกันอีกครั้งหนึ่งเพื่อให้ได้ผลลัพธ์ของการแก้ปัญหาที่ต้องการ ซึ่งการแบ่งแยกลักษณะนี้เรียกว่า “การแตกเพื่อชัยชนะ ซึ่งเรียกโปรแกรมที่แตกออกมาในแต่ละส่วนนี้ว่าโมดูล โดยแต่ละงานย่อยที่แยกออกมานั้นจะมีหน้าที่ชัดเจนในการแก้ปัญหาหนึ่งปัญหาใดเท่านั้น ซึ่งเป็นปัญหาย่อยๆ ของระบบทั้งหมด ซึ่งวิธีการเขียนโปรแกรมแบบนี้มีข้อดีพอสรุปได้ดังนี้

1. ง่ายในการพัฒนาเพราะแต่ละโมดูลสามารถเขียนและทดสอบได้แยกต่างหากจากโมดูลอื่น

2. ง่ายในการตรวจหาข้อผิดพลาด เพราะการเขียนโปรแกรมในแต่ละส่วนนั้นจะสนใจเฉพาะการทำงานของแต่ละโมดูลที่กำลังพัฒนาอยู่เท่านั้น ไม่สนใจส่วนอื่น

3. โปรแกรมหนึ่งอาจช่วยกันทำได้หลายคน โดยแต่ละคนเขียนโมดูลของตนเอง และนำมารวมกันภายหลัง ทำให้มีส่วนช่วยให้งานเสร็จแล้วขึ้น

4. สามารถนำโมดูลที่พัฒนาเสร็จแล้ว ไปปรับปรุง เปลี่ยนแปลงใช้กับงานอื่นๆ ที่มีคล้ายกันได้

5. ทำให้ปัญหาที่มีความซับซ้อน สามารถลดความซับซ้อนออกเป็นส่วนๆ ในโมดูลได้ ทำให้เข้าใจโปรแกรมได้ง่ายขึ้น และสะดวกในการตรวจสอบภายหลังอีกด้วย

6. ผู้พัฒนาแต่ละโมดูลสามารถทดสอบโมดูลของตนเองได้อย่างเป็นอิสระ

ในการเขียนโปรแกรมคอมพิวเตอร์นั้นมีข้อแนะนำในการสร้างโมดูล พอจะสรุปได้ดังนี้

1) โมดูลควรมีขนาดเล็ก ความยาวไม่ควรเกิน 1-2 หน้ากระดาษ

2) โมดูลควรมีทางเข้าเพียงทางเดียวและสิ้นสุดการทำงานเพียงทางเดียว

3) โมดูลควรมีความเป็นอิสระ สมบูรณ์ในตัวเอง

4) แต่ละโมดูลควรมีหน้าที่เฉพาะอย่างใดอย่างหนึ่งเท่านั้น

5) แต่ละโมดูลควรมีส่วนเชื่อมโยงที่ชัดเจน คือข้อมูลนำเข้า มาจากโมดูลไหน และเป็นชนิดอะไร รวมถึงมีข้อมูลที่ต้องส่งมาจากโมดูลอื่นๆ จำนวนเท่าไร เป็นต้น

ในภาษาซี นั้นได้บังคับไว้ว่าจะต้องมีอย่างน้อย 1 โมดูลคือโมดูลชื่อ main() ซึ่งโปรแกรมจะเริ่มทำงานที่โมดูลนี้ก่อน จากนั้นหากโมดูลนี้ไปเรียกใช้โมดูลอื่นๆ ก็จะเป็นไปภายหลัง

ตัวอย่างที่ 9.1 ให้นักศึกษาเขียนผังงานของโปรแกรมในการคำนวณพื้นที่รูปเรขาคณิตดังต่อไปนี้

ก) รูปวงกลม

ข) รูปสามเหลี่ยม

ค) รูปสี่เหลี่ยม

วิธีคิด จะต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลพื้นที่ของรูปเรขาคณิต

2. วิเคราะห์ที่มาของข้อมูล รับมาทาง Keyboard

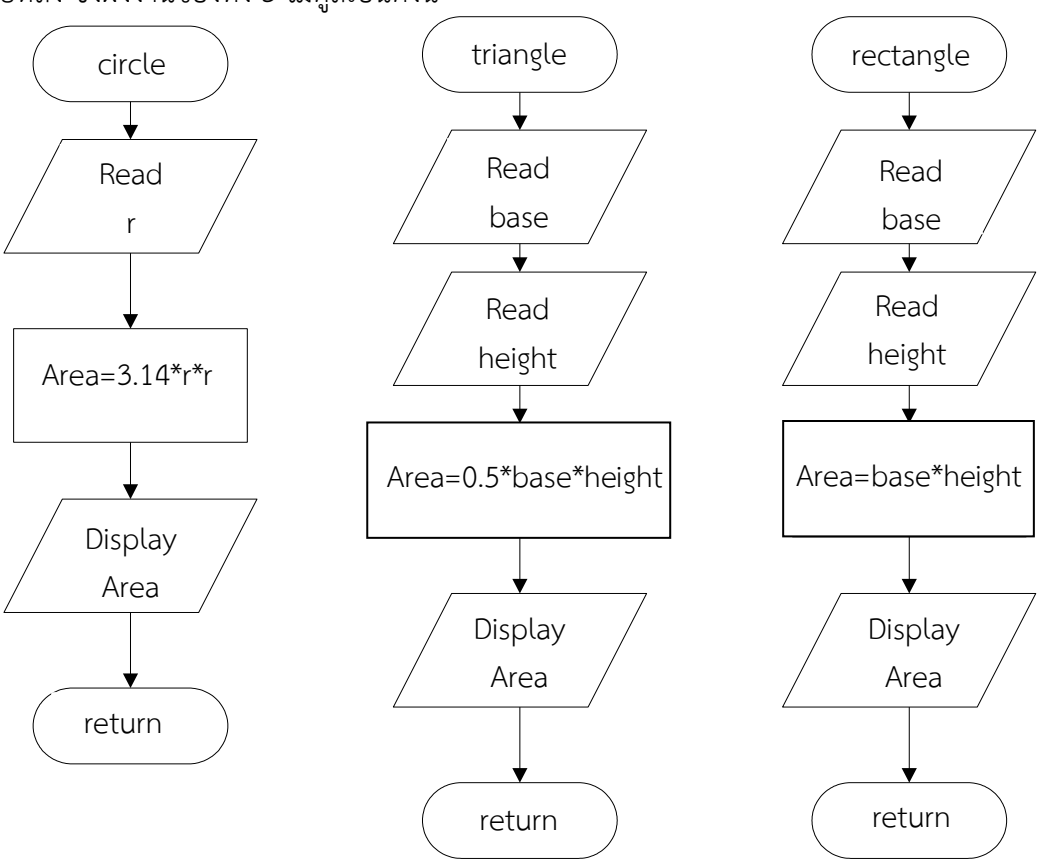
3. วิเคราะห์ความสัมพันธ์ของข้อมูล ความสัมพันธ์ของข้อมูลในข้อนี้แยกได้เป็น 3 กรณี ดังนี้

1) รูปวงกลม หาพื้นที่ได้จากสูตร $\pi \cdot r^2$ เมื่อ π เป็นค่าคงที่เท่ากับ 3.14 และ r เป็นรัศมี ของวงกลม

2) รูปสามเหลี่ยม หาพื้นที่ได้จากสูตร $\frac{1}{2} \times base * height$

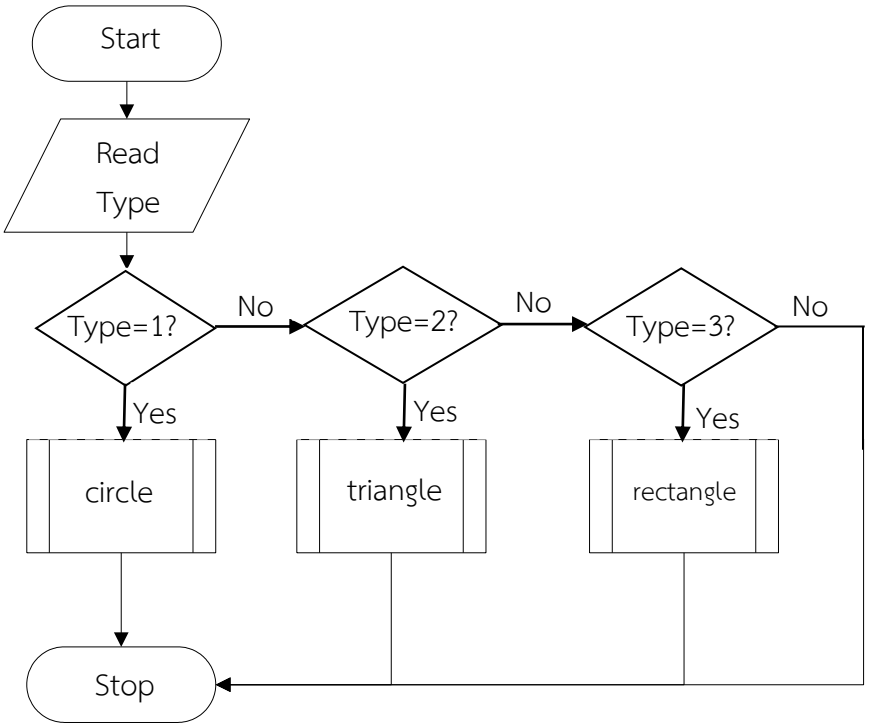
3) รูปสี่เหลี่ยม หาพื้นที่ได้จากสูตร $base * height$

ซึ่งจะเห็นได้ว่าแยกคิดได้เป็น 3 กรณี ดังนั้นหากจะเขียนทั้งหมดรวมเป็นโปรแกรมเดียวกันจะเกิดความไม่สัมพันธ์กันภายในโปรแกรม ดังนั้นจะทำการแยกออกเป็น 3 โมดูลแล้วรวมกันภายหลัง ซึ่งผังงานของทั้ง 3 โมดูลเป็นดังนี้



ภาพที่ 9.1 ผังงานโปรแกรมคำนวณพื้นที่รูปเรขาคณิต

จากภาพที่ 9.1 จะเห็นได้ว่าผังงานนี้แตกต่างจากผังงานที่เคยเรียนมาก่อน กล่าวคือจะเริ่มต้นด้วยชื่อซึ่งบ่งบอกถึงการทำงานของโมดูล (ชื่อโมดูล) เช่น circle, triangle, rectangle และสิ้นสุดด้วย reeturn เพื่อบ่งบอกถึงการสิ้นสุดการทำงานของโมดูล ไม่ใช่ของโปรแกรมหลัก ซึ่งจะเห็นได้ว่าโมดูลที่เขียนในภาพที่ 9.1 นี้ไม่สามารถทำงานได้ เนื่องจากไม่มีส่วนเริ่มต้นและสิ้นสุดของโปรแกรม ซึ่งต้องมีผังงานอีกลักษณะหนึ่งที่บอกว่าโมดูลใดเริ่มการทำงานเมื่อใด ซึ่งเขียนการทำงานของโมดูลหลักได้ดังภาพที่ 9.2



ภาพที่ 9.2 การทำงานของโมดูลหลักในการแก้ปัญหา

จากภาพที่ 9.2 จะเห็นได้ว่าการทำงานของโมดูลหลักนั้นจะเหมือนกับผังงานที่เคยเรียนไปแล้ว แต่จะต่างกันที่สัญลักษณ์ในการเรียกใช้โมดูลที่เพิ่มเข้ามาเพื่อให้รู้ว่าเป็นการใช้งานโปรแกรมในส่วนโมดูลย่อย

รูปแบบการประกาศฟังก์ชันในภาษาซี

ในการประกาศฟังก์ชันเพื่อให้ทำงานตามวัตถุประสงค์ที่ต้องการนั้น ฟังก์ชันหนึ่งๆ จะต้องมีการประกาศ 2 ส่วนได้แก่

1. ต้นแบบของฟังก์ชัน

ต้นแบบของฟังก์ชัน (Function Prototype) เพื่อช่วยให้การทำงานของแต่ละฟังก์ชันสามารถทำงานร่วมกันได้ เช่นการประกาศชนิดของข้อมูลที่ต้องนำเข้าสู่ฟังก์ชันเพื่อประมวลผล และชนิดของข้อมูลที่ต้องออกมาจากฟังก์ชัน เพื่อส่งให้โปรแกรมหลัก main() นำไปใช้ต่อไปนี้ การประกาศต้นแบบของฟังก์ชันจะต้องประกาศไว้ในตอนต้นของโปรแกรม โดยมีรูปแบบดังนี้

รูปแบบ : ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน(ชนิดข้อมูล ข้อมูล1,....., ชนิดข้อมูล ข้อมูลn);

หมายเหตุ : ขอให้สังเกตว่าในการประกาศต้นฉบับของฟังก์ชันนั้นจะมีเครื่องหมายอัฒภาค ;
: ข้อมูลนำเข้ามีได้หลายข้อมูล แต่ข้อมูลที่ส่งกลับมีได้เพียง 1 ข้อมูลเท่านั้น

ในส่วนของต้นแบบของฟังก์ชันนี้จะมีเฉพาะชนิดของข้อมูลที่เกี่ยวข้องกับฟังก์ชันเท่านั้น เช่น มีข้อมูลที่ต้องรับเข้าจำนวนกี่ตัว อะไรบ้าง เป็นชนิดตัวแปรอะไร รวมถึงชนิดข้อมูลที่ส่งกลับ โดยไม่มีรายละเอียดการทำงานของฟังก์ชัน

2. การนิยามการทำงานฟังก์ชัน

การนิยามการทำงานฟังก์ชัน (Function Definition) เป็นการกำหนดลำดับขั้นตอนการทำงานฟังก์ชันแต่ละฟังก์ชัน โดยมีรูปแบบนิยามการทำงานของฟังก์ชันดังนี้

```
รูปแบบ : ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน(ชนิดข้อมูล ข้อมูล1,....., ชนิดข้อมูล ข้อมูลn)
         {
              คำสั่งที่ให้ทำงานภายในฟังก์ชัน;
              return();
          }
หมายเหตุ : ขอให้สังเกตว่าในการนิยามการทำงานฟังก์ชันนั้นจะไม่มีเครื่องหมายอัฒภาค ; ที่หลัง ()
         : ข้อมูลนำเข้ามีได้หลายข้อมูล แต่ข้อมูลที่ส่งกลับมีได้เพียง 1 ข้อมูลเท่านั้น
```

สำหรับการนิยามการทำงานของฟังก์ชันนั้นจะคล้ายกันกับการประกาศต้นแบบของฟังก์ชันโดยเฉพาะบรรทัดที่เป็นชื่อฟังก์ชันจะมีรายละเอียดเหมือนกัน แต่ในส่วนของการนิยามการทำงานของฟังก์ชันนั้นจะมีคำสั่งสำหรับสิ้นสุดการทำงานของฟังก์ชันคือ return() เพื่อใช้ในการส่งค่ากลับจากการทำงานไปยังฟังก์ชันอื่นๆ เพื่อดำเนินการต่อไป โดยชนิดของตัวแปรจะที่ส่งกลับนั้นต้องเป็นชนิดเดียวกับชนิดข้อมูลที่ส่งกลับ ยกเว้นกรณีที่ฟังก์ชันไม่ต้องการที่จะส่งค่ากลับ ให้ใส่ชนิดข้อมูลที่ส่งกลับเป็น void ดังตัวอย่าง

- void main() หมายถึงฟังก์ชันชื่อ main() ไม่มีการรับข้อมูลจากฟังก์ชันอื่น เพราะเป็นฟังก์ชันเริ่มแรก และไม่มีการส่งค่ากลับยังฟังก์ชันอื่น เพราะเป็นฟังก์ชันสุดท้ายที่จะจบโปรแกรม
- int sum(int a,int b) หมายถึงฟังก์ชันชื่อ sum() รับข้อมูลเข้าสู่การประมวลผล 2 ตัวคือ a และ b เป็นข้อมูลชนิดเลขจำนวนเต็ม (int) และส่งค่ากลับเป็นชนิดเลขจำนวนเต็ม
- float fac(int a) หมายถึงฟังก์ชันชื่อ fac() รับข้อมูลเข้าสู่การประมวลผล 1 ตัวคือ a เป็นข้อมูลชนิดเลขจำนวนเต็ม (int) และส่งค่ากลับเป็นชนิดเลขทศนิยม (float)
- void swap(int a,int b) หมายถึงฟังก์ชันชื่อ swap() รับข้อมูลเข้าสู่การประมวลผล 2 ตัวคือ a และ b เป็นข้อมูลชนิดเลขจำนวนเต็ม (int) และโดยไม่ส่งค่ากลับ (void)

การเรียกใช้งานฟังก์ชัน

เมื่อประกาศฟังก์ชันเรียบร้อยแล้ว ฟังก์ชันนั้นจะยังไม่มีผลต่อการทำงาน เพราะยังไม่ถูกเรียกใช้โดย main() ซึ่งต้องเรียกใช้ฟังก์ชันในรูปแบบดังนี้

รูปแบบ	:	ตัวแปร = ชื่อฟังก์ชัน (ชนิดข้อมูล ข้อมูล1,....., ชนิดข้อมูล ข้อมูลn);
หมายเหตุ	:	ตัวแปรเป็นชื่อตัวแปรที่ต้องการเก็บผลลัพธ์จากการทำงานของฟังก์ชันไว้

ตัวอย่างการประกาศฟังก์ชัน เป็นดังแสดงในโปรแกรมที่ 9.1

```
1   #include <stdio.h>
2   #include <conio.h>
3   int sum(int,int);       /* Declare Function Prototype */
4   int sum(int a,int b)    /* Declare Function Definition */
5   {
6           int result;
7           result=a+b;
8           return(result);
9   }
10  void main()
11  {
12          int a,b;
13          int Total;
14          a=10;
15          b=20;
16          Total=sum(a,b);    /* Function Call */
17          printf("Total of %d + %d is %d",a,b,Total);
18  }
```

โปรแกรมที่ 9.1 การประกาศฟังก์ชัน และการเรียกใช้งาน

ในโปรแกรมที่ 9.1 การประกาศต้นแบบของฟังก์ชันในบรรทัดที่ 3 int sum(int,int); กำหนดให้ฟังก์ชันชื่อ sum ต้องการพารามิเตอร์ชนิดของตัวแปรประเภทตัวเลข (int) จำนวน 2 ตัวแปร และผลลัพธ์ของฟังก์ชัน ชนิดของตัวแปรประเภทตัวเลข (int)

ในบรรทัดที่ 6 ถึง 11 เป็นการแสดงการนิยามการทำงานของฟังก์ชัน โดยให้มีการส่งค่าตัวแปร result กลับสู่โปรแกรมหลัก ให้สังเกตว่าตัวแปร result จะเป็นชนิดเดียวกับชนิดของตัวแปรที่ต้องการส่งค่ากลับ (int) และในบรรทัดที่ 19 จะมีการเรียกใช้ฟังก์ชันโดยส่งค่าตัวแปร a และ b ไปยังฟังก์ชัน sum เพื่อประมวลผลโดย a มีค่า 10 และ b มีค่า 20 และเมื่อฟังก์ชัน sum ทำงานเสร็จแล้วจะส่งค่าตัวแปร result กลับมายังตัวแปร Total จะเห็นว่าจะเกิดตัวแปร a และ b ใน 2 ส่วนซึ่งอาจเกิดความสับสนในการใช้งานของตัวแปรได้ดังนั้นตัวแปลภาษาซี จึงได้กำหนดให้การเข้าถึงตัวแปรแบ่งเป็น 2 ลักษณะคือตัวแปรทั่วไป (Global Variable) และตัวแปรท้องถิ่น (Local Variable)

ขอบเขตการใช้งานของตัวแปร

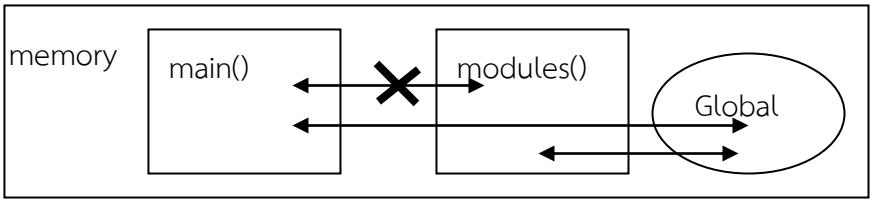
เมื่อทำการแบ่งแยกโปรแกรมออกเป็นส่วนย่อยๆ แล้วดังนั้นการจัดสรรหน่วยความจำของโปรแกรมก็จะแยกออกจากกันด้วย ดังนั้นจึงต้องศึกษาเรื่องของขอบเขตการใช้งานของตัวแปรซึ่งแบ่งได้เป็น 2 ประเภทคือ

1.ตัวแปรส่วนกลาง

ตัวแปรส่วนกลาง (Global Variable) เป็นตัวแปรที่ใช้งานได้ตลอดทั้งโปรแกรม ทุกฟังก์ชันสามารถเข้าถึงและเปลี่ยนแปลงข้อมูลของตัวแปรชนิดนี้ได้ตลอดเวลา ตัวแปรประเภทนี้จะเป็นตัวแปรทุกๆ ตัวที่ไม่อยู่ภายใต้การประกาศของฟังก์ชันใด ฟังก์ชันหนึ่ง (รวมถึง main()) นั่นคือประกาศในตอนต้นของโปรแกรม หลังจาก #include, #define เป็นต้น

2.ตัวแปรเฉพาะที่

ตัวแปรเฉพาะที่ (Local Variable) เป็นตัวแปรที่ใช้งานได้เฉพาะในฟังก์ชันนั้นๆ ไม่สามารถเข้าถึงตัวแปรประเภทนี้ได้จากฟังก์ชันอื่น ได้แก่ตัวแปรทุกตัวที่ประกาศภายใต้เครื่องหมาย { และ } ของฟังก์ชันใดฟังก์ชันหนึ่ง



ภาพที่ 9.3 การแบ่งหน่วยความจำในการเข้าถึงตัวแปร

จากภาพที่ 9.3 จะเห็นแสดงการแบ่งหน่วยความจำ ซึ่งฟังก์ชัน main() และแต่ละโมดูลซึ่งเขียนเป็นฟังก์ชันย่อยๆ นั้นจะสามารถเข้าถึงข้อมูลในตัวแปร Global ได้ แต่ไม่สามารถเข้าถึงข้อมูลซึ่งกันและกัน (ระหว่าง main() และ modules() ได้)

ในโปรแกรมที่ 9.1 นี้ตัวแปร a และ b เป็นตัวแปรเฉพาะที่ ซึ่งเป็นตัวแปรเฉพาะที่ ทั้งในฟังก์ชัน main() และในฟังก์ชัน sum() ด้วยดังนั้นตัวแปร a และ b ไม่สามารถใช้งานร่วมกันได้

ในกรณีที่ตัวแปรทั้งชนิดตัวแปรส่วนกลาง และตัวแปรเฉพาะที่มีชื่อเหมือนกันก็จะพิจารณาเฉพาะตัวแปรเฉพาะที่ก่อนจึงจะพิจารณาตัวแปรส่วนกลางในลำดับถัดจากตัวแปรเฉพาะที่

ตัวอย่างที่ 9.2 เขียนฟังก์ชันคำนวณพื้นที่รูปเรขาคณิตดังต่อไปนี้

ก) รูปวงกลม
ข) รูปสามเหลี่ยม
ค) รูปสี่เหลี่ยม

วิธีคิด ให้ดูผังงานในภาพที่ 9.1 และ ภาพที่ 9.2 ประกอบจะเขียนโปรแกรมได้ดังโปรแกรมที่ 9.2

```
1  #include <stdio.h>
2  void circle(void);    // ประกาศต้นแบบของฟังก์ชันทั้ง circle
3  void triangle(void);
4  void rectangle(void);
5  void main()
6  {
7      int type;
8      printf("1. Circle\n");  // เพื่อให้ง่ายจะแทนแต่ละรูปแบบโดยการใช้ตัวเลข
9      printf("2. Triangle\n");
10     printf("3. Rectangle\n");
11     printf("Select Polygon type: ");
12     scanf("%d",&type);              // รับค่าตัวเลขของแต่ละรูปเรขาคณิต
13     switch(type)           //  ตรวจสอบว่าเป็นรูปเรขาคณิตใด
14     {
15        case 1: circle();     // วงกลม เรียกใช้ฟังก์ชันในการหาพื้นที่วงกลม
16                break;
17        case 2: triangle();  // สามเหลี่ยม เรียกใช้พื้นที่สามเหลี่ยม
18                break;
19        case 3: rectangle();  // สี่เหลี่ยม เรียกใช้ฟังก์ชันในการหาพื้นที่สี่เหลี่ยม
20                break;
21        default: printf("Invalid shape\n");
22     }
23 }
24 void circle()   //  ฟังก์ชันในการหาพื้นที่วงกลม
25 {
26     float area;
27     float r;
28     printf("Enter Radius: ");     scanf("%f",&r);
29     area=3.14*r*r;
30     printf("Area of the circle = %.2f",area);
31 }
32 void triangle()  //  ฟังก์ชันในการหาพื้นที่สามเหลี่ยม
33 {
34     float area;
35     float h,b;
36     printf("Enter height: "); scanf("%f",&h);
```

```
37      printf("Enter base: ");    scanf("%f",&b);
38      area=0.5*h*b;
39      printf("Area of the triangle = %.2f",area);
40  }
41  void rectangle()  //  ฟังก์ชันในการหาพื้นที่สี่เหลี่ยม
42  {
43      float area;
44      float h,b;
45      printf("Enter height: ");  scanf("%f",&h);
46      printf("Enter base: ");    scanf("%f",&b);
47      area= h*b;
48      printf("Area of the rectangle = %.2f",area);
49  }
```

โปรแกรมที่ 9.2 โปรแกรมในการแก้ปัญหาการคำนวณพื้นที่รูปเรขาคณิต

ตัวอย่างที่ 9.3 ให้นักศึกษาเขียนโปรแกรมให้ประกอบด้วยฟังก์ชันชื่อ power(int a,int b) เพื่อทำการคำนวณค่าของ a^b เมื่อ a และ b มีค่ามากกว่า 0 และ b เป็นจำนวนเต็ม

วิธีคิด จะต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือผลคูณของ a จำนวน b ครั้ง

2. วิเคราะห์ที่มาของข้อมูล โจทย์ไม่ได้บอกว่ารับข้อมูลอะไรมาบ้างแต่ให้สูตรมาว่าให้หาค่า a^b โดยบอกว่า a และ b มีค่ามากกว่า 0 และ b เป็นจำนวนเต็ม ดังนั้นตัวแปร a, b ควรเป็น int

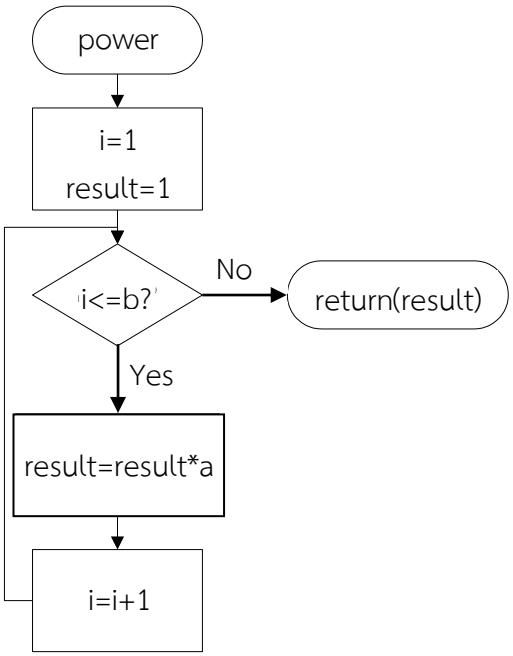
แต่โจทย์บอกว่าให้ทำการคำนวณในฟังก์ชัน ดังนั้น a,b ต้องเป็น input ของฟังก์ชัน โดยรับค่ามาจากโปรแกรมหลัก main()

3. วิเคราะห์ความสัมพันธ์ของข้อมูล

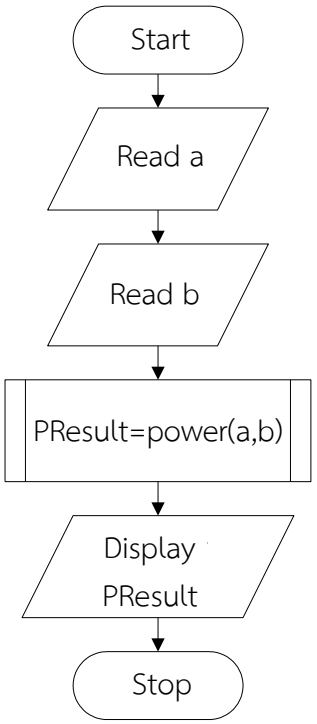
หลักการ a^b จะต้องมี a คูณกัน b ครั้ง ดังนั้นวิธีที่ง่ายที่สุดก็คือใช้ Loop for คล้ายๆ กับการหาผลรวม แต่เปลี่ยนแปลงเล็กน้อย ดังนี้

```
for (i=1; i<=b; i++)
    result = result *a;
```

จากนั้นก็เขียนผังงานของโปรแกรมจะได้ผังงานดังภาพที่ 9.4



(ก) ผังงานโมดูลย่อย power



ข) ผังงานของโมดูลควบคุม

ภาพที่ 9.4 ผังงานของการแก้ปัญหาในการหาค่า power (a,b)

ซึ่งเมื่อนำมาเขียนโปรแกรมจะได้โปรแกรมดังโปรแกรมที่ 9.3

```
1   #include <stdio.h>
2   float power(int a,int b);
3   void main()
4   {
5      int a,b;
6      float PResult;
7      clrscr();
8      printf("Enter base (a) : "); scanf("%d",&a);
9      printf("Enter power (b): "); scanf("%d",&b);
10     PResult = power (a,b);
11     printf("Result of %d^%d is %f",a,b,PResult);
12
13  }
14
15  float  power (int a,int b)
16  {
17     int i;
18     float result;
19     result =1;
20     for(i=1;i<b;i++)
21          result=result*a;
22     return(result);
23  }
```

โปรแกรมที่ 9.3 การเขียนโปรแกรมในการหาค่า a ยกกำลัง b

ฟังก์ชันการคำนวณทางคณิตศาสตร์

ภาษาซีได้สร้างฟังก์ชันมาตรฐานมากมาย อยู่ในคลังโปรแกรมต่าง ๆ เช่น ในคลังโปรแกรม math.h มีฟังก์ชันทางคณิตศาสตร์ที่น่าสนใจคือ sin(), cos(), tan() และ ฟังก์ชันอื่น ๆ โดยมีรายละเอียดดังตารางที่ 9.1

ตารางที่ 9.1 ฟังก์ชันทางคณิตศาสตร์

ฟังก์ชัน	รายละเอียดของฟังก์ชัน
sin(x)	double sin(double x); ฟังก์ชันตรีโกณมิติหาค่า sin ของมุม โดยที่ x เป็นมุมที่ต้องการหามีหน่วยเป็นเรเดียน
cos(x)	double cos(double x); ฟังก์ชันตรีโกณมิติหาค่า cos ของมุม โดยที่ x เป็นมุมที่ต้องการหามีหน่วยเป็นเรเดียน
tan(x)	double tan(double x); ฟังก์ชันตรีโกณมิติหาค่า tan ของมุม โดยที่ x เป็นมุมที่ต้องการหา มีหน่วยเป็นเรเดียน
asin()	double asin(double x); ฟังก์ชันตรีโกณมิติ ฟังก์ชันผกผัน (Inverse) ของ sin หรือ arc sine โดยที่ x เป็นมุมที่ต้องการหา มีหน่วยเป็นเรเดียน
acos()	double acos(double x); ฟังก์ชันตรีโกณมิติ ฟังก์ชันผกผัน (Inverse) ของ cos หรือ arc cos โดยที่ x เป็นมุมที่ต้องการหา มีหน่วยเป็นเรเดียน
atan()	double atan(double x); ฟังก์ชันตรีโกณมิติ ฟังก์ชันผกผัน (Inverse) ของ tan หรือ arc tan โดยที่ x เป็นมุมที่ต้องการหา มีหน่วยเป็นเรเดียน
sqrt(x)	double sqrt(double x); ฟังก์ชันหาค่ารากที่สองของค่า x ($\sqrt{x}$) โดยที่ x เป็นตัวแปรหรือค่าคงที่ ซึ่งเป็นจำนวนเต็มบวก หรือจำนวนเต็มศูนย์
abs()	ฟังก์ชันหาค่าสัมบูรณ์ของเลขจำนวนเต็ม ค่าสัมบูรณ์ของเลขจำนวนเต็ม เช่น I=abs(-20); จะได้ I มีค่า 20
pow(x,y)	double pow(double x, double y); ฟังก์ชันหาค่ายกกำลังของ x^y โดยที่ x,y เป็นตัวแปรหรือค่าคงที่ ซึ่งเป็นเลขฐาน จำนวนเต็มบวก หรือจำนวนเต็มศูนย์
log(x)	ฟังก์ชันหาค่า log ฐาน e โดยที่ x เป็นตัวแปรหรือค่าคงที่ ซึ่งเป็นจำนวนเต็มบวก หรือจำนวนเต็มศูนย์
log10(x)	ฟังก์ชันหาค่า log ฐาน 10 โดยที่ x เป็นตัวแปรหรือค่าคงที่ ซึ่งเป็นจำนวนเต็มบวก หรือจำนวนเต็มศูนย์

ที่มา: ไกรศร ตั้งโอภากุล. 2554: 163

ตัวอย่างที่ 9.4 การใช้งานฟังก์ชันตรีโกณมิติ

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <math.h>
4   void main()
5   {
6      float x , y , z ;
7      clrscr();
8      x = 3 ;
9      y = 2 ;
10     z = power(2,3) ;
11     printf("sin (%.0f) = %.4f \n", x, y );
12     x = 60 ;
13     y = cos(x) ;
14     printf("cos (%.0f) = %.4f \n", x, y );
15     x = 45 ;
16     y = tan(x) ;
17     printf("tan (%.0f) = %.4f \n", x, y );
18     getch() ;
19  }
```

โปรแกรมที่ 9.4 การใช้งานฟังก์ชันตรีโกณมิติ

ผลลัพธ์

sin (30) = -0.9880
cos (60) = -0.9524
tan (45) = 1.6198

โปรแกรมที่ 9.4 แสดงให้เห็นการเรียกใช้งานฟังก์ชัน sin() ฟังก์ชัน cos() และฟังก์ชัน tan() เช่น double sin(double x); ฟังก์ชันตรีโกณมิติหาค่า sin ของมุม โดยที่ x เป็นมุมที่ต้องการหามีหน่วยเป็นเรเดียน ค่าของ sin(30) มีค่าเท่ากับ -0.9880 โดยส่งค่ามุมเป็น ได้ผลลัพธ์เป็นค่า sin ส่วนฟังก์ชัน cos() และ ฟังก์ชัน tan() เช่นเดียวกัน การใช้งานคล้านคลึงกัน

ตัวอย่างที่ 9.5 การใช้งานฟังก์ชันทางคณิตศาสตร์

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <math.h>
4   void main()
5   {
6      float  x , y , z ;
7      clrscr() ;
8      x = 2 ;
9      y = 3 ;
10     z = pow(2,3) ;
11     printf (“pow ( %.0f , %.0f ) = %.0f \n” , x , y , z) ;
12     y = -45 ;
13     z = abs(y) ;
14     printf (“abs ( %.0f ) = %.0f \n” ,  y , z) ;
15     y = 16 ;
16     z = sqrt(y) ;
17     printf (“sqrt ( %.0f ) = %.0f \n” , y , z) ;
18     x = 100 ;
19     y = log10(x) ;
20     z = log(y) ;
21     printf (“log10 ( %.0f ) = %.2f \n” , x , y) ;
22     printf (“log ( %.0f ) = %.2f \n” , x , z) ;
23  }
```

โปรแกรมที่ 9.5 การใช้งานฟังก์ชันทางคณิตศาสตร์

ผลลัพธ์

pow (2 , 3) = 8
abs (-45) = 45
sqrt (16) = 4
log10 (100) = 10
log (100) = 4.61

โปรแกรมที่ 9.5 แสดงให้เห็นการใช้งานฟังก์ชันทางคณิตศาสตร์ ฟังก์ชัน pow (2 , 3) หมายถึง 2 ยกกำลัง 3 มีค่าเท่ากับ 8 ฟังก์ชัน abs (-45) หมายถึงค่าสัมบูรณ์ของ -45 มีค่าเท่ากับ 45 ฟังก์ชัน sqrt (16) หมายถึงค่ารากที่ 2 ของ 16 มีค่าเท่ากับ 4 การคำนวณหาค่ารากที่ 2 ควรหาค่ารากที่ 2 ของค่าตัวเลขที่เป็นบวกเท่านั้น ฟังก์ชัน log10 (100) = 10 เป็นการหาค่า log ฐานสิบ ฟังก์ชัน log (100) = 4.61 เป็นการหาค่า ln (100)

ตัวอย่างที่ 9.6 การส่งค่าตัวแปรแบบแถวลำดับผ่านฟังก์ชัน

```
1   #include <stdio.h>
2   #include <conio.h>
3   void  FunctionCall( int a[5]) ;
4
5   void main()
6   {
7      int  x[5] = {1, 2, 3, 4, 5}  ;
8      clrscr() ;
9      FunctionCall (x) ;
10  }
11  void  FunctionCall(int a[5])
12  {
13      int i ;
14      for ( i=0; i< 5; i++)
15           printf (“a[%d] = %d \n” , i, a[i]) ;
16  }
```

โปรแกรมที่ 9.6 การส่งค่าตัวแปรแบบแถวลำดับผ่านฟังก์ชัน

ผลลัพธ์

a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5

โปรแกรมที่ 9.6 แสดงให้เห็นการส่งค่าตัวแปรแบบแถวลำดับผ่านฟังก์ชัน โดยสามารถส่งค่าตัวแปรแบบแถวลำดับ ผลลัพธ์แสดงค่าของตัวแปรแบบแถวลำดับที่กำหนดค่าจากฟังก์ชัน main() ส่งมาให้ฟังก์ชัน FunctionCall() แสดงค่าสมาชิกในแต่ละตัวของตัวแปรแบบแถวลำดับ

สรุป

การเขียนโปรแกรมในเบื้องต้นนักศึกษาอาจจะไม่ยุ่งยากนักในการอ่านโปรแกรมในการเขียนโปรแกรมเพราะโปรแกรมยังเป็นขนาดเล็กการทำงานมีเพียงอย่างเดียว หรือสองอย่างยังไม่ซับซ้อนนัก การเขียนเป็นฟังก์ชันอาจจะยังไม่เห็นประโยชน์ชัดเจนนัก แต่เมื่อนักศึกษาเขียนโปรแกรมใหญ่ขึ้น การทำงานซับซ้อนขึ้นมีการทำงานซ้ำไปซ้ำมาคล้าย ๆ กันนักศึกษาควรเขียนเป็นฟังก์ชันเพื่อการเรียกใช้งานได้ง่าย และโปรแกรมเป็นระเบียบอ่านเข้าใจง่ายและหากเป็นการทำงานจริง การทำงานจะทำเป็นทีมในระบบงานหนึ่งมีผู้เขียนโปรแกรมหลายคน ดังนั้นการเขียนฟังก์ชันเพื่อการทำงานต่าง ๆ จะเป็นประโยชน์มากในการอ่านโปรแกรม เพื่อให้ผู้อื่นอ่านเข้าใจได้ง่ายและบางครั้งมีการแก้ไขโปรแกรมเก่าก็สามารถกลับมาทำความเข้าใจได้ไม่ยาก

แบบฝึกหัด

1. ให้นักศึกษาตอบคำถามต่อไปนี้

ก) ฟังก์ชัน power() ในโปรแกรม 9.3 ทำไมตัวแปร result จึงต้องเป็นตัวแปร float อยากทราบว่าเป็น int ได้หรือไม่ เพราะเหตุใด

ข) ถ้าผู้ใช้ป้อนค่า b น้อยกว่า 0 โปรแกรมนี้จะทำงานได้ถูกต้องหรือไม่ ถ้าไม่ถูกต้องจะแก้ให้ถูกต้องได้อย่างไร

2. เขียนโปรแกรมในการหาค่า factorial โดยกำหนดว่า

$n! = n(n-1)(n-2).....3 \cdot 2 \cdot 1 \quad ; \quad n \geq 0$ โดยกำหนดให้ 0! และ 1! =1

โดยให้มีฟังก์ชันชื่อว่า factorial() และมีการทำงานลักษณะเดียวกับตัวอย่างที่ 9.3

3. เขียนฟังก์ชันหาผลบวกโดยส่งตัวตั้งและตัวบวก
4. เขียนฟังก์ชันเพื่อคำนวณหาเส้นรอบวงของวงกลม โดยส่งค่ารัศมีเป็นพารามิเตอร์
5. เขียนฟังก์ชันเพื่อการสลับค่าของตัวแปร 2 ตัว
6. เขียนฟังก์ชันเพื่อหาค่ามากที่สุด โดยส่งอาร์เรย์ของเลขจำนวนเต็มเป็นพารามิเตอร์
7. เขียนฟังก์ชันเพื่อหาค้นหาตำแหน่งของค่าที่ต้องการค้นหา โดยส่งอาร์เรย์ของเลขจำนวนเต็มและค่าที่ต้องการค้นหาเป็นพารามิเตอร์ หากไม่พบให้ส่งค่า 0 กลับฟังก์ชันหลัก
8. เขียนฟังก์ชันเพื่อแปลงอุณหภูมิจากองศาเซลเซียส เป็นองศาฟาเรนไฮท์ โดยส่งองศาเซลเซียส เป็นพารามิเตอร์ โดยใช้สูตร F = 9 * C / 5 + 32
9. เขียนฟังก์ชันเพื่อหาผลลัพธ์ของเลข a^n โดยส่ง a, n เป็นพารามิเตอร์
10. เขียนฟังก์ชันเพื่อใส่, ให้กับเลขที่มากกว่า 1000 โดยส่งตัวเลขเป็นพารามิเตอร์

11. ให้นักศึกษาพิจารณาโปรแกรมข้างล่าง แล้วตอบคำถามต่อไปนี้

11.1 ให้เติมโปรแกรมในจุดที่ขาดหายให้ทำงานได้ตามคำอธิบายที่เขียนไว้

11.2 หากป้อนค่า n เท่ากับ 7 ให้บอกผลลัพธ์ (หน้าจอ)

11.3 ตัวแปร flag ทำหน้าที่อะไรในโปรแกรมนี้ ให้อธิบายไม่เกิน 2 บรรทัด

```
#include <stdio.h>
void main()
{
     .....1.  ประกาศตัวแปร n,i และ flag เป็นชนิดข้อมูลตัวเลข ไม่มีทศนิยม
     clrscr();
     flag=1;
     printf("Enter value of n: ");     2 ..... รับค่าของตัวแปร n
     for(i=2;  3..... ตรวจสอบว่า i น้อยกว่า n และ flag มีค่าเป็น 1 ;i++)
     {
          4  ..... ตรวจสอบว่า n หารด้วย i เหลือเศษ 0 หรือไม่
               flag=0 ;
     }
     5..... ตรวจสอบว่า flag มีค่าเป็น 1 จริงหรือไม่
          printf("This number is prime.");
     else
          printf("This number is NOT prime.");
     getch();
}
```

12. ให้แก้ไขโปรแกรมต่อไปนี้ซึ่งมีข้อผิดพลาดอยู่ 10 จุด ให้ถูกต้องตามไวยกรณ์ (Syntax) ของภาษาซี

```
1  #include <stdio.h>
2  #include <conio.h>
3  #define MAX 3
4  void main()
5  {
6     int Matrix1[MAX][MAX], Matrix2[MAX][MAX];
7     int row,col;
8     clrscr();
9     printf("====Enter Matrix 1====\n")
10    for(row=0;row< MAX;row+)
11       for(col=0;col< MAX;col++) {
12            printf(" Matrix[%d][%d]: ",row,col);
13            scanf("%d",Matrix1[row][col]);
14            Matrix2[col][row]=Matrix1[row][col];
15       }
16
17    printf(===== Matrix 1 is ====\n");
18    for(row=0;row< MAX;row++) {
19       for(col=0;col< MAX ;col++) {
20            printf("%d\t",Matrix1[row][col]);
21       }
22       printf("\n);
23    }
24    printf("===== Calculated Matrix 2====\n");
25    for(row=0;row< MAX;row++)) {
26       for(col=0;col< MAX col++) {
27            printf("%d\t,Matrix2[row][col]");
28       }
29       print("\n");
30    }
31    gech();
32 }
```

เอกสารอ้างอิง

ไกรศร ตั้งโอภากุล. (2554). คู่มือเรียนเขียนโปรแกรมภาษา C. นนทบุรี: ไอดีซี พรีเมียร์.

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: ดวงกมลสมัย

ประภาพร ช่างไม้.(2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for Engineering and Technology.* London: Prentice Hall.

บทที่ 10
ตัวแปรชนิดตัวชี้

ในบทก่อนหน้านี้ได้กล่าวถึงตัวแปร การเก็บข้อมูลไว้ในตัวแปร ว่าจะต้องทำการเก็บไว้ในหน่วยความจำโดยมีการจัดสรรพื้นที่ในหน่วยความจำให้โดยระบบปฏิบัติการ โดยมีตัวแปลภาษาซีเป็นตัวกลางในการจัดสรรอีกชั้นหนึ่ง ดังนั้นหากต้องการที่จะเข้าถึงข้อมูลในหน่วยความจำโดยตรงจำเป็นจะต้องอ้างอิงถึงตำแหน่งของหน่วยความจำ แต่ไม่สามารถทราบได้ว่าตัวแปรนั้นๆ ถูกเก็บอยู่ในตำแหน่งใดในหน่วยความจำ ดังนั้นตัวแปลภาษาซี จึงมีการกำหนดตัวแปรชนิดพิเศษที่เรียกว่า “ตัวชี้ (Pointer)” ขึ้นมาเพื่อใช้ในการนี้โดยเฉพาะ

การประกาศตัวแปรตัวชี้

การประกาศตัวแปรตัวชี้ จะใช้วิธีเดียวกันกับการประกาศตัวแปรชนิดอื่นๆ แต่จะมีการเพิ่มเครื่องหมายพิเศษเพื่อบ่งบอกถึงความเป็นตัวแปรตัวชี้ นั่นคือต้องเติมเครื่องหมาย * ไว้หน้าชื่อตัวแปร ในรูปแบบ

รูปแบบ	:	ชนิดตัวแปร * ชื่อตัวแปร;
หมายเหตุ	:	ตัวแปรตัวชี้ (pointer) ต้องมีเครื่องหมาย * นำหน้าชื่อตัวแปร

โดยชนิดตัวแปรเป็นชนิดข้อมูลใดๆ ที่ต้องการเข้าถึง อาจจะเป็น int, char, หรือชนิดข้อมูลที่กำหนดเองอื่นๆ เช่น

int *ip; หมายถึงประกาศตัวแปรชื่อว่า ip ให้เป็นตัวแปรตัวชี้ของข้อมูลชนิด integer

float *fp; หมายถึงประกาศตัวแปรชื่อว่า fp ให้เป็น ตัวแปรตัวชี้ของข้อมูลชนิด float

หมายเหตุ ในการใช้ตัวแปรตัวชี้ นั้นหากต้องการที่จะเข้าถึงข้อมูลชนิดใด ต้องประกาศ ตัวแปรตัวชี้ให้เป็น ตัวแปรตัวชี้ ของชนิดข้อมูลนั้นๆ เสมอ ทั้งนี้เนื่องจากข้อมูลแต่ละประเภทใช้พื้นที่ในหน่วยความจำไม่เท่ากัน

ในเอกสารนี้จะแทนตำแหน่งของหน่วยความจำด้วยเลขฐานสิบหกนำหน้าด้วย 0x เพื่อความสะดวกในการเขียน แต่จริงๆ แล้วตำแหน่งของหน่วยความจำจะแทนด้วยเลขฐานสองในเครื่องคอมพิวเตอร์

การใช้งานตัวแปรตัวชี้

เนื่องจากตัวแปรตัวชี้ ไม่ได้ทำการเก็บค่าไว้ในตัวแปรนั้นๆ แต่จะเก็บค่าตำแหน่งของหน่วยความจำที่มันชี้อยู่แทน ดังนั้นการใช้งานตัวแปรตัวชี้ จึงแบ่งได้ 2 ลักษณะ คือ เพื่อเข้าถึงตำแหน่งของหน่วยความจำ และเพื่อเข้าถึงข้อมูลที่ชี้อยู่

1. การใช้งานตัวแปรตัวชี้ เพื่อเข้าถึงตำแหน่งของหน่วยความจำ

การใช้งานตัวแปรตัวชี้ เพื่อเข้าถึงตำแหน่งของหน่วยความจำ ในการนี้จะใช้ตัวแปรตัวชี้โดยระบุเฉพาะชื่อของตัวแปรตัวชี้ ไม่มีเครื่องหมาย * ข้างหน้า และในกรณีที่ต้องการอ้างถึงตำแหน่งของหน่วยความจำของตัวแปรอื่นๆ ให้ใช้เครื่องหมาย & นำหน้าตัวแปรนั้น

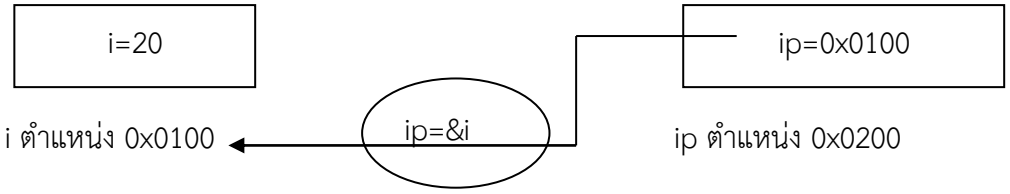
ตัวอย่าง

```
int   i=20;
int   *ip;
```

จากคำสั่งข้างต้นสามารถอธิบายได้ว่า หากตัวแปร i ได้รับการจัดสรรหน่วยความจำให้ที่ตำแหน่ง 0x0100 ตัวแปร ip ได้รับการจัดสรรหน่วยความจำที่ตำแหน่ง 0x0200 สามารถเขียนตำแหน่งของหน่วยความจำได้ดังภาพที่ 10.1

ภาพที่ 10.1 การประกาศตัวแปร i และ ip เริ่มต้น

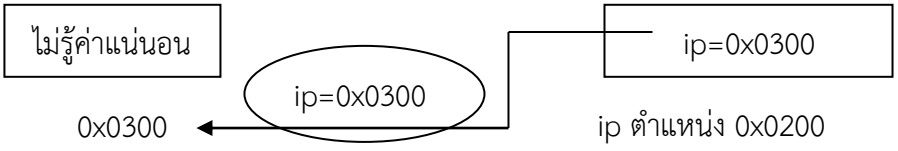
หากใช้คำสั่ง ip = &i; จะทำให้เกิดการเปลี่ยนแปลงดัง



ภาพที่ 10.2 ค่าที่เก็บในหน่วยความจำของตัวแปร i และ ip เมื่อมีการใช้คำสั่ง ip=&i

นั่นคือตัวแปร ip จะเก็บค่าตำแหน่งของหน่วยความของตัวแปร i เอาไว้

หรือหากต้องการสั่งให้ตัวแปร ip ชี้ไปยังตำแหน่งของหน่วยความจำที่ต้องการก็สามารถทำได้โดยการใช้คำสั่ง ip = ตำแหน่งหน่วยความจำ เช่น ต้องการให้ ip ชี้ไปที่ตำแหน่งของหน่วยความจำที่ 0x0300 ก็ใช้คำสั่ง ip=0x0300 ดังตัวอย่างที่แสดงใน



ภาพที่ 10.3 ค่าที่เก็บในหน่วยความจำของตัวแปร i และ ip เมื่อมีการใช้คำสั่ง ip=0x0300

แต่โดยมากแล้วกรณีหลังนี้ไม่นิยมใช้กันเพราะไม่สามารถรับประกันได้ว่าตำแหน่งที่กำหนดนั้นเก็บค่าอะไรอยู่ เป็นของตัวแปรใด

2. การเข้าถึงข้อมูลที่ชี้อยู่โดยตัวแปรตัวชี้

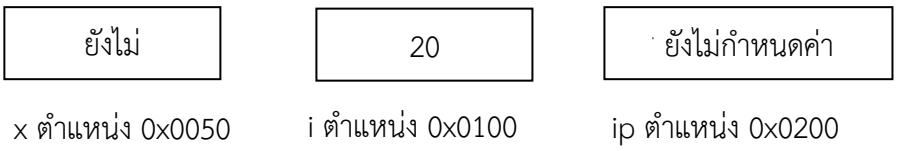
การเข้าถึงข้อมูลที่ชี้อยู่โดยตัวแปรตัวชี้ เพื่อเรียกใช้ค่าที่เก็บอยู่ในตำแหน่งของหน่วยความจำที่ชี้อยู่โดยตัวแปรตัวชี้ ใดๆ จะต้องทำการใส่เครื่องหมาย * (Indirector Operator) นำหน้าชื่อตัวแปรตัวชี้ นั้นๆ ดังรูปแบบต่อไปนี้

variable = *pointer;

แต่ไม่จำเป็นว่าต้องเอาค่ามาเก็บไว้ในตัวแปรเสมอไป อาจนำไปคำนวณ หรือทำอย่างอื่น ดังตัวอย่าง

```
int   i=20,x;
int   *ip;
```

จากคำสั่งข้างต้น หากตัวแปร i ได้รับการจัดสรรหน่วยความจำให้ที่ตำแหน่ง 0x0100 ตัวแปร ip ได้รับการจัดสรรหน่วยความจำที่ตำแหน่ง 0x0200 และตัวแปร x ได้รับการจัดสรรหน่วยความจำที่ตำแหน่ง 0x0050 สามารถเขียนตำแหน่งของหน่วยความจำได้ดังแสดงในภาพที่ 10.4

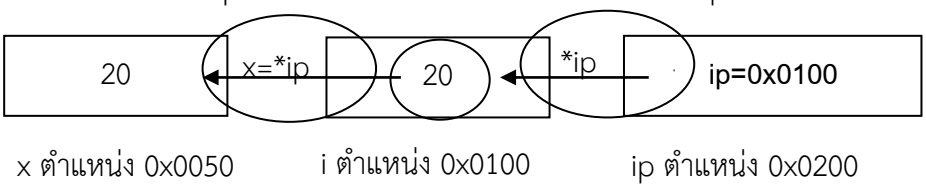


ภาพที่ 10.4 การประกาศตัวแปร i และ ip

หากกำหนดให้ตัวแปร ip เก็บค่าตำแหน่งหน่วยความจำของตัวแปร i โดยใช้คำสั่ง ip=&i จะได้ว่าตำแหน่งหน่วยความจำและค่าต่างๆ จะเป็นดังแสดงในภาพที่ 10.5

ภาพที่ 10.5 ค่าที่เก็บในหน่วยความจำของตัวแปร i,x และ ip เมื่อมีการใช้คำสั่ง ip=&i

จากนั้นหากกำหนดให้ตัวแปร x เก็บค่าเหมือนกับค่าที่อยู่ในตำแหน่งหน่วยความจำของตัวแปร i โดยใช้คำสั่ง x = *ipจะได้ว่าตำแหน่งหน่วยความจำและค่าต่างๆ ดังแสดงในภาพที่ 10.6



ภาพที่ 10.6 ค่าที่เก็บในหน่วยความจำของตัวแปร i, x และ ip เมื่อมีการใช้คำสั่ง x=*ip

ซึ่งการทำงานดังกล่าวนั้นสามารถเขียนโปรแกรมด้วยคำสั่งดังโปรแกรมที่ 10.1 ได้ดังนี้

```
1   #include <stdio.h>
2   #include <conio.h>
3   void main()
4   {
5           int i,x;
6           int *ip;
7           clrscr();
8           i=20;
9           printf ("Now i = %d\n",i);
10          ip= &i;
11          printf("Now ip = 0x%xh equal to &i=0x%xh (ip=&i)\n",ip,&i);
12          printf("But *ip is %d",i);
13  }
```

โปรแกรมที่ 10.1 การใช้งานตัวแปรตัวชี้ในลักษณะต่างๆ

ซึ่งเมื่อทดลองประมวลผลโปรแกรมแล้วจะได้ผลลัพธ์ดังนี้

```
Now i = 20
Now ip = 0xfff4h equal to &i=0xfff4h (ip=&i)
But *ip is 20
```

การส่งผ่านค่าระหว่างฟังก์ชัน

การส่งผ่านค่าในระหว่างหน่วยความจำในภาษาซี นั้นแบ่งออกได้เป็น 2 ลักษณะคือ

1. Pass by Value

Pass by Value เป็นวิธีการส่งผ่านค่าตัวแปรที่นักศึกษาได้เรียนไปเรียกว่าการส่งค่าโดย Pass by Value เนื่องจากเมื่อทำการส่งผ่านค่าไปแล้วตัวแปรระหว่างฟังก์ชันหลัก กับฟังก์ชันที่ถูกเรียกไปจะเป็นอิสระจากกัน ไม่สามารถใช้งานได้อีก ทำให้หากต้องส่งค่ากลับต้องส่งกลับโดยใช้คำสั่ง return() เท่านั้น ทำให้ในบางกรณีที่ต้องการส่งค่ากลับมากกว่า 1 ค่าไม่สามารถใช้งานได้ ต้องเลี่ยงไปใช้ตัวแปรส่วนกลาง (Global variable) แทน เช่นในการเขียนโปรแกรมค่าระหว่างตัวแปร 2 จำนวนดังโปรแกรมที่ 10.2

```
1  #include <stdio.h>
2  #include <conio.h>
3  void swap(int a,int b);
4  void main()
5  {
6      int a,b;
7      clrscr();
8      printf("Enter Value of a: "); scanf("%d",&a);
9      printf("Enter Value of b: "); scanf("%d",&b);
10     printf("  Before Call swap(). Value of a= %d, b=%d\n",a,b);
11     swap(a,b);
12     printf("  After Call swap().. Value of a= %d, b=%d\n",a,b);
13 }
14 void swap(int a,int b)
15 {
16    int c;
17    c= a;
18    a= b;
19    b= c;
20 }
```

โปรแกรมที่ 10.2 การใช้งานฟังก์ชันในการสลับค่า swap() โดยมีการประกาศตัวแปรท้องถิ่น

ในโปรแกรมที่ 10.2 นี้จะเห็นว่าฟังก์ชัน swap() จะทำการสลับค่าระหว่างตัวแปร a และ b ซึ่งเป็นตัวแปรเฉพาะที่ ซึ่งไม่สามารถทำให้การเปลี่ยนแปลงนี้ส่งผลถึงตัวแปร a และ b ใน main() ได้ ทำให้ในฟังก์ชัน main() ไม่สามารถสลับค่าตัวแปร a และ b ได้ตามต้องการเนื่องจากขอบเขตการใช้งานของตัวแปร

ดังนั้นจึงต้องเปลี่ยนโปรแกรมนี้ไปใช้วิธีการใช้ตัวแปรทั่วไปตามโปรแกรมที่ 10.3

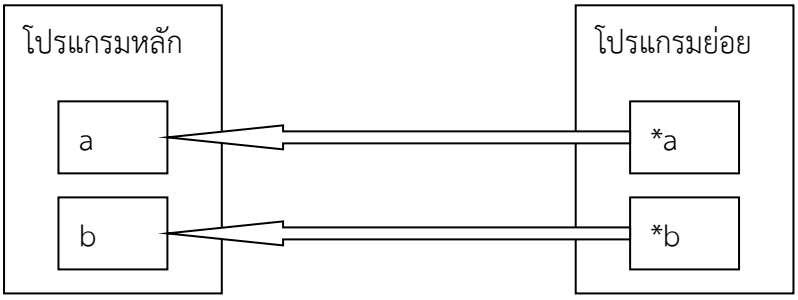
```
1   #include <stdio.h>
2   #include <conio.h>
3   int a,b;          <-------------  ตัวแปรทั่วไป a,b
4   void swap(void);
5   void main()
6   {
7       clrscr();
8       printf("Enter Value of a: "); scanf("%d",&a);
9       printf("Enter Value of b: "); scanf("%d",&b);
10      printf("Before Call swap(). Value of a= %d, b=%d\n",a,b);
11      swap();
12      printf("After Call swap().. Value of a= %d, b=%d\n",a,b);
13  }
14
15  void swap()
16  {
17     int c;
18     c= a;
19     a= b;
20     b= c;
21  }
```

โปรแกรมที่ 10.3 การใช้งาน Function ในการสลับค่า swap() โดยมีการประกาศตัวแปรทั่วไป

ในโปรแกรมที่ 10.3 นี้จะเป็นการแสดงตัวอย่างของการใช้งานตัวแปรส่วนกลาง (Global Variable) โดยจะทำการประกาศตัวแปร a,b เป็นชนิด int และให้เป็นที่รู้จักกันทั่วทั้งโปรแกรม (Global Variable เพราะมีการประกาศก่อนเข้าสู่ฟังก์ชันอื่นใด) ดังนั้นเนื่องจาก a,b เป็นตัวแปรทั่วไป การเปลี่ยนแปลงค่าของ a และ b ใน swap() จึงส่งผลต่อฟังก์ชัน main() ด้วย

2. Pass by Reference

ในกรณีนี้ต้องการส่งค่ากลับจากโปรแกรมย่อยไปยังโปรแกรมหลักมากกว่า 1 ค่า หรือต้องการให้มีการปรับเปลี่ยนค่าในฟังก์ชันย่อยแล้ว จะทำให้ค่าในฟังก์ชันหลักเปลี่ยนแปลงตามไปด้วย ซึ่งไม่สามารถทำได้ ดังนั้นอาจจำเป็นต้องการส่งค่ากลับโดยการใช้วิธีการส่งค่าอีกวิธีหนึ่งเรียกว่า วิธีการส่งค่าแบบ Pass by Reference ซึ่งหลักการของการส่งค่าแบบ Pass by Reference คือการใช้ ตัวแปรตัวชี้ (pointer) ดังแสดงในภาพที่ 10.7



ภาพที่ 10.7 การใช้ตัวชี้ในการชี้ตำแหน่งของตัวแปร

จากโปรแกรมที่ 10.3 อาจทำการแก้ไขให้ค่าของตัวแปร a และ b ในฟังก์ชัน main() ปรับเปลี่ยนค่าเป็นผลลัพธ์จากการสลับค่าได้โดยแก้โปรแกรมดังโปรแกรมที่ 10.4 ดังนี้

```
1   #include <stdio.h>
2   #include <conio.h>
3   void swap(int *x, int *y);
4   void main()
5   {
6       int a,b;
7       printf("Enter Value of a: ");  scanf("%d",&a);
8       printf("Enter Value of b: ");  scanf("%d",&b);
9       printf("  Before Call swap(). Value of a= %d, b=%d\n",a,b);
10      printf("Address of a is %p, Address of b is %p\n",&a,&b);
11      swap(&a,&b);
12      printf("In main() after calling swap() \n");
13      printf("  After Call swap().. Value of a= %d, b=%d\n",a,b);
14  }
15  void swap(int *x,int *y)
16  {
17     int c;
18     printf ("\n\n==== In swap() ====\n");
19     printf ("  x is point to %p, y is point to %p\n",x,y);
20     c= (*x);
21     (*x) = (*y);
22     (*y) = c;
23     printf ("==== End of swap() ====\n\n");
24  }
```

โปรแกรมที่ 10.4 การใช้งาน Function ในการสลับค่า swap() โดยการ Pass by Reference

ตัวอย่างที่ 10.1 การส่งค่าตัวแปรตัวชี้ผ่านฟังก์ชัน

```
1   #include <stdio.h>
2   #include <conio.h>
3   void  FunctionCall( int  *a , int  number) ;
4
5   void main()
6   {
7      int  x[5] = {1, 2, 3, 4, 5}  ;
8      clrscr() ;
9      FunctionCall (x , 5) ;
10  }
11  void  FunctionCall(int  *a , int  number)
12  {
13      int i ;
14      for ( i=0; i< number; i++)
15            printf (“a[%d] = %d \n” , i, a[i]) ;
16  }
```

โปรแกรมที่ 10.5 การส่งค่าตัวแปรตัวชี้ผ่านฟังก์ชัน

ผลลัพธ์

a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5

โปรแกรมที่ 10.6 แสดงให้เห็นวิธีการส่งค่าตัวแปรตัวชี้ผ่านฟังก์ชัน โดยสามารถส่งค่าตัวแปรตัวชี้ ผลลัพธ์แสดงค่าของตัวแปรแบบแถวลำดับที่กำหนดค่าจากฟังก์ชัน main() ส่งมาให้ฟังก์ชัน FunctionCall() แสดงค่าสมาชิกในแต่ละตัวของตัวแปรแบบแถวลำดับ

สรุป

ตัวแปรตัวชี้เป็นตัวแปรเพื่อใช้เป็นตัวชี้ในตำแหน่งหน่วยความจำ มีวิธีการประกาศตัวแปรโดยมีเครื่องหมาย * อยู่ข้างหน้าชื่อตัวแปรโดยตัวแปรตัวชี้สามารถใช้เป็นตัวชี้ตำแหน่งในหน่วยความจำได้ทั้งตัวเลขจำนวนเต็ม เลขทศนิยมและตัวอักษร โดยค่าตำแหน่งในหน่วยความจำที่เก็บในตัวแปรตัวชี้จะเก็บเป็นเลขฐานสิบหก ตัวแปรตัวชี้นิยมใช้เป็นพารามิเตอร์ในการส่งผ่านค่าระหว่างฟังก์ชันในกรณีที่ต้องการเข้าถึงข้อมูลในหน่วยความจำโดยตรง

แบบฝึกหัด

1. เขียนฟังก์ชันหาค่า factorial โดยกำหนดพารามิเตอร์เป็นแบบตัวแปรตัวชี้

   $n! = n(n-1)(n-2).....3 \cdot 2 \cdot 1 \quad ; \quad n \geq 0$ โดยกำหนดให้ 0! และ 1! =1

   โดยให้มีฟังก์ชันชื่อว่า factorial() และมีการทำงานลักษณะเดียวกับตัวอย่างที่ 10.1

2. เขียนโปรแกรมในการต่ออักษร 2 ชุดเข้าด้วยกัน และให้แสดงข้อความที่ต่อแล้วนั้นออกทางจอภาพ ดังตัวอย่างการทำงาน

   Enter astring: ***Introduction***

   Enter bstring: ***C Programming***

The new string is:"IntroductionC Programming

หมายเหตุ ข้อมูลที่พิมพ์ด้วย***ตัวหนาเอียงและขีดเส้น***ใต้เป็นข้อมูลที่ป้อนเข้าไป

3. เขียนฟังก์ชันหาผลบวกโดยส่งตัวตั้งและตัวบวกเป็นแบบตัวแปร Pointer
4. เขียนฟังก์ชันเพื่อคำนวณหาเส้นรอบวงของวงกลมโดยส่งค่าตำแหน่งในหน่วยความจำของตัวแปรรัศมี เป็นพารามิเตอร์
5. เขียนฟังก์ชันเพื่อการสลับค่าของตัวแปร 2 ตัว โดยส่งตำแหน่งในหน่วยความจำของตัวแปรมาเป็นพารามิเตอร์
6. เขียนฟังก์ชันเพื่อหาค่ามากที่สุด โดยส่งอาร์เรย์ของเลขจำนวนเต็มเป็นพารามิเตอร์แบบตัวแปร Pointer
7. เขียนฟังก์ชันเพื่อหาค้นหาตำแหน่งของค่าที่ต้องการค้นหา โดยส่งตำแหน่งในหน่วยความจำของอาร์เรย์เลขจำนวนเต็มและค่าที่ต้องการค้นหาเป็นพารามิเตอร์ หากไม่พบให้ส่งค่า 0 กลับฟังก์ชันหลัก
8. เขียนฟังก์ชันเพื่อแปลงอุณหภูมิจากองศาเซลเซียส เป็นองศาฟาเรนไฮท์ โดยส่งองศาเซลเซียส เป็นพารามิเตอร์ แบบตัวแปรตัวชี้ (pointer) โดยใช้สูตร F = 9 * C / 5 + 32
9. เขียนฟังก์ชันเพื่อหาผลลัพธ์ของเลข a^n โดยส่ง a, n เป็นพารามิเตอร์ แบบตัวแปรตัวชี้ (pointer)
10. เขียนฟังก์ชันเพื่อใส่, ให้กับเลขที่มากกว่า 1000 โดยส่งตำแหน่งในหน่วยความจำตัวเลขเป็นพารามิเตอร์ และส่งผลลัพธ์เป็นแบบตัวแปร Pointer

11. จากโปรแกรมข้างล่างนี้ให้นักศึกษาตอบคำถามต่อไปนี้

11.1 วาดรูปแสดงการชี้หน่วยความของตัวแปรตัวชี้ (pointer) a และ b

11.2 เขียนผลการประมวลผลโปรแกรมนี้ทั้งหมด พร้อมอธิบาย

11.3 โปรแกรมนี้ทำงานอะไร ให้อธิบายสั้นๆ ไม่เกิน 2 บรรทัด

11.4 เขียนคำสั่งภาษาซี ที่ใช้งานได้เทียบเท่ากับโปรแกรมนี้ โดยไม่ใช้ตัวแปรแบบตัวแปรตัวชี้ (pointer)

```
#include <stdio.h>
void main()
{
    int x=20,y=80,z;
    int *a,*b;
    clrscr();
    printf("BEFORE: x=%d y=%d\n",x,y);
    a= &x;     b= &y;
    z= (*a);   (*a) = y;   (*b) = z;
    printf("AFTER: x=%d y=%d\n",x,y);
    getch();
}
```

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: ดวงกมลสมัย

ประภาพร ช่างไม้.(2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร:อินโฟเพรส,

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for Engineering and Technology.* London: Prentice Hall.

บทที่ 11
ตัวแปรแบบโครงสร้าง

การเขียนโปรแกรมที่ผ่านมาเป็นการเขียนโปรแกรมเพื่อรับข้อมูล ตัวแปรแต่ละตัว โดยตัวแปรแต่ละตัวไม่มีการผูกโยงความสัมพันธ์กัน แต่ในการสร้างระบบงานจริง ๆ นั้น ข้อมูลที่ใช้นั้นบางข้อมูลจะมีความสัมพันธ์เกี่ยวข้องกันเป็นกลุ่ม ๆ เช่นข้อมูลนักศึกษา ประกอบด้วย รหัสนักศึกษา ชื่อนักศึกษา คณะ ชั้นปี และ อื่น ๆ อีกที่เกี่ยวกับนักศึกษาแต่ละคน ดังนั้น ตัวแปรที่ใช้เก็บข้อมูลประเภทนี้ ควรเป็นตัวแปรที่สามารถอ้างถึงข้อมูลของนักศึกษาในแต่ละคนได้ ในภาษาซี มีตัวแปรชนิดโครงสร้าง ที่สามารถรองรับการใช้งานแบบนี้ได้

ความหมายของตัวแปรแบบโครงสร้าง

ตัวแปรโครงสร้าง หมายถึง กลุ่มของตัวแปรที่มีความเกี่ยวข้องกันมาอยู่ด้วยกัน โดยตัวแปรภายในโครงสร้างถือว่าเป็นส่วนหนึ่งของโครงสร้าง หรือเป็นสมาชิกของโครงสร้าง โครงสร้างนี้ถือว่าเป็นชนิดของข้อมูลที่กำหนดเอง การนิยามโครงสร้างมักนำไปใช้เพื่อนิยามระเบียน (record) ระเบียนเป็นโครงสร้างข้อมูลประกอบด้วยเขตข้อมูล (field) ตั้งแต่หนึ่งเขตข้อมูลขึ้นไป หนึ่งเขตข้อมูลค่าใดๆ ที่แทนคุณสมบัติอันหนึ่งของสิ่งที่สนใจ เพื่อใช้ในการติดต่อกับแฟ้มหรือใช้กับงานด้านโครงสร้างข้อมูลต่อไป

ในการเขียนโปรแกรมคอมพิวเตอร์บางกรณีนั้นต้องการเก็บข้อมูลหลายประเภทเช่น ข้อมูลนักศึกษาประกอบด้วยข้อมูลดังต่อไปนี้

- รหัสนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชื่อนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชั้นปี เป็นชนิดตัวเลข
- เกรดเฉลี่ย เป็นชนิดตัวเลขทศนิยม

ซึ่งหากต้องการที่จะเก็บข้อมูลของนักศึกษาไว้ เห็นว่าต้องใช้ตัวแปรจำนวน 4 ตัวดังนี้

- `char ID[12];`
- `char Name[100];`
- `int year;`
- `float gpa;`

การประกาศโครงสร้างข้อมูล

การประกาศตัวแปรโครงสร้างสามารถทำได้โดยการใช้คำสั่ง struct ซึ่งมี 2 ขั้นตอนดังนี้

1. การนิยามโครงสร้างข้อมูล

การนิยามโครงสร้างข้อมูลนั้นมีรูปแบบดังนี้

```
รูปแบบ    :  struct  ชื่ออ้างอิง
                  {
                        ชนิดข้อมูล    ชื่อฟิลด์1;
                        ชนิดข้อมูล    ชื่อฟิลด์2;
                              ⋮              ⋮
                        ชนิดข้อมูล    ชื่อฟิลด์n;
                  };
หมายเหตุ: ขอให้สังเกตว่าในการนิยามโครงสร้างข้อมูลนั้น หลัง struct จะไม่มีเครื่องหมายอัฒภาค ;
          แต่ ; จะอยู่หลัง } ของคำสั่ง struct
```

2. การประกาศตัวแปรตามชนิดชนิดใหม่ที่สร้างตามโครงสร้างข้อมูล

การประกาศตัวแปรตามชนิดชนิดใหม่ที่สร้างตามโครงสร้างข้อมูล ขั้นตอนนี้มีลักษณะคล้ายการประกาศตัวแปรตามปกติที่ได้เรียนไปแล้วแต่เปลี่ยนจากชนิดข้อมูล int, char เป็นชนิดข้อมูลที่ประกาศ ในรูปแบบดังนี้

```
รูปแบบ    :  struct  ชื่ออ้างอิง  ชื่อตัวแปร;
```

ตัวอย่างที่ 11.1 ให้นักศึกษาประกาศตัวแปร struct สำหรับเก็บข้อมูลนักศึกษาซึ่งประกอบด้วยข้อมูลดังนี้

- รหัสนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชื่อนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชั้นปี เป็นชนิด integer
- เกรดเฉลี่ย เป็นชนิด float

วิธีคิด กำหนดโครงสร้างข้อมูลก่อนดังนี้

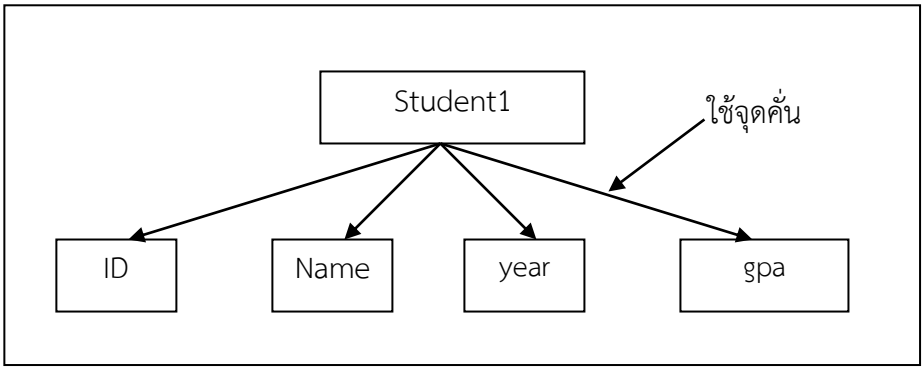
```
struct student {
        char ID[12];
        char Name[100];
        int year;
        float gpa;
};
```

จากนั้นประกาศตัวแปรโดยใช้คำสั่ง

```
struct student  student1;
```

เพื่อเป็นการประกาศตัวแปรชื่อว่า student1 ให้มีชนิดของข้อมูลเหมือนกับที่ประกาศไว้ใน struct student

ซึ่งจากการประกาศเช่นนี้ทำให้แสดงการใช้งานของตัวแปร student1 เป็นดังนี้



ภาพที่ 11.1 การเข้าถึงข้อมูลโครงสร้าง struct student

จากภาพที่ 11.1 หากต้องการหมายถึงข้อมูลต่างๆของนักศึกษาคนที่ 1 สามารถทำได้โดยการอ้างถึงข้อมูลดังนี้

student1.ID	หมายถึงรหัสนักศึกษาของนักศึกษาคนที่ 1
student1.Name	หมายถึงชื่อนักศึกษาของนักศึกษาคนที่ 1
student1.year	หมายถึงชั้นปีของนักศึกษาคนที่ 1
student1.gpa	หมายถึงเกรดเฉลี่ย ของนักศึกษาคนที่ 1

ตัวอย่างที่ 11.2 จากตัวอย่าง 11.1 ต้องการเก็บข้อมูลของนักศึกษา 10 คนสามารถเขียนโปรแกรมได้อย่างไร

- รหัสนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชื่อนักศึกษา เป็นตัวแปรแบบแถวลำดับของอักขระ
- ชั้นปี เป็นชนิด integer
- เกรดเฉลี่ย เป็นชนิด float

วิธีคิด กำหนดโครงสร้างข้อมูลตามโจทย์กำหนดก่อนดังนี้

```
struct student {
        char ID[12];
        char Name[100];
        int year;
        float gpa;
};
```

แต่โจทย์ให้ประกาศตัวแปรเพื่อเก็บข้อมูลนักศึกษาจำนวน 10 คนซึ่งต้องการให้แต่ละคนมีโครงสร้างเหมือน stuct student อาจทำได้โดยใช้คำสั่ง

```
struct student  student1;
struct student  student2;
      ⋮              ⋮
struct student  student10;
```

ซึ่งจะเกิดปัญหาว่าไม่สามารถใช้ประโยชน์ในการทำงานได้เต็มที่เพราะไม่สามารถใช้ กา รวนซ้ำเข้าช่วยได้ในการรับ ดังนั้นจึงควรประกาศเป็นอาร์เรย์ดังนี้

```
struct student  TenStudent[10];
```

ในที่นี้เปลี่ยนมาใช้ชื่อว่า TenStudent แทนเพื่อให้หมายถึงนักศึกษา 10 คน เนื่องจากคำว่า student ไม่สามารถใช้เป็นชื่อตัวแปรได้อีก เพราะเอาไปใช้เป็นชื่ออ้างอิงของรูปแบบของใหม่เสียแล้ว ซึ่งจากคำสั่งดังกล่าวเปรียบเสมือนมีพื้นที่ ในการเก็บข้อมูลนักศึกษา 10 คนตามโจทย์ต้องการแล้ว

สำหรับการเข้าถึงข้อมูลที่ประกาศไว้นั้นต้องปรับจากตัวอย่าง 11.1 เล็กน้อย เนื่องจากใช้ตัวแปรแถวลำดับ ดังนั้นจึงต้องบอกตำแหน่งดรรชนี (Index) ของตำแหน่งข้อมูลที่ต้องการด้วย เช่น

ตัวแปร	ความหมาย
TenStudent[0].ID	รหัสนักศึกษาของนักศึกษาในลำดับที่ 0
TenStudent[0].Name	ชื่อนักศึกษาของนักศึกษาในลำดับที่ 0
TenStudent[0].year	ชั้นปีของนักศึกษา ของนักศึกษาในลำดับที่ 0
TenStudent[0].gpa	เกรดเฉลี่ยของนักศึกษา ของนักศึกษาในลำดับที่ 0
TenStudent[3].ID	รหัสนักศึกษาของนักศึกษาในลำดับที่ 3
TenStudent[3].Name	ชื่อนักศึกษาของนักศึกษาในลำดับที่ 3
TenStudent[3].year	ชั้นปีของนักศึกษา ของนักศึกษาในลำดับที่ 3
TenStudent[3].gpa	เกรดเฉลี่ยของนักศึกษา ของนักศึกษาในลำดับที่ 3

การประกาศชื่อแฝงโครงสร้างข้อมูลโดยใช้ typedef

คำสั่ง typedef เป็นคำสั่งในการตั้งชื่อแฝงให้กับโครงสร้างข้อมูลที่สร้างขึ้นเพื่อให้การอ้างอิงเป็นไปได้สะดวกยิ่งขึ้น เพราะถ้าไม่ประกาศชื่อแฝงจะต้องอ้างอิงโดยใช้ struct ตามด้วยชื่ออ้างอิงเสมอ เช่น

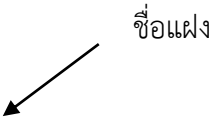
```
struct  student  student1  เป็นต้น
```

การประกาศชื่อแฝงนั้นมีรูปแบบดังนี้

```
รูปแบบ    :  typedef  struct ชื่ออ้างอิง  ชื่อแฝง;
```

เช่นจากตัวอย่างข้างต้นอาจประกาศชื่อแฝงได้ดังนี้

```
struct student {
        char ID[12];
        char Name[100];
        int year;
        float gpa;
};
typedef  struct student  StudentType;
```



ตัวอย่างที่ 11.3 ให้นักศึกษาเขียนโปรแกรมรับข้อมูลนักศึกษา 1 คนซึ่งประกอบด้วยข้อมูลดังนี้

- char ID[12];
- char name[50];
- int midterm;
- int project;
- int final;
- int total;
- char grade;

วิธีคิด

ข้อนี้เหมือนโจทย์ตัวอย่างที่กล่าวมา เพียงแต่เปลี่ยนโครงสร้างข้อมูลเท่านั้น ซึ่งในข้อนี้ประกาศตัวแปรตามที่โจทย์กำหนดให้ได้เลย ดังนี้

```
struct student {
   char ID[12];
   char name[50];
   int  midterm;
   int project;
   int final;
   int total;
   char grade;
};
typedef   struct student  StudentType;
```

จากนั้นเขียนโปรแกรมรับข้อมูลโดยใช้ฟังก์ชัน printf() ในการแสดงข้อมูล และรับข้อมูลโดยใช้ฟังก์ชัน scanf() รับข้อมูลตามปกติ ส่วนรูปแบบในการอ่านค่านั้นก็ดูตามรูปแบบต้นฉบับของข้อมูลเดิมเป็นหลัก เช่น

char ID[12];	รับข้อมูลโดยgets เพราะเป็นตัวแปรแบบแถวลำดับของอักขระ
char name[50];	รับข้อมูลโดย gets() เพราะเป็นตัวแปรแถวลำดับของอักขระ
int midterm;	รับข้อมูลโดย scanf(“%d”) เพราะเป็น int
int project;	รับข้อมูลโดย scanf(“%d”) เพราะเป็น int
int final;	รับข้อมูลโดย scanf(“%d”) เพราะเป็น int
int total;	รับข้อมูลโดย scanf(“%d”) เพราะเป็น int
char grade;	รับข้อมูลโดย scanf(“%c”) เพราะเป็น char

ดังนั้นจะเขียนโปรแกรมได้ดังโปรแกรมที่ 11.1

```
1   #include <stdio.h>
2   struct student {
3       char ID[12];
4       char name[50];
5       int  midterm;
6       int project;
7       int final;
8       int total;
9       char grade;
10  };
11  typedef struct student  StudentType;
12  void main()
13  {
14      StudentType std;
15      int t,check;
16      clrscr();
17      printf("Enter your ID: ");
18      gets(std.ID);
19      printf("Enter your name: ");
20      gets(std.name);
21      printf("Enter midterm score: ");
22      scanf("%d",&std.midterm);
23      printf("Enter project score: ");
24      scanf("%d",&std.project);
25      printf("Enter final score  :");
26      scanf("%d",&std.final);
27      t= std.midterm + std.project + std.final;
28      std.total=t;
29      printf("Total Score = %d ",t);
30      getch();
31  }
```

โปรแกรมที่ 11.1 ตัวอย่างการใช้งาน struct ตามตัวอย่าง 11.3

สรุป

ตัวแปรแบบโครงสร้างใช้ในการเก็บข้อมูลที่มีความเกี่ยวข้องกัน เช่น ข้อมูลนักศึกษา ข้อมูลสินค้า ข้อมูลพนักงาน ตัวแปรแบบโครงสร้างประกอบด้วยตัวแปรชนิดต่าง ๆ เพื่อเก็บรายละเอียดของข้อมูล โดยมีรูปแบบประกาศตัวแปรแบบโครงสร้าง ใช้คำสั่ง struct โดยการประกาศตัวแปรแบบโครงสร้าง ไม่ได้ทำให้ใช้หน่วยความจำเพิ่มขึ้นจากการประกาศตัวแปรแบบปกติ และหากข้อมูลมีหลายจำนวน หลายชุด สามารถกำหนดเป็นตัวแปรแบบแถวลำดับได้ ซึ่งสามารถรับข้อมูลนักศึกษาจำนวนหลาย ๆ คนได้ ซึ่งการใช้ตัวแปรแบบปกติมีปัญหาในการอ้างถึงข้อมูลของแต่ละคน การรับข้อมูลและการแสดงผลข้อมูล สามารถใช้ได้กับฟังก์ชันที่ใช้กับตัวแปรแบบปกติ และสามารถใช้คำสั่ง typedef เพื่อประกาศชื่อแฝงโครงสร้างข้อมูล ในการอ้างอิงต่อไป สามารถใช้ร่วมกันกับข้อมูลที่เกี่ยวข้องกันได้

แบบฝึกหัด

1. ให้นักศึกษาแก้ไขโปรแกรม 11.1 ให้สามารถตรวจสอบความผิดพลาดของการป้อนคะแนนได้ โดยกำหนดให้
   - คะแนนกลางภาคมีคะแนนเต็ม 30 คะแนน
   - คะแนนโครงงานมีคะแนนเต็ม 30 คะแนน
   - คะแนนปลายภาคมีคะแนนเต็ม 40 คะแนน
2. ให้นักศึกษาแก้ไขโปรแกรม 11.1 ให้สามารถรองรับข้อมูลของนักศึกษาได้ 10 คนและต้องตรวจสอบความผิดพลาดของการป้อนคะแนนได้
3. ให้ประกาศตัวแปรแบบโครงสร้างของข้อมูลพนักงาน ซึ่งประกอบด้วยชื่อ ที่อยู่ เงินเดือน และ อายุ
4. ให้เขียนโปรแกรมเพื่อรับข้อมูลพนักงาน
5. ให้เขียนโปรแกรมเพื่อแสดงข้อมูลพนักงาน
6. ให้เขียนโปรแกรมเพื่อรับข้อมูลพนักงานจำนวน 10 คน แล้วแสดงข้อมูลดังกล่าว
7. ให้ประกาศตัวแปรแบบโครงสร้างของข้อมูลสินค้า ซึ่งประกอบด้วยรหัสสินค้า ชื่อสินค้า ราคาต่อหน่วย จำนวนสินค้า
8. ให้เขียนโปรแกรมเพื่อรับข้อมูลสินค้า
9. ให้เขียนโปรแกรมเพื่อแสดงข้อมูลสินค้า
10. ให้เขียนโปรแกรมเพื่อรับข้อมูลสินค้าจำนวน 10 รายการ แล้วแสดงข้อมูลดังกล่าว

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: ดวงกมลสมัย

ประภาพร ช่างไม้. (2545). *หลักการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร:อินโฟเพรส

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for engineering and technology.* London: Prentice Hall.

บทที่ 12
แฟ้ม

การทำงานโดยส่วนใหญ่ จำเป็นต้องมีการเก็บข้อมูลเดิมไว้ เพื่อไม่ต้องคีย์ข้อมูลเข้าไปใหม่ทุกครั้งที่มีการทำงาน หรือมีข้อมูลเดิมเพื่อตรวจสอบข้อมูลบางอย่าง ดังนั้นระบบงานส่วนใหญ่ไม่ว่าจะเป็นระบบเล็กหรือระบบใหญ่ การเก็บข้อมูลมีความสำคัญมาก ดังนั้นในภาษาซี ระบบจัดการแฟ้มและมีฟังก์ชันที่เกี่ยวข้องกับแฟ้ม (File) มากมาย ไม่ว่าจะเป็นการเปิดแฟ้มหรือปิดแฟ้ม การอ่านและเขียนข้อมูลลงแฟ้ม ในบทนี้จะอธิบายถึงแฟ้ม บัฟเฟอร์ ตัวชี้แฟ้ม ประเภทของแฟ้ม การเปิดแฟ้ม การปิดแฟ้ม การอ่านอักขระจากแฟ้ม การเขียนอักขระลงแฟ้ม การอ่านข้อมูลแบบมีรูปแบบจากแฟ้ม การเขียนข้อมูลแบบมีรูปแบบลงแฟ้ม

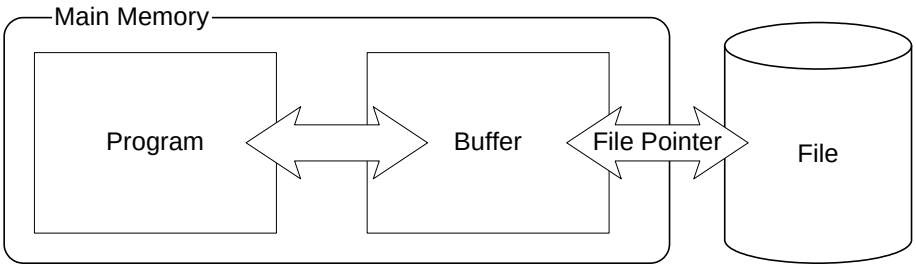
แฟ้ม

แฟ้มเป็นกลุ่มของข้อมูลที่เก็บอยู่บนสื่อบันทึก เช่น จานบันทึกแบบแข็ง (Hard disk) เป็นต้น สำหรับหน่วยการเก็บภายในจะใช้รหัส 0 และ 1 ซึ่งถือว่าเป็นหน่วยที่เล็กที่สุด แต่ละค่าเรียกว่า “บิต (Bit)” กลุ่มของบิตเรียกว่า “ไบต์ (Byte)” ซึ่งประกอบด้วย 8 บิต หน่วยที่ใหญ่ขึ้นมาอีกคือ “เขตข้อมูล (Field)” ซึ่งใช้เก็บข้อมูลชนิดใดชนิดหนึ่ง เช่น รหัสนักศึกษา ชื่อนักศึกษา เป็นต้น หากนำหลายๆ เขตข้อมูลมาประกอบกันเรียกว่า “ระเบียน (Record)” เช่นข้อมูลของนักศึกษา 1 คน เรียกว่า เป็น 1 ระเบียน

หลายๆ ระเบียนรวมกันเป็น 1 แฟ้ม ซึ่งสามารถเก็บข้อมูลของนักศึกษาได้หลายๆ คน สิ้นสุดการบันทึกข้อมูลโดยมีสัญลักษณ์สิ้นสุดแฟ้ม (End of File) ซึ่งเป็นอักขระพิเศษในการบ่งชี้ถึงการสิ้นสุดของแฟ้ม

บัฟเฟอร์และตัวชี้แฟ้ม

บัฟเฟอร์ (Buffer) จะทำหน้าที่เป็นที่พักถ่ายเทข้อมูลระหว่างโปรแกรมกับแฟ้มที่อยู่บนสื่อบันทึก เนื่องจากความเร็วของการทำงานของสื่อบันทึกและหน่วยความจำหลัก และ ซีพียู นั้นต่างกันมาก ดังภาพที่ 12.1



ภาพที่ 12.1 การทำงานของโปรแกรมและแฟ้ม

จากภาพที่ 12.1 จะเห็นว่าโปรแกรมจะทำการติดต่อกับกับแฟ้ม (file) โดยผ่านทางระบบปฏิบัติการ ซึ่งจะต้องสร้างตัวชี้ไปตำแหน่งทางกายภาพของแฟ้มนั้นๆ ผ่านทางสิ่งที่เรียกว่า “ตัว

ชี้แฟ้ม (File Pointer)” ดังนั้นขั้นตอนแรกในการทำงานกับแฟ้มคือการสร้างตัวชี้แฟ้ม (File Pointer) สำหรับในภาษาซี นั้นมีรูปแบบดังนี้

รูปแบบ : FILE *ชื่อตัวแปร; หมายเหตุ FILE เป็นชนิดตัวแปรเฉพาะสำหรับสร้างแฟ้ม (file) ตัวแปรตัวชี้ (pointer) ต้องเป็นพิมพ์ใหญ่ทั้งหมดและต้องเป็น FILE * เสมอ

ประเภทแฟ้มจำแนกตามลักษณะการเข้าถึง

ประเภทของแฟ้มจำแนกตามลักษณะการเข้าถึงของภาษาซีนั้นออกแบ่งเป็น 2 ประเภท คือ

1. แฟ้มแบบลำดับ

แฟ้มแบบลำดับ (Sequential File) เป็นแฟ้มที่มีลักษณะการเข้าถึงแบบลำดับ ทีละไบต์ หรือทีละเรคคอร์ด ตามลำดับ เรคคอร์ดที่ 1 – 2 – 3 –4 –5 ตามลำดับไม่สามารถกระโดดข้ามได้

แฟ้มแบบนี้แต่ละเรคคอร์ดจะต้องคั่นด้วยเครื่องหมายในการแบ่งเรคคอร์ดเสมอ ส่วนมากจะใช้อักษรที่เรียกว่า Carriage Return (CR) และ Line Feed (LF) ควบคู่กันเป็น CRLF (Carriage Return Line Feed)

การเข้าถึงแฟ้มแบบนี้ในภาษาซี โดยการใช้คำสั่ง fgetc(), fputc(), fgets(), fputs(), fprintf() ซึ่งจะได้กล่าวต่อไป

2. แฟ้มเข้าถึงแบบสุ่ม

แฟ้มเข้าถึงแบบสุ่ม (Random Access File) เป็นแฟ้มที่มีลักษณะการเข้าถึงข้อมูลแบบสุ่ม กล่าวคือถ้าจะอ่านคือเขียนข้อมูลที่เรคคอร์ดใดๆ ก็สามารถเข้าถึงเรคคอร์ดนั้นๆ ได้ทันที ไม่ต้องเริ่มจากเรคคอร์ดแรกเสมอเหมือนแฟ้มแบบลำดับ โดยการอ้างถึงโดยใช้เลขที่อยู่ทางตรรกะ (Logical Address) ในการคำนวณหาตำแหน่งที่อยู่ของเรคคอร์ดใดๆ ดังนี้

เลขที่อยู่ของเรคคอร์ดที่ n = เลขที่อยู่ของเรคคอร์ดแรก + ขนาดของแต่ละเรคคอร์ด * (n-1)

จากวิธีการคำนวณข้างต้นทำให้ได้ข้อสรุปว่าการจะคำนวณแบบนี้ได้นั้นจะต้องมีขนาดของทุกๆ เรคคอร์ดที่เท่ากันทั้งหมดจึงจะสามารถคำนวณได้

ซึ่งวิธีการทำงานกับแฟ้มประเภทนี้นักศึกษาจะได้เรียนในรายวิชาต่อไป

การเปิดแฟ้ม

เมื่อทำการสร้างตัวชี้แฟ้ม (File Pointer) แล้วต่อไปจะต้องทำการเปิดการติดต่อระหว่างแฟ้มใดในดิสก์ที่จะให้ตัวชี้แฟ้มอ้างอิงถึง โดยใช้ฟังก์ชัน fopen() ซึ่งมีรูปแบบดังนี้

รูปแบบ : ตัวแปร File ตัวแปรตัวชี้ (pointer) = fopen(“ชื่อแฟ้ม”, “mode”);

ซึ่ง mode เป็นการบ่งบอกถึงวิธีการในการเข้าถึงแฟ้ม ซึ่งมี mode ที่ใช้มากดังนี้

- “r” เป็น การเปิดอ่านข้อมูลจากแฟ้ม ถ้าไม่มีแฟ้มอยู่จะถือว่าเปิดไม่สำเร็จ
- “w” เป็นการเปิดแฟ้มเพื่อเขียนข้อมูลใหม่ ถ้ามีแฟ้มเดิมอยู่แล้วจะลบข้อมูลทิ้งทั้งหมด ถ้าไม่มีแฟ้มอยู่ก่อนก็จะสร้างแฟ้มใหม่ขึ้นมา
- “a” เป็นการเปิดแฟ้มเพื่อเขียน ถ้าแฟ้มมีอยู่แล้วจะไปต่อท้ายแฟ้มเสมอ แต่ถ้ายังไม่มีแฟ้มก็จะสร้างแฟ้มขึ้นใหม่

การปิดแฟ้ม

การปิดแฟ้มเป็นการให้โปรแกรมบอกระบบปฏิบัติการว่าการติดต่อกับแฟ้มนั้นสิ้นสุดแล้ว ให้ระบบปฏิบัติการทำงานเขียนข้อมูลทั้งหมดที่มีอยู่ในบัฟเฟอร์ไปสู่จานบันทึกได้เลย และจากนั้นให้สิ้นสุดการทำงานกับแฟ้มนั้น

การปิดแฟ้มในภาษาซี ทำได้โดยการใช้คำสั่ง fclose() มีรูปแบบการใช้งานดังนี้

รูปแบบ : fclose(ชื่อ File Pointer);

ตัวอย่างที่ 12.1 ให้นักศึกษาเปิดแฟ้มชื่อ student.txt ใน Drive C เพื่อทำการอ่านข้อมูล

วิธีคิด

โจทย์บอกให้เปิดแฟ้มชื่อ student.txt ใน Drive C ดังนั้นชื่อ File ที่ถูกต้องจึงเป็น C:\student.txt ตามบทที่ 1 และทำการอ่านข้อมูลต้องใช้ mode เป็น “r”

ดังนั้นการเปิดแฟ้มจึงเป็นดังนี้

```
FILE *fp;
fp = fopen(“C:\\student.txt”, “r”);
```

FILE *fp; เพื่อสร้างตัวแปร File Pointer

fp = fopen(“C:\\student.txt”, “r”); เปิดแฟ้ม “student.txt” เพื่อการอ่าน

ซึ่งหากเปิดได้สำเร็จตัวแปร fp (ซึ่งเป็น File Pointer) จะต้องมีค่าดังนี้

- NULL แสดงว่าการเปิดแฟ้มไม่สำเร็จ
- ไม่ใช่ NULL แสดงว่าแฟ้มเปิดได้สำเร็จ

ดังนั้นหากต้องการที่จะตรวจสอบความสำเร็จของการเปิดไฟล์ก็สามารถทำได้ดังโปรแกรมที่ 12.1

```
1  #include <stdio.h>
2  void main()
3  {
4          FILE *fp;
5          fp = fopen(“C:\\student.txt”, “r”);
6          if (fp==NULL) {
7                  printf(“Open File FAILED!!!”);
8                  exit(1);
9          }
10         else
11         {
12                 printf(“File Open OK.”);
13                 fclose(fp);
14         }
15 }
```

โปรแกรมที่ 12.1 ตัวอย่างการใช้งานเปิดและปิดแฟ้ม

จะเห็นว่าในการระบุชื่อแฟ้มนั้นต้องใช้เครื่องหมาย \ เพื่อแยกระหว่างชื่อโฟล์เดอร์ และชื่อแฟ้มในที่นี้ C:\ หมายถึงโฟล์เดอร์หลักของ Drive C แต่ตัวแปลภาษาซี นั้นจะมองเครื่องหมาย \ เป็นเครื่องหมายพิเศษ ดังนั้นจึงต้องใช้เครื่องหมาย \\ แทนที่จะใช้ \ หรือใช้ C:/student.txt แทนความหมายเช่นเดียวกัน

การอ่านและเขียนข้อมูลครั้งละ 1 อักขระ

การอ่านและเขียนข้อมูลครั้งละ 1 อักขระนั้นสามารถทำได้โดยการใช้คำสั่ง fgetc() และ fput() ซึ่งทำหน้าที่ในการอ่านและเขียนข้อมูลตามลำดับ ซึ่งมีรูปแบบการใช้งานดังนี้

1. การใช้งาน fgetc()

รูปแบบ	:	ตัวแปร = fgetc(ชื่อ File Pointer)
Header file	:	stdio.h
หน้าที่	:	อ่านข้อมูลครั้งละ 1 ตัวอักษรในตำแหน่งทีตัวแปรตัวชี้แฟ้มชี้อยู่ และเลื่อนตัวแปรตัวชี้แฟ้มไปยังตำแหน่งถัดไปอีก 1 ไบต์

2. การใช้งาน fputc()

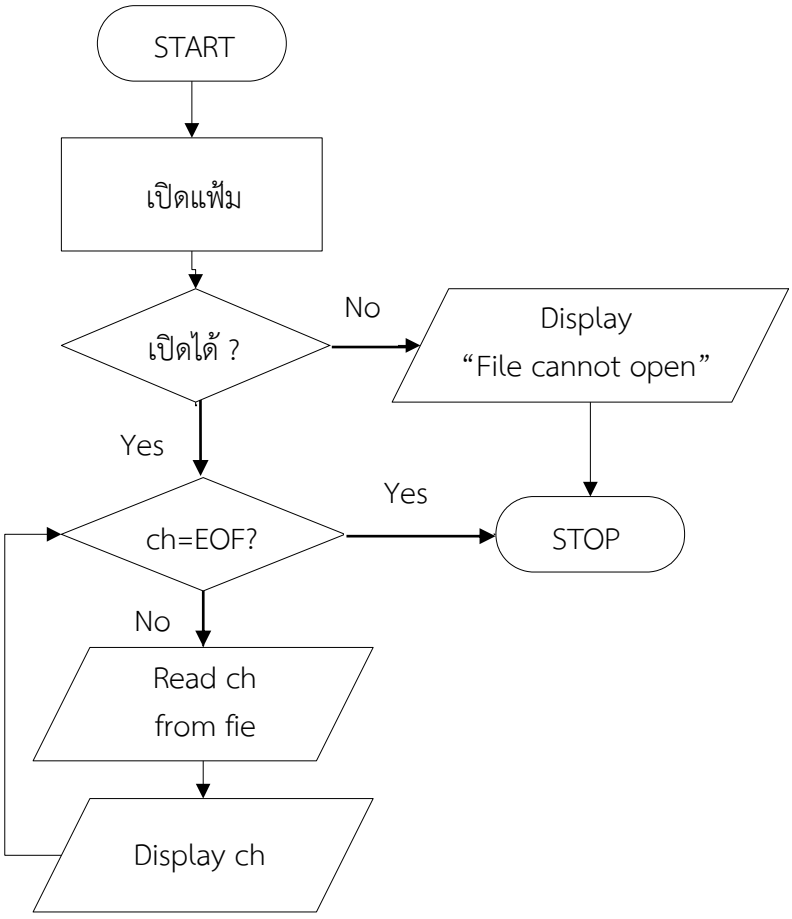
รูปแบบ	:	fgetc (ตัวอักษรที่ต้องการบันทึกลงแฟ้ม, ชื่อ File Pointer)
Header file	:	stdio.h
หน้าที่	:	บันทึกข้อมูลครั้งละ 1 ตัวอักษรในตำแหน่งทีตัวแปรตัวชี้แฟ้มชี้อยู่ และเลื่อนตัวแปรตัวชี้แฟ้มไปยังตำแหน่งถัดไปอีก 1 ไบต์

จากรูปแบบการใช้งานของฟังก์ชันทั้งสองนั้นจะเห็นว่าเมื่อทำงานเสร็จแล้วจะเลื่อนตัวแปรตัวชี้แฟ้มไปยังตำแหน่งถัดไปทั้งสองฟังก์ชัน

ตัวอย่างที่ 12.2 ให้นักศึกษาเปิดแฟ้มชื่อ student.txt ใน Drive C ซึ่งมีอยู่แล้วเพื่อทำการอ่านข้อมูลครั้งละ 1 ไบต์ และแสดงข้อมูลนักศึกษาออกทางจอภาพ

วิธีคิด จะต้องทำการวิเคราะห์ปัญหาเสียก่อนดังนี้

1. วิเคราะห์ผลลัพธ์ ผลลัพธ์ที่ต้องการคือการแสดงข้อมูลออกทางจอภาพ
2. วิเคราะห์ที่มาของข้อมูล โจทย์บอกว่าให้อ่านข้อมูลจากแฟ้ม C:\student.txt ซึ่งมีอยู่แล้วดังนั้นจึงต้องเปิดแฟ้มใน mode “r”
3. วิเคราะห์ความสัมพันธ์ของข้อมูล เนื่องจากเป็นการอ่านข้อมูลจากแฟ้มจึงถือว่าเป็นข้อมูลชนิดอักขระทั้งหมดไม่ว่าจะเป็นเลข 0 ก็ต้องเป็นอักขระเพราะเป็นข้อกำหนดเฉพาะของการอ่านไฟล์ครั้งละ 1 อักขระ สามารถเขียนผังงานได้ดังนี้



ภาพที่ 12.2 การอ่านแฟ้มครั้งละ 1 ไบต์

ซึ่งสามารถเขียนโปรแกรมได้ดังโปรแกรมที่ 12.2

```
1   #include <stdio.h>
2   void main()
3   {
4       FILE *fp;
5       char ch;
6       /* 1. Open file */
7       fp=fopen(“C:\\student.txt”,"r");
8       if (fp==NULL)
9       {
10        printf(“Cannot open file C:\\student.txt”);
11        exit(1);
12      }
13      else
14      {
15            do {
16                    ch=fgetc(fp);
17                    putchar(ch);
18                }
19       }
20  }
```

โปรแกรมที่ 12.2 การเปิดอ่านแฟ้มครั้งละ 1 ไบต์

การอ่านและเขียนข้อมูลแบบมีรูปแบบ

การอ่านและเขียนข้อมูลที่ผ่านมานั้นไม่สามารถกำหนดรูปแบบการอ่านหรือเขียนได้ เพราะต้องอ่านเป็น char ตลอด ดังนั้นหากต้องการอ่านค่าตัวเลขจะทำให้เกิดความไม่สะดวก จึงมีฟังก์ชันสำหรับการอ่านและเขียนข้อมูลอย่างมีรูปแบบโดยฟังก์ชัน fprintf() และ fscanf() ดังรูปแบบต่อไปนี้

1. ฟังก์ชัน fprintf()

รูปแบบ : fprintf(ตัวแปรตัวชี้แฟ้ม,“รูปแบบการบันทึก”, ชื่อตัวแปรที่เก็บข้อมูล)
Header File : stdio.h

2. ฟังก์ชัน fscanf()

รูปแบบ : fscanf(ตัวแปรตัวชี้แฟ้ม,“รูปแบบการอ่าน”, &ชื่อตัวแปรที่จะเก็บข้อมูล)
Header File : stdio.h

จะเห็นได้ว่าฟังก์ชัน fprintf() และ fscanf() นั้นมีลักษณะคล้ายกับ printf() และ scanf() ที่ได้เรียนแล้วมาก เพียงแต่เพิ่มส่วนของ File ตัวแปรตัวชี้ (pointer) เข้าไปเท่านั้น สำหรับฟังก์ชันอย่าลืมว่าต้องมีเครื่องหมาย & หน้าชื่อตัวแปรด้วย

ตัวอย่างที่ 12.3 ให้นักศึกษาเปิดแฟ้มชื่อ student.txt ใน Drive C ซึ่งมีอยู่แล้วเพื่อทำการบันทึกข้อมูลนักศึกษาซึ่งมีข้อมูล รหัสนักศึกษา คะแนนกลางภาค และคะแนนปลายภาคโดยโปรแกรมที่เขียนจะต้องสามารถรับข้อมูลนักศึกษาจำนวน 3 คน

วิธีคิด ใช้หลักการประกาศ struct เพื่อเก็บข้อมูลนักศึกษา และใช้ Array ช่วยเพื่อให้เก็บข้อมูลนักศึกษาได้ 3 คนจากนั้นจึงบันทึกข้อมูลดังกล่าวลงสู่แฟ้ม ดังตัวอย่างในโปรแกรมที่ 12.3

```
1  #include <stdio.h>
2  struct student {
3      int ID;
4      int midterm;
5      int final;
6  };
7  void main()
8  {
9     FILE *fp;
10    struct student std[3];
11    int i;
12    fp=fopen("C:\\student.txt","w");
13    if (fp==NULL)
14    {
15        printf("File Open FAILED!!!!.");
16    }
17    else
18        printf("File Open OK.");
19     for (i=0;i<5;i++)
20    {
21        printf("Enter Student #%d\n",i);
22        printf("   ID: ");              scanf("%d",&std[i].ID);
23        printf("   midterm: ");    scanf("%d",&std[i].midterm);
24        printf("   final: ");          scanf("%d",&std[i].final);
25        fprintf(fp,"%d %d %d\n",std[i].ID,std[i].midterm,std[i].final);
26    }
27    fclose(fp);
28 }
```

โปรแกรมที่ 12.3 การเขียนแฟ้มอย่างมีรูปแบบ

ตัวอย่างที่ 12.4 การอ่านสายอักขระจากแฟ้ม

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4
5   void main()
6   {
7      FILE *fp;
8      char  str1[10] , str2[20] , str3[30] ;
9      fp=fopen("C:\\student.txt","r");
10     if (fp==NULL)
11     {
12        printf( “File cannot open”);
13     }
14     else {
15        fscanf(fp , “%s%s%s” , str1,str2,str3) ;
16        printf(“%s\n”,str1);
17        printf(“%s\n”,str2);
18        printf(“%s\n”,str3);
19        fclose(fp);
20     }
21  }
```

โปรแกรมที่ 12.4 การอ่านสายอักขระจากแฟ้ม

ตัวอย่างที่ 12.5 การอ่านค่าตัวเลขจากแฟ้ม

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4
5   void main()
6   {
7      FILE  *fp;
8      int  day, month, year ;
9      fp=fopen("C:\\birthday.txt", "r");
10     if (fp==NULL)
11     {
12        printf( “File cannot open”);
13     }
14     else {
15        fscanf(fp , “%d%d%d” , &day , &month , &year) ;
16        printf(“%d\n”,day);
17        printf(“%d\n”,month);
18        printf(“%d\n”,year);
19        fclose(fp);
20     }
21     getch();
22  }
```

โปรแกรมที่ 12.5 การอ่านค่าตัวเลขจากแฟ้ม

ตัวอย่างที่ 12.6 การคัดลอกไฟล์

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   void main()
5   {
6      FILE  *fin, *fout;
7      char   ch , fileIn[30] , fileOut[30] ;
8      clrscr();
9      printf (“Enter source filename : ”) ;
10     gets(fileIn) ;
11     fin=fopen(fileIn, "r");
12     if (fin==NULL)
13     {
14        printf(“File cannot open for reading. ”) ;
15        exit(0) ;
16     }
17     printf (“Enter target filename : ”) ;
18     gets(fileIn) ;
19
20     fout=fopen(fileIn, "w");
21     if (fout==NULL)
22     {
23        printf(“File cannot open for writing. ”) ;
24        exit(0) ;
25     }
26     while ( !feof ( fin ) ) {
27       ch = fgetc ( fin ) ;
28       fputc (ch , fout) ;
29     }
30      fclose(fin);
31      fclose(fout);
32  }
```

โปรแกรมที่ 12.6 การคัดลอกไฟล์

ตัวอย่างที่ 12.7 การใช้ฟังก์ชัน fprintf ()

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   void  main()
5   {
6      FILE  *fp ;
7      char   name[30] ;
8      float   salary ;
9      int      i , n ;
10     clrscr() ;
11     fp=fopen("data.txt", "w") ;
12     if (fp==NULL)
13     {
14        printf("File cannot open for writing  " ) ;
15        exit(0) ;
16     }
17     printf ("Enter number of record  : ") ;
18     scanf (“%d”,&n) ;
19     for (i = 1; i<=n ; i++)  {
20        printf (“Enter employee name : ”) ;
21        scanf (“%s”, name) ;
22        printf (“Enter salary : ”) ;
23        scanf (“%f”, &salary) ;
24        fprintf (fp, “%s %.2f \n” , name , salary ) ;
25     }
26     fclose(fp) ;
27  }
```

โปรแกรมที่ 12.7 การใช้ฟังก์ชัน fprintf ()

ตัวอย่างที่ 12.8 การใช้ฟังก์ชัน fscanf ()

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   void  main()
5   {
6      FILE  *fp ;
7      char   name[30] ;
8      float   salary ;
9      int      i , n ;
10     clrscr() ;
11     fp=fopen("data.txt", "r") ;
12     if (fp==NULL)
13     {
14        printf("File cannot open for reading  " ) ;
15        exit(0) ;
16     }
17     while ( fscanf (fp , “%s %f” , name , &salary) != EOF ) {
18        printf (“Name : %s \n” , name) ;
19        printf (“Salary : %.2f \n” , salary) ;
20     }
21     fclose(fp) ;
22  }
```

โปรแกรมที่ 12.8 การใช้ฟังก์ชัน fscanf ()

ตัวอย่างที่ 12.9 การใช้ฟังก์ชัน fwrite ()

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   void  main()
5   {
6      FILE  *fp ;
7      struct  employee {
8         char  name[30] ;
9         float  salary ;
10     } personal ;
11     int  i , n ;
12     clrscr() ;
13      if  ( fp=fopen("data.txt", "w") == NULL)
14     {
15         printf("File cannot open for writing " ) ;
16         exit(0) ;
17     }
18     printf ("Enter number of record  : ") ;
19     scanf (“%d”,&n) ;
20     for (i = 1; i<=n ; i++)  {
21        printf (“Enter employee name : ”) ;
22        scanf (“%s”, personal.name) ;
23        printf (“Enter salary : ”) ;
24        scanf (“%f”, &personal.salary) ;
25        fwrite (&personal , sizeof(personal) , 1, fp) ;
26     }
27     fclose(fp) ;
28  }
```

โปรแกรมที่ 12.9 การใช้ฟังก์ชัน fwrite ()

ตัวอย่างที่ 12.10 การใช้ฟังก์ชัน fread ()

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   void  main()
5   {
6     FILE  *fp ;
7     struct  employee {
8        char  name[30] ;
9        float  salary ;
10    } personal ;
11    int  i , n ;
12    clrscr() ;
13     if  ( fp=fopen("data.txt", "r") == NULL)
14     {
15        printf("File cannot open for reading " ) ;
16        exit(0) ;
17     }
18    fread (&personal , sizeof(personal) , 1, fp) ;
19    while ( !feof(fp) ) {
20       printf (“Name : %s \n” , personal.name) ;
21       printf (“Salary : %.2f \n” , personal.salary) ;
22       fread (&personal , sizeof(personal) , 1, fp) ;
23     }
24    fclose(fp) ;
25  }
```

โปรแกรมที่ 12.10 การใช้ฟังก์ชัน fread()

นอกจากนี้ยังมีฟังก์ชันที่เกี่ยวกับแฟ้ม เช่น fclose() โดยมีรายละเอียดในตารางที่ 12.1

ตารางที่ 12.1 ฟังก์ชันเกี่ยวกับไฟล์ในคลังโปรแกรม stdio.h

ฟังก์ชัน	รายละเอียดการใช้งาน
fclose()	int fclose(FILE *stream); ฟังก์ชันปิดแฟ้มปัจจุบัน จะส่งค่ากลับเป็น NULL ถ้าไม่สามารถปิดแฟ้มได้ ตัวอย่าง fclose(fp);
feof()	int feof(FILE *stream); ฟังก์ชันตรวจสอบการสิ้นสุดของแฟ้ม ส่งค่ากลับเป็นจริง เมื่อสิ้นสุดแฟ้ม ตัวอย่าง res = feof(fp);
fgetc()	int fgetc(FILE *stream); ฟังก์ชันอ่านอักขระจากแฟ้มครั้งละ 1 ไบต์และเลื่อนตัวแปรตัวชี้ (pointer) ไปยังตำแหน่งถัดไป ส่งค่า EOF ถ้าถึงจุดสิ้นสุดของแฟ้ม ตัวอย่าง ch = fgetc(fp);
fgets()	char *fgets (char *s, int n, FILE *stream); ฟังก์ชันอ่านสายอักขระจากแฟ้มครั้งละ n ไบต์และเลื่อนตัวแปรตัวชี้ (pointer) ไปยังตำแหน่งถัดไปจากการอ่าน จะส่งค่า NULL ถ้าไม่สามารถอ่านข้อมูลได้ ตัวอย่าง fgets(str,100,fp); อ่านข้อมูล 100 ไบต์ เก็บค่าที่อ่านได้ที่ตัวแปร str

flushall()	int flushall();
	ลบข้อมูลในคีย์บอร์ดบัฟเฟอร์ (Keyboard buffer) ใช้ก่อนการรับค่าข้อมูลที่เป็น array ของตัวอักษร (char)
	ใช้ฟังก์ชัน flushall(); ก่อนที่จะอ่านข้อมูลใหม่
	flushall();
	gets(Name);

fopen()	FILE *fopen(const char *filename, const char *mode); ฟังก์ชันเปิดแฟ้ม ค่าที่ส่งกลับคือ ตัวแปรตัวชี้ (pointer) ที่ชี้ตำแหน่งเริ่มของไฟล์บัฟเฟอร์ซึ่งเชื่อมโยงกับตำแหน่งข้อมูลในดิสก์ ตัวอย่าง fp=fopen(“File.txt”, “r”);
fprintf()	int fprintf(FILE *stream, const char *format[, argument,...]); บันทึกข้อมูลลงแฟ้ม

ตารางที่ 12.1 ฟังก์ชันเกี่ยวกับไฟล์ในคลังโปรแกรม stdio.h (ต่อ)

ฟังก์ชัน	รายละเอียดการใช้งาน
fputc()	int fputc(int c, FILE *stream); ฟังก์ชันบันทึกข้อลงไปสู่แฟ้มครั้งละ 1 ไบต์และเลื่อนตัวแปรตัวชี้ (pointer) ไปยังตำแหน่งถัดไป ฟังก์ชันจะส่งค่า EOF เมื่อเกิดข้อผิดพลาด เช่น fputc(ch,fp);
fputs()	char *fputs(char *s, FILE *stream); บันทึกแฟ้มและเลื่อนตัวแปรตัวชี้ (pointer) ไปยังตำแหน่งถัดไปจากการบันทึกอักขระตัวสุดท้ายที่บันทึกได้ถ้าไม่ผิดพลาด EOF เมื่อเกิดข้อผิดพลาด เช่น fputs(“Hello”,100,fp); บันทึกข้อมูล “Hello”ไปยังแฟ้ม fp
fread()	size_t fread(void *ptr, size_t size, size_t n, FILE *stream); อ่านข้อมูลจากแฟ้มในรูปแบบ binary ส่งค่า จำนวนข้อมูลที่อ่านได้สำเร็จ 0 เมื่อถึงจุดสิ้นสุดของแฟ้ม
fscanf()	int fscanf(FILE *stream, const char *format[, address, ...]); อ่านข้อมูลลงสู่แฟ้มในรูปแบบ text ส่งค่า จำนวนข้อมูลที่อ่านได้สำเร็จ (%) ถ้าไม่มีข้อมูลที่อ่านได้จะให้ค่า 0
fseek()	int fseek(FILE *stream, long offset, int whence); เลื่อนตำแหน่งของ File ตัวแปรตัวชี้ (pointer) มักนิยมใน Binary File ส่งค่า 0 ถ้าเลื่อน File ตัวแปรตัวชี้ (pointer) ได้สำเร็จ ค่าอื่นๆ ถ้าเลื่อนไม่สำเร็จ เช่น fseek(fp,+100,SEEK_SET); เลื่อน File ตัวแปรตัวชี้ (pointer) ไปข้างหน้า (+) 1000 ไบต์นับจากตำแหน่งต้นของแฟ้ม
fwrite()	บันทึกข้อมูลจากสู่ข้อมูลในรูปแบบ binary size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream); จำนวนข้อมูลที่บันทึกได้สำเร็จ

สรุป

ภาษาซีได้สร้างฟังก์ชันการทำงานเกี่ยวกับการเปิดแฟ้มคือ fopen() ซึ่งสามารถกำหนดโหมดในการเปิดได้ โดยกำหนดเป็นโหมดในการอ่าน เขียน หรือเพิ่มเติม และฟังก์ชันเพื่อการปิดแฟ้ม คือ fclose() และยังมีฟังก์ชันเพื่อการอ่านข้อมูลจากแฟ้มคือ fgetc() และ fscanf() ซึ่งการทำงานแตกต่างกันออกไป ควรเลือกให้เหมาะสมกับข้อมูลที่ใช้ในการอ่าน และฟังก์ชันที่ใช้ในการเขียนข้อมูล คือ fputc() และ fprintf() ซึ่งการทำงานมีรูปแบบที่แตกต่างกันออกไป ควรเลือกการใช้งานให้เหมาะสม

แบบฝึกหัด

1. ให้ประมวลผลโปรแกรมที่12.2 โดยสร้างแฟ้มของนักศึกษา C:\student.txt ด้วยโปรแกรม Notepad โดย ใส่ข้อมูลนักศึกษา และให้สังเกตการทำงานว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องจะแก้ไขอย่างไร
2. ให้สร้างโปรแกรมเพื่อบันทึกข้อมูลนักศึกษาแทนที่จะใช้โปรแกรม Notepad และเปรียบเทียบการอ่านข้อมูลกับแบบฝึกหัดที่ 12.1
3. ให้สร้างโปรแกรมเพื่อแสดงข้อมูลนักศึกษาที่เขียนโดยโปรแกรมที่ 12.3 ออกบนจอภาพ และลองตรวจสอบดูว่าถูกต้องหรือไม่ เพราะเหตุใด
4. หากการทำงานในแบบฝึกหัดที่ 12.3 ไม่ถูกต้องให้แก้ไขให้ถูกต้อง และอธิบายว่าเพราะเหตุใดจึงเกิดข้อผิดพลาดขึ้น
5. จงเขียนโปรแกรมเพื่ออ่านข้อมูลจากไฟล์แล้วนับจำนวนตัวอักษรที่มีอยู่ในไฟล์นั้น
6. จงเขียนโปรแกรมเพื่ออ่านข้อมูลจากไฟล์แล้วนับจำนวนตัวอักษร ‘a’ ที่มีอยู่ในไฟล์นั้นว่ามีอยู่กี่ตัวอักษร
7. จงเขียนโปรแกรมเพื่ออ่านข้อมูลจากไฟล์แล้วนับจำนวนคำ ที่มีอยู่ในไฟล์ว่ามีคำกี่คำ
8. จงอ่านข้อมูลจากไฟล์แล้วเปลี่ยนให้เป็นตัวอักษรพิมพ์ใหญ่ทั้งหมด
9. เขียนโปรแกรมเพื่อคัดลอดไฟล์จากไฟล์หนึ่งไปอีกไฟล์หนึ่ง โดยรับชื่อไฟล์ที่จะทำการคัดลอก และไฟล์ที่จะเขียนใหม่
10. ข้อมูลของนักศึกษาประกอบไปด้วย รหัสนักศึกษา (11 หลัก) ชื่อ-นามสกุล คะแนนการบ้านรวม คะแนนสอบกลางภาค และคะแนนสอบปลายภาค โดยมีเงื่อนไขว่า

ชื่อ-นามสกุล	เก็บข้อมูล ไม่เกิน	50 ตัวอักษร
คะแนนเก็บ	ไม่เกิน	30 คะแนน
คะแนนสอบกลางภาค	ไม่เกิน	30 คะแนน
คะแนนสอบปลายภาค	ไม่เกิน	40 คะแนน

ให้นักศึกษากำหนดโครงสร้างในการเก็บข้อมูลนักศึกษากำหนดขึ้นและให้นักศึกษาทำการเขียนฟังก์ชันเพื่อแสดงข้อมูลนักศึกษา ซึ่งบันทึกอยู่ในแฟ้มชื่อ C:\STUDENT.DAT โดยแฟ้มที่เก็บอยู่นั้นเป็นแฟ้มแบบ Text

ให้เขียนโปรแกรมหลักซึ่งเรียกใช้โปรแกรมย่อยซึ่งมี 2 โปรแกรมย่อยทำหน้าที่ดังนี้

10.1. โปรแกรมย่อยทำหน้าที่บันทึกข้อมูลลงแฟ้ม รับค่าข้อมูลนักศึกษา แต่ละข้อมูลของนักศึกษาแต่ละคนมาบันทึกลงแฟ้ม โดยจะวนรับไปเรื่อยๆ จนกว่าจะป้อนรหัสนักศึกษาเป็น 0 จำนวน 11 ตัว

10.2. โปรแกรมย่อยทำหน้าที่แสดงข้อมูลจากแฟ้ม จะแสดงข้อมูลนักศึกษาแต่ละคนออกทางจอภาพ ครั้งละ 10 คน เมื่อครบ 10 คนจะหยุดรอการกดแป้นพิมพ์จากผู้ใช้ ก่อนจึงจะแสดงข้อมูลถัดไปจนครบจำนวน (หากหน้าจอสุดท้ายแสดงไม่ถึง 10 ข้อมูล ก็ให้แสดงตามจริง)

11. ให้เขียนโปรแกรมการคิดราคาสินค้าในร้านค้า โดยเก็บข้อมูลในแฟ้ม

โปรแกรมการคิดราคาสินค้าในร้านค้า

Menu:

Product

11 Create New Product File
12 Add Product
13 List Product

Customer

21 Create New Customer File
22 Add Customer
23 List Customer

Invoice

31 Invoice

ให้นักศึกษาเขียนโปรแกรมเพื่อออกใบกำกับภาษีของร้านแห่งหนึ่ง โดยมีการทำงานดังนี้

1. ใส่รหัสลูกค้า แล้วโปรแกรมจะแสดงรายละเอียดลูกค้า โดยอ่านมาจากไฟล์ shop.txt

2. ใส่รหัสของสินค้า โปรแกรมจะแสดงชื่อ และ ราคาสินค้าดังกล่าว (อ่านจากไฟล์ Product.txt)

3. ใส่จำนวนสินค้าที่ต้องการซื้อ (ป้อนค่าทำได้ทีละรายการจนกว่าจะป้อนค่ารหัสสินค้าเป็นศูนย์)

จากนั้นให้โปรแกรมคำนวณหาราคารวมที่ต้องชำระเงิน แล้วสรุปเป็นรายการสินค้าที่ซื้อ ราคาภาษีมูลค่าเพิ่ม และราคารวมที่ผู้ซื้อต้องจ่ายทั้งหมด (หากสั่งซื้อรายการเดิมหลายครั้งให้แสดงผลรวมเพียงรายการเดียว) นอกจากนี้ให้มีการเช็คสต๊อกของสินค้าด้วย คือให้มีการกำหนดจำนวนต่ำสุดของสินค้าในคลังสินค้า เมื่อมีการซื้อสินค้าแล้วจำนวนที่เหลือต่ำกว่าที่กำหนดไว้ ให้พิมพ์

ข้อความเตือนว่าสินค้าต่ำกว่าจุดที่กำหนด เมื่อจบการทำงานให้มีการเขียนไฟล์เพื่อทำการ update ข้อมูลสินค้าคงเหลือในไฟล์ Product.txt

หมายเหตุ ให้โปรแกรมทำการอ่านข้อมูลเริ่มต้นจากไฟล์ 2 ไฟล์ ซึ่งประกอบด้วย

1. ไฟล์รายละเอียดสินค้า (Product.txt) ได้แก่ รหัส ชื่อ ราคา จำนวนสินค้า ขีดต่ำสุดในการสั่งซื้อ
2. ไฟล์รายละเอียดของร้าน (Shop.txt) ได้แก่ รหัสลูกค้า ชื่อร้าน ที่อยู่ เบอร์โทรศัพท์ อัตราส่วนลด และ อัตราภาษีมูลค่าเพิ่ม

ตัวอย่างข้อมูลในไฟล์ Product.txt

```
1001  Paper-A4      50.00   30   20
1002  White-Board   100.00  200  50
1003  Calendar      20.00   150  10
1004  Book          20.00   175  5
```

ตัวอย่างข้อมูลในไฟล์ Shop.txt

```
2001  Abc Co.ltd.   123 Sukumwit Road  Bangkok 10800   02-526-5236      20  7
2002  Dusit Shop    3 Pracharad  Road  Bangkok 10300   01-648-5236      30  10
2003  DTAC Shop     985 Pibulsongkam Road  Nonthaburi   09-574-1254      10  10
```

ตัวอย่างการสรุปรายการสินค้าที่ซื้อ

รหัสลูกค้า 2003
ชื่อร้าน .....DTAC Shop
ที่อยู่ .....985 Pibulsongkam Road Nonthaburi 10300
เบอร์โทรศัพท์ 09-574-1254

****

Product Code	Product name	Unit Price	Amount	Price
1001	Paper-A4	50	20	1000
1002	White-Board	100	5	500
1003	Calendar	20	10	200

****

Total	1700
Discount 10 %	170
After Discount	1530
VAT 10 %	153
Total Price	1683

เอกสารอ้างอิง

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: ดวงกมลสมัย.

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึ่ม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

ประภาพร ช่างไม้. (2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรส.

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for engineering and Technology.* London: Prentice Hall.

Kris Jamsa. (2002). *Jamsa's C/C++/C# programmer's bible: The ultimate guide to C/C++/C# programming.* 2nd Edition. USA: Onword Press.

บทที่ 13
การเรียงลำดับ

การทำงานโดยส่วนใหญ่ จำเป็นต้องมีการเรียงลำดับข้อมูลก่อนการเก็บข้อมูล เพื่อสะดวกในการค้นหา การเรียงลำดับ มีหลายวิธี เช่น การเรียงลำดับแบบฟอง (Bubble Sort) การเรียงเรียงลำดับแบบเลือก (Selection Sort) การเรียงลำดับโดยการแทรกตำแหน่ง (Insertion Sort) การเรียงลำดับแบบเร็ว (Quick Sort) การเรียงลำดับเชิงสถิติ (Order Statistics) การเรียงลำดับโดยใช้ฮีป (Heap Sort) การเรียงลำดับโดยวิธีของ Shell (Shell Sort) การเรียงลำดับโดยอาศัยฐานตัวเลข (Radix Sort) และอื่น ๆ อีกมากมาย แต่ในที่นี้ จะนำมาศึกษา 3 วิธี คือ การเรียงลำดับแบบฟอง (Bubble Sort) การเรียงลำดับแบบเลือก (Selection Sort) และการเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) เพื่อใช้ในการศึกษาวิธีการเรียงลำดับ

การเรียงลำดับแบบฟอง

การเรียงลำดับแบบฟอง (Bubble Sort) เป็นการเรียงลำดับที่มีการตรวจสอบค่าทีละค่าแล้วสลับค่าให้ถูกต้อง โดยในแต่ละรอบจะได้ค่าที่ถูกต้อง ไว้หลังสุด ในกรณีที่ต้องการเรียงลำดับจากน้อยไปมาก จะมีการตรวจสอบค่าที่ติดกัน หากเรียงไว้ไม่ถูกต้องจะสลับค่ากัน โดยในแต่ละรอบจะได้ค่าที่มากที่สุด อยู่หลังสุด เสมือนกับการลอยค่ามากที่สุด ลอยขึ้นเรื่อย ๆ เหมือนกับฟอง โดยมีอัลกอริทึมดังนี้

```
procedure bubbleSort ( A: list of sortable items ) defined as:
    for each i in 1 to length(A) do:
        for each j in length(A) downto  i + 1 do:
            if A[ j - 1 ] > A[ j ] then
                swap( A[ j - 1],  A[ j ] )
            end if
        end for
    end for
end procedure
```

วิธีการเรียงลำดับแบบฟอง มีการเขียนโปรแกรมลูป 2 ลูป คือ ลูป i รันจำนวนรอบตามจำนวนของข้อมูล และ ลูป j เป็นลูปที่เปรียบเทียบค่าแต่ละตัวว่าเรียงลำดับถูกต้องหรือไม่ หากค่ามากอยู่ก่อนค่าน้อย ก็จะทำการสลับค่า เมื่อสลับค่าในแต่ละรอบ จะได้ค่ามากที่สุดอยู่ท้ายสุด ทีละตัวทีละตัวในแต่ละรอบ

ดังนั้นหากต้องการเขียนโปรแกรมภาษาซีเพื่อเรียงลำดับแบบฟอง (Bubble Sort) สามารถทำได้ดังโปรแกรม ที่ 13.1

```
1   #define MAX_DATA 5
2   void main()
3   {
4       int number[MAX_DATA];
5       int  i, j, c;
6       for(j=0; j <= MAX_DATA-1; j++)
7           for(i=0; i <= MAX_DATA-1; i++)
8               if(number[i]>number[i+1])
9               {
10                      c=number[i];
11                      number[i]=number[i+1];
12                      number[i+1]=c;
13              }
14  }
```

โปรแกรมที่ 13.1 โปรแกรมเรียงลำดับแบบฟอง

ตัวอย่างที่ 13.1 กำหนดข้อมูลให้ มี 5 ค่า คือ 9, 4, 8, 7 และ 2 ต้องการเรียงลำดับจากน้อยไปมาก

9	4	8	7	2

รอบที่ 1

9	4	8	7	2
4	9 สลับ	8	7	2
4	8	9 สลับ	7	2
4	8	7	9 สลับ	2
4	8	7	2	9

ในรอบที่ 1 จะมีการเปรียบเทียบค่า 9 กับ 4 ซึ่ง อยู่ในตำแหน่งไม่ถูกต้องเพราะ 4 น้อยกว่า 9 เพราะฉะนั้น 4 ควรจะอยู่ก่อน 9 จึงเกิดการสลับค่า เมื่อสลับค่า 9 แล้ว 9 เปรียบเทียบกับ 8 ซึ่ง 8 น้อยกว่า 9 จึงเกิดการสลับค่า และ 9 เปรียบเทียบกับ 7 อีก 7 ก็น้อยกว่า 9 อีก เกิดการสลับค่า และเมื่อเปรียบเทียบกับ 2 แน่นอน 2 น้อยกว่า 9 ก็เกิดการสลับค่าอีก

จะเห็นว่า เมื่อจบรอบที่ 1 จะได้ ค่ามากที่สุด อยู่ท้ายสุด ถูกต้อง 1 ตำแหน่ง เสมือนกับว่า ค่ามากที่สุดถูกลอยขึ้นเรื่อย ๆ จนอยู่ในตำแหน่งที่ถูกต้อง

รอบที่ 2

4	8	7	2	9
4	8 สลับ	7	2	9
4	7	8 สลับ	2	9
4	7	2	8	9

ในรอบที่ 2 เกิดการเปรียบเทียบเช่นกัน หากค่าอยู่ในตำแหน่งที่ถูกต้อง คือค่าน้อยอยู่ก่อน ค่ามาก ก็จะไม่มีการสลับค่า หากผิดตำแหน่ง ก็จะมีการสลับค่า จะเห็นว่า เมื่อจบรอบที่ 2 จะได้ ค่ามากที่สุด อยู่ท้ายสุด ถูกต้อง 2 ตำแหน่ง เสมือนกับว่า ค่ามากที่สุดถูกลอยขึ้นเรื่อย ๆ จนอยู่ในตำแหน่งที่ถูกต้อง

รอบที่ 3

4	7	2	8	9
4	7 สลับ	2	8	9
4	2	7	8	9

ในรอบที่ 3 เกิดการเปรียบเทียบเช่นกัน หากค่าอยู่ในตำแหน่งที่ถูกต้อง คือค่าน้อยอยู่ก่อน ค่ามาก ก็จะไม่มีการสลับค่า หากผิดตำแหน่ง ก็จะมีการสลับค่า จะเห็นว่า เมื่อจบรอบที่ 2 จะได้ ค่ามากที่สุด อยู่ท้ายสุด ถูกต้อง 3 ตำแหน่ง

รอบที่ 4

4 สลับ	2	7	8	9
2	4	7	8	9

ในรอบที่ 4 ข้อมูลเกิดการสลับที่ ถูกต้องครบทุกตำแหน่ง จะพบว่า ถ้าข้อมูลมี 5 ตัว จะสามารถเรียงลำดับ ได้ถูกต้อง ในรอบที่ 4 คือ n -1 รอบและ ในรอบแรกมีการเปรียบเทียบ 4 ครั้ง และลดลงเรื่อย ในแต่ละรอบ เหลือ 3, 2 และ 1 ตามลำดับ

ผลลัพธ์

2	4	7	8	9

การเรียงลำดับแบบเลือก

การเรียงลำดับแบบเลือก (Selection Sort) เป็นวิธีการการเรียงลำดับที่มีอยู่ในสัญชาตญาณโดยธรรมชาติ คือ ค้นหาค่าตัวเลขน้อยที่สุด แล้วเปรียบเทียบค่าน้อยที่สุดกับข้อมูลแต่ละตัว แล้วสลับตำแหน่งให้ค่าน้อยที่สุด อยู่ในตำแหน่งที่ถูกต้อง แล้วหาค่าน้อยที่สุดตัวต่อไปเรื่อย ๆ จนถึงตัวสุดท้าย โดยมีขั้นตอนวิธีดังนี้

```
procedure  SelectionSort( A: list of sortable items ) defined as:
for i ⟵ 0 to n-2 do
      min ⟵ i
       for j ⟵ (i + 1) to n-1 do
            if A[j] < A[min]
                 min ⟵ j
       swap A[i] and A[min]
      end for
end for
end procedure
```

วิธีการเรียงลำดับแบบเลือก มีการเขียนโปรแกรมเพื่อคำนวณหาค่าที่ถูกต้อง ทีละตำแหน่ง ตั้งแต่ตำแหน่งที่ 1 ไปเรื่อย ๆ จนครบทุกตัว โดยมีการทำงานลูป 2 ลูป คือ ลูป i รันตำแหน่งเช่น รอบแรก I = 0 กำหนดให้เท่ากับ Min นั้นหมายความว่า ต้องการหาตัวเลข ที่มาแทนที่ตำแหน่งที่ 1โดยการวนหาตัวเลขที่น้อยที่สุด หรือน้อยกว่า ตัวแรก หากพบว่าตัวเลขใด มีค่าน้อยกว่า ตัวแรก ก็จะได้ตำแหน่งของตัวเลขที่น้อยที่สุด โดยในลูปนี้ วนจนครบข้อมูล แล้วเมื่อออกจากลูป J นั้นหมายความว่า ได้ตำแหน่งของตัวเลขที่น้อยที่สุดแล้ว จากนั้นก็ทำการสลับค่า ให้ตัวเลขที่น้อยที่สุด อยู่ในตำแหน่งแรก ซึ่งเป็นตำแหน่งที่ถูกต้อง เมื่อได้ตัวแรกถูกต้องก็จะหาตำแหน่งที่ 2 ต่อไป ว่ามีตัวเลขที่น้อยที่สุด ตัวต่อไปคือตัวเลขใด แล้วสลับค่า ไป เรื่อย ๆ จน ครบทุกตำแหน่ง

ดังนั้นหากต้องการเขียนโปรแกรมภาษาซีเพื่อเรียงลำดับแบบเลือก (Selection Sort) สามารถทำได้ ดังโปรแกรมที่ 13.2

```
1   #define MAX_DATA 5
2   void main()
3   {
4           for( I=0; I <= MAX_DATA-2; I++)
5           {
6                   Min = I ;
7                   for( J =I+1 ; J <= MAX_DATA-1; J++)
8                   {
9                       if  (number[Min] > number[J])
10                                      Min = J ;
11                  }
12                  If  (Min != I)
13                  {
14                      temp=number[I] ;
15                      number[I]=number[Min] ;
16                      number[Min]=temp ;
17                  }
18          }
19  }
```

โปรแกรมที่ 13.2 โปรแกรมการเรียงลำดับแบบเลือก (Selection Sort)

ตัวอย่างที่ 13.2 กำหนดข้อมูลให้ มี 5 ค่า คือ 9, 4, 8, 7 และ 2 ต้องการเรียงลำดับจากน้อยไปมาก

9	4	8	7	2

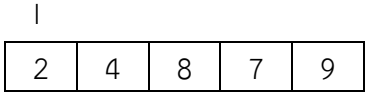
รอบที่ 1 min = 0 หาตัวเลขที่น้อยที่สุด ได้ 2 สลับค่า 2 ไว้ที่ Number [0]

				I
9	4	8	7	2

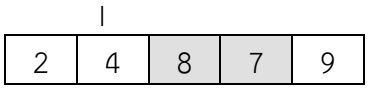
ผลลัพธ์ รอบที่ 1

2	4	8	7	9

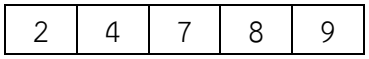
รอบที่ 2 min = 1 หาตัวเลขที่น้อยที่สุด จากตำแหน่งที่ I + 1 ซึ่งไม่มีค่าใดน้อยกว่าค่า number[Min] จึงไม่มีการสลับค่า



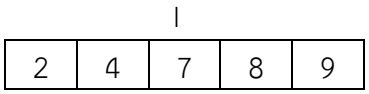
รอบที่ 3 min = 2 หาตัวเลขที่น้อยที่สุด จากตำแหน่งที่ I + 1 ซึ่งได้ค่า 7 จึงมีการสลับค่า ระหว่าง 7 กับ 8



ผลลัพธ์ รอบที่ 3



รอบที่ 4 min = 3 หาตัวเลขที่น้อยที่สุด จากตำแหน่งที่ I + 1 ซึ่งไม่มีค่าใดน้อยกว่าค่า number[Min] จึงไม่มีการสลับค่า ซึ่งจะได้ผลลัพธ์ ถูกต้องทุกตำแหน่ง



การเรียงลำดับแบบแทรกตำแหน่ง

การเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) เป็นวิธีที่จะเอาข้อมูลปัจจุบันไปวางไว้ตรงตำแหน่งที่เหมาะสมที่สุด เพื่อให้เกิดลำดับใหม่ขึ้น ในขณะที่เอาข้อมูลตัวปัจจุบันไปวาง จะทำการเลื่อนข้อมูลที่อยู่ระหว่างตำแหน่งใหม่ของมันกับตำแหน่งปัจจุบันของมันไปทางขวา (ชิดชนก เหลือสินทรัพย์. 2543: 82-83) โดยมีขั้นตอนวิธีดังนี้

```
procedure InsertionSort( A: list of sortable items ) defined as:
for I ⟵ 2 to n do
begin
    a ⟵ D[I]
    J = I -1
    while J > 0 and D[J] > a do
     begin
        D[J + 1] = D[J]
        J = J – 1
     end while
     D[J + 1] = a
end for
end procedure
```

ตัวอย่างที่ 13.3 กำหนดข้อมูล มี 9 4 8 7 2 เรียงจากน้อยไปมาก

รอบที่ 1 I = 2 คือพิจารณา 2 ตำแหน่งแรกโดยใช้ตัวเลข 4 เป็นข้อมูลที่ใช้เป็นข้อมูลปัจจุบัน แล้วเปรียบเทียบกับข้อมูลก่อนหน้า ถ้าค่ามากกว่า ข้อมูลปัจจุบัน จะเลื่อนตำแหน่งเพิ่มขึ้น 1 ตำแหน่ง

I=2

9	4	8	7	2
4	9	8	7	2

รอบที่ 2 I = 3 คือพิจารณา 3 ตำแหน่งแรกโดยใช้ตัวเลข 8 เป็นข้อมูลที่ใช้เป็นข้อมูลปัจจุบัน แล้วเปรียบเทียบกับข้อมูลก่อนหน้า 9 มากกว่า 8 เลื่อนตำแหน่งเพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 2 แล้วเปรียบตัวต่อไป คือ 4 ซึ่งไม่มากกว่า 8 จึงหยุด ซึ่ง J=1 กำหนดค่า 8 ให้ D[J + 1] หลังออกจากลูป ได้ D[2] = 8

I=3

4	9	8	7	2
4	9	9	7	2
4	8	9	7	2

D[J + 1] = D[J] เลื่อนตำแหน่งไปทางขวา

D [J + 1] = a

รอบที่ 3 I = 4 คือพิจารณา 4 ตำแหน่งแรกโดยใช้ตัวเลข 7 เป็นข้อมูลที่ใช้เป็นข้อมูลปัจจุบัน แล้วเปรียบเทียบกับข้อมูลก่อนหน้า 9 มากกว่า 7 เลื่อนตำแหน่งของ 9 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 3 แล้ว เปรียบตัวต่อไป คือ 8 ซึ่งมากกว่า 7 เลื่อนตำแหน่งของ 8 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 2 แล้ว เปรียบตัวต่อไป คือ 4 ซึ่งไม่มากกว่า 7 จึงหยุด ซึ่ง J=1 กำหนดค่า 7 ให้ D[J + 1] หลังออกจากลูป ได้ D[2] = 7

I=4

4	8	9	7	2	
4	8	9	9	2	D[J + 1] = D[J] เลื่อนตำแหน่งไปทางขวา
4	8	8	9	2	D[J + 1] = D[J] เลื่อนตำแหน่งไปทางขวา
4	7	8	9	2	D [J + 1] = a

รอบที่ 4 I = 5 คือพิจารณา 5 ตำแหน่งแรกโดยใช้ตัวเลข 2 เป็นข้อมูลที่ใช้เป็นข้อมูลปัจจุบัน แล้วเปรียบเทียบกับข้อมูลก่อนหน้า 9 มากกว่า 2 เลื่อนตำแหน่งของ 9 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 4 แล้ว เปรียบตัวต่อไป คือ 8 ซึ่งมากกว่า 2 เลื่อนตำแหน่งของ 8 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 2 แล้ว เปรียบตัวต่อไป คือ 7 ซึ่งมากกว่า 2 เลื่อนตำแหน่งของ 7 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 1 แล้วเปรียบตัวต่อไป คือ 4 ซึ่งมากกว่า 2 เลื่อนตำแหน่งของ 4 เพิ่มขึ้น 1 ตำแหน่ง ซึ่ง J = 0 กำหนดค่า 2 ให้ D[J + 1] หลังออกจากลูป ได้ D[1] = 2

I=5

4	7	8	9	2	
4	7	8	9	9	D[J+1] = D[J] เลื่อนตำแหน่งของ 9 ไปทางขวา
4	7	8	8	9	D[J+1] = D[J] เลื่อนตำแหน่งของ 8 ไปทางขวา
4	7	7	8	9	D[J+1] = D[J] เลื่อนตำแหน่งของ 7 ไปทางขวา
4	4	7	8	9	D[J+1] = D[J] เลื่อนตำแหน่งของ 4 ไปทางขวา
2	4	7	8	9	D[J+1] = a

วิธีการเรียงลำดับแบบแทรกตำแหน่ง มีวิธีการพิจารณาค่าทีละค่าว่าอยู่ในตำแหน่งที่ถูกต้องหรือไม่ หากไม่ถูกก็เลือกตำแหน่งตัวเลขอื่นออกไป เพื่อแทรกตัวเลขที่กำลังพิจารณาอยู่เข้าไปพิจารณาทีละค่า ไปเรื่อย ๆ จนครบทุกค่า

ดังนั้นหากต้องการเขียนโปรแกรมภาษาซีเพื่อเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) สามารถทำได้ ดังโปรแกรมที่ 13.3

```
1   #define MAX_DATA 5
2   void main()
3   {
4           for( I= 2; I <= MAX_DATA-2; I++)
5           {
6                   a = D[I]  ;
7                   J = I – 1 ;
8                 while  ( J > 0  &  D[J]  > a )
9                    {
10                         D[J + 1] =  D[J]  ;
11                         J = J - 1 ;
12                    }
13                  D[J + 1] =  a  ;
14          }
15  }
```

โปรแกรมที่ 13.3 โปรแกรมการเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort)

สรุป

การเรียงลำดับแบบฟอง (Bubble Sort) มีหลักการในการเรียงข้อมูล โดยจัดการข้อมูลให้อยู่ในตำแหน่งที่ถูกต้อง จนได้ข้อมูลเรียงถูกต้องทุกจำนวน โดยเปรียบเทียบข้อมูลครั้งละคู่เท่ากับจำนวนของข้อมูล n-1 ครั้ง เปรียบเทียบจำนวน n-1 รอบ

การเรียงลำดับแบบเลือก (Selection Sort) มีหลักการคือหาตัวเลขที่น้อยที่สุด มาเรียงทีละตำแหน่ง โดยเริ่มจากตำแหน่งที่ 1 ไปเรื่อย ๆ จนครบได้ข้อมูลทุกตัว โดยการหาจำนวนตัวเลขที่น้อยที่สุด สับค่าให้อยู่ในตำแหน่งแรก และหาจำนวนที่น้อยลำดับต่อไป สลับให้อยู่ในตำแหน่งถัดไปเรื่อย ๆ จนครบทุกจำนวน

การเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) มีหลักการคือพิจารณาตัวเลขทีละค่า แล้วหาตำแหน่งที่เหมาะสมให้กับตัวเลขนั้นทีละตัว จนครบทุกตัว โดยนำข้อมูลปัจจุบันไปวางไว้ตรงตำแหน่งที่เหมาะสมที่สุด เพื่อให้เกิดลำดับใหม่ขึ้น ในขณะที่เอาข้อมูลตัวปัจจุบันไปวาง จะทำการเลื่อนข้อมูลที่อยู่ระหว่างตำแหน่งใหม่กับตำแหน่งปัจจุบันไปทางขวา

แบบฝึกหัด

1. ให้วิเคราะห์วิธีการเรียงลำดับแบบฟอง (Bubble Sort) ว่ามีวิธีปรับเปลี่ยนโปรแกรมอย่างไร เพื่อให้ มีจำนวนครั้งของการเปรียบเทียบลดลงมากที่สุด แล้วยังสามารถทำงานได้เหมือนเดิม

2. ให้วิเคราะห์วิธีการเรียงลำดับแบบเลือก (Selection Sort) ว่ามีวิธีปรับเปลี่ยนโปรแกรมอย่างไร เพื่อให้ มีจำนวนครั้งของการเปรียบเทียบลดลงมากที่สุด แล้วยังสามารถทำงานได้เหมือนเดิม

3. ให้วิเคราะห์วิธีการเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) ว่ามีวิธีปรับเปลี่ยนโปรแกรมอย่างไร เพื่อให้ มีจำนวนครั้งของการเปรียบเทียบลดลงมากที่สุด แล้วยังคงมีหลักการทำงานเหมือนเดิม

4. ให้เขียนโปรแกรมการเรียงลำดับของนักศึกษาเอง โดยนำวิธีการเรียงลำดับแบบฟองและวิธีการเรียงลำดับแบบเลือก มาประยุกต์ใช้ โดยต้องมีจำนวนครั้งของการเปรียบเทียบ น้อยที่สุดและยังสามารถเรียงลำดับได้ถูกต้องเหมือนเดิม

5. ให้เขียนโปรแกรมการเรียงลำดับของนักศึกษาเอง โดยนำวิธีการเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) และวิธีการเรียงลำดับแบบเลือก (Selection Sort) มาประยุกต์ใช้ โดยต้องมีจำนวนครั้งของการเปรียบเทียบน้อยที่สุด และยังสามารถเรียงลำดับได้ถูกต้องเหมือนเดิม

เอกสารอ้างอิง

ชิดชนก เหลือสินทรัพย์. (2543). *Analysis & Design of Algorithms.* กรุงเทพมหานคร: Sum System Company Limited.

บทที่ 14
การค้นหา

การค้นหาข้อมูล เป็นอีกเรื่องที่มีความสำคัญกับการทำงานเป็นอย่างมาก การค้นหาข้อมูลเป็นการค้นหาตำแหน่งที่อยู่ ของข้อมูลที่ต้องการ วิธีการค้นหาข้อมูลมีหลายวิธี เช่น การค้นหาแบบเรียงลำดับ (Sequential Search) การค้นหาแบบทวิภาค (Binary Search) และอื่น ๆ อีกมากมาย สำหรับในบทนี้ ได้นำวิธีการค้นหา มา 2 วิธี คือ การค้นหาแบบเรียงลำดับ (Sequential Search) และการค้นหาแบบทวิภาค (Binary Search) เพื่อใช้ในการศึกษาวิธีการค้นหา

การค้นหาแบบเรียงลำดับ

การค้นหาแบบเรียงลำดับ (Sequential Search) หรือเรียกอีกชื่อหนึ่งว่า Linear Search เป็นวิธีการค้นหาที่ง่ายที่สุด โดยมีหลักการค้นหา คือ ใช้ค่าที่ต้องการค้นหาเปรียบเทียบกับข้อมูลทีละค่า แล้วตรงกับข้อมูลตัวใด ก็จะแสดงตำแหน่งนั้นออกมา โดยมีขั้นตอนวิธีดังนี้

```
Function  SeqSearch ( A: list of   items, target: valueToFind ) as integer
for   I = 1   to   n
begin
    if (target == A[I] )
       return  I;
end for
return -1;
end  function
```

วิธีการค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) เริ่มจากข้อมูลตัวที่ 1 แล้วจากนั้นตรวจสอบว่าตรงกับข้อมูลที่ต้องการค้นหา หรือไม่ ถ้าตรงก็จะส่งค่าตำแหน่งที่ 1 กลับให้กับฟังก์ชัน หากไม่ตรงกัน ก็เพิ่มค่าตำแหน่งเป็น 2 และ 3 ไปเรื่อย ๆ แล้วเปรียบเทียบถ้าค่าใดตรงกับค่าที่ต้องการค้นหา ก็ส่งค่าตำแหน่งที่ตรงนั้นให้กับฟังก์ชัน หากเพิ่มค่าตำแหน่งจนครบทุกข้อมูลแล้วไม่พบค่าใดตรงกับค่าที่ต้องการค้นหา ก็ส่งค่า (-1) ให้กับฟังก์ชัน แสดงว่าในข้อมูลทั้งหมดไม่มีข้อมูลที่ต้องการค้นหา หรือค้นหาไม่พบ

ดังนั้นหากต้องการเขียนโปรแกรมภาษาซีเพื่อค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) สามารถทำได้ ดังโปรแกรมที่ 14.1

```
1   #define MAX_DATA 5
2   int    SeqSearch ( int  A[MAX_DATA] , int  target )
3   {
4         for( I= 0; I < MAX_DATA  ;  I++)
5         {
6               if (target == A[I] )
7                     return  I;
8          }
9          return 0 ;
10  }
```

โปรแกรมที่ 14.1 โปรแกรมการค้นหาข้อมูลแบบเรียงลำดับ

ตัวอย่างที่ 14.1 กำหนดข้อมูลให้ มี 5 ค่า คือ 9, 4, 8, 7 และ 2 ต้องการค้นหาค่า 8

9	4	8	7	2

โปรแกรมจะตรวจสอบค่าทีละค่า เริ่มจาก 9 ว่าเท่ากับ 8 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่ง ตรวจสอบ ค่า 4 ว่าเท่ากับ 8 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่งไปอีก ตรวจสอบค่า 8 ได้เท่ากับ 8 ก็จะ ส่งค่า 3 แสดงว่า พบ 8 ในข้อมูล ตำแหน่งที่ 3 เป็นต้น

ตัวอย่างที่ 14.2 กำหนดข้อมูลให้ มี 5 ค่า คือ 9, 4, 8, 7 และ 2 ต้องการค้นหาค่า 5

9	4	8	7	2

โปรแกรมจะตรวจสอบค่าทีละค่า เริ่มจาก 9 ว่าเท่ากับ 5 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่ง ตรวจสอบ ค่า 4 ว่าเท่ากับ 5 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่งไปอีก ตรวจสอบค่า 8 ว่าเท่ากับ 5 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่งไปอีก ตรวจสอบค่า 7 ว่าเท่ากับ 5 หรือไม่ ซึ่งไม่เท่า ก็จะเลื่อนตำแหน่งไปอีก ตรวจสอบค่า 2 ว่าเท่ากับ 5 หรือไม่ ซึ่งไม่เท่า ซึ่งข้อมูลหมด ตรวจสอบครบทุกตัว ออกจากลูป โปรแกรมก็จะส่ง ค่า 0 ให้กับโปรแกรม แสดงว่า ไม่พบค่าที่ต้องการค้นหา

การค้นหาแบบทวิภาค

การค้นหาแบบทวิภาค (Binary Search) เป็นวิธีการค้นหาที่ข้อมูลต้องมีการเรียงลำดับให้เรียบร้อยเสียก่อน โดยการค้นหาจากค่าที่อยู่ตรงกลาง โดยหาข้อมูลตรงกลาง ไปเรื่อย ๆ จนกว่าข้อมูลจะครบ หากค้นหาครบแล้วไม่เจอ แสดงว่า ไม่พบข้อมูลดังกล่าว โดยมีขั้นตอนวิธีดังนี้

```
Function BinarySearch  ( A: array of data ,  n: integer, value: integer) as integer
begin
    int mid = 0 ;
    int upper = n ;
    int lower = 1 ;
    mid = (lower + upper) / 2;
    while( mid > 0 )
    begin
        mid = (lower + upper) / 2;
        if( value == A[mid] )
                return mid ;
        else  if(lower > upper)
                return  0 ;
        else
                if  (value  >   A[mid]  )
                        lower = mid + 1;        { ค่าที่ต้องการอยู่ในส่วนหลัง }
                else
                        upper = mid - 1;        { ค่าที่ต้องการอยู่ในส่วนหน้า  }
    end
end
```

วิธีการค้นหาแบบทวิภาค (Binary Search) มีวิธีการค้นหาจากค่าที่อยู่ตรงกลางของข้อมูล จนกว่าตำแหน่งตรงกลางคือ 0 โดยเปรียบเทียบค่าที่อยู่ตรงกลาง ถ้าค่าที่ต้องการหามีค่าน้อยกว่าค่าที่อยู่ตรงกลาง แสดงค่า ค่าที่ต้องการอาจอยู่ในส่วนหน้า จะกำหนด จำนวนข้อมูลให้ลดลงครึ่งหนึ่ง โดยกำหนดจำนวนข้อมูลเป็นค่าตรงกลางแทน เพราะไม่สนใจค่าที่อยู่มากกว่าแล้ว แต่ถ้าค่าที่ต้องการค้นหามีค่ามากกว่า จะกำหนดค่าเริ่มต้น เป็นค่ากลางแทน เพราะส่วนแรกมีค่าน้อยกว่าจะไม่สนใจ เพราะค่าที่ต้องการค้นหา มีค่ามากกว่า ตรงกลาง แต่ทั้งนี้ทั้งนั้น ข้อมูลต้องมีการเรียงลำดับแล้ว จึงจะทำวิธีนี้ได้ ดังตัวอย่างที่ 14.3

ตัวอย่างที่ 14.3 กำหนดข้อมูลให้ มี 5 ค่า คือ 2, 4, 7, 8 และ 9 ต้องการค้นหาค่า 2

mid=3	2	4	**7**	8	9
mid=2	2	**4**	7	8	9
mid=1	**2**	4	7	8	9

ข้อมูลมีทั้งหมด 5 ตัว ได้ upper = 5 คำนวณหาค่ากลาง mid = (1 + 5) /2 = 3 ค่าที่อยู่ตรงกลาง คือ 7 (x[3]) ค่าที่จะค้นหา คือ 2 ซึ่งค่าที่ต้องการน้อยกว่าค่าที่อยู่ตรงกลาง แสดงว่า ค่าที่ต้องการอยู่ส่วนหน้า ปรับค่า upper ใหม่ เป็น 3 – 1 = 2 คำนวณหาตำแหน่งกลางใหม่ = (1 + 2) /2 = 2 (ปัดขึ้น) ซึ่งได้ค่าในตำแหน่งตรงกลางคือ 4 (x[2]) เปรียบเทียบกับค่าที่ต้องการคือ 2 ซึ่งค่าที่ต้องการน้อยกว่าค่าที่อยู่ตรงกลาง แสดงว่า ค่าที่ต้องการอยู่ส่วนหน้า ปรับค่า upper ใหม่ เป็น 2 – 1 = 1 คำนวณหาตำแหน่งกลางใหม่ = (1 + 1) /2 = 1 ซึ่งได้ค่าในตำแหน่งตรงกลาง คือ 2 (x[1]) เปรียบเทียบกับค่าที่ต้องการคือ 2 ซึ่งมีค่าเท่ากันเพราะฉะนั้น ส่งค่าตำแหน่ง 1 ให้กับฟังก์ชัน

ตัวอย่างที่ 14.4 กำหนดข้อมูลให้ มี 5 ค่า คือ 2, 4, 7, 8 และ 9 ต้องการค้นหาค่า 9

mid=3	2	4	**7**	8	9
mid=5	2	4	7	8	**9**

ข้อมูลมีทั้งหมด 5 ตัว ได้ upper = 5 คำนวณหาค่ากลาง mid = (1 + 5) /2 = 3 ค่าที่อยู่ตรงกลาง คือ 7 (x[3]) ค่าที่จะค้นหา คือ 9 ซึ่งค่าที่ต้องการมกกว่า แสดงว่า ค่าที่ต้องการอยู่ส่วนหลัง ปรับค่า lower ใหม่ เป็น 3 + 1 = 4 คำนวณหาตำแหน่งกลางใหม่ = (4 + 5) /2 = 5 (ปัดขึ้น) ซึ่งได้ค่าในตำแหน่งตรงกลางคือ 9 (x[5]) เปรียบเทียบกับค่าที่ต้องการคือ 9 ซึ่งมีค่าเท่ากันเพราะฉะนั้น ส่งค่าตำแหน่ง 5 ให้กับฟังก์ชัน

ดังนั้นหากต้องการเขียนโปรแกรมภาษาซีเพื่อการค้นหาแบบเลือกทวิภาค (Binary Search) สามารถทำได้ดังโปรแกรมที่ 14.2

```
1   #define MAX_DATA 5
2   int   BinarySearch  ( int  A[MAX_DATA] , int  target )
3   {
4       int  mid = 0 ;
5       int  upper = n ;
6       int  lower = 1 ;
7       mid = (lower + upper) / 2;
8       while( mid > 0 )
9       {
10          mid = (lower + upper) / 2;
11          if  ( value == A[mid] )
12                  return mid ;
13          else  if  (lower > upper)
14                  return  0 ;
15          else
16                  if  (value  >   A[mid]  )
17                          lower = mid + 1;
18                  else
19                          upper = mid - 1;
20      }
22  }
```

โปรแกรมที่ 14.2 โปรแกรมการค้นหาแบบเลือกทวิภาค

ตัวอย่างที่ 14.4 กำหนดข้อมูลให้ มี 5 ค่า คือ 2, 4, 7, 8 และ 9 ต้องการค้นหาค่า 5

mid=3	lower=1	upper=5	2	4	**7**	8	9
mid=2	lower=1	upper=2	2	**4**	7	8	9
mid=2	lower=3	upper=2	2	**4**	7	8	9

ข้อมูลมีทั้งหมด 5 ตัว ได้ upper = 5 คำนวณหาค่ากลาง mid = (1 + 5) /2 = 3 ค่าที่อยู่ตรงกลาง คือ 7 (x[3]) ค่าที่จะค้นหา คือ 5 ซึ่งค่าที่ต้องการน้อยกว่าค่าที่อยู่ตรงกลาง แสดงว่าค่าที่ต้องการอยู่ส่วนหน้า ปรับค่า upper ใหม่ เป็น 3 – 1 = 2 คำนวณหาตำแหน่งกลางใหม่ = (1 + 2) /2 = 2 (ปัดขึ้น) ซึ่งได้ค่าในตำแหน่งตรงกลางคือ 4 (x[2]) เปรียบเทียบกับค่าที่ต้องการคือ 5 ซึ่งค่าที่ต้องการมากกว่าค่าที่อยู่ตรงกลาง แสดงว่า ค่าที่ต้องการอยู่ส่วนหลัง ปรับค่า lower = mid – 1 = lower ใหม่ เป็น 2 + 1 = 3 คำนวณหาตำแหน่งกลางใหม่ = (3 + 2) /2 = 2 ซึ่งได้ค่าในตำแหน่งตรงกลางคือ 4 (x[2]) เปรียบเทียบกับค่าที่ต้องการคือ 5 ซึ่งยังไม่เท่ากันแต่ค่า lower = 3 มากกว่าค่า upper = 2 ส่งค่าตำแหน่ง 0 ให้กับฟังก์ชัน แสดงว่าไม่พบข้อมูล

การปรับปรุงประสิทธิภาพการค้นหาแบบเรียงลำดับ

การค้นหาแบบเรียงลำดับ จะเสียเวลามากกรณีที่มีข้อมูลจำนวนมาก และบางครั้งข้อมูลใช้บ่อย ๆ อยู่ท้ายสุด เพื่อให้การค้นหาข้อมูลแบบเรียงลำดับมีประสิทธิภาพมากขึ้น มีขั้นตอนวิธีหลายแบบที่นำมาใช้และเพิ่มเข้าไปในอัลกอริทึมการค้นหาแบบรียงลำดับ โดยมีขั้นตอนวิธีดังนี้

1. การย้ายข้อมูลไปไว้ด้านหน้า (Move to the Front) ในการค้นหาข้อมูลแบบเรียงลำดับ ถ้ามีข้อมูลจำนวนมากและข้อมูลที่ต้องการอยู่ข้างหลัง ทำให้การค้นหาในแต่ละครั้งใช้เวลาในการเปรียบเทียบจำนวนมาก หากถ้าข้อมูลที่ต้องการใช้อยู่ด้านหน้า จะสามารถค้นหาได้เร็วขึ้น วิธีนี้ เป็นวิธีที่ย้ายข้อมูลที่ค้นหาพบย้ายมากข้างหน้า แล้วเลื่อนตำแหน่งข้อมูลออกไป 1 ตำแหน่ง โดยใช้วิธีการเดียวกับวิธีการเรียงลำดับแบบแทรกตำแหน่ง (Insertion Sort) อัลกอริทึมที่ใช้ในการเปลี่ยนแปลงเพิ่มเติม มีดังนี้

```
Tmp=key[i];
for (i= n ; i >0 ; i--)
{
        key[i]=key[i-1];
key[0] =tmp;
}
```

2. การเปลี่ยนตำแหน่ง (Transposition) การปรับปรุงโดยวิธีการย้ายไปข้างหน้าแล้วเลื่อนตำแหน่งทุกค่า สามารถทำให้ข้อมูลที่ใช้อยู่ด้านหน้า แต่ต้องเลื่อนตำแหน่งข้อมูลทุกตัวออกไป 1 ตำแหน่ง ซึ่งทำให้เสียเวลาอยู่ไม่น้อย เท่ากับ จำนวนข้อมูลทั้งหมด ซึ่งไม่เกิดประโยชน์มากนัก วิธีการที่ 2 คือเปลี่ยนตำแหน่งเฉพาะ ตัวที่ต้องการค้นหากับตัวแรกก็เพียงพอ ขั้นตอนวิธีที่ใช้ในการสลับค่า มีดังนี้

```
Tmp = Key [i] ;
Key[i] =Key[0] ;
Key[0] = Tmp ;
```

การปรับปรุงประสิทธิภาพโดยวิธีการย้ายไปข้างหน้า หรือสลับตำแหน่งจะทำให้ข้อมูลที่ใช้ต้องการอยู่ด้านหน้า ทำให้สามารถค้นหาข้อมูลได้เร็วขึ้น

ตัวอย่างที่ 14.5 กำหนดข้อมูลให้ มี 5 ค่า คือ 1, 2, 6, 8 และ 9 ต้องการค้นหาค่า 6 ให้นักศึกษาเขียนโปรแกรมเพื่อค้นหาค่า 6 ว่าอยู่ในตำแหน่งที่เท่าไร

วิธีคิด โจทย์กำหนดข้อมูลให้ มี 5 ค่า คือ 1, 2, 6, 8 และ 9 กำหนดให้เป็นตัวแปร Global เขียนฟังก์ชัน Binary Search รับข้อมูล จำนวนข้อมูล และ ข้อมูลที่ต้องการค้นหา ใช้ตัวแปร pos เป็นตัวรับค่าตำแหน่ง แล้วแสดงค่าตำแหน่งออกมา ได้ดังโปรแกรมที่ 14.3

โปรแกรมที่ 14.3 โปรแกรมภาษาซี กำหนดข้อมูลให้ มี 5 ค่า คือ 1, 2, 6, 8 และ 9 ต้องการค้นหาค่า 6

```
1   #define MAX_DATA 5
2   int A[5] = { 1, 2, 6, 8, 9 } ;
3   int   BinarySearch  ( int  A[MAX_DATA] , int n, int  target )
4   {
5       int  mid = 0 ;
6       int  upper = n ;
7       int  lower = 1 ;
8       mid = (lower + upper) / 2;
9       while( mid > 0 )
10      {
11          mid = (lower + upper) / 2;
12          if( target == A[mid] )
13                  return mid +1  ;
14          else  if(lower > upper)
15                  return  0 ;
16          else
17                  if  (target  >   A[mid]  )
18                          lower = mid + 1;
19                  else
20                          upper = mid - 1;
22      }
23      return 0;
24  }
25  void main()
26  {
27          int value = 6;
28          int pos ;
29          pos = BinarySearch(A,MAX_DATA,value) ;
30          printf("Found at Position: %d",pos) ;
31  }
```

โปรแกรมที่ 14.3 โปรแกรมการค้นหาข้อมูล

สรุป

การค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) เป็นการค้นหาข้อมูลทีละตำแหน่งไปเรื่อย ๆ จนกว่าจะพบข้อมูล ที่ต้องการค้นหา หรือค้นหาข้อมูลจนครบแสดงไม่พบข้อมูล ส่วนการค้นหาข้อมูลแบบทวิภาค (Binary Search) เป็นการค้นหาข้อมูลจากข้อมูลที่เรียงลำดับแล้ว สามารถค้นหาได้รวดเร็วกว่า การค้นหาแบบเรียงลำดับเพราะค้นหาจากข้อมูลทีละครึ่ง สามารถลดข้อมูล ทีละครึ่ง ทีละครึ่ง ได้ช่วยลดเวลาในการค้นหาได้มาก แต่ทั้งนี้ทั้งนั้น ขึ้นอยู่กับข้อมูลที่ใช้กับลักษณะการใช้ข้อมูลว่าเป็นลักษณะใด และเทคนิคในการค้นหายังมีอีกมากมาย

แบบฝึกหัด

1. ให้วิเคราะห์วิธีการค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) ว่ามีวิธีปรับเปลี่ยนโปรแกรมอย่างไร เพื่อให้ มีจำนวนครั้งของการเปรียบเทียบลดลงมากที่สุด แล้วยังสามารถทำงานได้เหมือนเดิม

2. ให้วิเคราะห์วิธีการค้นหาข้อมูลแบบทวิภาค (Binary Search) ว่ามีวิธีปรับเปลี่ยนโปรแกรมอย่างไร เพื่อให้ มีจำนวนครั้งของการเปรียบเทียบลดลงมากที่สุด แล้วยังสามารถทำงานได้เหมือนเดิม

3. ให้เขียนโปรแกรมการค้นหาข้อมูลของนักศึกษาเอง โดยนำวิธีการค้นหาข้อมูลแบบเรียงลำดับ (Sequential Search) และวิธีการค้นหาข้อมูลแบบทวิภาค (Binary Search) มาประยุกต์ใช้ โดยต้องมีจำนวนครั้งของการเปรียบเทียบ น้อยที่สุด และยังสามารถเรียงลำดับได้ถูกต้องเหมือนเดิม

4. เขียนโปรแกรมเพื่อรับข้อมูล 10 จำนวน แล้วรับค่าที่ต้องการค้นหาจากแป้นพิมพ์ แล้วแสดงตำแหน่งของของค่าที่ต้องการที่ค้นหา

เอกสารอ้างอิง

ชิดชนก เหลือสินทรัพย์. (2543). *Analysis & Design of Algorithms.* กรุงเทพมหานคร: Sum System Company Limited.

บรรณานุกรม

ไกรศร ตั้งโอภากุล. (2554). คู่มือเรียนเขียนโปรแกรมภาษา C. นนทบุรี: ไอดีซีพรีเมียร์.

เจนวิทย์ เหลืองอร่าม. (2537). *การใช้ Turbo C++ เขียนโปรแกรมภาษา C.* กรุงเทพมหานคร: สุขภาพใจ.

เฉลียว เพ็ชกิ่ง. (2546). *เอกสารประกอบการสอนการเขียนโปรแกรมคอมพิวเตอร์และอัลกอริทึม.* กรุงเทพมหานคร: มหาวิทยาลัยราชภัฏสวนดุสิต.

ชิดชนก เหลือสินทรัพย์. (2543). *Analysis & Design of Algorithms.* กรุงเทพมหานคร: Sum System Company Limited.

ประภาพร ช่างไม้. (2545). *คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.* กรุงเทพมหานคร: อินโฟเพรสใ

ปัญญาพล หอระตะ. (2545). *หลักการเขียนโปรแกรมภาษา C.* ขอนแก่น: คลังนานาวิทยา.

มนตรี พจนารถลาวัลย์. (2521). *การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี.* กรุงเทพมหานคร: ซีเอ็ดยูเคชั่น.

มัณฑนา ปราการสมุทร. (2534). *การเขียนชุดคำสั่งภาษาซี.* กรุงเทพมหานคร: ดวงกมลสมัย.

วราภรณ์ โกวิทวรางกูร. (2544). *ระบบฐานข้อมูลและการออกแบบ.* กรุงเทพมหานคร: พิทักษ์อักษร.

สานนท์ เจริญฉาย. (2552). *การเขียนโปรแกรมและอัลกอริทึ่ม (กรณีตัวอย่างภาษาซี).* พิมพ์ครั้งที่ 8. กรุงเทพมหานคร: มหาจุฬาลงกรณราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2547). *คู่มือเรียนภาษาซี.* กรุงเทพมหานคร: ซีเอ็ดยูเคชั่น.

โอภาส เอี่ยมสิริวงศ์. (2546). *การวิเคราะห์และออกแบบระบบ.* กรุงเทพมหานคร: ซีเอ็ดยูเคชั่น.

Brian W. Kernighan, Dennis M. Ritchie. (1988). *The C Programming Language*. 2nd Edition. London: Prentice Hall.

Borland International, Inc. (1990). *Turbo C++ Version 1.00.*

CQUniversity. (2012). *Study Material*. Retrieved July 15, 2005, from http://webclass.cqu.au/Units/81120_FOCT_Hardware/Study_Material/Study_Guidechap1/chapter1.html.

Delores M. Etter. (2013). *Engineering Problem Solving with C.* 4th Edition. London: Prentice Hall.

Gary B. Shelly. (2003). *Systems Analysis and Design.* 5th Edition. USA: Tomson Learning.

James L. Antonakos, Kenneth C. Mansfield JR.(1998). *Structured C for Engineering and Technology.* London: Prentice Hall.

Harvey M. Deitel, Paul J. Deitel. (2004). *C How to program*. London: Prentice Hall.

Kris Jamsa. (2002). *Jamsa's C/C++/C# Programmer's Bible: The Ultimate Guide to C/C++/C# Programming.* 2nd Edition. USA: Onword Press.

Microsoft Corporation. (2014). *Naming Files, Paths and Namespaces*. Retrieved Jan 14, 2014, from https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx.

Stephen G. Kochan. (1994). *Programming in ANSI C Revised Edition.* USA: SAMS Publishing.

University of Rhode Island. (2014). *The CPU and Memory*. Retrieved April 5, 2014, from http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm.

ดรรชนี