เอกสารประกอบการสอน

โปรแกรมคอมพิวเตอร์สำหรับวิศวกร
Computer Programming for Engineers

รายวิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

วนายุทธ์ แสนเงิน

ภาควิชาวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม

สารบัญ

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 1 แนะนำเกี่ยวกับเนื้อหาในรายวิชาและการประยุกต์ใช้งาน

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจกรอบแนวคิดของโครงการสอน แผนการสอน และสามารถนำไปประยุกต์แนวทางในการเรียนของนักศึกษาในวิชานี้ได้
- เพื่อให้อาจารย์ผู้สอนและนักศึกษาได้ทำความคุ้นเคย พูดคุยทำความเข้าใจกับนักศึกษา
- เพื่อให้นักศึกษาได้ทำความเข้าใจ กฎระเบียบของมหาวิทยาลัย
- เพื่อให้นักศึกษาเข้าใจ พื้นฐานและหลักการทำงานของระบบคอมพิวเตอร์ ความสำคัญในการเรียนรู้และการนำไปประยุกต์ใช้งานในภาคอุตสาหกรรม

เนื้อหา

- กรอบแนวคิดของโครงการสอนและแผนการสอน เนื้อหาสาระในแต่ละสัปดาห์ของการเรียนการสอนและกฎระเบียบของมหาวิทยาลัย
- ภาษาคอมพิวเตอร์ ระบบควบคุม ระบบควบคุมอัตโนมัติในภาคอุตสาหกรรม

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- กล่าวต้อนรับและอธิบายความสำคัญของวิชาโปรแกรมคอมพิวเตอร์สำหรับวิศวกร
- บรรยายเนื้อหารายวิชาพร้อมยกตัวอย่างประยุกต์ใช้งานในภาคอุตสาหกรรม

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- ระบบ e-learning
- อธิบายประกอบบนกระดาน

การบรรยายสัปดาห์ที่ 1

แนะนำเกี่ยวกับเนื้อหาในรายวิชาและการประยุกต์ใช้งาน

Introduction to the Computer Programming for engineer and application

ในปัจจุบันคอมพิวเตอร์หรือระบบสมองกลอัจฉริยะ เป็นเครื่องมือที่จำเป็นในระบบควบคุมอัตโนมัติ การประมวลผลข้อมูลที่ซับซ้อน รวดเร็วเมื่อเทียบกับมนุษย์ ซึ่งเป็นการหลีกเลี่ยงไม่ได้ที่จะนำไปประยุกต์ใช้งานในภาคส่วนต่างๆ เช่น ในภาคอุตสาหกรรมการผลิต อุตสาหกรรมเกษตรแม่นยำ อุตสาหกรรมอาหาร อุตสาหกรรมเครื่องมือแพทย์ เป็นต้น เพื่อตอบสนองการทำงานของระบบควบคุมให้มีประสิทธิภาพมากยิ่งขึ้น ลดความผิดพลาดที่อาจจะเกิดขึ้นจากมนุษย์ ทั้งยังสามารถทำงานได้อย่างต่อเนื่อง

1.1 คอมพิวเตอร์อุตสาหกรรม

คอมพิวเตอร์ ถือว่าเป็นส่วนหนึ่งในเครื่องจักรกลที่ทำงานในระบบอัตโนมัติ มีการประมวลผลที่รวดเร็ว สามารถประมวลผลหลายๆอย่างพร้อมกันได้ ทั้งยังสามารถกำหนดไว้เป็นลำดับขั้นตอนและมีประสิทธิภาพสูงในการประมวลผลข้อมูลที่มีความซับซ้อนและทำงานได้อย่างต่อเนื่อง จึงทำให้คอมพิวเตอร์มีบทบาทสำคัญในทุกภาคส่วนรวมถึงในชีวิตประจำวัน ซึ่งปัจจุบันคอมพิวเตอร์เป็นการพัฒนาอย่างก้าวกระโดด ด้วยการพัฒนาเทคโนโลยีอย่างไร้ขีดจำกัดอย่างต่อเนื่อง จึงทำให้คอมพิวเตอร์มีประสิทธิภาพสูง ขนาดเล็กและเบา เช่น Embedded PC, Industrial PC, Smart Phone เป็นต้น

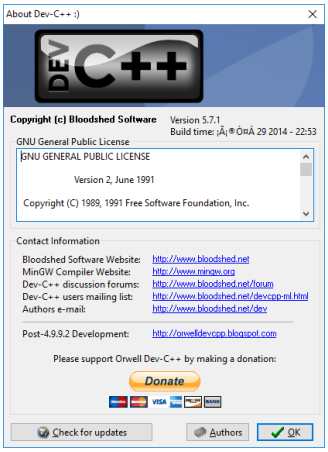
1.2 โปรแกรมคอมพิวเตอร์

โปรแกรมคอมพิวเตอร์ เป็นเครื่องมือที่สำคัญทำให้ระบบคอมพิวเตอร์สามารถทำงานได้ตามผู้ใช้งานหรือผู้พัฒนาโปรแกรมได้อย่างมีประสิทธิภาพ ถือได้ว่าเป็นการสร้างสมองกลให้คอมพิวเตอร์ทำงานได้ตามความต้องการ ภายใต้เครื่องมือของแต่ละภาษาคอมพิวเตอร์นั้นๆ ซึ่งมีหลากหลายภาษาที่ได้มีการพัฒนามาอย่างต่อเนื่อง เช่น C Language, Assembly Language, Python เป็นต้น ซึ่งการพัฒนาโปรแกรมสำหรับภาษาที่แตกต่างกันขึ้นอยู่กับความเชี่ยวชาญของผู้พัฒนาโปรแกรม

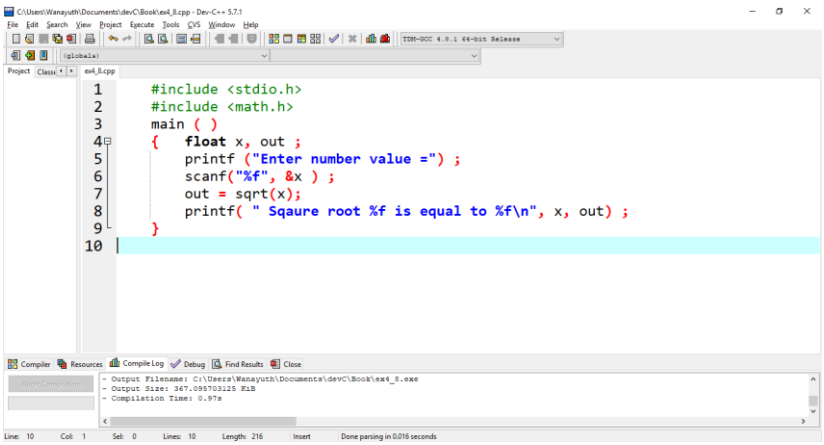
รูปที่ 1.1 ภาพรวมภาษาคอมพิวเตอร์ที่ประยุกต์ใช้งานในระบบคอมพิวเตอร์

https://www.quora.com/What-is-beautiful-programming-in-computer-programming

ในเนื้อหารายวิชาจะกล่าวถึงการใช้ภาษาซี ด้วยการใช้โปรแกรมคอมพิวเตอร์ DEV C++ ประกอบเนื้อหารายวิชา เพื่อให้เข้าใจถึงรูปแบบการใช้งาน โครงสร้างภาษาซี การเขียนคำสั่งต่างๆ รวมถึงการเรียกใช้ฟังก์ชั่นพื้นฐานในการติดต่อข้อมูลอินพุตและเอาต์พุต ทั้งยังฟังก์ชั่นทางคณิตศาสตร์ในการคำนวณและการแก้ไขปัญหาทางด้านวิศวกรรมด้วยโปรแกรมคอมพิวเตอร์



รูปที่ 1.2 โปรแกรมคอมพิวเตอร์ Dev C++



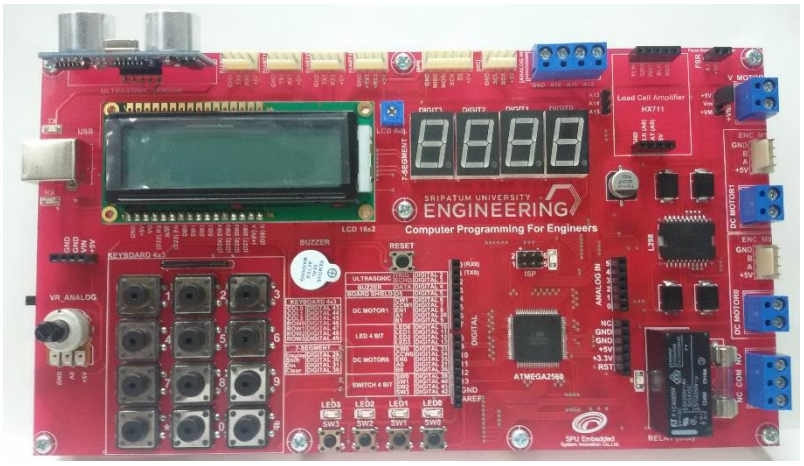
รูปที่ 1.3 Editor ของโปรแกรมคอมพิวเตอร์ Dev C++

โปรแกรม Dev C++ เป็น Free Software พัฒนาโดยบริษัท Bloodshed Co.,LTD สามารถพัฒนาโปรแกรมภาษาซี โดยใช้ editor ในการเขียนโปรแกรมตามโครงสร้างภาษาซี รวมถึงการใช้ฟังก์ชั่นต่างๆ ของภาษาซี สามารถดูข้อมูลและรายละเอียดได้ที่ http://www.cplusplus.com/reference/ ซึ่งมีฟังก์ชั่นที่หลากหลายในการนำไปพัฒนาโปรแกรมภายใต้ข้อกำหนดของโครงสร้างภาษาซีเท่านั้น หากต้องการนำไปเขียนภาษาอื่นๆ จะไม่สามารถรองรับการทำงาน

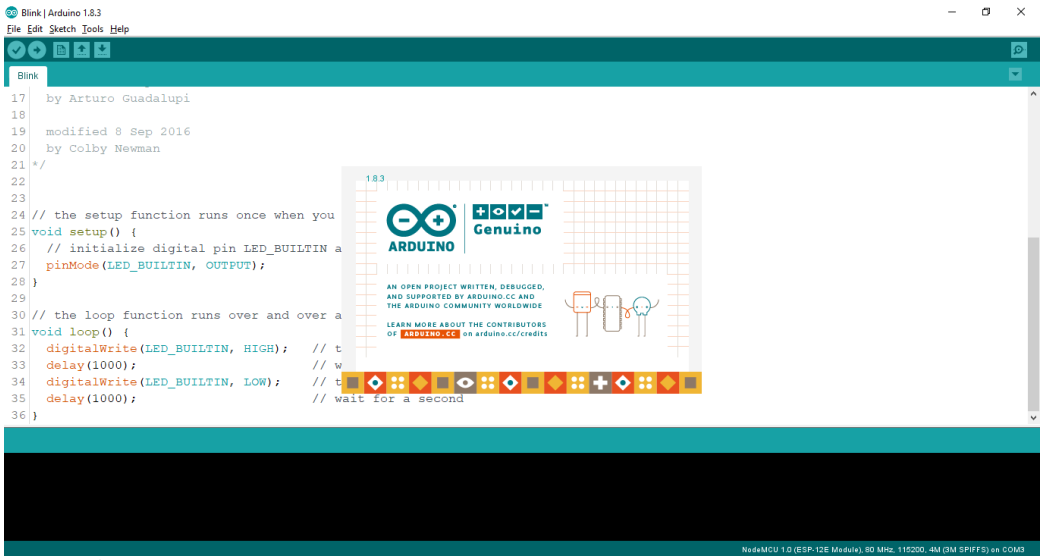
ภาษาซี ถือว่าเป็นภาษาที่มียืดหยุ่นในการพัฒนาโปรแกรมร่วมกับภาษาอื่นๆได้ และสามารถเข้าถึงอุปกรณ์ทางอินพุตและเอาต์พุตในระดับบิตเพื่อเชื่อมต่อการทำงานทางด้านอุปกรณ์ ทำให้เป็นที่นิยมในการพัฒนาโปรแกรมสำหรับการควบคุมผ่านช่องทางการสื่อสารอนุกรม(Serial Port, USB Port) การสื่อสารแบบขนาน (Parallel Port) ทั้งยังเป็นภาษาที่มีการพัฒนาต่อยอดนำไปสู่ภาษาคอมพิวเตอร์ในรูปแบบของโปรแกรมต่างๆ เช่น Visual C++, Visual C#, MATLAB, JAVA เป็นต้น โดยการนำโครงสร้างการเขียนโปรแกรมภาษาซีทำให้ใช้งานได้ง่ายขึ้น และรองรับการเชื่อมประสานสัมพันธ์กับผู้ใช้ผ่าน GUI (Graphic User Interfacing) สำหรับการพัฒนาโปรแกรมขั้นสูงต่อไป

1.3 อุปกรณ์อินพุตและเอาต์พุต

ด้วยรายวิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร ในภาคปฏิบัติได้พัฒนาชุดอุปกรณ์ฝึกการเขียนโปรแกรมสำหรับเชื่อมต่อกับอุปกรณ์อินพุตและเอาต์พุต แสดงดังรูปที่ 0.2 เพื่อให้เห็นถึงการนำความรู้ทางภาคทฤษฎีไปใช้งานภาคปฏิบัติจริง โดยใช้โปรแกรม Arduino สำหรับการพัฒนาร่วมภายใต้โครงสร้างของภาษาซี แสดงดังรูปที่ 0.3 เป็นโปรแกรม Free Software สามารถดาวน์โหลดได้ที่ https://www.arduino.cc/ โดยมีตัวอย่างการพัฒนาใช้งานอย่างแพร่หลาย



รูปที่ 1.4 ชุดอุปกรณ์ฝึกการเขียนโปรแกรมสำหรับเชื่อมต่อกับอุปกรณ์อินพุตและเอาต์พุต



รูปที่ 1.5 โปรแกรม Arduino สำหรับพัฒนาภาษาซีรวมกับอุปกรณ์อินพุตและเอาต์พุต

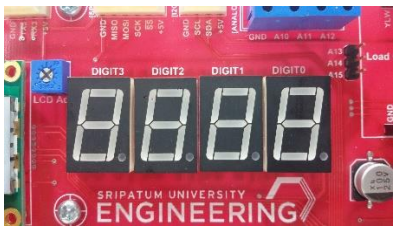
ด้วยบอร์ดไมโครคอนโทรลเลอร์ ถูกแบบเพื่อให้พัฒนาโปรแกรมภาษาซี สำหรับการเขียนโปรแกรมการเชื่อมต่ออุปกรณ์อินพุตและเอาต์พุต เป็นพื้นฐานของการทำงานในภาคอุตสาหกรรมอัตโนมัติ ภายในบอร์ดประกอบด้วย ไมโครคอนโทรลเลอร์ อุปกรณ์ทางอินพุตและเอาต์พุต วงจรอิเล็กทรอนิกส์ที่เกี่ยวข้องกับอุปกรณ์ซึ่งได้ทำการเชื่อมต่อแบบสมบูรณ์ จึงมุ่งเน้นให้พัฒนาโปรแกรมภาษาซีเพื่อเขียนโปรแกรมชุดคำสั่งต่างๆ ที่ใช้ในการสื่อสารข้อมูล ทั้งในรูปแบบดิจิตอลและแบบอนาล็อก เพื่อนำองค์ความรู้ที่ได้ไปพัฒนาต่อยอดในอนาคต ทั้งยังเป็นประยุกต์ใช้งานสมองกลฝั่งตัวสำหรับภาคอุตสาหกรรมอัตโนมัติ

ส่วนประกอบหลักของบอร์ดประกอบด้วย ตัวประมวลผลหลักด้วยไมโครคอนโทรลเลอร์ ชุดอุปกรณ์อินพุตและเอาต์พุต ซึ่งจะประกอบไปด้วยส่วนของอินพุต อาร์เรย์สวิตช์อินพุต 4x3 จำนวน แบบกดติดปล่อยดับ ตัวปรับค่าความต้านทาน (VR: Variable Resistance) และเซนเซอร์วัดระยะทาง (Ultrasonic Distance Sensor) และในส่วนของเอาต์พุตประกอบด้วย หน้าจอแสดงผล LCD จำนวน 16 อักษร 2 บรรทัด จอแสดงแบบ Seven segment และชุดอุปกรณ์ Relay สำหรับการควบคุมการเปิดปิด แสดงดังรูปที่ 1.6

(ก)

(ข)

(ค)



(ง)

(จ)

รูปที่ 1.6 อุปกรณ์อินพุตและเอาต์พุต (ก) อัลตร้าโซนิคเซนเซอร์วัดระยะ (ข) อาเรย์สวิตช์อินพุต 12 จำนวน (ค) หน้าจอแสดงผลแบบ LCD 2x16 อักษร (ง) จอแสดง Seven segment (จ) ชุดอุปกรณ์ Relay

1.4 การประยุกต์ใช้งาน

ด้วยองค์ความรู้การเขียนโปรแกรมภาษาซี และโปรแกรม Arduino แสดงให้เห็นถึงการนำไปประยุกต์ใช้งาน สร้างสรรค์สิ่งประดิษฐ์และนวัตกรรม ชื่อผลงาน “หุ่นยนต์ช่วยเดินสำหรับผู้ทุพพลภาพครึ่งท่อนล่าง” โดยทีมนักศึกษา ภาควิชาวิศวกรรรมไฟฟ้าและอิเล็กทรอนิกส์ประยุกต์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม

รูปที่ 1.7 ต้นแบบหุ่นยนต์ช่วยเดินสำหรับผู้ทุพพลภาพครึ่งท่อนล่าง

รูปที่ 1.8 เข้าร่วมนำเสนอผลงานในกิจกรรมต่างๆ

<u>รางวัลระดับประเทศ</u>

รางวัลชนะเลิศ, หัวข้อ“หุ่นยนต์ช่วยเดินสำหรับผู้ทุพพลภาพครึ่งท่อนล่าง” การประกวดผลงานด้านหุ่นยนต์เพื่อกระบวนการผลิต (RACMP2014) ประเภทเพื่อสังคม ประจำปี 2557

รางวัลชนะเลิศ, กลุ่มสิ่งประดิษฐ์เพื่อประโยชน์ทางการแพทย์และสาธารณสุข ประเภทอาชีวศึกษาและอุดมศึกษา หัวข้อ“หุ่นยนต์ช่วยเดินสำหรับผู้ทุพพลภาพครึ่งท่อนล่าง” โดยสำนักงานคณะกรรมการวิจัยแห่งชาติ (วช.) ประจำปี 2558

<u>ระดับนานาชาติ</u>

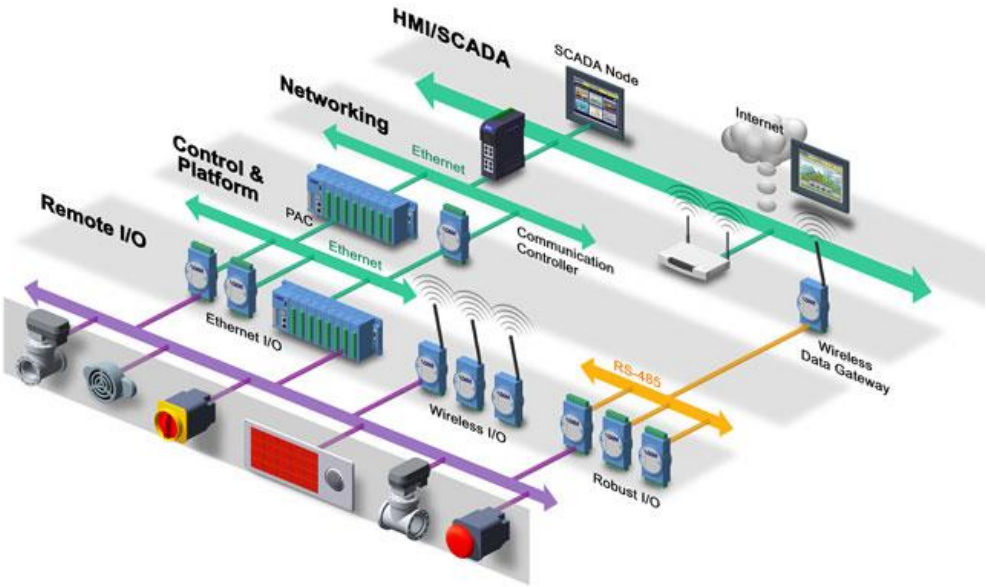
“Leading Innovation Award” by International Intellectual Property Network Forum (IIPNF) at Taipei International Invention Show & Technomart” (INST 2015) ณ กรุงไทเป ประเทศไต้หวัน

“Honorable Mention” by Taipei International Invention Show & Technomart” (INST) at Taipei International Invention Show & Technomart” (INST 2015) ณ กรุงไทเป ประเทศไต้หวัน

การประยุกต์ใช้งานระบบควบคุมอัตโนมัติของภาคอุตสาหกรรม ด้วยโปรแกรมคอมพิวเตอร์

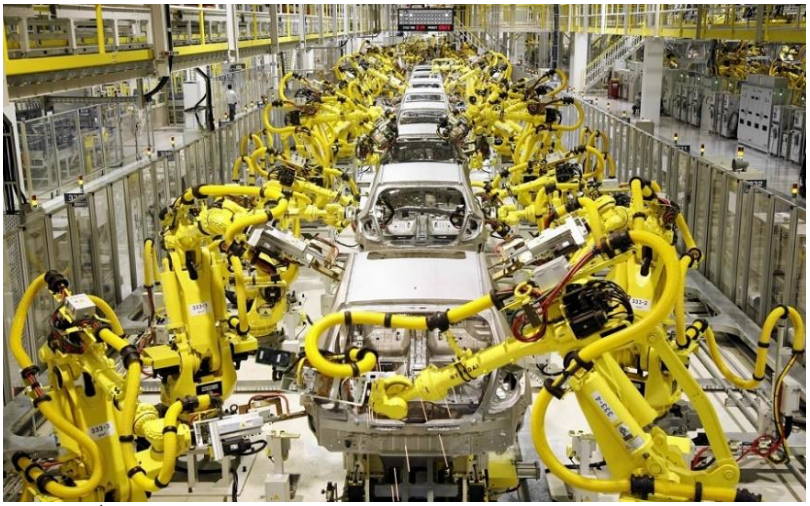
รูปที่ 1.9 โปรแกรมสำหรับควบคุมหุ่นยนต์อัตโนมัติในภาคอุตสาหกรรม

ที่มา: https://www.smashingrobotics.com/artiminds-rps-faster-intuitive-way-programming-industrial-robots/



รูปที่ 1.10 ระบบสมองกลฝังตัวเชื่อมต่ออุปกรณ์อินพุตเอาต์พุตและระบบ SCADA

ที่มา: http://www.advantech.eu/eAutomation/Remote-IO/Introduction.aspx

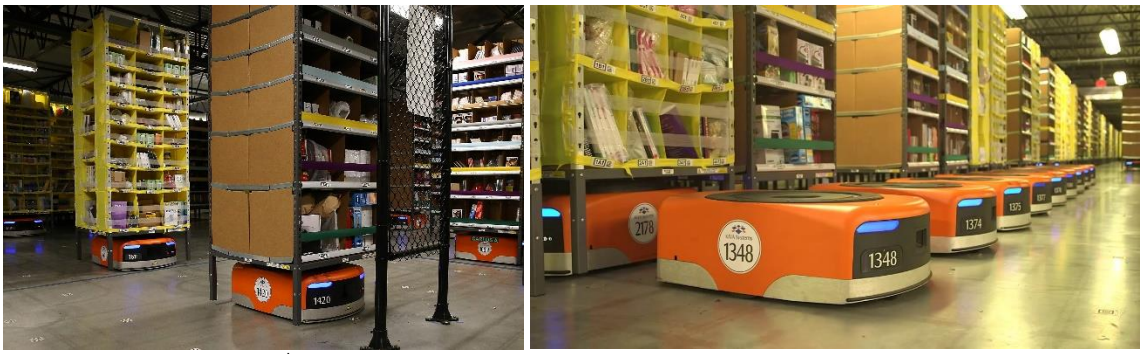


รูปที่ 1.11 หุ่นยนต์อัตโนมัติในภาคอุตสาหกรรมประกอบรถยนต์

ที่มา: http://www.metalworkingworldmagazine.com/growth-forecast-for-robotics-market-to-2020/

รูปที่ 1.12 หุ่นยนต์อัตโนมัติสำหรับระบบโลจิสติกส์

ที่มา:https://mycomputerlessons.com/exotec-debuts-autonomous-skypod-robots-in-french-warehouse/



รูปที่ 1.13 หุ่นยนต์อัตโนมัติสำหรับคลังสินค้า (Warehouse)

ที่มา: http://fortune.com/2016/03/24/americans-feel-more-pressured-by-job-competition-now-and-they-like-it/

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 2 ระบบคอมพิวเตอร์ และคอมไพเลอร์

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจ พื้นฐานและหลักการทำงานของระบบคอมพิวเตอร์ โครงสร้าง ส่วนประกอบอินพุตเอาต์พุต และคอมไพเลอร์

เนื้อหา

- พื้นฐานของระบบคอมพิวเตอร์ องค์ประกอบ อุปกรณ์อินพุตและเอาต์พุตของระบบคอมพิวเตอร์ รวมถึงกระบวนการเปลี่ยนภาษาคอมพิวเตอร์ให้เป็นภาษาเครื่อง

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความสำคัญระบบคอมพิวเตอร์ โครงสร้าง ส่วนประกอบอินพุตเอาต์พุต และคอมไพเลอร์
- บรรยายเนื้อหารายวิชาพร้อมยกตัวอย่างประกอบ

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

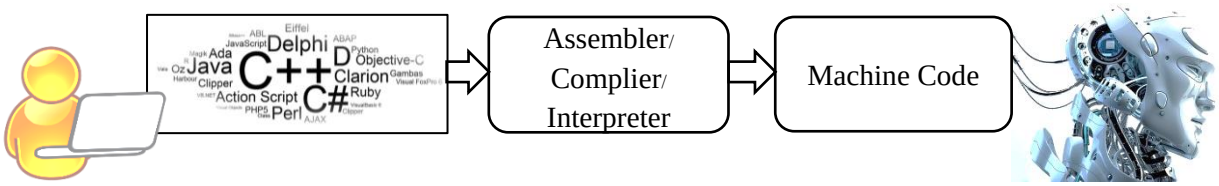
การบรรยายสัปดาห์ที่ 2

ระบบคอมพิวเตอร์และคอมไพเลอร์

Computer System and Compiler

ปัจจุบันการประมวลผลด้วยระบบคอมพิวเตอร์ เป็นส่วนหนึ่งในการใช้ชีวิตประจำวัน เป็นเสมือนเครื่องมืออำนวยความสะดวกที่จำเป็นต่อการดำรงค์ชีวิต เช่น สมาร์ทโฟน คอมพิวเตอร์แบบพกพา เป็นต้น รวมถึงการนำไปใช้ในการทำงานในด้านต่างๆ อย่างแพร่หลาย ทั้งในด้านธุรกิจเก็บข้อมูลสินค้า ระบบบัญชี การสื่อสารโทรคมนาคม การบันเทิง ด้านการควบคุมการผลิตของภาคอุตสาหกรรม เป็นต้น การเข้ามามีบทบาทของคอมพิวเตอร์นั้น ทำให้การทำงานมีประสิทธิภาพ ถูกต้อง รวดเร็ว ทดแทนการทำงานของมนุษย์

สำหรับงานด้านวิศวกรรมนั้นได้รับอิทธิพลจากความก้าวหน้าทางด้านเทคโนโลยีคอมพิวเตอร์เป็นอย่างมาก เนื่องจากงานด้านวิศวกรรมนั้นเป็นงานที่เกี่ยวข้องกันการคำนวณทางคณิตศาสตร์ และตรรกศาสตร์ โดยการประมวลผลด้วยคอมพิวเตอร์สามารถทำงานได้เป็นอย่างมีประสิทธิภาพสูง เช่น ระบบควบคุมอัตโนมัติ งานออกแบบและการวิเคราะห์ด้านวิศวกรรมศาสตร์ รวมถึงการจำลองระบบการทำงาน เป็นต้น ทำให้งานทางด้านวิศวกรรม มีประสิทธิภาพและแม่นยำสูงขึ้น เป็นส่วนช่วยในเพิ่มประสิทธิภาพการทำงานให้รวดเร็วขึ้น



รูปที่ 2.1 กระบวนการทำงานของภาษาคอมพิวเตอร์

2.1 ประวัติความเป็นมาของภาษาซี

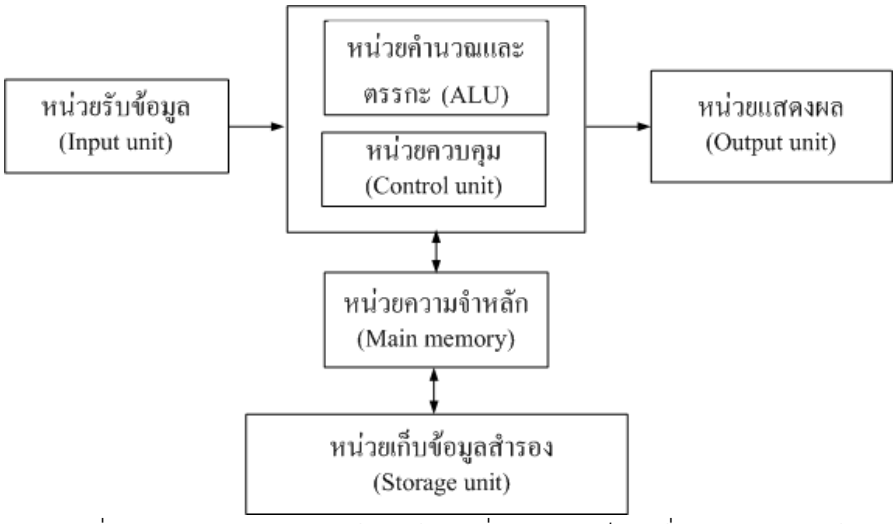
ภาษาซี (C Language) เกิดขึ้นจากการดัดแปลงภาษาบี (B Language) ภาษาซีเป็นภาษาที่คิดค้นโดย เดนนิส ริสซี่ (Dennis Ritchie) ที่ศูนย์วิจัย เบล (Bell Laboratories) ในสหรัฐอเมริการาวปี ค.ศ. 1972 เป็นภาษาที่ใช้เขียนระบบปฏิบัติการยูนิกส์ (UNIX Operating System) ซึ่งเป็นระบบปฏิบัติการในระบบเครือข่ายคอมพิวเตอร์ที่แพร่หลายในปัจจุบัน

ในช่วงการสร้างซีคอมไพเลอร์ (C compiler) มีการพัฒนาโปรแกรมและผลิตออกมาแข่งขันในท้องตลาดอย่างแพร่หลาย ทำให้รูปแบบข้อกำหนดเริ่มมีการผิดเพี้ยนไป จนกระทั่ง American National Standard Institute (ANSI) ได้กำหนดมาตรฐานกลางขึ้น ทำให้ส่วนประกอบหลักๆ ของ C compiler เป็นมาตรฐานเดียวกันทั้งหมด จะมีแตกต่างออกไปบ้างก็เป็นส่วนปลีกย่อยๆ ที่เป็นคุณสมบัติเฉพาะตัวของ C compiler นั้นๆ ดังนั้น มาตรฐานของภาษาซี รู้จักกันในชื่อของ “ANSI C”

ในช่วงต้นปี ค.ศ. 1980 ได้มีการพัฒนาภาษาระดับสูง (high-level language) ซึ่งเป็นอีกภาษาหนึ่งเรียกว่า ภาษาซีพลัสพลัส (C++ Language) ซึ่งถูกสร้างโดยนาย Bjarne Stroustrup ที่ห้องปฏิบัติการ Bell Laboratory เช่นกัน แต่เนื่องจากภาษา C++ ถูกพัฒนามาจากพื้นฐานภาษา C ดังนั้นคุณสมบัติต่างๆ จะมีความสอดคล้องกับภาษา C อย่างไรก็ตามภาษา C++ ไม่ใช่ส่วนเพิ่มเติมที่ต่อยอดจากภาษา C แต่ภาษา C++ ได้รวบรวมแนวทางรากฐานใหม่ ที่รู้จักกันในปัจจุบันว่า Object-oriented programming (OOP) ซึ่งทำให้ programmer มองทุกอย่างอยู่ในรูปของ object ทั้งหมด มีการถ่ายทอดพันธุกรรมที่มีโครงสร้างและหลักการที่แน่นอน ในช่วงเวลาต่อมาได้มีการพัฒนา C# เพื่อทำงานบน platform .Net ของบริษัทไมโครซอฟต์ ทำให้ผู้ใช้สามารถนำภาษา C ไปพัฒนาบน Microsoft platform ที่เป็นที่แพร่หลายในปัจจุบัน ทำให้ลดเวลาการพัฒนาซอฟท์แวร์ได้มาก

2.2 ส่วนประกอบของเครื่องคอมพิวเตอร์

คอมพิวเตอร์สามารถจำแนกองค์ประกอบหลักเป็น 2 ส่วนคือ ฮาร์ดแวร์ (Hard ware) หรือส่วนประกอบของตัวเครื่องคอมพิวเตอร์ และซอร์ฟแวร์(Software) หรือชุดคำสั่งที่บุคคลทั่วไปอาจเรียกว่าโปรแกรมคอมพิวเตอร์



รูปที่ 2.2 แสดงกลุ่มของฮาร์ดแวร์หลักที่ประกอบเป็นเครื่องคอมพิวเตอร์

2.2.1 หน่วยรับข้อมูล (Input) ทำหน้าที่ในการรับข้อมูลจากภายนอกระบบ เพื่อส่งให้ CPU ทำการประมวลผล เช่น แป้นพิมพ์ (Keyboard), เมาส์ (Mouse), สแกนเนอร์ (Scanner), ระบบปากกา (pen-based system) และ จอสัมผัส (Touch screen)

2.2.2 หน่วยแสดงผล (Output) ทำหน้าที่ส่งข้อมูลในรูปแบบต่างๆสู่ภายนอกระบบ ตัวอย่างเช่น จอภาพ (Monitor), เครื่องพิมพ์ (Printer) ลำโพง เป็นต้น

2.2.3 หน่วยประมวลผล (Central Processing Unit: CPU) ทำหน้าที่ในการคิดคำนวณ และประมวลผลข้อมูล ภายในจะประกอบด้วยสองส่วนหลักคือส่วนที่ทำหน้าที่คำนวณ เรียกว่า ALU (Arithmetic and Logic Unit) และส่วนที่ทำหน้าที่ประมวลผล/จัดการข้อมูลต่างๆ (Processing Unit)

2.2.4 หน่วยความจำ (Memory) หน่วยความจำแบ่งเป็นสองประเภทคือ หน่วยความจำหลัก (Main memory) หรือ แบบโวลาไทล์ (Volatile) หน่วยความจำกลุ่มนี้เป็นหน่วยความจำที่ทำงานร่วมกับ CPU ในการคำนวณและประมวลผล หน่วยความจำกลุ่มนี้มีความเร็วในการเข้าถึงข้อมูลสูงแต่มีพื้นที่ในการจัดเก็บน้อยกว่าเมื่อเทียบกับหน่วยความจำอีกลุ่มหนึ่งและเหตุที่เรียกว่า volatile (แปลว่า ลบเลือนได้) เพราะหน่วยความจำประเภทนี้จะไม่สามารถบันทึกข้อมูลไว้ เมื่อไม่มีการจ่ายกระแสไฟฟ้ามาเลี้ยงวงจร ตัวอย่างหน่วยความจำที่จัดอยู่ในประเภทนี้คือ แรม (RAM)

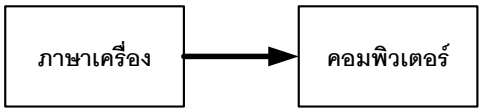
2.2.5 หน่วยเก็บข้อมูลสำรอง (Storage unit) ที่ทำหน้าที่เก็บข้อมูล เนื่องจากหน่วยความจำหลักไม่สามารถเก็บข้อมูลไว้ได้นาน หรือเมื่อปิดไฟข้อมูลก็จะหายไป จึงจำเป็นต้องมีหน่วยเก็บข้อมูลสำรองมาบันทึกข้อมูล เช่น ฮาร์ดดิสก์ (Hard disk), CD-ROM เป็นต้น

2.3 ภาษาสั่งงานเครื่องคอมพิวเตอร์

ภาษาสั่งงานคอมพิวเตอร์ หมายถึง ชุดคำสั่งที่เขียนขึ้นตามรูปแบบและโครงสร้างของภาษา เพื่อสั่งงานให้คอมพิวเตอร์ทำงานตามชุดคำสั่ง ภาษาสั่งงานคอมพิวเตอร์สามารถจำแนกออกได้ 3 ระดับดังนี้

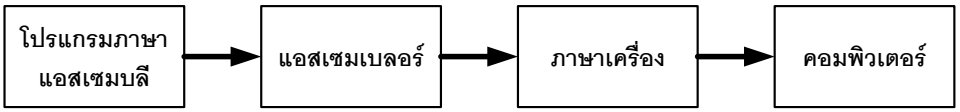
2.3.1 ภาษาระดับล่าง (Low Level Language) เป็นภาษาที่คอมพิวเตอร์สามารถเข้าใจคำสั่งได้ง่ายแต่มนุษย์เข้าใจได้ยากใช้เวลาในการศึกษาเพื่อเขียนโปรแกรมนานและต้องเข้าใจหลักการทำงานของฮาร์ดแวร์ ภาษาระดับล่างสามารถติดต่อกับฮาร์ดแวร์ได้ดี ทำให้ทำงานอย่างรวดเร็ว ซึ่งภาษาระดับล่าง มีอยู่ 2 ภาษาคือ

- **ภาษาเครื่อง (Machine Language)** เป็นชุดคำสั่งที่อยู่ในรูปแบบเลขฐานสองติดต่อกับฮาร์ดแวร์ได้โดยตรง คอมพิวเตอร์สามารถเข้าใจคำสั่งได้ทันทีโดยไม่ต้องใช้ตัวแปลภาษา ทำให้คอมพิวเตอร์สามารถทำงานได้อย่างรวดเร็ว แต่มนุษย์เข้าใจยากและใช้เวลาในการเขียนนานมาก



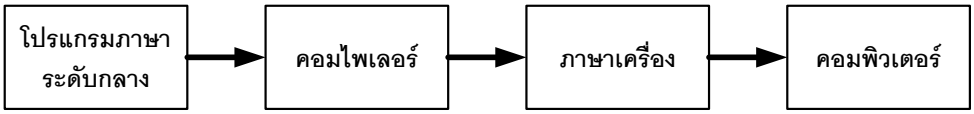
รูปที่ 2.3 ไดอะแกรมแกรมการทำงานของภาษาเครื่อง

- ภาษาแอสเซมบลี (Assembly Language) เป็นภาษาที่อยู่ในรูปแบบของชุดคำสั่งสั้นๆ มนุษย์เข้าใจได้ง่ายกว่าภาษาเครื่อง แต่ก็ยังเข้าใจได้ยากกว่าภาษาดับกลางและระดับสูง ภาษาแอสเซมบลี สามารถทำงานได้เร็ว การติดต่อทางฮาร์ดแวร์ทำได้ดี และการสั่งงานให้คอมพิวเตอร์ทำงานต้องมีการแปลความหมายให้เป็นภาษาเครื่องก่อนโดยใช้ตัวแปลภาษาที่เรียกว่า แอสเซมเบอร์ (Assembler)



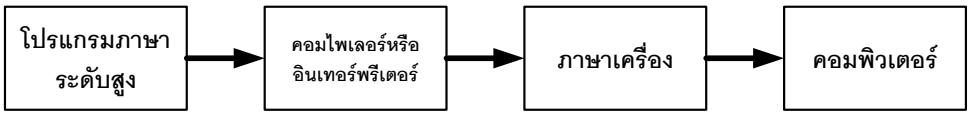
รูปที่ 2.4 ไดอะแกรมแกรมการทำงานของภาษาแอสเซมบลี

2.3.2 ภาษาระดับกลาง (Medium Level Language) เป็นภาษาที่มีลักษณะผสมกันระหว่างภาษาระดับสูงกับระดับล่าง คือมีลักษณะของคำสั่งคล้ายประโยคทางภาษาอังกฤษ แล้วยังมีบางคำสั่งไปคล้ายภาษาระดับล่าง ซึ่งสามารถทำงานได้เร็วใช้เวลาในการศึกษาและเขียนโปรแกรมน้อยกว่าภาษาระดับล่าง การสั่งงานให้คอมพิวเตอร์ ทำงานต้องมีการแปลความหมายเป็นภาษาเครื่องก่อนโดยใช้ตัวแปลภาษาที่เรียกว่าคอมไพเลอร์ (Compiler) ตัวอย่างของภาษาระดับกลางตัวอย่างเช่น ภาษาซี เป็นต้น



รูปที่ 2.5 ไดอะแกรมแกรมการทำงานของภาษาระดับกลาง

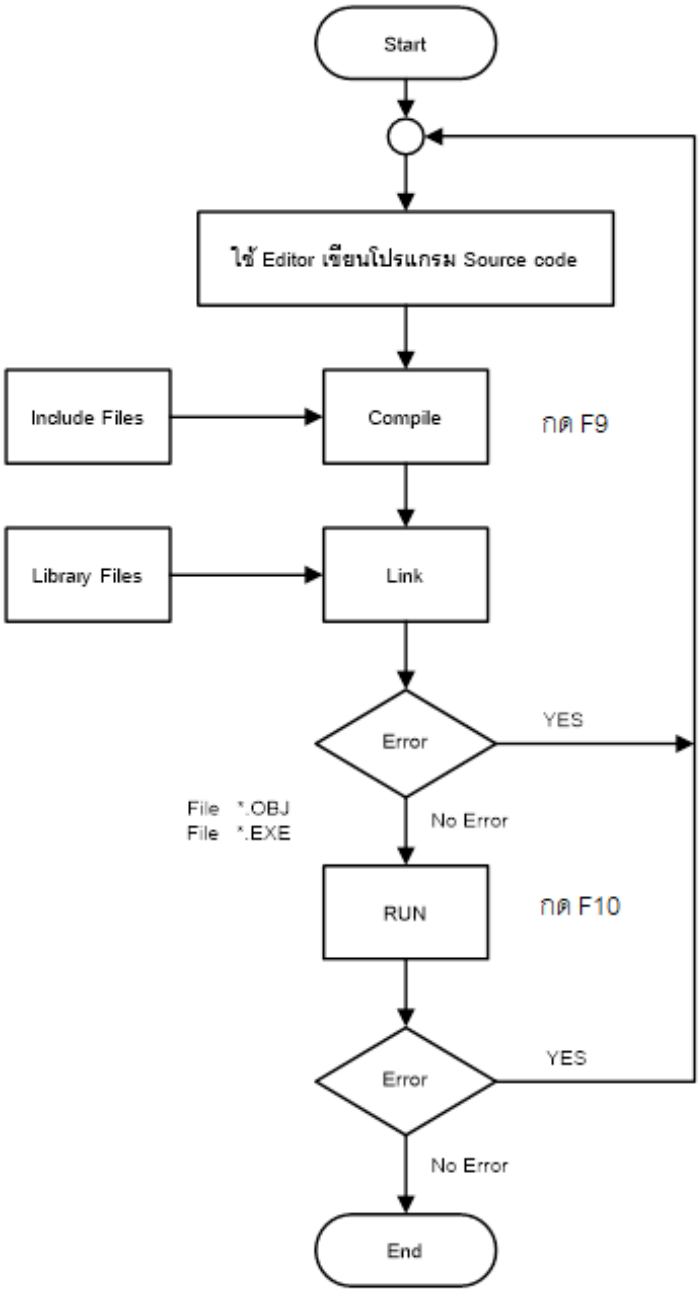
2.3.3 ภาษาระดับสูง (High-Level Language) เป็นภาษาที่สามารถศึกษาและเข้าใจได้ง่ายมีลักษณะของคำสั่งคล้ายกับประโยคทางภาษาอังกฤษ ซึ่งง่ายต่อการทำความเข้าใจและใช้เวลาในการเขียนโปรแกรมน้อยแต่การสั่งงานให้คอมพิวเตอร์ทำงานได้ช้ากว่าและสั่งงานได้เพียงบางส่วนของคอมพิวเตอร์เท่านั้น การสั่งงานให้คอมพิวเตอร์ทำงานต้องมีการแปลความหมายให้เป็นภาษาเครื่องก่อนโดยใช้ตัวแปลภาษาที่เรียกว่าอินเทอร์พรีเทอร์ (Interpreter) หรือคอมไพล์เลอร์ (Compiler) ตัวอย่างเช่น ภาษาเบสิค (BASIC) ปาสคาล (PASCAL) และดีเบส (Dbase) เป็นต้น



รูปที่ 2.6 ไดอะแกรมแกรมการทำงานของภาษาระดับสูง

2.4 การประมวลผลของคอมไพล์เลอร์ภาษาซี

การทำงานของภาษาซีเริ่มจากใช้ Editor เขียนโปรแกรม (Source Code) ซึ่งต้องเขียนตามลักษณะโครงสร้างของภาษาซี แล้วบันทึก (Save) เป็นไฟล์ที่มีนามสกุลเป็น *.C (File *.C) จากนั้นทำการคอมไพล์ถ้ามีข้อผิดพลาด (Error) ให้กลับไปแก้ไขที่โปรแกรมแต่ถ้าไม่มีข้อผิดพลาดจะได้ไฟล์ .OBJ แล้วทำการ Link ไฟล์ *.OBJ เข้ากับไฟล์ Libraries ถ้าไม่มีข้อผิดพลาดจะได้ไฟล์ *.EXE ขั้นตอนการทำงานของคอมไพล์เลอร์แสดงอยู่ในรูปที่ 2.7



รูปที่ 2.7 ลำดับขั้นตอนการทำงานของคอมไพล์เลอร์

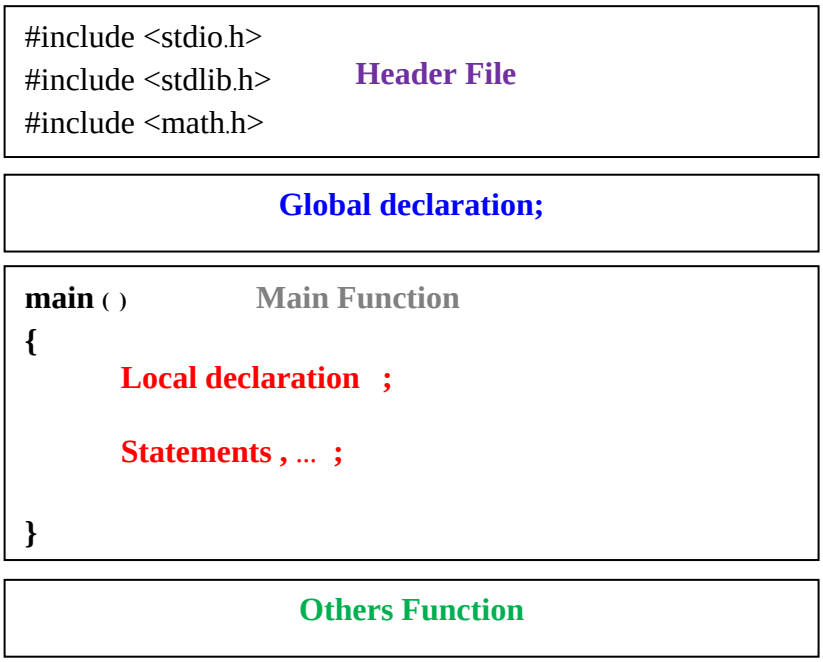
2.5 โครงสร้างในการเขียนโปรแกรมภาษาซี

ไฟล์ที่เขียนในภาษาซี ต้องประกอบด้วยการเขียนโปรแกรมสามารถสร้างไฟล์แบ่งเป็น 2 กลุ่มหลักๆ คือ เฮดเดอร์ไฟล์ (Header files) และซอร์ซไฟล์ (Source File)

Header Files เป็นการสร้างไฟล์นามสกุล .h สำหรับการเขียนโปรแกรมการเรียกใช้งานคำสั่งต่างๆ ที่ผู้เขียนสามารถออกแบบการทำงานได้อย่างอิสระรวมถึงการสร้างไฟล์เฉพาะเจาะจงการทำงานได้ โดยโปรแกรม Source File อื่นๆ สามารถเรียกใช้งานได้

Source File เป็นการสร้างไฟล์นามสกุล .cpp สำหรับการเขียนโปรแกรมที่ต้องการ โดยเรียกใช้คำสั่งต่างๆ จากไฟล์ .h เพื่อใช้คำสั่งต่างๆที่อยู่ภายในเฮดเดอร์ไฟล์นั้นๆ รวมถึงการทำงานรวมกับฮาร์ดแวร์ที่เป็นทั้งอินพุตและเอาต์พุตของระบบคอมพิวเตอร์

ทั้งนี้การเขียนโปรแกรมจำเป็นต้องคำนึกถึงรูปแบบโครงสร้างของภาษาซี แสดงดังรูปที่ 2.8 รวมถึงการจำแนกโครงสร้าง ขั้นตอนการสร้างไฟล์ โดยในเนื้อหานี้จะได้ศึกษาการเขียนโปรแกรม Source ไฟล์ที่มีนามสกุลเป็น .cpp



รูปที่ 2.8 โครงสร้างซอร์ซไฟล์ (Source File) ของภาษาซี

2.5.1 โปรแกรมส่วนพรีโพรเซส (Preprocessor)

โปรแกรมส่วนนี้เป็นส่วนที่ทำหน้าที่ในการติดต่อกับไฟล์ที่มีส่วนขยายเป็น ***.h** หรือไลบรารี (Library) ต่างๆ ของภาษาซี เพื่อให้สามารถใช้คำสั่งต่างๆ สำหรับไฟล์ของไลบรารี ซึ่งสามารถเปรียบเสมือนได้กับการจัดเตรียมกล่องเครื่องมือให้ตรงกับประเภทของงานที่จะทำนั่นเอง ไฟล์ต่างๆที่เก็บชุดคำสั่งมาตรฐานจะมีนามสกุลเป็น **.h** เช่น **stdio.h** เป็นไฟล์ที่ใช้เก็บไลบรารีมาตรฐานเกี่ยวกับการรับข้อมูลและการแสดงผล ดังเช่นฟังก์ชั่น **printf( );** ซึ่งได้ถูกนิยามไว้ใน **stdio.h** ดังนั้นการเขียนโปรแกรมครั้งใดที่มีการเรียกใช้ ฟังก์ชั่น **printf( );** จึงต้องทำการประกาศไฟล์ **stdio.h** เพื่อใช้รวมในการคอมไพล์โปรแกรม นอกจากนี้ผู้เขียนโปรแกรมสามารถสร้างไลบรารี ของตัวเองได้เช่นกัน ซึ่งเป็นการเขียนโปรแกรมระดับที่สูงขึ้น

2.5.2 ส่วนโปรแกรมหลัก (Main program)

โปรแกรมหลักคือส่วนที่จะทำงานต่างๆตามที่ผู้เขียนได้วางแผน ออกแบบและเขียนขึ้น โดยใช้ชุดคำสั่งต่างๆตามความเหมาะสมและตามที่ได้ประกาศไว้ในส่วนพรีโพรเซส โปรแกรมส่วนนี้สามารถสังเกตุได้ง่ายคือจะเริ่มต้นด้วยฟังก์ชั่น main ( ) ซึ่งเป็นการเริ่มต้นฟังก์ชั่นหลักของโปรแกรม และมีเครื่องหมาย ปีกกาเปิด **{** เป็นเครื่องหมายเริ่มต้นการเขียนโปรแกรม และเครื่องหมายปีกกาปิด **}** เป็นเครื่องหมายการสิ้นสุดโปรแกรม

ภายในฟังก์ชั่น main( ) จะประกอบไปด้วยชุดคำสั่งและฟังก์ชั่นต่างๆ ซึ่งเกือบทุกบรรทัดจะสิ้นสุดด้วยเครื่องหมายเซมิคอลอน (;) เพื่อเป็นสัญญลักษณ์ให้คอมไพเลอร์แยกคำสั่งในแต่ละชุดออกจากกันได้

การเขียนโปรแกรมภาษาซีจะเขียนด้วนตัวอักษรภาษาอังกฤษพิมพ์เล็ก คอมไพล์เลอร์ภาษาซีจะพิจารณาตัวอักษรพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่เป็นอักษรที่แตกต่างกัน (Case sensitive)

โปรแกรมตัวอย่างที่ 1	คำอธิบาย
/*Example program */	ส่วนของคำอธิบาย ไม่มีผลต่อการโปรแกรม
#include <stdio.h>	ประกาศการเรียกใช้คำสั่งจากไฟล์ stdio.h
main( )	ชื่อฟังก์ชั่นหลัก เริ่มต้นส่วนหลักของโปรแกรม
{	ปีกกาเปิด เริ่มต้นการเขียนโปรแกรม
printf("********************\n");	คำสั่งให้เครื่องแสดงข้อความใน “ ”
printf("Welcome To EGR205\n");	คำสั่งให้เครื่องแสดงข้อความใน “ ”
printf("********************\n");	คำสั่งให้เครื่องแสดงข้อความใน “ ”
}	ปีกกาปิด จบโปรแกรม

<u>สรุปท้ายบท</u>

ภาษาซีเป็นการเปลี่ยนภาษาคอมพิวเตอร์เป็นภาษาเครื่อง หรือแมตซีนโค้ด (Machine code) ด้วยตัวคอมไพเลอร์ภาษาซี เพื่อให้คอมพิวเตอร์เข้าใจในโปรแกรมภาษาซีที่สร้างขึ้น ทำให้การประมวลผลคอมพิวเตอร์เข้าใจในรูปแบบการทำงาน โดยโครงสร้างของการเขียนโปรแกรมภาษาซี จะต้องประกอบด้วยส่วนสำคัญคือ Header file และ Main function หากไม่มีสองส่วนนี้โปรแกรมจะไม่สามารถทำงานได้

ในการพัฒนาโปรแกรมให้มีประสิทธิภาพขึ้นอยู่กับนำชุดฟังก์ชั่นหรือคำสั่งต่างๆ ไปประยุกต์ใช้งานในส่วนงานการคำนวณ วิเคราะห์ ประมวลผลสมการต่างๆ ทางด้านวิศวกรรมศาสตร์ ด้วยกระบวนการและภายใต้โครงสร้างของภาษาซี

คำถามท้ายบท

1. ส่วนประกอบของคอมพิวเตอร์มีอะไรบ้าง?
2. ระบบคอมพิวเตอร์ที่ใช้ในการประมวลผลแตกต่างจากหน่วยความจำอย่างไร?
3. ภาษาคอมพิวเตอร์แบ่งเป็นกี่ประเภท อะไรบ้าง?
4. โปรแกรมส่วนพรีโพรเซส (Preprocessor) หมายถึงอะไร?
5. Header File หมายถึงอะไร?
6. Source File หมายถึงอะไร?
7. โปรแกรมหลัก main ทำหน้าที่อะไร?
8. เครื่องหมายเซมิโคลอน (;) มีความสำคัญอย่างไร?
9. เครื่องหมายที่ใช้ในการเริ่มต้นการทำงานและจบการทำงานของโปรแกรมหลัก คือ?
10. Global Declaration และ Local Declaration มีความแตกต่างกันอย่างไร?

แบบฝึกหัดท้ายบท

1. จงอธิบายการทำงานของคอมไพเลอร์ภาษาซี พร้อมทั้งวาดรูปประกอบ
2. จงจำแนกและยกตัวอย่างอุปกรณ์อินพุตและเอาต์พุต ที่ใช้ในระบบคอมพิวเตอร์ พร้อมคำอธิบายประกอบ
3. จงเขียนคำสั่งสำหรับเรียกใช้งาน Header File พร้อมยกตัวอย่างประกอบ
4. จงเขียนผังการทำงานของคอมไพเลอร์ภาษาซี
5. จงเขียนโครงสร้างซอร์ซไฟล์ (Source File) ของภาษาซี

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 3 รูปแบบการทำงานของโปรแกรม การเขียนเพอร์ซูโด-โค้ดและการเขียนผังงาน

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาโครงสร้างการทำงานของโปรแกรมภาษาซี
- เพื่อให้นักศึกษาเข้าใจการเขียนลำดับการทำงานของโปรแกรมด้วยเพอร์ซูโด-โค้ด
- เพื่อให้นักศึกษาเข้าใจหลักการทำงานของระบบคอมพิวเตอร์ด้วยผังงาน

เนื้อหา

- ขั้นตอนการประมวลผลของระบบคอมพิวเตอร์
- การเขียนเพอร์ซูโด-โค้ด และสัญลักษณ์ของผังงาน
- วิธีการแก้ไขปัญหาโจทย์ทางคณิตศาสตร์ ด้วยการเขียนเพอร์ซูโด-โค้ด การเขียนผังงาน

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความสำคัญของการเขียนโปรแกรมด้วยเพอร์ซูโด-โค้ด รวมถึงผังงาน
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

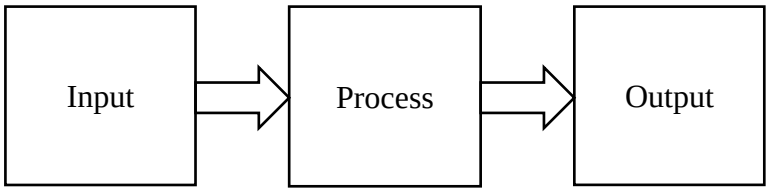
- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 3

รูปแบบการทำงานของโปรแกรม การเขียนเพอร์ซูโด-โค้ดและการเขียนผังงาน
Computer Programming Concept, Pseudo-code and Flow Chart

ในการแก้ปัญหาทางวิศวกรรมโดยเขียนโปรแกรมคอมพิวเตอร์นั้นมีความจำเป็นอย่างยิ่งที่ผู้ที่ทำการเขียนโปรแกรมจะต้องทำความเข้าใจหลักการและแนวคิดในการวิเคราะห์ปัญหา เพื่อให้สามารถแก้ไขปัญหาให้ตรงตามความต้องการให้ถูกต้อง สามารถเข้าใจแนวทางการแก้ปัญหาที่ผ่านกระบวนการวิเคราะห์อย่างเป็นระบบ และตามลำดับขั้นตอนที่อย่างถูกต้องอย่างเหมาะสมได้

ขั้นตอนการประมวลผลสำหรับการวิเคราะห์และการคำนวณทางวิศวกรรม เพื่อให้จำแนกพารามิเตอร์หรือระบุกระบวนการทำงาน คำสั่งการคำนวณต่างๆ ประกอบด้วยส่วนของอินพุต (Input) โพรเซส (Process) และ เอาท์พุต (Output) แสดงดังรูปที่ 3.1 ในส่วนของอินพุตหมายถึงการกำหนดค่าข้อมูล ระบุค่าจากผู้ใช้งานหรือได้ค่าข้อมูลจากอุปกรณ์ทางด้านอินพุตต่างๆ โดยนำข้อมูลอินพุตมาใช้ในส่วนของการดำเนินการตามคำสั่งเพื่อประมวลผลคำสั่งต่างๆ รวมถึงการคำนวณตามสมการทางด้านคณิตศาสตร์ และส่งผลลัพธ์การทำงานแสดงผลในรูปแบบต่างๆ เช่น แสดงผลทางจอภาพ รวมถึงอุปกรณ์ทางด้านเอาท์พุทต่างๆ เพื่อให้ผู้ใช้งานสามารถทราบถึงการทำงานอย่างเป็นลำดับขั้นตอน



รูปที่ 3.1 ระบบประมวลผลการทำงานของโปรแกรมคอมพิวเตอร์

3.1 การเขียนเพอร์ซูโด-โค้ด

พื้นฐานการเขียนโปรแกรมสามารถเขียนเริ่มต้นจากการเขียน เพอร์ซูโด-โค้ด (Pseudo-code) หรือการเขียนตามลำดับความคิดเป็นประโยคที่เข้าใจง่าย กระชับ มีใจความสำคัญ ซึ่งเปรียบเสมือนภาษาสื่อสารกับกับคอมพิวเตอร์ด้วยภาษามนุษย์ เพื่อให้จัดรูปแบบหรือเรียบเรียงลำดับขั้นตอนให้เข้าใจง่าย

โดยเป็นการเขียนลำดับขั้นตอนเพื่อให้สามารถเขียนผังการทำงานได้ง่ายขึ้น เมื่อเข้าใจรูปแบบลำดับขั้นตอนสามารถเขียนผังงานได้อย่างถูกต้องและสอดคล้องกับการเขียน Pseudo-code

3.2 การเขียนผังงาน

ผังงาน (Flow Chart) เป็นที่นิยมใช้เพื่อการถ่ายทอดแนวคิดหรือช่วยในการออกแบบโปรแกรม ดังนั้น นักศึกษาจึงจำเป็นต้องศึกษาหลักการเขียน และการนำผังงานไปประยุกต์ใช้ในการช่วยออกแบบโปรแกรม ก่อนที่จะลงมือเขียนโปรแกรม ขอให้นักศึกษาให้ความสำคัญกับการเขียนผังงาน เนื่องจากการเขียนโปรแกรมคอมพิวเตอร์ที่ดีจำเป็นต้องมาจากการออกแบบที่มีความสมบูรณ์

การเขียนผังงาน หมายถึงการเขียนแผนภาพที่เป็นลำดับเพื่อแสดงการทำงานของโปรแกรมหรือระบบงานอย่างใดอย่างหนึ่งเพื่อให้ง่ายต่อการเข้าใจและการแก้ไขปัญหา การเขียนผังงาน ควรเขียนจากบนลงล่าง โดย มีจุดเริ่มต้น และจุดสิ้นสุดจุดเดียว

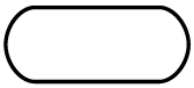
ความหมายของผังงาน

ผังงาน (Flowchart) คือ รูปภาพหรือรูปเขียนสัญลักษณ์ (Symbol) ที่มีความหมายที่ใช้เขียนแทนขั้นตอน คำอธิบาย ข้อความประโยค หรือคำพูด เพื่อให้การนำเสนอขั้นตอนของระบบงานให้เข้าใจตรงกันระหว่างผู้ปฏิบัติงานที่มีความแตกต่างด้านภาษาสื่อสาร โดยทดแทนด้วยสัญลักษณ์ เช่น การเขียนผังงานระบบควบคุมไฟฟ้า การเขียนผังงานระบบสมองกล การเขียนผังงานของอัลกอริทึม (Algorithm) สำหรับปัญญาประดิษฐ์ เป็นต้น ซึ่งหากใช้ภาษาสื่อสารระหว่างผู้เกี่ยวข้อง ด้วยคำพูดหรือข้อความ ยุ่งยากกว่าเมื่อใช้รูปภาพหรือสัญลักษณ์

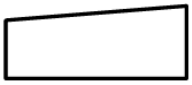
ประโยชน์ของผังงาน

1) ทำให้เข้าใจ และแยกแยะปัญหาได้ง่าย (Problem Defined)
2) แสดงลำดับการทำงาน (Step Flow)
3) หาข้อผิดพลาดได้ง่าย (Easy to Debug)
4) ทำความเข้าใจโปรแกรมได้ง่าย (Easy to Read)
5) ไม่ขึ้นกับภาษาใดภาษาหนึ่ง (Flexible Language)

3.3 การเขียนสัญลักษณ์พื้นฐานของผังงาน

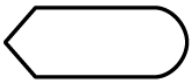


หมายถึงจุดเริ่มต้น และจุดสิ้นสุด (Start / End)

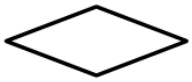


หมายถึง การรอรับข้อมูลจากผู้ใช้ (Manual input) จากทางด้านคีย์บอร์ด

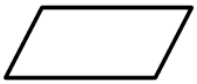
หมายถึง การกำหนดค่า การคำนวณและการประมวลผลต่างๆ (Process)



หมายถึง การแสดงผลต่างๆ (Display) ออกทางจอภาพมอนิเตอร์

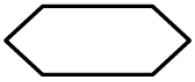


หมายถึง การตรวจสอบเงื่อนไข และการเปรียบเทียบหรือการตัดสินใจ (Decision)



หมายถึง กล่องรับหรือแสดงข้อมูลโดยไม่ระบุชนิดของอุปกรณ์

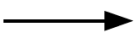
หมายถึง กล่อง predefined process หรือการเรียกใช้ฟังก์ชั่นย่อย (Sub function)



หมายถึง กล่องประกาศ การกำหนดค่า , ตัวแปร , ส่วนประกอบต่างๆ

หมายถึง จุดต่อเชื่อมภายในหน้า (Connection)

หมายถึง จุดต่อเชื่อมระหว่างหน้า (Off-page reference)



หมายถึง ทิศทางการทำงาน (Direction)

3.4 โครงสร้างการเขียนผังงาน

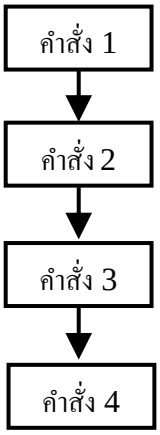
การเขียนรูปแบบโครงสร้างการเขียนผังงาน มีลำดับขั้นตอนตามรูปแบบการเขียนผังงาน และสามารถทำให้การออกแบบผังงานเข้าใจง่ายขึ้นและเป็นระเบียบแบบแผน เพื่อทำให้การตรวจสอบขั้นตอนการทำงานทำได้ง่ายและเข้าใจภาพรวมของระบบ ซึ่งมีรูปแบบโครงสร้างการเขียนผังงาน จำนวน 3 รูปแบบ ดังนี้

1) แบบลำดับขั้น (Sequence)
2) แบบทางเลือก (Selection)
3) แบบการทำซ้ำ (Iteration)

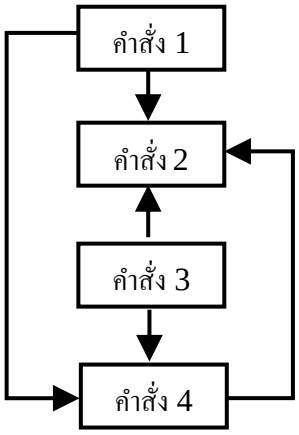
โดยรูปแบบผังงานดังกล่าวสามารถนำมาผสมผสานรวมกันได้ เพื่อออกแบบผังงานได้อย่างเหมาะสมกับโปรแกรม จนถึงรูปแบบผังงานที่ซับซ้อนยิ่งขึ้น ง่ายต่อการเข้าใจมากกว่าการเขียนโปรแกรมภาษาซี

3.4.1 การเขียนผังงานแบบลำดับขั้น (Sequence)

การเรียงเป็นลำดับขั้นตอนของผังงาน และทิศทางของการทำงาน ด้วยลูกศรชี้ตามลำดับเพื่อให้การเขียนเป็นลำดับ เช่น จากคำสั่งที่ 1 เมื่อเสร็จสิ้นให้ไปทำคำสั่งที่ 2 เมื่อเสร็จสิ้นให้ทำคำสั่งที่ 3 และคำสั่งที่ 4 ตามลำดับ แสดงดังรูปที่ 3.2 การเขียนทิศทางที่ทำให้ลำดับการทำงานผิดพลาด ทำให้การเขียนผังงานไม่เป็นไปตามรูปแบบลำดับขั้น แสดงดังรูปที่ 3.3



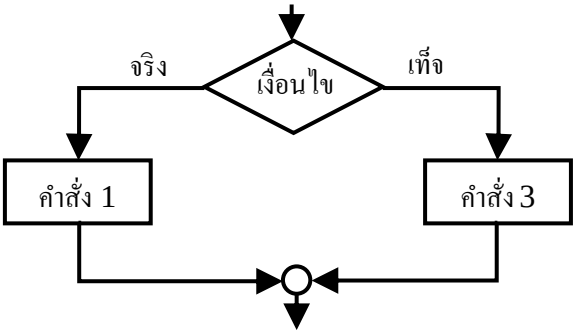
รูปที่ 3.2 ลำดับผังการทำงานที่ถูกต้อง



รูปที่ 3.3 ลำดับผังการทำงานที่ไม่ถูกต้อง

3.4.2 การเขียนผังงานแบบทางเลือก (Selection)

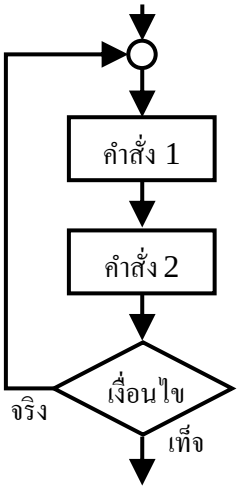
การเขียนผังงานแบบทางเลือกจะใช้ในส่วนของเงื่อนไข หรือเรียกว่าการตัดสินใจ ซึ่งการเขียนผังงานให้สามารถกำหนดทางเลือกภายใต้เงื่อนไข 2 กรณี คือ กรณีเงื่อนไขเป็นจริงและเงื่อนไขเป็นเท็จ รวมถึงทิศทางของตัวผังงานจะดำเนินการทำในส่วนต่างๆ ตามเงื่อนไขที่กำหนดไว้ หลังดำเนินการเสร็จสิ้นทิศทางของแต่ละส่วนทางเลือกทั้งสองต้องมาบรรจบกันและทำงานในขั้นตอนถัดไปหรือคำสั่งต่อไป แสดงดังรูปที่ 3.4



รูปที่ 3.4 ผังงานและทิศทางแบบทางเลือก

3.4.3 การเขียนผังงานแบบวนซ้ำ (Iteration)

การทำซ้ำเป็นการเขียนผังงาน ให้กลับมาทำงานซ้ำๆในรูปแบบเดิมๆ จะเห็นว่า ผังงาน มีการทำงานซ้ำเรียกว่าลูป (Loop) จากรูปที่ 3.5 การทำงานเป็นลำดับจากบนลงล่าง และทำงานตรวจสอบเงื่อนไข เมื่อเงื่อนไขเป็นจริงจะทำการวนกลับไปที่จุดเชื่อมต่อเพื่อทำงานซ้ำ และจะออกจากการทำงานซ้ำเมื่อเงื่อนไขเป็นเท็จ ทั้งนี้การทำงานซ้ำใน loop ขึ้นอยู่กับการตรวจสอบเงื่อนไขที่กำหนด ดังจะได้กล่าวในบทเรียนเรื่องการวนลูป ต่อไป



รูปที่ 3.5 การเขียนผังงานรูปแบบการทำงานซ้ำหรือวนลูป

3.5 ตัวอย่างการเขียน Pseudo-code และการเขียนผังงาน

ตัวอย่างที่ 3.1 จงเขียนผังงานสำหรับแสดงข้อความออกทางหน้าจอคอมพิวเตอร์ ดังต่อนี้

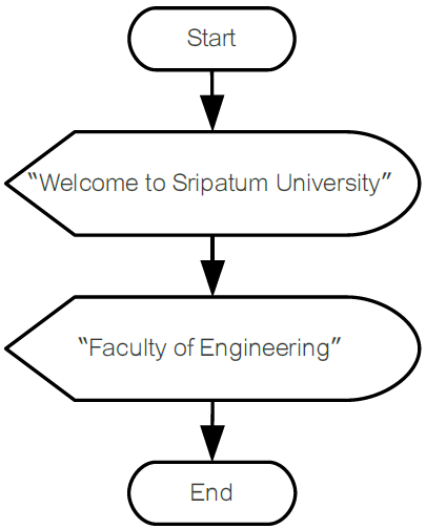
```
Welcome to Sripatum University
Faculty of Engineering
```

วิธีทำ จากโจทย์ข้างต้นสามารถวิเคราะห์ขั้นตอนการทำงาน และแบ่งได้เป็นส่วนๆได้ดังนี้

การเขียน Pseudo-code

1. เริ่มการทำงาน
2. แสดงข้อความว่า “ **Welcome To Sripatum University** ”
3. แสดงข้อความว่า “ **Faculty of Engineering** ”
4. จบการทำงาน

การเขียนผังงานตัวอย่าง 3.1



รูปที่ 3.6 แสดงผังงาน ในการแสดงข้อความออกทางหน้าจอ

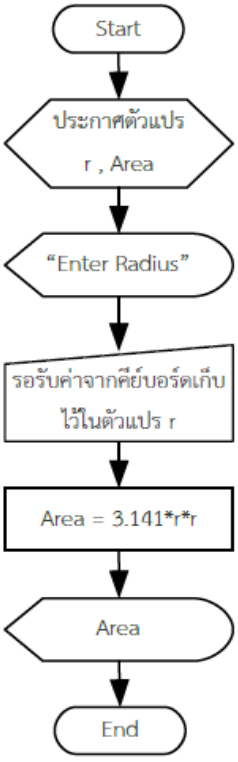
ตัวอย่างที่ 3.2 จงเขียนผังงาน สำหรับโปรแกรมการคำนวณหาพื้นที่ของวงกลม(πr^2) โดยรับค่ารัศมี **r** จากผู้ใช้งานผ่านคีย์บอร์ด

วิธีทำ จากโจทย์ข้างต้นสามารถวิเคราะห์ขั้นตอนผังงานโดยแบ่งได้เป็นส่วนๆได้ดังนี้

การเขียน Pseudo-code

1. เริ่มต้นทำงาน
2. ทำการประกาศตัวแปรเพื่อที่จะนำมาใช้งาน ชื่อว่า “ **r** ” และ “พื้นที่ ”
3. แสดงข้อความเพื่อให้รู้ว่าต้องการค่า รัศมี
4. รอรับค่ารัศมีจากคีย์บอร์ดมาเก็บไว้ที่ตัวแปร **r**
5. คำนวณหาพื้นที่ของวงกลมตามสูตร พื้นที่ = πr^2
6. แสดงค่าพื้นที่ของวงกลมที่ได้จากการคำนวณ
7. จบการทำงาน

การเขียนผังงานตัวอย่าง 3.2



รูปที่ 3.7 ผังงานการคำนวณหาพื้นที่วงกลม

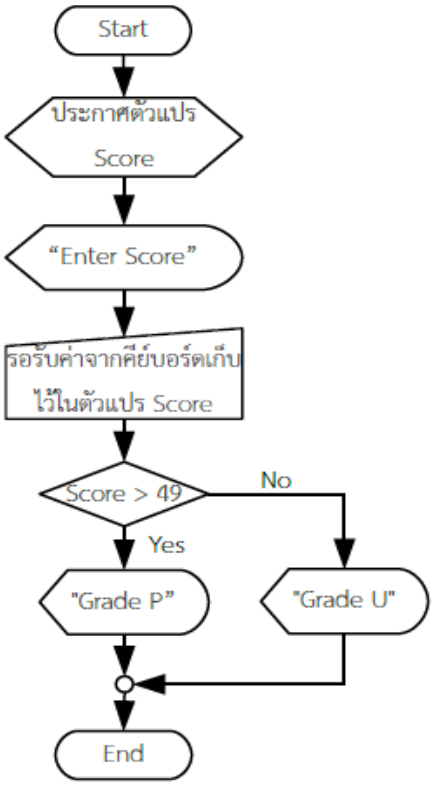
ตัวอย่างที่ 3.3 จงเขียนผังงานแสดงการตัดเกรดนักศึกษาแบบ 2 เกรด โดยรับค่าคะแนน (Score) ที่จะทำการตรวจสอบจากทางคีย์บอร์ดคือ **ถ้า**ได้คะแนนมากกว่า 49 คะแนน ให้แสดงข้อความ “Grade S” (ผ่าน) และ **ถ้า**น้อยกว่า 50 ให้แสดงข้อความ “Grade U” (ไม่ผ่าน)

วิธีทำ จากโจทย์ข้างต้นสามารถเขียนขั้นตอนผังงาน โดยแบ่งเป็นส่วนๆได้ดังนี้

การเขียน Pseudo-code

1. เริ่มต้นทำงาน
2. ทำการประกาศตัวแปรเพื่อที่จะนำมาใช้งาน ชื่อว่า “ **Score** ”
3. แสดงข้อความเพื่อให้รู้ว่าต้องการค่า Score
3. รอรับค่าคะแนนมาเก็บไว้ที่ตัวแปร Score
4. นำค่าคะแนนที่เก็บไว้ในตัวแปร Score มาเปรียบเทียบกับ 49
    4.1 ถ้าคะแนนที่ได้มากกว่า 49 ให้พิมพ์ Grade S
    4.3 ถ้าคะแนนที่ได้น้อยกว่า 50 ให้พิมพ์ Grade U
5. จบการทำงาน

การเขียนผังงาน ตัวอย่างที่ 3.3



รูปที่ 3.8 แสดงผังงานการตัดเกรดนักศึกษาแบบเกรด S และเกรด U

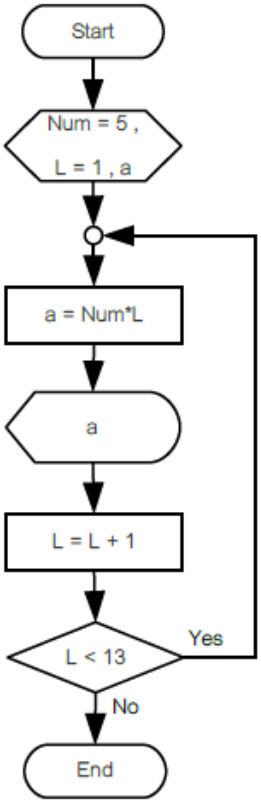
ตัวอย่างที่ 3.4 จงเขียนผังงาน แสดง**การคำนวณสูตรคูณ มาตรา 5**

วิธีทำ จากโจทย์ข้างต้นสามารถเขียนขั้นตอนผังงาน โดยแบ่งเป็นส่วนๆได้ดังนี้

การเขียน Pseudo-code

1. เริ่มต้นทำงาน
2. กำหนดมาตราสูตรคูณ Num = 5
   กำหนด L = 1 เพื่อกำหนดรอบที่ 1
3. คำนวณหาค่า a เท่ากับผลคูณระหว่าง Num และ L (a = Num*L)
4. แสดงผลค่า a ที่คำนวณได้
5. L=L+1 หมายถึง ทำการเพิ่มค่า L ขึ้นอีก 1
6. L<13 หมายถึง ตรวจสอบค่า L ว่าน้อยกว่า 13 หรือไม่
   6.1 ถ้าค่า L น้อยกว่า 13 จะวนกลับไปทำที่ ข้อ 3, 4, 5
   6.3 ถ้าค่า L เท่ากับหรือมากกว่า 13 จึงออกจากลูป
7. จบการทำงาน

การเขียนผังงาน ตัวอย่าง 3.4



รูปที่ 3.9 แสดงการคำนวณสูตรคูณมาตรา 5

สรุปท้ายบท

การเขียนเพอร์ซูโด-โค้ด (Pseudo-code) ทำให้เข้าใจลำดับการทำงานให้ง่ายขึ้น โดยเขียนเป็นลำดับขั้นตอน และทำให้การสร้างผังงานได้สะดวก ตามเงื่อนไขรูปแบบการใช้สัญลักษณ์ผังงานต่างๆ หากเข้าใจหลักการเขียนเพอร์ซูโด-โค้ด ก็จะสามารถเขียนผังงานของโปรแกรมหรือระบบต่างๆ ได้ ทั้งนี้การเขียนผังงานสามารถเข้าใจถึงการออกแบบโปรแกรมหรือระบบที่ต้องการได้โดยใช้สัญลักษณ์ เพื่อลดความผิดพลาดในการสื่อสารระหว่างผู้ใช้งาน ซึ่งการเขียนผังงานนั้นมีความสำคัญอย่างยิ่ง ในการออกแบบโปรแกรมขนาดใหญ่และการทำงานร่วมกับคนจำนวนมาก ทั้งยังให้เข้าใจถึงภาพรวมของโปรแกรมและระบบ

คำถามท้ายบท

1. จงระบุอุปกรณ์ที่เป็นอินพุต ไม่ต่ำกว่า 5 อุปกรณ์ ที่เกี่ยวข้องกับคอมพิวเตอร์มีอะไรบ้าง?
2. จงระบุอุปกรณ์ที่เป็นเอาท์พุต ไม่ต่ำกว่า 5 อุปกรณ์ ที่เกี่ยวข้องกับคอมพิวเตอร์มีอะไรบ้าง?
3. Pseudo-code คืออะไร?
4. การเขียนผังงาน (Flow Chart) มีความสำคัญอย่างไร?
5. รูปแบบโครงสร้างการเขียนผังงานมีกี่ประเภท อะไรบ้าง พร้อมยกตัวอย่างประกอบ?
6. จงระบุประโยชน์ของผังงาน มีอะไรบ้าง?
7. สัญลักษณ์ผังงานพื้นฐานมีกี่ชนิด และอะไรบ้าง?

แบบฝึกหัดท้ายบท

1. จงเขียนผังงาน การคำนวณหาปริมาตรทรงกระบอกและพื้นที่ผิวทรงกระบอก โดยกำหนดรับค่าข้อมูลความสูง h และรัศมี r จากผู้ใช้ทางคีย์บอร์ด

   กำหนดให้ ประมาตรทรงกระบอก = $\pi r^2 h$

   พื้นที่ผิวทรงกระบอก = $2\pi rh + 2\,\pi r^2$

2. จงเขียนผังงานแบบการตรวจสอบเงื่อนไขของผลลัพธ์ของการโยนเหรียญจำนวน 2 เหรียญพร้อมกัน โดยกำหนดการรับอินพุตเหรียญทั้งสองทางคีย์บอร์ด และให้แสดงผลข้อความตามตารางที่กำหนดให้ต่อไปนี้

   กำหนดให้ 1 เหรียญ มี 2 ด้าน ประกอบด้วยด้านหัว (Heads) และด้านก้อย (Tails)

เหรียญที่หนึ่ง (1st Coin)	เหรียญที่สอง (2nd Coin)	แสดงข้อความ
H	H	Double Heads
H	T	Flip two coin again
T	H	
T	T	Double Tails

3. จงเขียนผังงานแบบทำงานซ้ำ การแสดงข้อความ **“Computer Programming for Engineer”** ทางจอภาพมอนิเตอร์ จำนวน 30 ครั้ง และเมื่อครบจำนวนแล้วเสร็จ ให้แสดงข้อความ **“Completed 30 Round”** ออกทางจอภาพมอนิเตอร์ จำนวน 1 ครั้ง

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 4 ชนิดตัวแปรและตัวดำเนินการ

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจการกำหนดชนิดข้อมูลและกำหนดชื่อตัวแปร
- เพื่อให้นักศึกษาเข้าใจลำดับความสำคัญการใช้ตัวดำเนินการและเงื่อนไข
- เพื่อให้นักศึกษาเข้าใจหลักการตัวดำเนินการและขั้นตอนการทำงานของโปรแกรม

เนื้อหา

- การกำหนดชนิดของตัวแปร
- การกำหนดชื่อตัวแปร ตามข้อกำหนดของภาษาซี
- การใช้เครื่องหมายทางคณิตศาสตร์และการกำหนดเงื่อนไข

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความสำคัญของข้อกำหนดชนิดข้อมูล การระบุชนิดของตัวแปร และการประกาศตัวแปรใช้งาน รวมถึงเงื่อนไขและข้อกำหนด
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

การบรรยายสัปดาห์ที่ 4

ชนิดตัวแปรและตัวดำเนินการ

Data type and Operators

ในเนื้อหาบทเรียนนี้ เข้าใจรูปแบบตัวดำเนินการ เครื่องหมายการเปรียบเทียบ เครื่องหมายทางตรรกะ รวมถึงการสร้างตัวแปร ชนิดของตัวแปร และตัวดำเนินการต่างๆ ที่ใช้งานสำหรับโปรแกรมคอมพิวเตอร์ เพื่อให้เข้าใจถึงการสร้างตัวแปรหรือการประกาศตัวแปร (Variable Declaration) ที่มีรูปแบบการเก็บข้อมูลที่แตกต่างกัน เช่น จำนวนเต็ม จำนวนจริง ตัวอักษรหรืออักขระ เป็นต้น โดยให้ผู้เรียนได้เข้าใจวิธีการใช้งานให้ถูกต้องและเหมาะสมกับการทำงานของโปรแกรม

4.1 หลักการตั้งชื่อตัวแปรในภาษาซี

ตัวแปร (Variable) หมายถึง ตัวแปรที่สร้างขึ้น เป็นตัวอักษร เพื่อรองรับการเก็บข้อมูลชนิดต่างๆ เช่น การคำนวณทางคณิตศาสตร์ การเก็บประโยคข้อความหรือตัวอักษร เป็นต้น โดยนำข้อมูลที่ถูกจัดเก็บในหน่วยความจำเพื่อนำไปใช้งานต่อไป การสร้างตัวแปรสำหรับการจัดเก็บข้อมูลในโปรแกรมภาษาซี มีข้อกำหนดการสร้างชื่อตัวแปร ซึ่งผู้ใช้งานโปรแกรมสามารถสร้างตัวแปร ภายใต้ข้อกำหนดการสร้างตัวแปรดังนี้

ข้อกำหนดการตั้งชื่อตัวแปร

1. ต้องขึ้นต้นด้วยอักษร a-z หรือ A-Z หรือเครื่องหมายอันเดอร์สกอร์ _ (Underscore) เท่านั้น เช่น **key , Num , score , A123 , _sum , _value**
2. ไม่มีการเว้นวรรค หรือเว้นว่าง ระหว่างตัวอักษร แต่สามารถใช้เครื่องหมายอันเดอร์สกอร์ (ขีดล่าง) ได้ เช่น **chk_sw , stop_motor , t_msec , Total_time , a_1**
3. ตัวอักษรตัวพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่ มีความแตกต่างกันถือว่าเป็นตัวแปรคนละตัวกัน เช่น **x** และ **X , answer** และ **Answer**
4. ไม่สร้างชื่อตัวแปรซ้ำซ้อนกับคำสงวน (Reserved Word) ของภาษาซี (ตารางที่ 4.1)

ตารางที่ 4.1 ตารางคำสงวนของภาษาซี

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

ข้อแนะนำ

1. การสร้างตัวแปรหรือประกาศตัวแปร ควรสร้างตัวแปรให้สื่อกับวัตถุประสงค์ของการทำงานของตัวแปร เพื่อสื่อความหมายที่ตรงกัน เช่น h คือตัวแปรความสูง, w คือตัวแปรความกว้าง, area คือตัวแปรพื้นที่ เป็นต้น
2. การตั้งชื่อตัวแปรควรตั้งชื่อให้ถูกต้อง โดยไม่มีเครื่องหมายทางคณิตศาสตร์ กำหนดชื่อไม่ซ้ำกับคำสงวนของภาษาซี เช่น Count test123 A_1 ELSE เป็นต้น

4.2 ชนิดข้อมูลในภาษาซี

ข้อมูลที่ใช้แต่ละรูปงานในภาษาซี มีหลากหลายรูปแบบและมีคุณสมบัติเฉพาะตัวที่แตกต่างกันออกไป ผู้ใช้ควรทราบถึงคุณสมบัติและข้อจำกัดข้างต้นเพื่อให้การใช้งานเป็นไปอย่างมีประสิทธิภาพ ชนิดของข้อมูลที่กำหนดไว้ในภาษาซี สามารถแบ่งเป็น 6 ชนิด คือ ข้อมูลชนิดจำนวนเต็ม (Integer), ข้อมูลชนิดจำนวนจริงหรือทศนิยม (Floating Point), ข้อมูลชนิดตัวอักขระ (Character), ข้อมูลชนิดข้อความ (String) แสดงตามตารางที่ 4.2 และข้อมูลแบบพิเศษคือ ข้อมูลชนิดไม่กำหนดค่าใดๆ (void) และข้อมูลชนิดเลขฐาน (Binary, Octal, Hexadecimal)

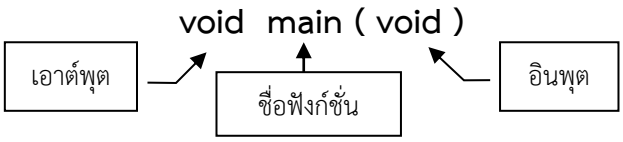
ตารางที่ 4.2 ชนิดข้อมูลของการกำหนดตัวแปร

ชนิดข้อมูล	**ชนิดข้อมูลเฉพาะ**
Integer -> **int, long, unsigned int**	Void -> **void**
Real (Floating Point) -> **float, double, long, double**	Binary -> **01010110_2** หรือ **10110111b**
Character -> **char, unsigned char**	Octal -> **205_8** หรือ **0127**
String -> **string**	Hexadecimal -> **$7EF0_{16}$** หรือ **0x3E9A**

4.3 ชนิดของตัวแปรในภาษาซี

ในการประกาศตัวแปร นอกจากการระบุด้วยชื่อเป็นส่วนแรก จะต้องประกอบด้วยประเภทของตัวแปรของตัวระบุ ข้อมูลพื้นฐานในภาษาซี มีดังนี้

void เรียกว่า “วอยด์” ตัวแปรชนิด void นี้ หมายถึงการระบุในส่วนที่ไม่มีค่าใดๆ ไม่สามารถนำมาคำนวณใดๆ ได้ อย่างไรก็ตาม void เป็นตัวแปรที่มีความสำคัญในการระบุอินพุตและเอาต์พุตของฟังก์ชั่น เมื่อกำหนด void ที่อินพุตของฟังก์ชั่น ฟังก์ชั่นนั้นจะไม่มีการรับอินพุตจากภายนอก และเมื่อกำหนด void ที่เอาต์พุตของฟังก์ชั่น ฟังก์ชั่นนั้นจะไม่มีการส่งข้อมูลออกภายนอก (รายละเอียดในเนื้อหาเรื่องฟังก์ชั่น)



<u>จำนวนเต็ม</u> (Integer)

ข้อมูลจำนวนเต็มในภาษาซี คือ ตัวเลขจำนวนเต็มบวก จำนวนเต็มลบ จำนวนเต็มศูนย์ ตัวเลขปกติทั่วไป ซึ่งข้อมูลจำนวนเต็มสามารถนำไปคำนวณทางคณิตศาสตร์ได้ ตัวอย่างชนิดข้อมูลจำนวนเต็ม เช่น 205, 10, -12, 0 , 45000 เป็นต้น โดยชนิดข้อมูลของตัวแปรจำนวนเต็ม แบ่งได้ดังตารางที่ 4.3

ตารางที่ 4.3 ชนิดข้อมูลของตัวแปรจำนวนเต็ม

ประเภทของข้อมูล	จำนวนบิต	ขอบเขตของค่าข้อมูล	การกำหนดชนิดของตัวแปร
Integer	16	-32,768 ถึง 32,767	**int**
Short Integer	16	-32,768 ถึง 32,767	**short int**
Long Integer	32	-2,147,483,648 ถึง 2,147,483,647	**long**
Unsigned Integer	16	0 ถึง 65,535	**unsigned int**
Unsigned Short	16	0 ถึง 65,535	**unsigned short**
Unsigned Long	32	0 ถึง 4,294,967,295	**unsigned long**

โดยปกติ short และ long จะเก็บค่าจำนวนเต็ม ตามที่ผู้ใช้กำหนดเมื่อกำหนดประเภทข้อมูลเป็น int, short และ long จะหมายถึงจำนวนเต็มที่มีเครื่องหมายกำหนด(signed integer) ในการเก็บจำนวนเต็มเหล่านี้จะต้องใช้บิตแรกสำหรับเก็บเครื่องหมายบวกหรือลบ ส่วนบิตที่เหลือใช้เก็บค่าของข้อมูล สำหรับจำนวนเต็มที่ไม่ระบุเครื่องหมาย (unsigned integer)ได้แก่ unsigned int, unsigned short และ unsigned long จะเก็บข้อมูลมีค่าเป็นบวกเท่านั้น ในกรณีนี้ทำให้สามารถใช้เนื้อที่ทั้งหมดในการเก็บค่าของข้อมูล ซึ่งทำให้เก็บข้อมูลที่มีค่ามากขึ้นได้

<u>จำนวนจริง</u> (Real) จำนวนจริงแบ่งออกเป็น 2 ประเภท คือ float และ double การจัดเก็บจำนวนจริงจะต้องแปลงจำนวนจริงให้อยู่ในรูปของ $Ax10^n$ โดยที่ A ≥1 ซึ่ง A เป็นจำนวนเต็ม กำหนดให้ A คือ Fraction และ n คือ Exponent ลักษณะการจัดเก็บจำนวนจริง float ใช้เนื้อที่ทั้งหมด 32 บิต โดยแบ่งออกเป็นสามส่วน คือส่วนที่เป็น Fraction ใช้เนื้อที่ 23 บิต ส่วนที่เป็น exponent ใช้เนื้อที่ 8 บิต และ เครื่องหมายและค่าของ Exponent จำนวน 1 บิต โดยแสดงการเก็บข้อมูลจำนวนจริง สำหรับการจัดเก็บจำนวนจริง float จะมีค่านัยสำคัญประมาณ 7 ตำแหน่ง แสดงได้ดังตารางที่ 4.4 และตารางที่ 4.5

ตารางที่ 4.4 จำนวนบิตสำหรับข้อมูลจำนวนจริง

บิตที่ 0	**บิตที่ 1-8**	**บิตที่ 9-31**
เครื่องหมาย	**Exponent**	**Fraction**

ตารางที่ 4.5 ข้อมูลจำนวนจริงในรูปแบบของ Fraction และ Exponent

จำนวนจริง	**จำนวนจริงยกกำลัง**	**บิตเครื่องหมาย**	**Exponent**	**Fraction**
12.45	1.245×10^{1}	0	1	1.245
-211.0	-2.110×10^{2}	1	2	2.11
0.0056	5.600×10^{-3}	0	-3	5.6

สำหรับการจัดเก็บจำนวนจริง double จะใช้เนื้อที่ในการจัดเก็บทั้งหมด 64 บิต ทำให้สามารถจัดเก็บจำนวนจริงที่มีค่ามากขึ้น และมีค่านัยสำคัญประมาณ 14 ตำแหน่ง ในขณะที่จำนวนจริง float มีค่านัยสำคัญ 7 ตำแหน่ง แสดงดังตารางที่ 4.6

ตารางที่ 4.6 ตัวอย่างข้อมูลชนิดจำนวนจริง

ประเภทของข้อมูล	**จำนวนบิต**	**ขอบเขตของค่าข้อมูล**	**การกำหนดชนิดของตัวแปร**
Float	32	3.4×10^{-38} ถึง 3.4×10^{38}	**float**
Double	64	1.7×10^{-308} ถึง 1.7×10^{308}	**double**
Long Double	80	3.4×10^{-4932} ถึง 1.1×10^{4932}	**long double**

ข้อมูลชนิดอักขระ (Character) การสร้างตัวแปรชนิดนี้สามารถจัดเก็บตัวอักษร ตัวเลข และตัวอักษรพิเศษ โดยจัดเก็บได้ที่ละ 1 ตัวอักษร และสามารถการจัดเก็บตัวอักขระในเครื่องหมายคอมพิวเตอร์จะจัดเก็บในรูปของรหัสแอสกี (ASCII) และ เอ็บซิดิก (EBCDIC) ขึ้นอยู่กับชนิดของระบบคอมพิวเตอร์ โดยข้อมูลมีขนาดเท่ากับ 1 ไบต์ การกำหนดค่าเป็นตัวอักษรนี้มีการกำหนดได้ 2 แบบ แสดงดังตารางที่ 4.7

ตารางที่ 4.7 ข้อมูลชนิดอักขระ

ประเภทของข้อมูล	จำนวนบิต	ขอบเขตของค่าข้อมูล	การกำหนดชนิดของตัวแปร
Char	8	-128 ถึง 127	**char**
Unsigned Char	8	0 ถึง 255	**unsigned char**

ข้อมูลชนิดข้อความ (String) การสร้างตัวแปรชนิดนี้จะสามารถจัดเก็บตัวอักษรที่มากกว่า 1 ตัว หรือเป็นประโยคข้อความตัวอักษร โดยการสร้างตัวแปรชนิดนี้จะนำไปใช้ในการแสดงข้อความเป็นประโยคและจัดการข้อมูลต่างๆ โดยการสร้างตัวแปรชนิดนี้ในภาษาซี จะนิยมการใช้ตัวแปรแบบอาเรย์ ที่สามารถจัดเก็บตัวแปรแบบข้อความหรือข้อมูลอักษรต่างๆ (อธิบายในบทเรียนเรื่องตัวแปรอาเรย์) ในการเขียนโปรแกรมด้วยภาษาซี จะระบุตัวสุดท้ายของอักษรว่าง NULL (\0) เป็นตัวสุดท้ายของข้อความ

ตัวอย่างเช่น

char a[9] = “COMPUTER” ;

แนวคิด โดยตัวแปรกำหนดเป็น char ชนิดอาร์เรย์จำนวน 9 จำนวน ข้อมูลของตัวแปรในแต่ละจำนวนข้อมูลซึ่ง COMPUTER มีอักษรจำนวน 8 ตัวอักษร รวมกับค่า \0 1 จำนวน รวม 9 จำนวน แสดงได้ดังนี้

C	O	M	P	U	T	E	R	**\0**

รูปที่ 4.1 ข้อมูลชนิดข้อความ string

ตารางที่ 4.8 ASCII มาตรฐานสำหรับ 127 characters

ASCII		ASCII		ASCII		ASCII	
Value	Character	Value	Character	value	character	value	character
0	NUL	32	(Blank)	64	@	96	‘
1	SOH	33	!	65	A	97	a
2	STX	34	“	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	‘	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

4.4 รูปแบบการประกาศตัวแปร

การสร้างตัวแปรขึ้นมาใช้งานในโปรแกรมภาษาซี เรียกว่า การประกาศตัวแปร (Variable Declaration) โดยต้องทำให้ถูกต้องตามรูปแบบคำสั่งภาษาซีกำหนดไว้ แสดงดังนี้

```
datatype    variable ;          //ชนิดของตัวแปร   ชื่อตัวแปร
```

datatype คือ กำหนดชนิดของตัวแปรที่จะสร้างขึ้น เช่น int, float, double, char เป็นต้น

variable คือ ระบุชื่อตัวแปรที่ต้องการ เช่น area, a1, h, value, เป็นต้น

ตัวอย่างเช่น

int A; หมายถึง A เป็นตัวแปรที่ใช้เก็บค่าเลขจำนวนเต็มที่มีค่าอยู่ระหว่าง -32768 ถึง 32767

unsigned int Lower; หมายถึง Lower เป็นตัวแปรที่ใช้เก็บค่าที่เป็นเลขจำนวนเต็มมีค่าอยู่ระหว่าง 0 ถึง 65535

float woman , ratio ; หมายถึง woman และ ratio เป็นตัวแปรที่ใช้เก็บค่าที่เป็นทศนิยม โดยจะให้ตำแหน่งทศนิยมได้ ไม่เกิน 6 หลัก

double score ; หมายถึง score เป็นตัวแปรที่ใช้เก็บค่าที่เป็นทศนิยมที่จะให้ตำแหน่งทศนิยมได้ละเอียดถึง 12 หลัก

char x , y , num ; หมายถึง x , y และ num เป็นตัวแปรที่ใช้เก็บข้อมูลที่เป็นตัวอักษรขนาด 1 ตัวอักษร

4.5 เครื่องหมายดำเนินการ (Operators)

เครื่องหมายดำเนินการในโปรแกรมภาษาซี มีความสำคัญสำหรับการคำนวณ การตรวจสอบเงื่อนไขเปรียบเทียบ โดยสามารถใช้เครื่องหมายดำเนินการ ซึ่งแบ่งออกเป็น 3 ชนิดคือ

- เครื่องหมายทางคณิตศาสตร์ (Arithmetic Operator)
- เครื่องหมายเปรียบเทียบ (Relational Operator)
- เครื่องหมายทางตรรกะ (Logical Operators)

4.5.1 เครื่องหมายทางคณิตศาสตร์ (Arithmetic Operator)

เครื่องหมายดำเนินการใช้สำหรับการคำนวณทางคณิตศาสตร์ระหว่างตัวแปร เพื่อการคำนวณสมการทางคณิตศาสตร์ แสดงเครื่องหมายตารางที่ 4.9 ดังนี้

ตารางที่ 4.9 เครื่องหมายทางคณิตศาสตร์

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	การบวก	A + B
-	การลบ	A – B
*	การคูณ	A * B
/	การหาร	A / B
%	การหาร เก็บค่าเฉพาะเศษ (Modulus)	5%3 = 1 เศษ 2 ผลลัพธ์เก็บเศษ 2
- -	การลดค่าลงครั้งละ 1	A-- จะเหมือนกับ A = A – 1
+ +	การเพิ่มค่าขึ้นครั้งละ 1	A++ จะเหมือนกับ A = A + 1

4.5.2 การรวมเครื่องหมายทางคณิตศาสตร์ (Arithmetic-combination operators)

เครื่องหมายและลำดับความสำคัญของเครื่องหมายการกระทำทางคณิตศาสตร์ ซึ่งในการเขียนโปรแกรมภาษาซีบางกรณีผู้ใช้สามารถที่จะเขียนรวมเครื่องหมาย Operators เข้าด้วยกัน ตัวอย่างตารางที่ 4.10

ตารางที่ 4.10 รวมเครื่องหมายทางคณิตศาสตร์

รูปแบบที่ใช้ในภาษาซี	รูปแบบทางการการคำนวณทั่วไป
i + = 5	i = i + 5
f - = g	f = f – g
j * = (i-3)	j = j * (i – 3)
f /= 3	f = f/3
i %= (j – 2)	i = i % (j – 2)

ตัวอย่างที่ 4.1

Y = 5 ;

X = ++Y ;

หมายถึง มีการเพิ่มค่า Y ไปอีก 1 เป็น 6 แล้วจึงนำมาเก็บไว้ในค่าตัวแปร X ดังนั้น Y = 6, X = 6;

ตัวอย่างที่ 4.2

Y = 5 ;

X = Y++ ;

หมายถึง จะมีการนำ Y ซึ่งมีค่าเท่ากับ 5 ไปเก็บไว้ในตัวแปร X ก่อนแล้วจึงค่อยเพิ่มค่า Y อีก 1 เป็น 6 ดังนั้น Y = 6, X = 5;

ตัวอย่างที่ 4.3

A = 10 ;

A -= 5 ;

หมายถึง **A = A – 5** จะนำค่าในตัวแปร A ไปลบ 5 แล้วทำการเก็บค่าที่ได้ไว้ในตัวแปร A เช่นเดิม ดังนั้น **A = 5 ;**

4.5.2 เครื่องหมายเปรียบเทียบ (Relational Operator)

เครื่องหมายที่สำหรับการเปรียบเทียบ สำหรับเงื่อนไขการตัดสินใจของโปรแกรม ซึ่งผลลัพธ์จากการเปรียบเทียบสามารถแบ่งเป็น 2 กรณี คือ เงื่อนไขเป็นจริง โดยสมมุติให้มีค่าเป็น “1” และเงื่อนไขเป็นเท็จ โดยสมมุติให้มีค่าเป็น “0” โดยการใช้สัญลักษณ์เครื่องหมาย ให้ตรงตามรูปแบบคำสั่งของภาษาซี แสดงตารางที่ 4.11 ดังนี้

ตารางที่ 4.11 เครื่องหมายเปรียบเทียบ

เครื่องหมาย	ความหมาย	ตัวอย่าง
>	มากกว่า	A > B (A มากกว่า B เป็นจริง)
>=	มากกว่าหรือเท่ากับ	A >= B (A มากกว่าหรือเท่ากับ B เป็นจริง)
<	น้อยกว่า	A < B (A น้อยกว่า B เป็นจริง)
<=	น้อยกว่าหรือเท่ากับ	A <= B (A น้อยกว่าหรือเท่ากับ B เป็นจริง)
==	เท่ากับ	A == B (A เท่ากับ B เป็นจริง)
!=	ไม่เท่ากับ	A != B (A ไม่เท่ากับ B เป็นจริง)

4.5.3 เครื่องหมายทางตรรกะ (Logical Operators)

เครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ โดยอาศัยหลายเงื่อนไขที่นำมาเปรียบเทียบ (มากกว่า 2 เงื่อนไข) ซึ่งผลลัพธ์ของแต่ละเงื่อนไขจะนำมาเปรียบเทียบต่อไป โดยเครื่องหมายทางตรรกะ สามารถจำแนกออกเป็น 3 รูปแบบ คือ && (AND), || (OR), ! (NOT) อธิบายรายละเอียดดังนี้

เครื่องหมาย && (AND) หมายถึง การนำเงื่อนไขตั้งแต่ 2 เงื่อนไขขึ้นไปมาเปรียบเทียบกัน ผลลัพธ์จากการเปรียบเทียบนั้น จะเป็น**จริง**ได้ เมื่อเงื่อนไขที่นำมาเปรียบเทียบทุกเงื่อนไขต้องเป็น**จริง**ทั้งหมด แต่หากมีเงื่อนไขใดเงื่อนไขหนึ่งเป็นเท็จ ผลลัพธ์จะเป็นเท็จทุกกรณี แสดงในตารางที่ 4.12 ตัวอย่างเช่น (A > B) && (C > D) จากการเปรียบเทียบเงื่อนไข จะได้ผลลัทธ์ตามตารางต่อไปนี้

ตารางที่ 4.12 ตารางค่าความจริงของ && (AND)

A > B	C > D	(A > B) && (C > D)
เท็จ (0)	เท็จ (0)	เท็จ (0)
เท็จ (0)	จริง (1)	เท็จ (0)
จริง (1)	เท็จ (0)	เท็จ (0)
จริง (1)	จริง (1)	จริง (1)

เครื่องหมาย || (OR) หมายถึง การนำเงื่อนไขตั้งแต่ 2 เงื่อนไขขึ้นไปมาเปรียบเทียบกัน ผลลัพธ์จากการเปรียบเทียบนั้น จะเป็น**จริง**ได้ เมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็น**จริง** แต่ถ้าหากมีเงื่อนไขทั้งสองเงื่อนไขเป็นเท็จ ผลลัพธ์จะเป็น**เท็จ** แสดงในตารางที่ 4.13
ตัวอย่างเช่น (A > B) || (C > D) จากการเปรียบเทียบเงื่อนไข จะได้ผลลัทธ์ตามตารางต่อไปนี้

ตารางที่ 4.13 ตารางค่าความจริงของ || (OR)

A > B	C > D	(A > B) || (C > D)
เท็จ (0)	เท็จ (0)	เท็จ (0)
เท็จ (0)	จริง (1)	จริง (1)
จริง (1)	เท็จ (0)	จริง (1)
จริง (1)	จริง (1)	จริง (1)

เครื่องหมาย ! (NOT) หมายถึง การนำเงื่อนไขมาทำการกลับสถานะให้ตรงข้ามจากเดิม จากผลลัพธ์ จริง ให้เป็น เท็จ หรือจากผลลัพธ์ เท็จ ให้เป็น จริง แสดงดังตารางที่ 4.14

ตารางที่ 4.14 ตารางค่าความจริงของ ! (NOT)

(A>B)	! (A>B)
จริง (1)	เท็จ (0)
เท็จ (0)	จริง (1)

ตัวอย่างที่ 4.4

กำหนดให้ int a = 3, b = 2, c = 4 , d = -1

4.4.1 (a > b) && (c > a) คำตอบ ผลลัพธ์เป็น จริง

4.4.2 (2d < d) && (c > a) คำตอบ ผลลัพธ์เป็น จริง

4.4.3 (c != ab) && (d > b) คำตอบ ผลลัพธ์เป็น เท็จ

4.4.4 (bc != 0) || (ab > 10) คำตอบ ผลลัพธ์เป็น จริง

4.4.5 (d < -1) || (a > c) คำตอบ ผลลัพธ์เป็น เท็จ

ลำดับขั้นตอนการทำงานของนิพจน์

นิพจน์ในภาษาซี จะทำงานจากซ้ายไปขวาตามลำดับความสำคัญของเครื่องหมายโดยพิจารณาการทำงานจากข้อ 1 ถึงข้อสุดท้าย

ลำดับที่ 1 คือเครื่องหมาย ( ) การคำนวณในวงเล็บก่อนและทำจากซ้ายไปขวาตามลำดับเครื่องหมาย
ลำดับที่ 2 คือเครื่องหมาย **! , ++a , --a**
ลำดับที่ 3 คือเครื่องหมาย *** , / , %**
ลำดับที่ 4 คือเครื่องหมาย **+ , -**
ลำดับที่ 5 คือเครื่องหมาย **a++ , a--**
ลำดับที่ 6 คือเครื่องหมาย **< , <= , > , >=**
ลำดับที่ 7 คือเครื่องหมาย **== , !=**
ลำดับที่ 8 คือเครื่องหมาย **&&**
ลำดับที่ 9 คือเครื่องหมาย **||**

ยกตัวอย่างการทำงานของนิพจน์ภาษาซี ดังนี้

ตัวอย่างที่ 4.5

4 + 4 * 2

วิธีคิด **1.** เนื่องจากนิพจน์เครื่องหมายคูณสำคัญกว่าเครื่องหมายบวกจึงทำการคูณก่อนได้ผลลัพธ์ **4 * 2 = 8**

2. จากนั้นค่อยนำ **4** มาบวกกับผลลัพธ์ที่**ได้** **4 + 8 = 12** ผลลัพธ์ สุดท้ายได้ **12**

ตัวอย่างที่ 4.6

(4 + 5) – 4 * 2

วิธีคิด **1.** ให้ทำการคำนวณหาผลลัพธ์ที่อยู่ในวงเล็บก่อน 4 + 5 = 8 จะได้นิพจน์เป็น **8 – 4 * 2**

2. ต่อมานำ **4 * 2** ก่อน เพราะเครื่องหมายคูณสำคัญกว่าเครื่องหมายลบ จะได้ **8 – 6**

4. สุดท้ายจึงมาลบกัน **8 – 6 = 2** ผลลัพธ์สุดท้ายได้ **2**

ตัวอย่างที่ 4.7

4 * ++5

วิธีคิด **1.** เครื่องหมาย ++ มีอันดับสูงกว่าเครื่องหมายคูณ เพราะฉะนั้นจึงต้องทำการคำนวณเครื่องหมายนี้ก่อน จะได้ **++5 = 6**

2. จากนั้น นำผลลัพธ์ที่ได้มาทำการคูณ จะได้ **4 * 6 = 24** ผลลัพธ์สุดท้ายได้ **24**

ตัวอย่างที่ 4.8

++(4 * 5)

วิธีคิด **1.** ให้ทำการคำนวณค่าที่อยู่ในวงเล็บก่อน คือ **4 * 5 = 20**

2. จากนั้น นำผลลัพธ์ที่จากการคำนวณในวงเล็บมาเพิ่มค่าไปอีก 1 ผลลัพธ์สุดท้ายได้ **21**

สรุปท้ายบท

ขั้นตอนการสร้างชื่อตัวแปรและกำหนดชนิดของตัวแปรให้ถูกต้องตามข้อกำหนดของภาษาซี โดยกำหนดชนิดตัวแปรสำหรับเก็บค่าข้อมูลชนิดจำนวนจริง จำนวนเต็ม และตัวอักษร ให้ตรงตามคุณสมบัติของตัวแปรชนิดนั้นๆ หากกำหนดผิดวัตถุประสงค์ จะส่งผลให้การจะทำให้โปรแกรมการทำงานมีความผิดพลาด ผลลัพธ์การคำนวณจะมีค่าผิดพลาดได้ ซึ่งตัวแปรที่สามารถระบุทศนิยมคือตัวแปรจำนวนจริงเท่านั้น รวมถึงการใช้เครื่องหมายทางคณิตศาสตร์ต้องคำนวณถึงรูปแบบการเขียนสัญลักษณ์เครื่องหมายทางภาษาซี เพื่อที่จะทำให้โปรแกรมสามารถทำงานได้อย่างถูกต้อง

คำถามท้ายบท

1. จงยกตัวอย่างการสร้างชื่อตัวแปรที่ผิดมา 5 ตัวอย่าง อะไรบ้าง?
2. จงยกตัวอย่างคำสงวนที่ห้ามการสร้างชื่อตัวแปรมา 5 ชนิด อะไรบ้าง?
3. เครื่องหมายการดำเนินการมีกี่ประเภท อะไรบ้าง?
4. เครื่องหมายการเปรียบเทียบเงื่อนไขมีกี่ประเภท อะไรบ้าง?
5. จงอธิบายตัวแปรชนิดจำนวนเต็ม int สามารถเก็บค่าได้เท่าใด?
6. จงอธิบายตัวแปรชนิดจำนวนจริง float สามารถเก็บค่าได้เท่าใด?
7. จงอธิบายตัวแปรชนิดจำนวน char สามารถเก็บค่าได้เท่าใดและอะไรบ้าง?
8. ตัวแปรที่กำหนดแบบ unsinged หมายถึงอะไร?
9. ตัวแปรที่ใช้กำหนด string คืออะไร?
10. ตัวแปรชนิด unsigned char จำนวน 2 byte สามารถเก็บข้อมูลได้จำนวนเต็มได้เท่าไร?

แบบฝึกหัดท้ายบท

1. จงระบุชนิดของตัวแปร ในแต่ละข้อย่อยที่กำหนดให้ดังต่อไปนี้

   1.1) A = 1.351

   1.2) B =100

   1.3) C = -0.0001

   1.4) D = 'Z'

   1.5) E= 2,000,000,000

   1.6) F = -50,000

   1.7) G = 32,090,790.141536

   1.8) H = ' & '

2. จงหาผลลัพธ์ จริง หรือ เท็จ ในแต่ละข้อย่อย ตามเงื่อนไขที่กำหนดให้ในแต่ละข้อย่อย

   กำหนดให้

   int a = 2, b = 4, c = 5, d = -2 , e = 'a', f = 'z'

   2.1) **(a + b) < (c + a)**

   2.2) **(f != 'A')**

   2.3) **(e == f) || (e != f)**

   2.4) **(2d > d) && (ab < d)**

   2.5) **(c > dd) && (b < c) && (a != d)**

   2.6) **(f == e) || (f != e)**

   2.7) **(2c >= (a + c + d))**

   2.8) **(a^b > c^a)**

   2.9) **! (a + b > 0)**

   2.10) **(! (c < d)) || (! (d < a))**

   2.11) **(b % a) + (c / a)**

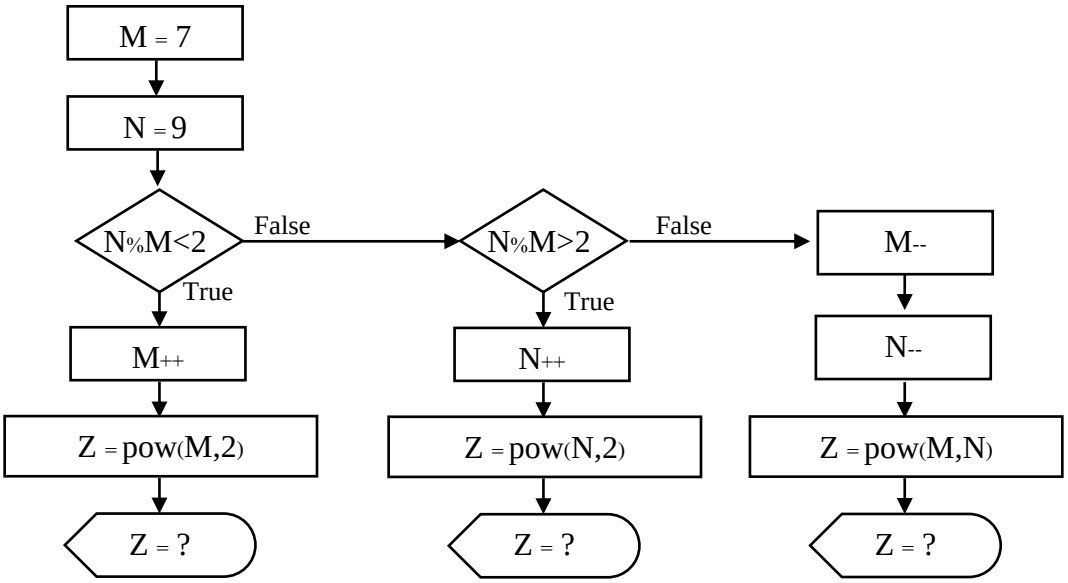
   2.12) **((a + b) % 2) > c)**

   2.13) **((a ^ b) > (d ^ b)) || (d < b)**

   2.14) **! ((2d + 2a) > (cd - ad))**

   2.15) **(! ((ab + d) / a)) && (a != a))**

3. จงหาผลลัพธ์ของ Z



แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 5 ฟังก์ชั่นพื้นฐานสำหรับโปรแกรมภาษาซี

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาสามารถเข้าใจสมการทางคณิตศาสตร์ในรูปแบบของการเขียนโปรแกรม
- เพื่อให้นักศึกษาเข้าใจรูปแบบการคำนวณทางคณิตศาสตร์
- เพื่อให้นักศึกษาเข้าใจการใช้ฟังก์ชั่นพื้นฐานสำหรับฟังก์ชั่นอินพุตและฟังก์ชั่นเอาต์พุต

เนื้อหา

- การเขียนโปรแกรมสำหรับการรับค่าทางอินพุตจากผู้ใช้งานผ่านแป้นพิมพ์
- การเขียนโปรแกรมสำหรับการแสดงผลข้อมูลทางเอาต์พุตทางหน้าจอคอมพิวเตอร์
- การเขียนโปรแกรมคำนวณโดยใช้ฟังก์ชั่นทางคณิตศาสตร์

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายการใช้ฟังก์ชั่นอินพุตและเอาต์พุต ฟังก์ชั่นพื้นฐานที่ใช้งานในการเขียนโปรแกรมการคำนวณ
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดาน

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 5

ฟังก์ชั่นพื้นฐานสำหรับโปรแกรมภาษาซี

Basic function for C language programming

ฟังก์ชั่นใช้งานในโปรแกรมภาษาซี มีความหลากหลายรูปแบบให้ใช้งาน โดยการใช้ฟังก์ชั่นใดๆ ก็ตาม ต้องทำการเรียก ส่วนหัวของโปรแกรม (Header Files) เป็นชนิดไฟล์นามสกุล .h สำหรับการเรียกใช้งานฟังก์ชั่นต่างๆ ของภาษาซี โดยฟังก์ชั่นพื้นฐานสำหรับการควบคุมอินพุตและเอาต์พุตของอุปกรณ์คอมพิวเตอร์ สามารถเรียก Header Files ของไฟล์ชื่อ <stdio.h> และ <stdlib.h> รวมถึงการใช้ฟังก์ชั่นการคำนวณทางคณิตศาสตร์สามารถเรียก Header Files ของไฟล์ชื่อ <math.h> ดังนั้นการเรียก Header Files มีความสำคัญในการเรียกใช้ฟังก์ชั่นต่างๆ ให้โปรแกรมสามารถทำงานได้

ในบทเรียนนี้จะกล่าวถึงชุดคำสั่งพื้นฐาน สำหรับฟังก์ชั่นที่ใช้การดำเนินการทางอินพุตและเอาต์พุต รวมถึงเริ่มต้นการเขียนโปรแกรมขั้นพื้นฐาน

5.1 ฟังก์ชั่นพื้นฐานโปรแกรมภาษาซี

ฟังก์ชั่นใช้งานในโปรแกรมภาษาซี มีฟังก์ชั่นที่หลากหลายในการเรียกใช้งานภายใต้การดำเนินการของโปรแกรมสำหรับไฟล์ส่วนหัวโปรแกรม โดยการเรียกใช้คำสั่ง #include < *Header File* > เพื่อกำหนดการใช้งานของฟังก์ชั่นต่างๆ ภายใต้ของ Header File นั้นๆ ซึ่งการเรียกใช้ไฟล์ส่วนหัวโปรแกรมต้องระบุนามสกุล .h เช่น stdio.h, stdlib.h, math.h, string.h เป็นต้น ด้วยฟังก์ชั่นพื้นฐานที่จะศึกษาในเนื้อหาบทนี้จะกล่าวถึงชุดคำสั่งพื้นฐานที่จำเป็นในการรับค่าทางอินพุตจากผู้ใช้ทางคีย์บอร์ด และแสดงผลลัพธ์ข้อมูลทางหน้าจอภาพคอมพิวเตอร์ การเรียก Header Files ของไฟล์ชื่อ <stdio.h> ซึ่งสามารถเรียกใช้ฟังก์ชั่นในตารางที่ 5.1 แสดงดังนี้

ตารางที่ 5.1 ฟังก์ชั่นการใช้งานของ Header Files ไฟล์ชื่อ <stdio.h>

fprintf	snprintf	vfscanf	vsprintf
fscanf	sprintf	vprintf	vsscanf
printf	sscanf	vscanf	
scanf	vfprintf	vsnprintf	

ฟังก์ชั่นพื้นฐานในการรับค่าข้อมูลอินพุตสามารถเรียกใช้ scanf และฟังก์ชั่นในการแสดงผลบนหน้าจอภาพ สามารถเรียกใช้คำสั่ง printf ในการแสดงผลลัพธ์จากการคำนวณ รวมถึงแสดงข้อความอักษรทางหน้าจอภาพ

5.1.1 การใช้ฟังก์ชั่น printf ( )

ฟังก์ชั่น printf( ); เป็นฟังก์ชั่นที่ใช้แสดงข้อความ แสดงผลลัพธ์ และแสดงข้อความพร้อมกับผลลัพธ์ออกทางจอภาพ โดยชุดคำสั่ง printf สามารถจำแนกใช้งานได้ดังนี้

รูปแบบที่ 1: การแสดงผลข้อความหรือประโยคข้อความ

printf (" *ข้อความหรือประโยคข้อความ* ") ;

รูปแบบที่ 2: การแสดงค่าข้อมูลตัวแปร

printf(" *ตัวกำหนดชนิดข้อมูล* " , *ตัวแปร*) ;

รูปแบบที่ 3: การแสดงข้อความและค่าข้อมูลตัวแปร

printf (" *ข้อความหรือประโยคข้อความและตัวกำหนดชนิดข้อมูล* " , *ตัวแปร*) ;

<u>รูปแบบที่ 1</u> แสดงเฉพาะข้อความ หรือ ประโยคข้อความ (อนุญาตเฉพาะภาษาอังกฤษเท่านั้น) โดยการแสดงข้อความเป็นตัวอักษรทั้งตัวพิมพ์เล็ก a-z ตัวพิมพ์ใหญ่ A-Z รวมถึงตัวเลข 0-9 ตัวอักขระต่างๆ หรือประโยคข้อความ ดังนั้นการใช้ฟังก์ชั่น printf รูปแบบนี้จะเป็นการแสดงข้อความต้องอยู่ภายในเครื่องหมายดับเบิ้ลโค้ท (double quote) เท่านั้น โดยข้อมูลใดๆที่อยู่ภายใต้เครื่องหมาย (" ") จะแสดงข้อความอักษรนั้นออกทางจอภาพ

รูปแบบการใช้งาน

printf(" *ข้อความอักษรหรือประโยคข้อความ* ") ;

ตัวอย่างโปรแกรมที่ 5.1

```
1. #include <stdio.h>
2. main ( )
3. {
4.     printf(" Computer programming for engineer ");  // แสดงข้อความอักษรหรือประโยค
5. }
```

ผลลัพธ์ของโปรแกรมจะแสดงข้อความ **Computer programming for engineer** ออกทางจอภาพ

<u>รูปแบบที่ 2</u> แสดงการแสดงค่าข้อมูลตัวแปร พร้อมตัวกำหนดชนิดข้อมูล (Control String) ที่ใช้ร่วมกับคำสั่ง printf( ) แสดงตารางที่ 5.2 โดยตัวกำหนดชนิดข้อมูล ให้ตรงกับการกำหนดชนิดของตัวแปร เช่น ตัวแปร a ประกาศเป็น int ต้องกำหนดชนิดข้อมูล %d เป็นต้น เพื่อกำหนดรูปแบบตัวเลขจำนวนเต็ม เป็นต้น โดยค่าข้อมูลใดๆ ที่อยู่ภายในเครื่องหมายดับเบิ้ลโค้ท (" ") จะแสดงค่าของข้อมูลนั้นออกทางจอภาพ

รูปแบบการใช้งาน

printf(" *ตัวกำหนดชนิดข้อมูล Control string*" , *ตัวแปร*) ;

ตารางที่ 5.2 ตัวกำหนดชนิดข้อมูล (Control String)

ตัวกำหนดชนิดข้อมูล (Control String)	ความหมายของตัวกำหนดชนิดข้อมูล
%c	แสดงข้อมูลในรูปตัวอักษร
%d	แสดงข้อมูลในรูปตัวเลขจำนวนเต็ม
%e	แสดงข้อมูลในรูปเอกซ์โพเนนเชียล
%f	แสดงข้อมูลในรูปของตัวเลขทศนิยม
%o	แสดงข้อมูลในรูปของตัวเลขฐาน 8
%x	แสดงข้อมูลในรูปของตัวเลขฐาน16
%u	แสดงข้อมูลในรูปของจำนวนเต็มบวก
%s	แสดงข้อมูลในรูปของข้อความ(string)
%p	แสดงข้อมูลในรูปของที่อยู่ของตัวแปร(Address)
%%	แสดงเครื่องหมาย %

ตัวอย่างโปรแกรมที่ 5.2

```
1. #include <stdio.h>
2. main( )
3. {       int A = 100 ;
4.         printf( " %d ", A);      // ตัวกำหนดชนิดข้อมูล Control string ของ ตัวแปร A
5. }
```

ผลลัพธ์ของโปรแกรม จะแสดงจำนวนเต็มมีค่าเท่ากับ **100** ออกมาแสดงผลบนจอภาพ

<u>รูปแบบที่ 3</u> แสดงการแสดงข้อความและค่าข้อมูลตัวแปร เป็นการแสดงข้อความอักษรพร้อมกับค่าตัวแปรออกทางหน้าจอภาพคอมพิวเตอร์ โดยต้องใช้ร่วมกับรูปแบบของตัวกำหนดชนิดข้อมูล ซึ่งข้อความและตัวกำหนดชนิดข้อมูล ใดๆ ที่อยู่ภายในเครื่องหมายดับเบิ้ลโค้ท (" ") จะแสดงข้อความและค่าของตัวแปรนั้นๆ นั้นออกทางจอภาพคอมพิวเตอร์

รูปแบบคำสั่ง

printf(" *ข้อความหรือประโยคข้อความและตัวกำหนดชนิดข้อมูล* **" ,** *ตัวแปร***) ;**

ตัวอย่างโปรแกรมที่ 5.3

```
1. #include <stdio.h>
2. main ( )
3. {        int a = 205 ;
4.          printf("Value A = %d ", a );
5. }
```

ผลลัพธ์ของโปรแกรม จะแสดงข้อความและค่าของตัวแปร A ซึ่งมีค่าเท่ากับ 205

แสดงดังนี้ **Value A = 205**

ตัวอย่างโปรแกรมที่ 5.4

```
1. #include <stdio.h>
2. main ( )
3. {        int a = 205 ;
4.          float b = 3.14159 ;
5.          char c = ' S ' ;
6.          printf("Value A = %d and Pi = %f and Output C = %c " , a, b, c );
7. }
```

ผลลัพธ์ของโปรแกรม จะแสดงข้อความและค่าของตัวแปร A ซึ่งมีค่าเท่ากับ 205

แสดงดังนี้ **Value A = 205 and Pi = 3.141590 and Output C = S**

***หมายเหตุ** %f จะแสดงทศนิยมจำนวน 6 ตำแหน่ง และเมื่อกำหนด %.3f จะแสดงทศนิยม 3 ตำแหน่ง

5.1.2 การใช้ตัวเครื่องหมาย \ (Backslash) ร่วมกับคำสั่ง printf( )

การใช้คำสั่ง printf แสดงข้อความหรือค่าของตัวแปร สามารถใช้เครื่องหมาย \ ในการแสดงการทำงานของเครื่องหมายต่างๆ (แสดงดังตารางที่ 5.3) โดยการใช้งานเครื่องหมายใดๆ อยู่ภายในเครื่องหมายดับเบิ้ลโค้ท (" ") ของคำสั่ง printf เท่านั้น

ตารางที่ 5.3 สัญลักษณ์เครื่องหมายที่ใช้รวมกับคำสั่ง printf ( )

สัญลักษณ์	ความหมาย
\a	กำหนดให้ลำโพงในคอมพิวเตอร์ส่งเสียง beep สั้นๆ 1 ครั้ง
\n	กำหนดให้มีการเลื่อน cursor ไปอยู่บรรทัดถัดไป (ขึ้นบรรทัดใหม่)
\t	กำหนดให้มีการ tap ในแนวนอน (6 ถึง 8 ตัวอักษร)
\b	กำหนดให้การเลื่อน cursor ย้อนกลับไป 1 ตัวอักษร
\v	กำหนดให้มีการ tap ในแนวตั้ง
\f	กำหนดให้มีการขึ้นหน้าใหม่
\r	กำหนดให้มีการ เลื่อน cursor ไปอยู่ที่ตำแหน่งตัวอักษรแรกของบรรทัด
\'	กำหนดให้มีการพิมพ์ตัวอักษร ' (single quote) 1 ตัว บนจอภาพ
\"	กำหนดให้มีการพิมพ์ตัวอักษร " (double quote) 1 ตัว บนจอภาพ
****	กำหนดให้มีการพิมพ์ตัวอักษร \ (backslash) 1 ตัว บนจอภาพ
\000	กำหนดให้มีการแทนที่ตัวอักษรที่มีค่า ASCII (American Standard Code for Information Interchage) เท่ากับ 000 ในระบบเลขฐาน 8 (octal number)
\xhh	กำหนดให้มีการแทนที่ตัวอักษรที่มีค่า ASCII เท่ากับ hh ในระบบเลขฐาน 16 (hexadecimal number)

ตัวอย่างโปรแกรมที่ 5.5

```
1. #include <stdio.h>
2. main ( )
3. {       int a = 12345 , b = 999 ;
4.         printf("Value A = \r%d\n", a ) ;
5.         printf("EGR205\b\b\b%d\n",b) ;
6. }
```

ผลลัพธ์ของโปรแกรมจะแสดงดังนี้

12345 A =

EGR999

5.1.3 การรับค่าอินพุททางแป้นพิมพ์ (keyboard) ด้วยฟังก์ชั่น scanf( )

คำสั่ง scanf( ) เป็นคำสั่งในภาษาซี ที่ทำหน้าที่รับข้อมูลจากแป้นพิมพ์ (keyboard) ในรูปแบบที่กำหนดของกลุ่มข้อมูล คือตัวคำสั่งจะรับกลุ่มข้อมูลจนกระทั่งมีการกดปุ่ม <enter> ส่วนรูปแบบของคำสั่ง scanf(...) มีรูปแบบโปรแกรมดังต่อไปนี้

```
scanf ( "control string" ,  arg1 , arg2 , arg4 , ....., argn);
```

arg1, arg2, ... คือการกำหนดข้อมูลทางแป้นพิมพ์ให้กับตัวแปร ต้องนำหน้าด้วยเครื่องหมาย **&** (AND) ทุกตัวแปร เพื่อระบุตำแหน่งในรีจีสเตอร์ของตัวแปรนั้นๆ

control string คือ เป็นตัวแปรที่กำหนดให้รองรับรูปแบบที่กำหนดใน control string ในการรับรูปแบบของตัวแปรที่มีประเภทข้อมูลต่างๆ กัน เช่น จำนวนเต็มกำหนดด้วย **%d** และจำนวนจริงกำหนดด้วย **%f** โดยมีรายละเอียดตามตารางที่กำหนดให้ดังนี้

ตารางที่ 5.4 รูปแบบประเภทข้อมูลในการอ่านและแสดงผล

ประเภทข้อมูล	Format String สำหรับคำสั่ง scanf	Format String สำหรับคำสั่ง printf
short	%hd	%d
int	%d	%d
long	%ld	%ld
unsigned short	%hu	%u
unsigned int	%u	%u
unsigned long	%lu	%lu
octal short	%ho	%o
octal int	%o	%o
octal long	%lo	%lo
hex short	%hx	%x
hex int	%x	%x
hex long	%lx	%lx
float	%f	%f
double	%lf	%f

ตัวอย่างโปรแกรมที่ 5.6 การรับค่าตัวแปร ด้วยคำสั่ง scanf( )

```
1. #include <stdio.h>
2. main( )
3. {      int  a , b ;                 // ประกาศตัวแปร a และ b เป็นจำนวนเต็ม
4.        printf(“Enter a = ”);        // แสดงข้อความ Enter a =
5.        scanf( “%d”, &a );           // รับค่าจำนวนเต็มทางแป้นพิมพ์ เก็บไว้ในตัวแปร a
6.        printf(“Enter b = ”);        // แสดงข้อความ Enter b =
7.        scanf( “%d”, &b );           // รับค่าจำนวนเต็มทางแป้นพิมพ์ เก็บไว้ในตัวแปร b
8. }
```

การรับค่าหลายตัวแปร ด้วยฟังก์ชั่น scanf( ) สามารถเขียนครั้งเดียวแล้วใช้ รับค่าพร้อมกันให้แก่ ตัวแปรได้หลายๆ ชนิดตัวแปร เช่น

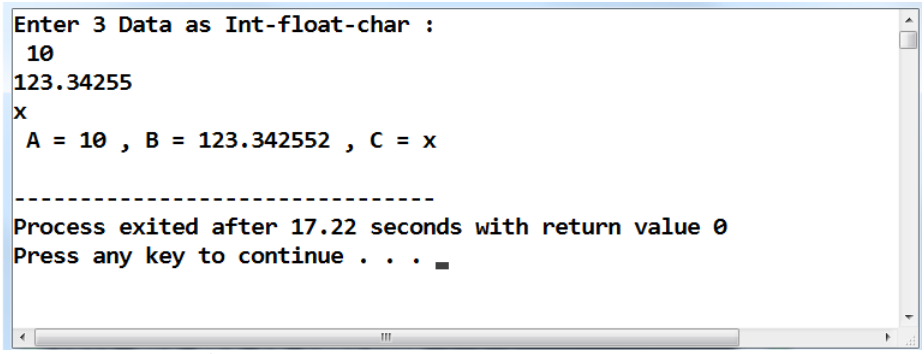
```
int A ;
float B;
char C;
scanf( “ %d %f %c ” , &A ,&B ,&C );
```

หมายความว่า เมื่อโปรแกรมทำงานมาถึง บรรทัดนี้แล้ว โปรแกรมจะทำการหยุดรอรับค่า โดยผู้ใช้งานต้องใส่ ข้อมูลทั้งหมด 3 ครั้ง โดยครั้งแรกจะรับค่าอินพุตเป็นเลขจำนวนเต็มเก็บไว้ในตัวแปร A ครั้งที่ 2 จะรับค่าอินพุตเป็นจำนวนจริงเก็บไว้ในตัวแปร B ครั้ง 3 จะรับข้อมูลเป็นแบบตัวอักขระ 1 ตัวอักษร เก็บไว้ในตัวแปร C

ตัวอย่างโปรแกรมที่ 5.7

```
1. #include <stdio.h>
2. main ( )
3. {      int  A ;
4.        float B ;
5.        char C ;
6.        printf ("Enter 4 Data as Int-float-char : \n ") ;
7.        scanf("%d %f %c", &A , &B , &C ) ;
8.        printf( " A = %d , B = %f , C = %c \n", A, B, C) ;
9. }
```

ผลลัพธ์อย่างที่ 5.7



รูปที่ 5.1 แสดงการใช้ scanf รับค่าพร้อมกันหลายๆตัว

จากรูปที่ 5.1 เห็นได้ว่า การใช้ฟังก์ชั่น scanf( ); เพียงครั้งเดียว สามารถรับค่าข้อมูลอินพุต พร้อมกันหลายตัวแปร โดยแต่ละตัวแปรสามารถกำหนดชนิดของตัวแปรที่แตกต่างกันได้

การรับค่าให้กับตัวแปรแต่ละตัวนั้นจะเกิดขึ้นต่อเมื่อผู้ใช้กดปุ่ม <Enter> เพื่อตอบสนองต่อค่าแต่ละค่าที่ป้อนให้กับคอมพิวเตอร์ ดังนั้นสำหรับคำสั่ง **scanf("%d %f %c", &A, &B, &C)** ซึ่งมีค่าที่จะรับทั้งหมด 3 ค่าจะต้องมีการกดปุ่ม <enter> จำนวน 3 ครั้ง

5.2 ฟังก์ชั่นการคำนวณทางคณิตศาสตร์

การเรียกใช้ฟังก์ชั่นการคำนวณทางคณิตศาสตร์ ต้องกำหนดในส่วนหัวของโปรแกรม หรือเรียกว่า Header file โดยการกำหนด #include <math.h> เพื่อสำหรับการเรียกใช้ฟังก์ชั่นทางการคำนวณคณิตศาสตร์ โดยแสดงตารางที่ 5.5 ดังนี้

ตารางที่ 5.5 ฟังก์ชั่นการใช้งานของ Header Files ของไฟล์ชื่อ math.h

cos	cosh	log10	logb
sin	tanh	modf	scalbn
tan	asinh	exp2	scalbln
acos	atanh	expm1	pow
asin	exp	ilogb	sqrt
atan	ldexp	log1p	cbrt
atan2	log	log2	hypot

เนื่องจากสมการทางคณิตศาสตร์ไม่สามารถเขียนสัญลักษณ์ได้โดยตรง ต้องใช้ฟังก์ชั่นของ math.h โดยฟังก์ชั่นต่างๆ จะมีรูปแบบการใช้งานที่ระบุค่าอินพุตให้กับฟังก์ชั่น ซึ่งการนำสมการทางคณิตศาสตร์มาใช้ในการคำนวณด้วยโปรแกรมจะต้องเปลี่ยนให้อยู่ในรูปของภาษาซี

ตัวอย่างสมการ

$y = \sqrt{x}$ สามารถเขียนในรูปแบบของภาษาซี โดยเรียกใช้ฟังก์ชั่น sqrt หมายถึงการหาค่ารากที่สอง สามารถเขียนได้ y = sqrt (x) ;

$y = \sqrt[3]{x}$ สามารถเขียนในรูปแบบของภาษาซี โดยเรียกใช้ฟังก์ชั่น cbrt หมายถึงการหาค่ารากที่สาม สามารถเขียนได้ y = cbrt (x) ;

$y = |\sin(a)\cos(b)|$ สามารถเขียนในรูปแบบของภาษาซี โดยเรียกใช้ฟังก์ชั่น abs sin และ cos หมายถึงการหาค่าสัมบูรณ์ของการคำนวณ sin(a) คูณกับ cos(b) สามารถเขียนได้ y = abs(sin(a)*cos(b)) ;

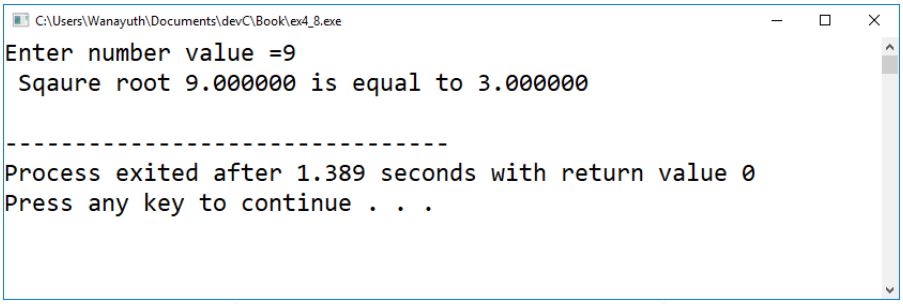
$y = \left|e^{sinx} + 2\sqrt{a^3 + b^3}\right|$ สามารถเขียนในรูปแบบของภาษาซีได้โดยเรียกใช้ฟังก์ชั่น abs exp sin และ sqrt หมายถึงการหาค่าสัมบูรณ์ของการคำนวณ exp ของ sin(x) บวกกับ 2 คูณกับ sqrt ของ pow(a,3) บวกกับ pow(b,3) สามารถเขียนได้ y = abs(exp(sin(x)) + 2*sqrt(pow(a,3) +pow(b,3))) ;

ในการกำหนดฟังก์ชั่นใช้งานจะประกอบด้วยวงเล็บ ดังนั้นควรตรวจสอบวงเล็บให้ครบทุกฟังก์ชั่น ผลลัพธ์การคำนวณจะทำตามลำดับความสำคัญ หากวงเล็บไม่ถูกต้องจะส่งผลให้การคำนวณผิดลำดับและผลลัพธ์มีความผิดพลาดได้

ตัวอย่างโปรแกรมที่5.8 การใช้ฟังก์ชั่นหารากที่สองของตัวแปร

```
1. #include <stdio.h>
2. #include <math.h>
3. main ( )
4. {       float x, out ;
5.         printf ("Enter number value =") ;
6.         scanf("%f", &x ) ;
7.         out = sqrt(x) ;
8.         printf( " Sqaure root %f is equal to %f\n", x, out) ;
9. }
```

ผลลัพธ์อย่างที่ 5.8



รูปที่ 5.2 ผลลัพธ์การทำงานของโปรแกรมที่ 5.8

5.3 ชุดคำสั่ง clear screen หรือคำสั่งล้างหน้าจอภาพ

ในการแสดงข้อความหรือผลลัพธ์ทางหน้าจอเอาท์พุต มีการแสดงผลลัพธ์ข้อมูลหรือข้อความต่างๆ ที่ปรากฏคงเดิมอยู่แล้ว หากต้องการลบข้อความที่แสดงหรือล้างข้อความเดิม ต้องใช้ชุดคำสั่งสำหรับล้างหน้าจอ (clear screen) เพื่อเริ่มต้นใหม่ในบรรทัดแรกของหน้าจอเอาท์พุต

คำสั่ง system ("cls")

ใช้คำสั่ง system("cls") โดยกำหนดให้เรียกใช้ไฟล์ stdlib.h สำหรับโปรแกรม Dev C++ และ ในกรณีใช้คำสั่ง clrscr( ) โดยกำหนดให้เรียกใช้ไฟล์ conio.h สำหรับโปรแกรม Turbo C++

ตัวอย่างการใช้งาน

```
#include <stdlib.h>          //  กำหนดไฟล์เพื่อใช้คำสั่ง clear screen
main ( )                     // โปรแกรมหลัก
{    system("cls") ;         // คำสั่งเคลียร์จอภาพ
     ...
}
```

<u>สรุปท้ายบท</u>

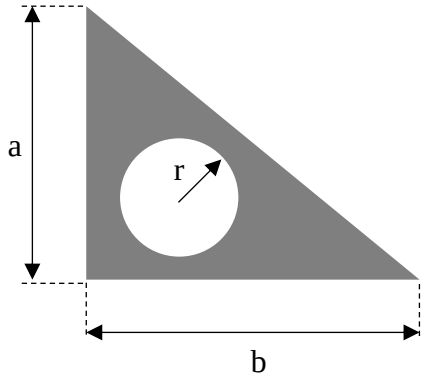
การเรียกใช้ฟังก์ชั่นต่างๆหรือคำสั่ง ในภาษาซี ต้องประกาศส่วนหัวของโปรแกรม (header file) เพื่อสามารถเรียกใช้ฟังก์ชั่นต่างๆ ภายใน header file นั้นๆ หากไม่ได้ระบุฟังก์ชั่นที่เรียกใช้งาน จะไม่สามารถทำงานได้ โปรแกรมไม่สามารถทำงานได้ รวมถึงการกำหนดเครื่องหมายการคำนวณทางคณิตศาสตร์ให้เป็นไปตามรูปแบบภาษาซี และการเปลี่ยนรูปแบบสมการทางคณิตศาสตร์ ให้เป็นสมการสำหรับโปรแกรมภาษาซี

คำถามท้ายบท

1. การกำหนดส่วนหัวของโปรแกรม (Header file) มีความสำคัญอย่างไร ?
2. Header file ที่ใช้สำหรับการควบคุมอินพุตและเอาท์พุตคือ?
3. Header file ที่ใช้สำหรับการคำนวณทางคณิตศาสตร์คือ?
4. ฟังก์ชั่นที่ใช้ในการแสดงผลทางด้านหน้าจอคือคำสั่งใด ?
5. ฟังก์ชั่นที่ใช้ในการรับค่าข้อมูลจากผู้ใช้งานทางแป้นพิมพ์ (Keyboard) คือคำสั่งใด ?
6. หากต้องการแสดงผลข้อความอักษร %EGR205% ทางหน้าจอคอมพิวเตอร์ต้องพิมพ์คำสั่งอย่างไร ?
7. กำหนดให้ลำโพงในคอมพิวเตอร์ส่งเสียง beep สั้นๆ 1 ครั้ง ต้องใช้คำสั่งอย่างไร ?
8. กำหนดให้การเลื่อน cursor ย้อนกลับไป 1 ตัวอักษร ต้องใช้คำสั่งอย่างไร ?
9. กำหนดให้มีการเลื่อน cursor ไปอยู่บรรทัดถัดไป (ขึ้นบรรทัดใหม่) ต้องใช้คำสั่งอย่างไร ?
10. กำหนดให้มีการ tap ในแนวนอน (6 ถึง 8 ตัวอักษร) ต้องใช้คำสั่งอย่างไร ?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมคำนวณหาพื้นที่แรงเงา ตามรูปที่กำหนดให้ต่อไปนี้ โดยตัวแปร a, b และ r รับค่าอินพุตจากผู้ใช้ทางคีย์บอร์ด



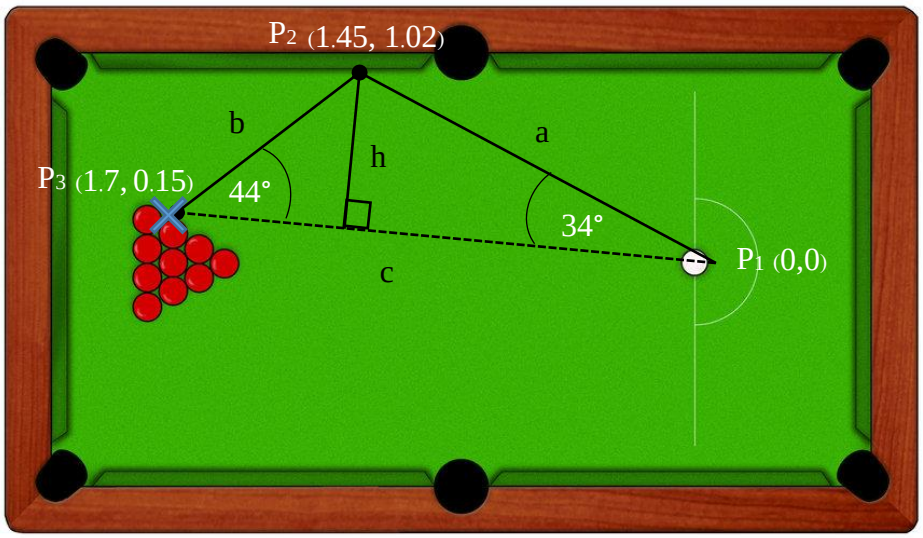
2. จงเขียนโปรแกรมคำนวณผลลัพธ์ของตัวแปร x ที่กำหนดให้ โดยตัวแปร a, b และ c รับค่าอินพุตจากผู้ใช้ทางคีย์บอร์ด

   กำหนดให้

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

3. จากรูปโต๊ะสนุกเกอร์ที่กำหนดให้ ต้องการให้ลูกสีขาวในตำแหน่ง x=0.0 และ y=0.0 ไปกระทบกับลูกสีแดง (เครื่องหมายกากบาท) ในตำแหน่งที่ x=1.7 และ y=0.15 ทำตามรูปที่กำหนดให้ต่อไปนี้ **จงเขียนโปรแกรมคำนวณคำนวณหาค่าของระยะ a , b , c และ h**

กำหนดต่อไปนี้ (Unit : เมตร) กำหนดให้จุด P_1, P_2 และ P_3 ดังรูป



แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 6 ตัวดำเนินการตัดสินใจ

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจตรรกะการเปรียบเทียบ รวมถึงการตรวจสอบเงื่อนไขเชิงลอจิก
- เพื่อให้นักศึกษาเข้าใจกระบวนการเขียนโปรแกรมแบบมีเงื่อนไขทางเลือก

เนื้อหา

- ตัวดำเนินการเงื่อนไข สัญลักษณ์ รูปแบบการใช้งานของโครงสร้างภาษาซี รวมถึงการใช้คำสั่งสำหรับตรวจสอบเงื่อนไขแบบหนึ่งทางเลือก if และแบบสองทางเลือก if-else
- กระบวนการเปรียบเทียบเงื่อนไข ตารางค่าความจริง แบบทางเลือกหนึ่งทางเลือกและหลายทางเลือก

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการตัดสินใจแบบมีเงื่อนไข
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 6

ตัวดำเนินการตัดสินใจ
Decision

บทเรียนนี้ กล่าวถึงตัวดำเนินการตัดสินใจ หรือทิศทางการทำงานของโปรแกรมตามเงื่อนไขที่กำหนด เพื่อให้โปรแกรมตัดสินใจ (Control Statements) การทำงานภายใต้เงื่อนไขที่กำหนดจากผู้ใช้งาน ในการเขียนโปรแกรมในภาษาซี จะมีการใช้คำสั่งที่เป็น Control Statements ต่างๆ โดยอาศัยผลลัพธ์จากการตรวจสอบภาวะต่างๆ ของ Control Statements เพื่อควบคุมลำดับการทำงานของโปรแกรม ให้สามารถเริ่มการทำงานในส่วนที่ต้องการ คือการแยกสาขาคำสั่ง (Branching) ในแต่ละชุดคำสั่ง หรือการเลือกให้คำนวณชุดคำสั่งตามที่กำหนด

ในการตรวจสอบเงื่อนไขภาวะต่างๆ สามารถใช้ตัวดำเนินการเปรียบเทียบ (Rational operators) เครื่องหมายทางคณิตศาสตร์ เช่น เครื่องหมายสัญลักษณ์ < (น้อยกว่า), <= (น้อยกว่าเท่ากับ), > (มากกว่า), >= (มากกว่าเท่ากับ) รวมทั้งเครื่องหมาย equality operators สำหรับเปรียบเทียบ ด้วยเครื่องหมาย == (เท่ากับ), และ != (ไม่เท่ากับ)

การเปรียบเทียบเงื่อนไขนั้น ให้ค่าผลลัพธ์ที่เป็นตรรกะ 2 ค่าผลลัพธ์ คือค่าผลลัพธ์เป็นจริง (True) เท่ากับ 1 และค่าผลลัพธ์เป็นเท็จ (False) เท่ากับ 0 ซึ่งผลการเปรียบเทียบเงื่อนไขสามารถอธิบายได้ดังนี้ ตัวอย่างเช่น

5 > 3 ผลที่ได้เป็น **จริง** เนื่องจาก 5 มากกว่า 3 จริง

4 >= 4 ผลที่ได้เป็น **จริง** เนื่องจาก 4 มากกว่าหรือเท่ากับ 4 จริง

10 != 5 ผลที่ได้เป็น **จริง** เนื่องจาก 10 ไม่เท่ากับ 5 จริง

6 >=15 ผลที่ได้เป็น **เท็จ** เนื่องจาก 6 ไม่ได้มากกว่าหรือเท่ากับ 15

รูปแบบที่ใช้ตรวจสอบเงื่อนไข สำหรับโปรแกรมภาษาซี คำสั่งที่ใช้สำหรับควบคุมทิศทางการทำงานของโปรแกรมมีคำสั่งในการตรวจสอบเงื่อนไข แบ่งเป็นการใช้งานดังนี้

1. if (expression)
2. if (expression) - else
3. if (expression) - else if(expression) - else
4. switch – case

6.1 เครื่องหมายเปรียบเทียบ (Relational Operator)

เครื่องหมายที่สำหรับการเปรียบเทียบ สำหรับเงื่อนไขการตัดสินใจของโปรแกรม ซึ่งผลลัพธ์จากการเปรียบเทียบสามารถจำแนกออกเป็น 2 กรณี คือ เงื่อนไขเป็นจริง โดยให้มีค่าเป็น “1” และเงื่อนไขเป็นเท็จ โดยให้มีค่าเป็น “0” โดยการใช้สัญลักษณ์เครื่องหมาย ให้ตรงตามรูปแบบคำสั่งของภาษาซี แสดงดังนี้

ตารางที่ 6.1 สัญลักษณ์ของการเปรียบเทียบ

สัญลักษณ์	ความหมาย	ตัวอย่าง
>	มากกว่า	A > B (A มากกว่า B เป็นจริง)
≥	มากกว่าหรือเท่ากับ	A >= B (A มากกว่าหรือเท่ากับ B เป็นจริง)
<	น้อยกว่า	A < B (A น้อยกว่า B เป็นจริง)
≤	น้อยกว่าหรือเท่ากับ	A <= B (A น้อยกว่าหรือเท่ากับ B เป็นจริง)
=	เท่ากับ	A == B (A เท่ากับ B เป็นจริง)
≠	ไม่เท่ากับ	A != B (A ไม่เท่ากับ B เป็นจริง)

6.2 เครื่องหมายทางตรรกะ (Logical Operators)

เครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ โดยอาศัยหลายเงื่อนไขที่นำมาเปรียบเทียบ (มากกว่า 2 เงื่อนไข) ซึ่งผลลัพธ์ของแต่ละเงื่อนไข จะนำมาเปรียบเทียบต่อไป โดยเครื่องหมายทางตรรกะสามารถจำแนกออกเป็น 3 รูปแบบ คือ && (AND), || (OR), ! (NOT) อธิบายรายละเอียดดังนี้

เครื่องหมาย && (AND) หมายถึง การนำเงื่อนไขมากกว่า 2 พจน์ มาเปรียบเทียบกัน โดยผลลัพธ์จากการเปรียบเทียบนั้น จะเป็นจริงได้ เมื่อเงื่อนไขที่นำมาเปรียบเทียบทุกเงื่อนไขจะต้องเป็นจริงทั้งหมด แต่หากมีเงื่อนไขใดเงื่อนไขหนึ่งเป็นเท็จ ผลลัพธ์จะเป็นเท็จทุกกรณี แสดงในตารางที่ 6.2

ตัวอย่างเช่น (A > B) && (C > D) จากการเปรียบเทียบเงื่อนไข จะได้ผลลัทธ์ตามตารางต่อไปนี้

ตารางที่ 6.2 ตารางค่าความจริงของ && (AND)

A > B	C > D	(A > B) && (C > D)
เท็จ (0)	เท็จ (0)	เท็จ (0)
เท็จ (0)	จริง (1)	เท็จ (0)
จริง (1)	เท็จ (0)	เท็จ (0)
จริง (1)	จริง (1)	จริง (1)

เครื่องหมาย || (OR) หมายถึง การนำเงื่อนไขมากกว่า 2 พจน์ ขึ้นไปมาเปรียบเทียบกัน ผลลัพธ์จากการเปรียบเทียบนั้น จะเป็น**จริง**ได้ เมื่อเงื่อนไขใดเงื่อนไขหนึ่งเป็น**จริง** แต่ถ้าหากมีเงื่อนไขทั้งสองเงื่อนไขเป็นเท็จ ผลลัพธ์จะเป็น**เท็จ** แสดงในตารางที่ 6.3

ตัวอย่างเช่น (A > B) || (C > D) จากการเปรียบเทียบเงื่อนไข จะได้ผลลัทธ์ตามตารางต่อไปนี้

ตารางที่ 6.3 ตารางค่าความจริงของ || (OR)

A > B	C > D	(A > B) || (C > D)
เท็จ (0)	เท็จ (0)	เท็จ (0)
เท็จ (0)	จริง (1)	จริง (1)
จริง (1)	เท็จ (0)	จริง (1)
จริง (1)	จริง (1)	จริง (1)

เครื่องหมาย ! (NOT) หมายถึง การนำผลลัพธ์ของเงื่อนไขมาทำการกลับสถานะให้ตรงข้ามจากเดิม จากผลลัพธ์ จริง ให้เป็น เท็จ หรือจากผลลัพธ์ เท็จ ให้เป็น จริง แสดงดังตารางที่ 6.4

ตารางที่ 6.4 ตารางค่าความจริงของ ! (NOT)

(A>B)	! (A>B)
จริง (1)	เท็จ (0)
เท็จ (0)	จริง (1)

ตัวอย่างเช่น กำหนดให้ตัวแปรชนิดจำนวนเต็ม a = 3, b = 2, c = 4 , d = -1

1.1. if ((a > b) && (c > a)) คำตอบ ผลลัพธ์เป็น จริง

1.2. if ((2d < d) && (c > a)) คำตอบ ผลลัพธ์เป็น จริง

1.3. if ((c != ab) && (d > b)) คำตอบ ผลลัพธ์เป็น เท็จ

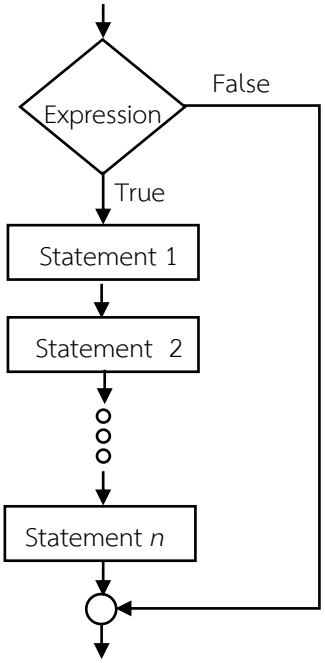
1.4. if ((bc != 0) || (ab > 10)) คำตอบ ผลลัพธ์เป็น จริง

1.5. if ((d < -1) || (a > c)) คำตอบ ผลลัพธ์เป็น เท็จ

การเปรียบเทียบเงื่อนไขสามารถกำหนดสัญลักษณ์ของการเปรียบเทียบได้ตามข้อกำหนดของภาษาซี โดยการเปรียบเทียบระหว่าง ตัวแปร ค่าคงที่ ข้อมูลอักษร โดยการเขียนผังงานเป็นสิ่งสำคัญทำให้สามารถออกแบบโปรแกรมได้อย่างถูกต้องและสามารถเข้าใจถึงรูปแบบการทำงานของกระบวนการตัดสินใจของโปรแกรม

6.3 รูปแบบการตรวจสอบเงื่อนไขด้วย if

การใช้คำสั่ง if จะใช้ในกรณีที่มีทางเลือกการทำงานอยู่ทางเลือกเดียว โดยผลลัพธ์จากการตรวจสอบเงื่อนไข แบ่งเป็น 2 กรณีผลลัพธ์เงื่อนไข เมื่อกรณีเงื่อนไขเป็นจริง จะทำทางเลือกในกลุ่มคำสั่ง Statement1, Statement2,... Statement *n* และกรณีผลลัพธ์เงื่อนไขเป็นเท็จ จะข้ามทางเลือกไปที่จุดสิ้นสุดของคำสั่ง if โดยจะไม่ทำกลุ่มคำสั่งใดๆ โดยผังงานการทำงานของคำสั่ง if สามารถเขียนผังงาน ได้ดังรูปที่ 6.1 และรูปแบบของโปรแกรม



รูปที่ 6.1 ผังงานการทำงานของคำสั่ง if

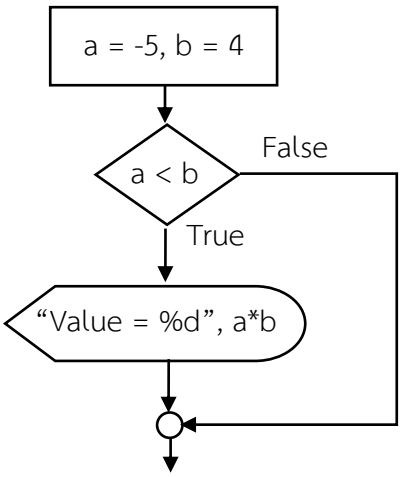
รูปแบบโปรแกรม if

```
if (Expression)                 // การตรวจสอบเงื่อนไข
{                               // เครื่องหมายปีกกาเปิด เริ่มการทำงานในคำสั่ง if
        Statement 1 ;
        Statement 2 ;           // กลุ่มคำสั่งหรือฟังก์ชั่น จะทำงานในส่วนนี้เมื่อ
        ...                     // เงื่อนไขเป็นจริง
        Statement n ;
}                               // เครื่องหมายปีกกาปิด สิ้นสุดการทำงานในคำสั่ง if
```

ตัวอย่างการตรวจสอบเงื่อนไขด้วยคำสั่ง if

6.1 กำหนดให้จำนวนเต็ม a = -5 และ b = 4 ทำการตรวจสอบเงื่อนไขของตัวแปรที่กำหนด a<b ถ้าเงื่อนไขเป็นจริงให้คำนวณหาผลคูณของตัวแปร a และ b

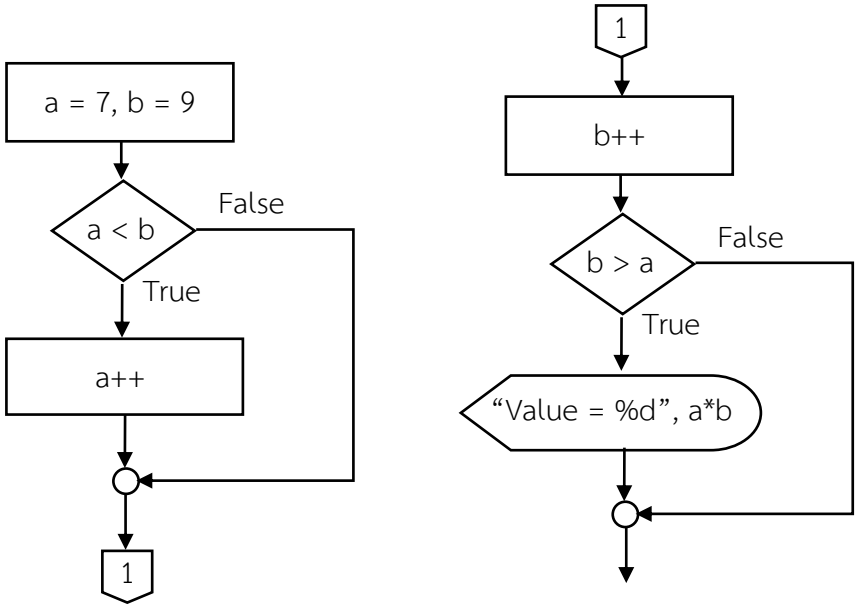
ผังงาน



ผลลัพธ์ **Value = -20**

6.2 กำหนดให้จำนวนเต็ม a = 7 และ b = 9 ทำการตรวจสอบเงื่อนไขของตัวแปรที่กำหนด a < b และ a ≠ b โดยตรวจสอบเงื่อนไขที่กำหนดให้ เมื่อเงื่อนไขเป็นจริงจะทำคำสั่งที่กำหนดให้

ผังงาน

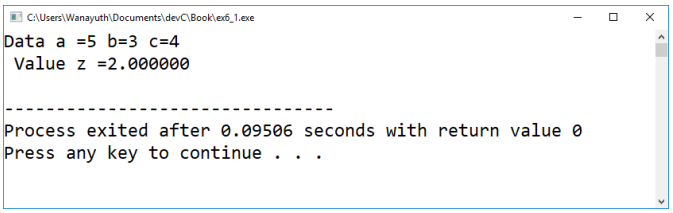


ผลลัพธ์ **Value = 80**

ตัวอย่างโปรแกรมที่ 6.1 การใช้คำสั่ง if ตรวจสอบเงื่อนการคำนวณรากที่สอง

```
1.  #include <stdio.h>                                   // เรียกใช้ไลบรารี stdio.h
2.  #include <math.h>                                    // เรียกใช้ไลบรารี math.h
3.  main( )                                  // เริ่มต้นการทำงานของโปรแกรมหลัก
4.  {       int a = 5 , b =3 , c =4 ;                   // ประกาศตัวแปรพร้อมระบุค่า
5.          printf("Data a =%d b=%d c=%d \n", a, b, c) ;  //แสดงผลลัพธ์ทางหน้าจอ
6.          if ( a + b < 10 )                           // ตรวจสอบเงื่อนไข
7.          {       float z = sqrt(c) ;                 // คำนวณหารากที่สองของตัวแปร c
8.                  printf(" Value z =%f \n", z ) ;     //แสดงผลลัพธ์ตตัวแปร z ทางหน้าจอ
9.          }                                           // สิ้นสุดการทำงานของ if
10. }                                                   // จบการทำงานของโปรแกรม
```

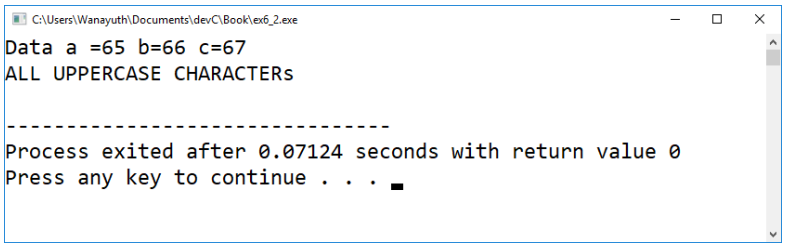
ผลลัพธ์โปรแกรมที่ 6.1



ตัวอย่างโปรแกรมที่ 6.2 การใช้คำสั่ง if ตรวจสอบตัวอักษร

```
1.  #include <stdio.h>                                   //เรียกใช้ไลบรารี stdio.h
2.  main( )                                             // เริ่มต้นการทำงานของ
    โปรแกรมหลัก
3.  {       char a = 'A' , b ='B' , c = 'C' ;           // ประกาศตัวแปร
4.          printf("Data a =%d b=%d c=%d \n", a, b, c) ;  // แสดงผลข้อมูล
5.          if (a != 'a' &&  b != 'b' && c != 'c')      // ตรวจสอบเงื่อนไข
6.          {       printf("ALL UPPERCASE CHARACTERs\n ");  // แสดงผลข้อมูล
7.          }                                           // สิ้นสุดการทำงานของ if
8.  }                                                   // จบการทำงานของโปรแกรม
```

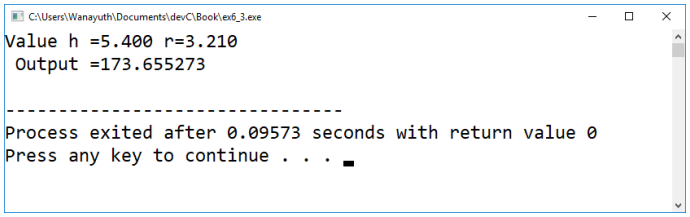
ผลลัพธ์โปรแกรมที่ 6.2



ตัวอย่างโปรแกรมที่ 6.3

```
1. #include <stdio.h>                                          // เรียกใช้ไลบรารี stdio.h
2. #include <math.h>                                          // เรียกใช้ไลบรารี math.h
3. main( )                                                    // เริ่มต้นการทำงานของ
โปรแกรมหลัก
4. {        float h = 5.4 , r =3.21 , pi = 3.141592 ;         // ประกาศตัวแปรพร้อมระบุค่า
5.          printf("Data h =%.3f r=%.3f \n", h, r ) ;         // แสดงผลข้อมูล
6.          if ( h > 0 && r > 3.0 )                           // ตรวจสอบเงื่อนไข
7.          {       float z = 2 * pi * r * h + 2 * pi * r * r ; // คำนวณหาพื้นที่ผิวทรงกระบอก
8.                  printf(" Output =%f \n", z ) ;            // แสดงผลลัพธ์ตัวแปร z ทางหน้าจอ
9.          }                                                 // จบการทำงานของ if
10. }                                                         // จบการทำงานของโปรแกรม
```

ผลลัพธ์โปรแกรมที่ 6.3



ตัวอย่างโปรแกรมที่ 6.4

```
1.  #include <stdio.h>                                         // เรียกใช้ไลบรารี stdio.h
2.  #include <math.h>                                         // เรียกใช้ไลบรารี math.h
3.  main( )                                                   // เริ่มต้นการทำงานของโปรแกรมหลัก
4.  {       int q = -1 , w = 12 ;    float  e = 1.5 , z ;     // ประกาศตัวแปรพร้อมระบุค่า
5.          printf("Data q =%d w=%d e=%.3f\n", q, w, e) ; // แสดงผลข้อมูล
6.          if ( abs(q*w) > abs( q + w /2 ) )                 // ตรวจสอบเงื่อนไข
7.          {       z =  q + w + e;                           // คำนวณผลรวมของตัวแปร q ,w
    และ e
8.                  printf(" Output =%f \n", z ) ;            // แสดงผลข้อมูลการคำนวณ
9.          }                                                 // จบการทำงานของ if
10. }                                                         // จบการทำงานของโปรแกรม
```

ผลลัพธ์โปรแกรมที่ 6.4

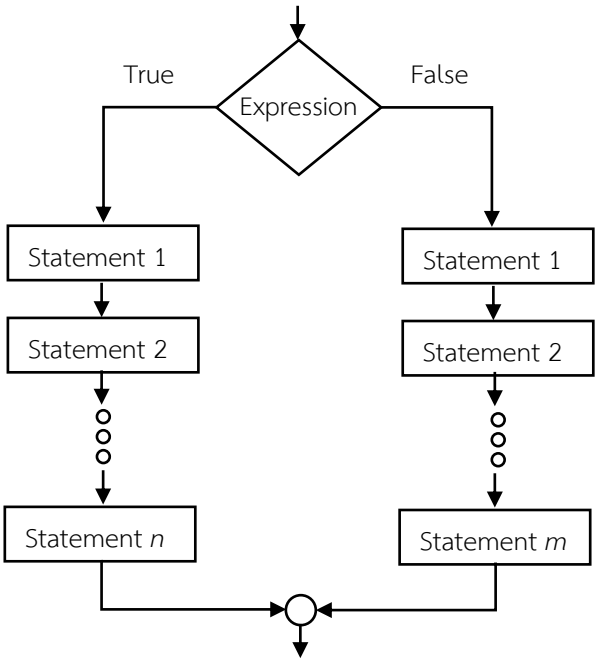
```
C:\Users\Wanayuth\Documents\devC\Book\ex6_4.exe
Data q =-1 w=12 e=1.500
 Output =12.500000

--------------------------------
Process exited after 0.07897 seconds with return value 0
Press any key to continue . . .
```

6.4 การใช้รูปแบบการตรวจสอบเงื่อนไขด้วย if-else

การตรวจสอบเงื่อนไขแบบ if จะทำเฉพาะทางเลือกของผลลัพธ์เงื่อนไขเป็นจริงเท่านั้น จึงทำให้การใช้งานคำสั่ง if มีขอบเขตในการใช้งาน ที่ใช้สำหรับการตรวจสอบเงื่อนไขแบบทางเดียว หากใช้การตรวจสอบเงื่อนไขที่มีทางเลือกสองทางเลือก สามารถใช้คำสั่ง if-else โดยผลลัพธ์เงื่อนไขทางเลือกเป็นจริงและผลลัพธ์เงื่อนไขทางเลือกเป็นเท็จ จะทำคำสั่งหรือกลุ่มคำสั่งของแต่ละทางเลือกนั้นๆ

การใช้คำสั่ง if-else สำหรับการใช้ทางเลือกสองทางเลือก จะใช้ในกรณีที่ต้องการให้ทำชุดคำสั่งของการตรวจสอบเงื่อนไขผลลัพธ์ที่เป็นจริงจะทำคำสั่งในกลุ่มของทางเลือกที่เป็นจริง และเป็นเท็จจะทำคำสั่งในกลุ่มของทางเลือกที่เป็นเท็จ โดยรูปแบบผังงาน ของการทำงานคำสั่ง if-else สามารถเขียนผังงานแสดงดังรูปที่ 6.2 นี้



รูปที่ 6.2 ผังงานการทำงานของคำสั่ง if-else

รูปแบบโปรแกรม if-else

```
if ( Expression )                    // การตรวจสอบเงื่อนไข
{       Statement 1 ;                // เริ่มต้นการทำงานของกลุ่มเงื่อนไขเป็นจริง
        Statement 2 ;
        ...
        Statement n ;
}                                    // สิ้นสุดการทำงานของกลุ่มเงื่อนไขเป็นจริง
else                                 // เมื่อเงื่อนไขเป็นเท็จจะทำกลุ่มคำสั่งของ else
{       Statement 1 ;                // เริ่มต้นการทำงานของกลุ่มเงื่อนไขเป็นเท็จ
        Statement 2 ;
        ...
        Statement m ;                // สิ้นสุดการทำงานของกลุ่มเงื่อนไขเป็นเท็จ
}
```

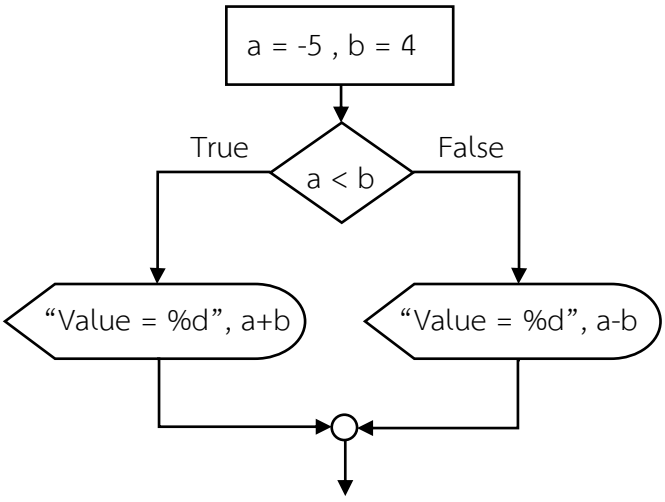
ตัวอย่างผังงานของคำสั่ง if-else

6.3 กำหนดให้จำนวนเต็ม a = -5 และ b = 4 ทำการตรวจสอบเงื่อนไขของตัวแปรที่กำหนด a<b

ถ้า <u>เงื่อนไขเป็นจริง</u> ให้คำนวณหาผลบวกของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

ถ้า <u>เงื่อนไขเป็นเท็จ</u> ให้คำนวณหาผลลบของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

<u>ผังงาน</u>



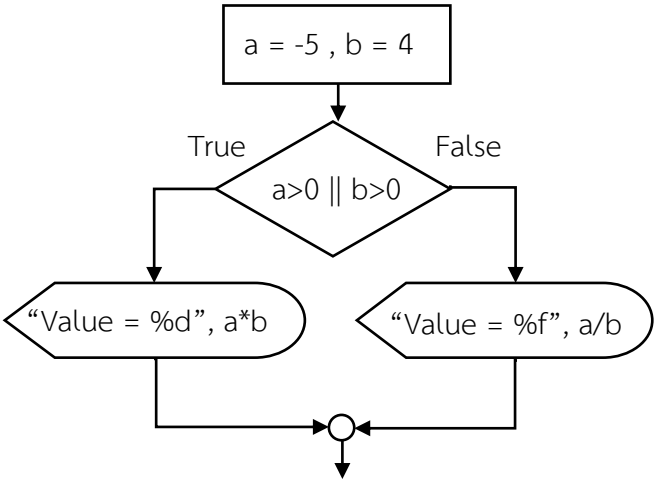
<u>ผลลัพธ์</u> **Value = -1**

6.4 กำหนดให้จำนวนเต็ม a = -5 และ b = 4 ทำการตรวจสอบเงื่อนไขของตัวแปรที่กำหนด a>0 || b>0

ถ้า เงื่อนไขเป็นจริง ให้คำนวณหาผลคุณของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

ถ้า เงื่อนไขเป็นเท็จ ให้คำนวณหาผลหารของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

ผังงาน



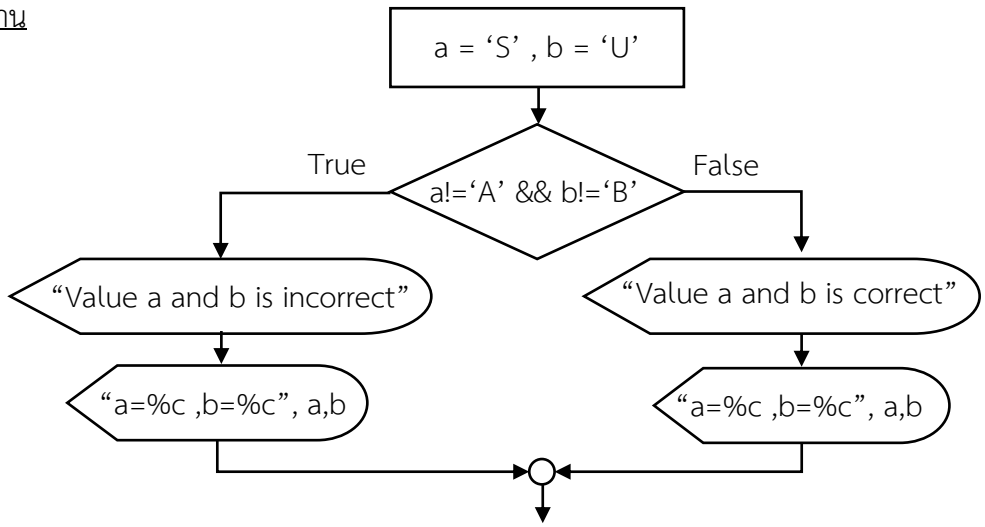
ผลลัพธ์ **Value = -20** และถ้ากำหนดให้ตัวแปร a = -12 และ b = -3 ผลลัพธ์จะเท่ากับ **Value = -4.000000**

6.5 กำหนดให้จำนวนเต็ม a = -5 และ b = 4 ทำการตรวจสอบเงื่อนไขของตัวแปรที่กำหนด a>0 || b>0

ถ้า เงื่อนไขเป็นจริง ให้คำนวณหาผลคุณของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

ถ้า เงื่อนไขเป็นเท็จ ให้คำนวณหาผลหารของตัวแปร a และ b พร้อมแสดงผลลัพธ์ทางหน้าจอภาพ

ผังงาน



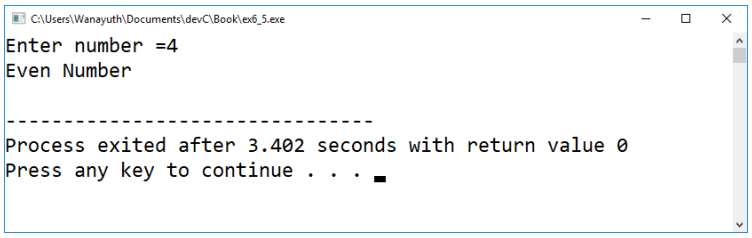
ผลลัพธ์ **Value a and b is incorrect a=S ,b=U** และถ้ากำหนดให้ตัวแปร a = ‘A’ และ b = ‘B’ ผลลัพธ์จะเท่ากับ **Value a and b is correct a=A ,b=B**

ตัวอย่างโปรแกรมที่ 6.5 การตรวจสอบตัวเลขจำนวน เลขคู่ หรือ เลขคี่ โดยรับค่าอินพุตจากแป้นพิมพ์

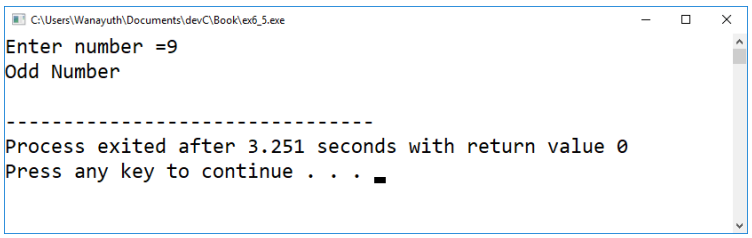
```
1.  #include <stdio.h>                          // เรียกใช้ไลบรารี stdio.h
2.  #include <math.h>                           // เรียกใช้ไลบรารี match.h
3.  main( )                                     // เริ่มต้นการทำงานของโปรแกรมหลัก
4.  {       int num = 0;                        // ประกาศตัวแปรใช้งาน
5.          printf("Enter number =") ;          // แสดงข้อความอักษร
6.          scanf( "%d",&num)                   // รับค่าอินพุตทางแป้นพิมพ์ไว้ในตัวแปร num
7.          if (num%2 == 0)                     // ตรวจสอบเงื่อนไข
8.          {                                   // เริ่มต้นการทำงานเมื่อเงื่อนไขเป็นจริง
9.                  printf("Even number\n");    // แสดงข้อความอักษร
10.         }                                   // สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นจริง
11.         else                                // เงื่อนไขเป็นเท็จให้มาทำในส่วนของ else
12.         {                                   // เริ่มต้นการทำงานเมื่อเงื่อนไขเป็นเท็จ
13.                 printf("Even number\n");    // แสดงข้อความอักษร
14.         }                                   // สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นเท็จ
15. }                                           // สิ้นสุดการทำงานโปรแกรมหลัก
```

ผลลัพธ์โปรแกรมที่ 6.5

เมื่อกำหนดรับค่าอินพุตทางแป้นพิมพ์เลขคู่เท่ากับ 4 ผลลัพธ์แสดงข้อความ Even Number



เมื่อกำหนดรับค่าอินพุตทางแป้นพิมพ์เลขคี่เท่ากับ 9 ผลลัพธ์แสดงข้อความ Even Number



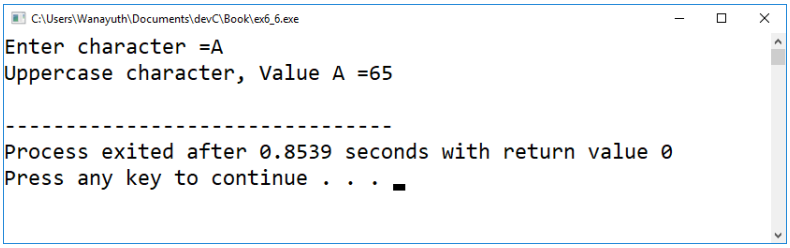
ตัวอย่างโปรแกรมที่ 6.6 โปรแกรมตรวจสอบตัวอักษรพิมพ์เล็กหรือตัวพิมพ์ใหญ่ โดยรับค่าอินพุตจากแป้นพิมพ์

```
1.  #include <stdio.h>                          // เรียกใช้ไลบรารี stdio.h
2.  #include <math.h>                          // เรียกใช้ไลบรารี math.h
3.  main( )                                    // เริ่มต้นการทำงานของโปรแกรมหลัก
4.  {       char ch ;                          // ประกาศตัวแปรใช้งาน
5.          printf("Enter character =") ;      // แสดงข้อความอักษร
6.          scanf("%c",&ch);                   // รับค่าอินพุตทางแป้นพิมพ์ไว้ในตัวแปร ch
7.          if (ch >=65 && ch <= 90)           // ตรวจสอบเงื่อนไข
8.          {                                  // เริ่มต้นการทำงานเมื่อเงื่อนไขเป็นจริง
9.                printf("Uppercase character, Value %c =%d\n",ch, ch);  //แสดงผลข้อความ
10.                                                                      //และตัวแปร
11.         }                                  // สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นจริง
12.         else                               // เงื่อนไขเป็นเท็จให้มาทำในส่วนของ else
13.         {                                  // เริ่มต้นการทำงานเมื่อเงื่อนไขเป็นเท็จ
14.          printf("Please insert only uppercase!!!\n");   // แสดงข้อความอักษร
15.         }                                  // สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นเท็จ
16. }                                          // สิ้นสุดการทำงานโปรแกรมหลัก
```

ผลลัพธ์โปรแกรมที่ 6.6

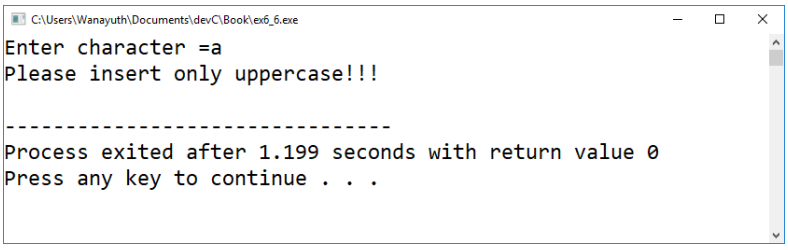
เมื่อกำหนดรับค่าอินพุตทางแป้นพิมพ์อักษรพิมพ์ใหญ่ A

ผลลัพธ์แสดงข้อความ Uppercase character, value A = 65



เมื่อกำหนดรับค่าอินพุตทางแป้นพิมพ์อักษรพิมพ์ใหญ่ a

ผลลัพธ์แสดงข้อความ Please insert only uppercase!!!



สรุปท้ายบท

ตัวดำเนินการตัดสินใจหรือเงื่อนไข if, if-else เป็นการตรวจสอบภายใต้เงื่อนไขและขอบเขตที่กำหนด เพื่อให้โปรแกรมดำเนินการตามทางเลือกของผลลัพธ์เงื่อนไขเป็นจริง หรือ ผลลัพธ์เงื่อนไขเป็นเท็จ ซึ่งแสดงถึงทางเลือกของโปรแกรมเมื่อเงื่อนไขเป็นจริง จะทำให้ส่วนของกลุ่มคำสั่งต่างๆที่อยู่ภายใต้เครื่องหมายปีกกาเปิดปิด หรือถ้าเงื่อนไขเป็นเท็จ จะทำให้ส่วนของกลุ่มคำสั่งต่างๆที่อยู่ภายใต้เครื่องหมายปีกกาเปิดปิด โดยโปรแกรมจะเลือกดำเนินการตามเงื่อนไขที่กำหนด

คำถามท้ายบท

1. สัญลักษณ์เงื่อนไขการเปรียบเทียบมีกี่ประเภท อะไรบ้าง?
2. ในการตรวจสอบเงื่อนไขเมื่อเป็นจริง ให้ทำคำสั่งใดๆ และหากเงื่อนไขเป็นเท็จ **ไม่**ทำคำสั่งใดๆ จะต้องเขียนใช้งานการตรวจสอบแบบใด?
3. ในการตรวจสอบเงื่อนไขเมื่อเป็นจริง ให้ทำคำสั่งใดๆ และหากเงื่อนไขเป็นเท็จ ทำกลุ่มคำสั่งใดๆ จะต้องเขียนใช้งานการตรวจสอบแบบใด?
4. สัญลักษณ์เงื่อนไขการเปรียบเทียบเงื่อนไขมากกว่า 2 เงื่อนไข มีอะไรบ้าง?
5. สัญลักษณ์ในการตรวจสอบเงื่อนไขแบบใดที่กลับสถานะเอาต์พุต?
6. เมื่อการเปรียบเทียบเงื่อนไขมากกว่า 2 เงื่อนไข หากต้องการให้ผลลัพธ์เงื่อนไขเป็นจริง **เท่านั้น** ต้องใช้สัญลักษณ์ใด?

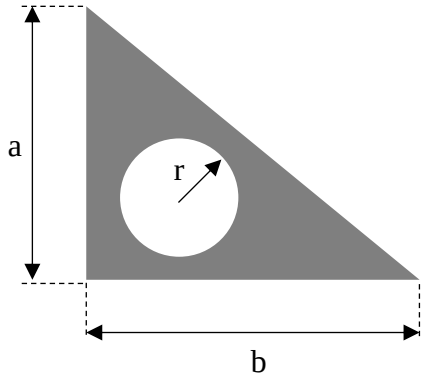
แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมคำนวณหาพื้นที่แรงเงา ตามรูปที่กำหนดให้ต่อไปนี้ โดยตัวแปร a, b และ r รับค่าอินพุตจากผู้ใช้ทางคีย์บอร์ด

   <u>กำหนดเงื่อนไข</u> ตัวแปร a และ b มีค่ามากกว่า r

   ถ้าผลลัพธ์เงื่อนไขเป็น **จริง** ให้คำนวณหาผลลัพธ์ของพื้นที่แรเงา และแสดงผลลัพธ์ทางหน้าจอ

   ถ้าผลลัพธ์เงื่อนไขเป็น **เท็จ** ให้แสดงขอความ **“Please insert values of a and b more than r value”**



2. จงเขียนโปรแกรมคำนวณผลลัพธ์ของตัวแปร ***x*** ที่กำหนดให้ โดยตัวแปร a, b และ c รับค่าอินพุตจากผู้ใช้ทางคีย์บอร์ด

   <u>กำหนดเงื่อนไข</u> $b^2 - 4ac \neq 0$ โดย a ,b, และ $c \neq 0$

   ถ้าผลลัพธ์เงื่อนไขเป็น **จริง** ให้คำนวณหาผลลัพธ์ของ x และแสดงผลลัพธ์ทางหน้าจอ

   ถ้าผลลัพธ์เงื่อนไขเป็น **เท็จ** ให้แสดงขอความ **“Please insert values of a, b and c more than zero”**

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 7 ตัวดำเนินการตัดสินใจหลายเงื่อนไข

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจตรรกะการเปรียบเทียบ รวมถึงการตรวจสอบเงื่อนไขเชิงลอจิก
- เพื่อให้นักศึกษาเข้าใจกระบวนการเขียนโปรแกรมแบบมีเงื่อนไขทางเลือกหลากทางเลือก

เนื้อหา

- ตัวดำเนินการเงื่อนไข สัญลักษณ์ รูปแบบการใช้งานของโครงสร้างภาษาซี รวมถึงการใช้คำสั่งสำหรับตรวจสอบเงื่อนไขแบบหนึ่งทางเลือก if-else if
- กระบวนการเปรียบเทียบเงื่อนไข ตารางค่าความจริง แบบหลายทางเลือก

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการตัดสินใจแบบมีเงื่อนไข
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 7

ตัวดำเนินการตัดสินใจแบบหลายเงื่อนไข

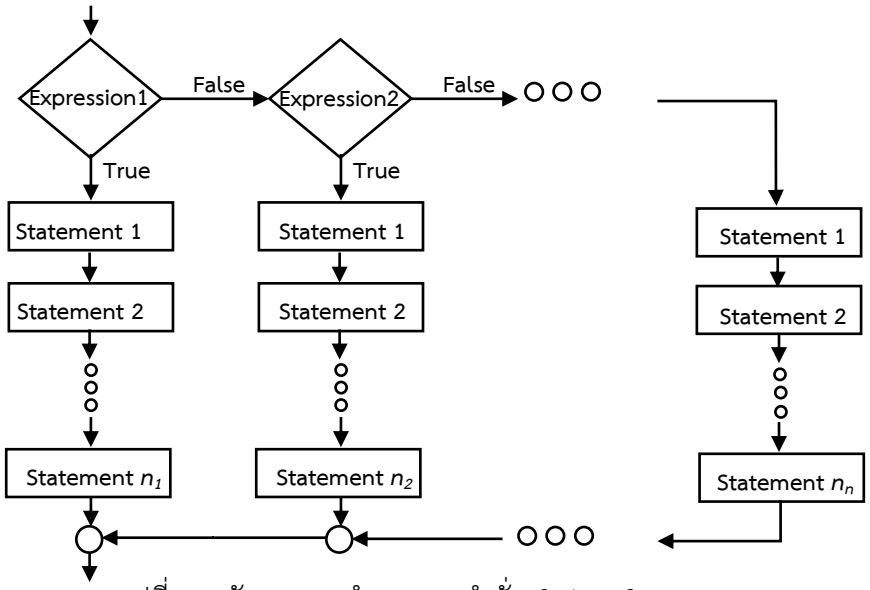
Multi-Decision

บทเรียนนี้ จะต่อเนื่องจากบทเรียนที่ผ่านมา จะกล่าวถึงตัวดำเนินการตัดสินใจแบบหลายทางเลือก โดยมีเงื่อนไขในการตรวจสอบหลายเงื่อนไขในหนึ่งชุดคำสั่ง ซึ่งเรียกว่า if-else if ,...., else โดยสามารถกำหนดตัวดำเนินการต่างๆ และเงื่อนไขที่ต้องการตรวจสอบได้หลายเงื่อนไข ภายใต้รูปแบบที่ของ if-else if สามารถกำหนดเงื่อนไขขึ้นอยู่กับดำเนินการของเงื่อนไขที่กำหนด

7.1 รูปแบบการตรวจสอบเงื่อนไขโดย if- else if -else

การใช้รูปแบบการตรวจสอบเงื่อนไขจะดำเนินการตรวจสอบเงื่อนไขแรก โดยใช้คำสั่ง if โดยผลลัพธ์จากการตรวจสอบเงื่อนไขใน Expression1 โดยผลลัพธ์จากการตรวจสอบเงื่อนไขแบ่งเป็น 2 กรณี คือ 1) กรณีเงื่อนไขเป็น**จริง** จะทำทางเลือกในกลุ่มคำสั่ง Statement1, Statement2,... Statement n_1 และ 2) กรณีผลลัพธ์เงื่อนไขเป็น**เท็จ** จะไปตรวจสอบเงื่อนไขที่สอง โดยเงื่อนไขที่สอง จะใช้เขียนรูปแบบ else if โดยจะทำหน้าที่ตรวจสอบเงื่อนไขใน Expression2 โดยผลลัพธ์จากการตรวจสอบเงื่อนไขแบ่งเป็น 2 กรณี คือ 1) กรณีเงื่อนไขเป็น**จริง** จะทำทางเลือกในกลุ่มคำสั่ง Statement1, Statement2,... Statement n_2 และ 2) กรณีผลลัพธ์เงื่อนไขเป็น**เท็จ** จะดำเนินการตรวจสอบเงื่อนไขที่ต่อไป โดยเงื่อนไขต่อไปจะใช้ else if เท่านั้น สามารถเพิ่มเงื่อนไขได้อย่างอิสระ ในกรณีที่สิ้นสุดเงื่อนไข นอกเหนือจากเงื่อนไขใดๆ สามารถระบุได้โดยใช้คำสั่ง else และกลุ่มคำสั่งภายใต้ Statement n_n โดยผังงานการทำงานของคำสั่ง if -else if -else if -else สามารถเขียนผังงาน ได้ดังรูปที่ 7.1 และรูปแบบของโปรแกรม

ผังงานการทำงานของคำสั่ง if- else if-else สามารถเขียนได้ดังนี้



รูปที่ 7.1 ผังงานการทำงานของคำสั่ง if-else if

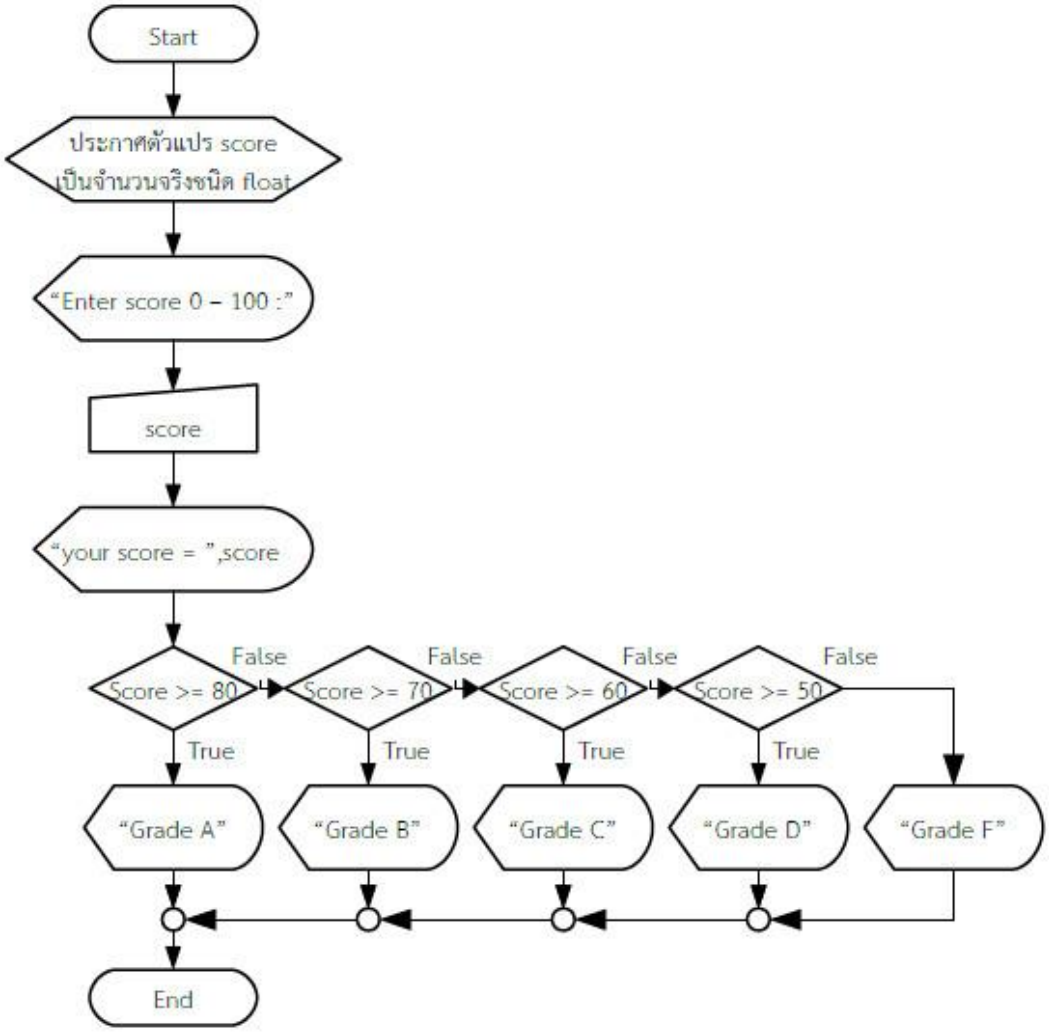
รูปแบบโปรแกรม

```
if ( Expression 1)
{       Statement 1 ;
        Statement 2 ;
        ...
        Statement n1 ;
} else if (Expression 2)
{       Statement 1 ;
        Statement 2 ;
        ...
        Statement n2 ;
} else
{       Statement 1 ;
        Statement 2 ;
        ...
        Statement nn ;
}
```

ตัวอย่างที่ 7.1 จงเขียนโปรแกรมคำนวณหาผลเกรดของนักศึกษา โดยกำหนดการรับค่าข้อมูลคะแนนผ่านทางคีย์บอร์ดผู้ใช้ ตามตารางที่กำหนดให้

ช่วงคะแนน	เกรด
มากกว่า 80	A
70 – 79	B
60 – 69	C
50 – 59	D
น้อยกว่า 50	F

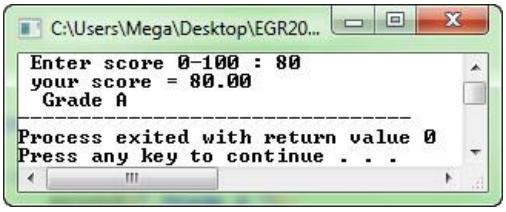
ตัวอย่างผังงานที่ 7.1



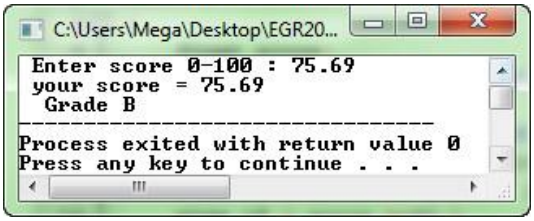
ตัวอย่างโปรแกรมที่ 7.1

```
1.  #include <stdio.h>
2.  main( )
3.  {       float score;
4.          printf (" Enter score 0-100 : ");
5.          scanf("%f",&score);
6.          printf(" your score = %5.2f \n ",score);
7.          if ( score >=80 )          printf(" Grade A ");
8.          else if ( score >=70 )     printf(" Grade B ");
9.          else if ( score >=60 )     printf(" Grade C ");
10.         else if ( score >=50 )     printf(" Grade D ");
11.         else                       printf(" Grade F ");
12. }
```

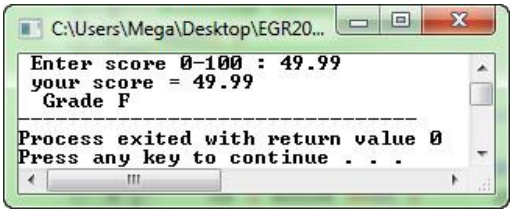
ผลการทดสอบโปรแกรมครั้งที่ 1 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ 80



ผลการทดสอบโปรแกรมครั้งที่ 2 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ 75.69



ผลการทดสอบโปรแกรมครั้งที่ 3 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ 49.99

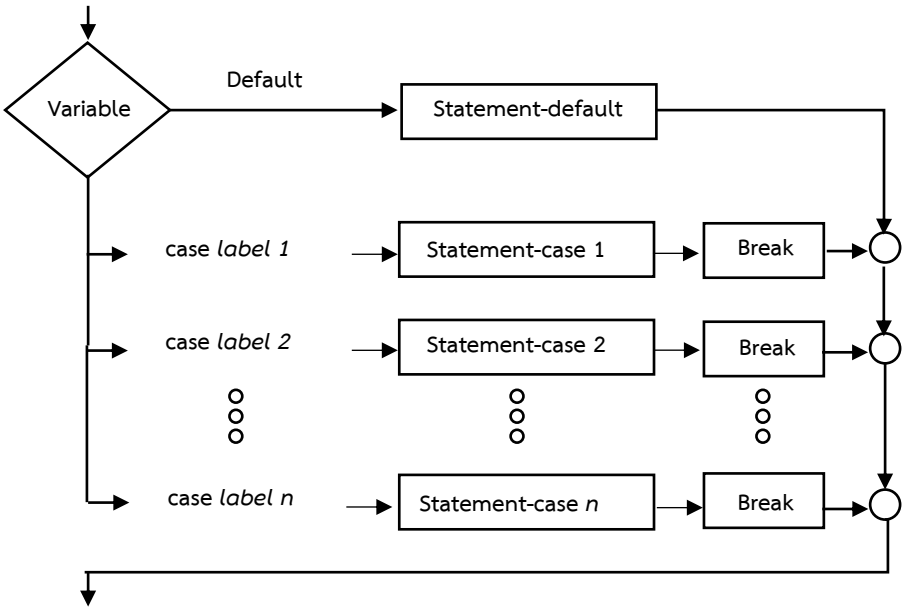


จากการทดสอบโปรแกรมเห็นได้ว่า เมื่อกำหนดค่าตัวเลขลงไป โปรแกรมจะทำการคำนวณหาเกรดที่กำหนดไว้แล้วแสดงออกมาว่า คะแนน ช่วงไหน ได้เกรดอะไร เป็นต้น

7.2 คำสั่ง Switch-Case

การใช้งานลักษณะของ switch case จะแตกต่างจากการตรวจสอบเงื่อนไข แบบ if elseif ในเนื้อหาที่ผ่านมา ซึ่งในรูปแบบของ switch case จะเป็นการกำหนดแบบทางเลือก โดยลักษณะทางเลือกตรงกับ case ที่กำหนดไว้ ด้วยรูปแบบลักษณะการทำงานของ switch case นี้จะทำงานได้รวดเร็วกว่าการตรวจสอบแบบอื่นๆ

ในการเขียน swtich-case สามารถเขียนตามรูปแบบโครงสร้างของภาษาซี การทดสอบแบบหลายทางเลือกได้ โดยใช้ switch - case แสดงดังผังงานการทำงานของคำสั่ง switch-case ดังนี้



รูปที่ 7.2 ผังงานการทำงานของคำสั่ง switch case

รูปแบบโปรแกรมของ switch-case

```
switch ( variable )
{       case  label 1  :  statement-case1 ; break ;
        case  label 2  :  statement-case 2 ;       break ;
        ...
        case  label n  :  statement-case n  ;      break ;
        default :         statement-default  ;
}
```

คำอธิบายโปรแกรม

variable คือ ตัวแปรที่ต้องการนำไปตรวจสอบใน case ต่างๆที่ต้องการระบุ

label คือ การระบุค่าคงที่หรือตัวอักษร สำหรับการตรวจสอบของแต่ละ case

statement-case คือ คำสั่งหรือกลุ่มคำสั่ง ภายในแต่ละ case ซึ่งจะทำงานภายใน case นั้น

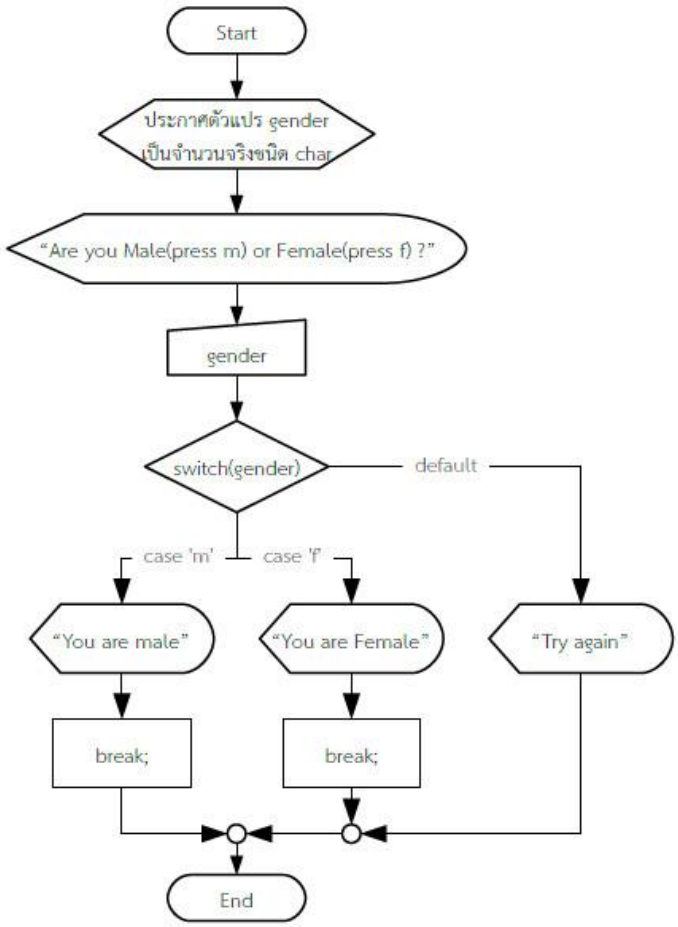
break คือ คำสั่งในการหยุดการทำงานหรือสิ้นสุดการทำงานของ case นั้นๆ

default คือ นอกเหนือจาก case ที่กำหนดโปรแกรมจะทำงานในส่วนนี้

การทำงานของ switch-case จะทำการทดสอบตัวแปรในวงเล็บหลังคำสั่ง switch โดยนำค่าในตัวแปรนั้นไปเปรียบเทียบกับ label ของแต่ละ case และหากไม่พบว่าตรงกับ label ใดเลย โปรแกรมก็จะไปทำงานที่ default ในกรณีที่พบว่า case ใดเป็นจริงโปรแกรมก็จะทำการดำเนินการใน statement ของ case นั้นๆ และเพื่อเป็นการแยก case แต่ละ case ออกจากกัน จะใช้คำสั่ง break เพื่อเป็นการแสดงจุดสิ้นสุดของแต่ละ case โดยหลังที่โปรแกรมทำงานใน case ใดแล้วก็จะอาศัยคำสั่ง break นี้เพื่อข้ามส่วนที่เหลือของโปรแกรมไปสู่จุดสิ้นสุดของ switch-case ที่เครื่องหมาย } (ข้อสังเกตุความแตกต่างจาก if ที่จะดำเนินการเพียงหนึ่งโปรแกรมภายใต้เครื่องหมายปีกกาเปิดปิดเท่านั้น { } ถ้าเงื่อนไขเป็นจริง)

ตัวอย่างที่ 7.2 จงเขียนผังงานและโปรแกรมเพื่อระบุอักษรย่อของผู้ใช้งานเป็นเพศชายหรือเพศหญิง โดยใช้รูปแบบของ switch –case กำหนดให้ ถ้าอักษร m (ตัวพิมพ์เล็ก) คือเพศชาย ถ้าอักษร f (ตัวพิมพ์เล็ก) คือเพศหญิง และถ้าอักษรอื่นๆที่นอกเหนือจากนี้ โปรแกรมจะแสดงคำว่า “ **Try again !!!!** ”

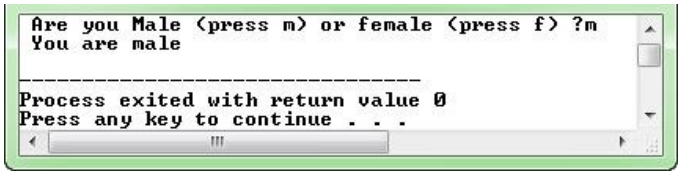
ตัวอย่างผังงานที่ 7.2



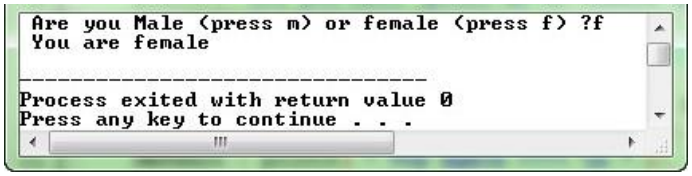
ตัวอย่างโปรแกรมที่ 7.2

```
1.    #include <stdio.h>
2.    main( )
3.    {       char gender ;
4.            printf (" Are you Male (press m) or female (press f) ?");
5.            scanf("%c", &gender);
6.            switch (gender)
7.            {
8.                    case 'm' : printf( " You are male \n " );  break ;
9.                    case 'f' : printf( " You are female \n " );  break ;
10.                   default : printf( " Try again !!!! \n " );
11.           }
12.   }
```

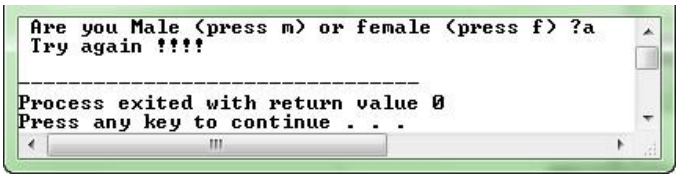
ผลการทดสอบโปรแกรมครั้งที่ 1 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ ‘**m**’



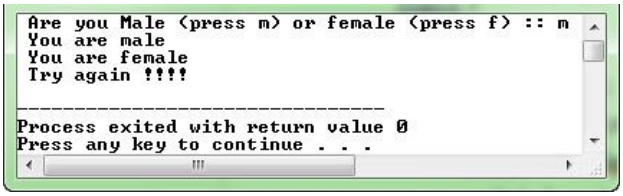
ผลการทดสอบโปรแกรมครั้งที่ 2 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ ‘**m**’



ผลการทดสอบโปรแกรมครั้งที่ 3 กำหนดค่าอินพุตทางคีย์บอร์ดเท่ากับ ‘**a**’



จากโปรแกรมของตัวอย่างที่ 7.2 โปรแกรมจะทำการรับค่ามา 1 ตัวอักษร และทำการเปรียบเทียบดูว่าเป็น m หรือ f หรือไม่ ถ้าเป็น m จะแสดงข้อความ “You are male” ถ้าเป็น f จะแสดงข้อความ “You are female” แต่ถ้าเป็นตัวอักษรอื่นๆที่นอกเหนือจากนั้น โปรแกรมจะแสดงข้อวาม “Try again !!!! ”



คำสั่ง break ; ที่ท้ายบรรทัด case ทั้งสองบรรทัดนั้น จะเป็นตัวสั่งให้หยุดทำคำสั่งในส่วนของ case นั้นๆ แล้วออกนอกคำสั่ง switch ทันที ซึ่งถ้าไม่มีคำสั่ง break ; ไว้โปรแกรมนั้นจะทำคำสั่ง case ที่อยู่ถัดไปตามลำดับ ยกตัวอย่างเช่น ไม่มีคำสั่ง break ; และทำการทดสอบโปรแกรมโดยป้อนตัวอักษร m ลงไป โปรแกรมจะทำการแสดงข้อความ ทั้ง “You are male” และ “You are female” และ “ Try again !!!!” ออกมาทั้งหมดดังรูป

สรุปท้ายบท

ตัวดำเนินการตัดสินใจหลายๆเงื่อนไข if if-else,... else และ Switch-case เป็นการตรวจสอบภายใต้หลายเงื่อนไขที่ซับซ้อนขึ้น เพื่อให้โปรแกรมทำงานได้ทุกๆ กรณีตามเงื่อนไขที่ได้กำหนดไว้ หากเงื่อนไขไม่ครอบคลุมจะทำให้การเขียนโปรแกรมไม่ข้อผิดพลาดหรือไม่ตรงขอบเขตที่กำหนด

ตัวดำเนินการตัดสินใจแบบ Switch-case ออกแบบเพื่อใช้สำหรับระบุทางเลือกแบบเจาะจง จะไม่เหมาะสำหรับช่วงค่าข้อมูล โดยในแต่ละ case จะสิ้นสุดเมื่อใช้คำสั่ง break หากไม่มีคำสั่งดังกล่าวโปรแกรมจะดำเนินการต่อใน case ถัดไป ในกรณีที่นอกเหนือจาก case ที่กำหนด จะไปทำงานที่ default ของโปรแกรม

ดังนั้นโดยการเปรียบเทียบการใช้งานระหว่าง if if-else และ Switch-case การทำงานของ Switch case จะทำงานได้รวดเร็วกว่าการตัดสินใจในแบบอื่นๆ

คำถามท้ายบท

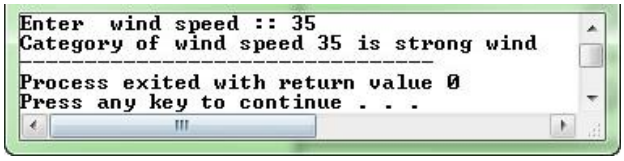
1. คำสั่ง if และ if else มีข้อแตกต่างการอย่างไร?
2. การใช้งานคำสั่ง else สามารถนำไปใช้กับเงื่อนไขแบบใด?
3. รูปแบบตัวดำเนินการตรวจสอบหลายๆเงื่อนไข ประเภทใดทำงานได้เร็วที่สุด?
4. คำสั่ง break ใน switch case ทำหน้าที่อะไร?
5. หากไม่มีคำสั่ง default ในโปรแกรม จะมีผลอย่างไร?
6. คำสั่ง default ใน switch case ทำหน้าที่อะไร?
7. การตรวจสอบเงื่อนไข เครื่องหมาย ^ ทำหน้าที่อะไร?
8. การตรวจสอบเงื่อนไข เครื่องหมาย && ทำหน้าที่อะไร?
9. การตรวจสอบเงื่อนไข เครื่องหมาย || ทำหน้าที่อะไร?
10. การตรวจสอบเงื่อนไข เครื่องหมาย ! ทำหน้าที่อะไร?

แบบฝึกหัดท้ายบท

1. จงเขียนผังการทำงานและโปรแกรม เพื่อแสดงประเภทของความเร็วลมดังตามตารางที่กำหนดให้ โดยกำหนดให้ผู้ใช้ป้อนค่าความเร็วลมทางคีย์บอร์ด
   ถ้าค่าที่รับมาน้อยกว่า 0 ให้แสดงข้อความ “ **Error !!!.**”

ความเร็วลม (ไมล์ / ชั่วโมง)	**ลำดับขั้น (Category)**
ต่ำกว่า 25	Not a strong wind
25 - 38	Strong wind
39 - 54	Gale
55 – 72	Whole gale
มากกว่า 72	Hurricane

ตัวอย่างอินพุตของและเอาต์พุตของโปรแกรม



2. จงเขียนผังการทำงานและโปรแกรม แสดงระดับผลการเรียนจากเกรดของนักศึกษาที่สอบได้ โดยโปรแกรมจะทำการรับอินพุตเป็นเกรดของนักศึกษา จากนั้นโปรแกรมจะทำการแสดงข้อความแจ้งระดับของเกรดขึ้นมาโดยมีข้อความดังนี้
   - เกรด A หรือ a ให้แสดง ข้อความ **Excellent**
   - เกรด B หรือ b ให้แสดง ข้อความ **Good**
   - เกรด C หรือ c ให้แสดง ข้อความ **Fair**
   - เกรด D หรือ d ให้แสดง ข้อความ **Poor**
   - เกรด F หรือ f ให้แสดง ข้อความ **Try Again**
   - ถ้าเป็นตัวอักษรอื่นนอกเหนือจากนี้ให้แสดงข้อความ **Invalid letter grade**

3. จากตารางการเดินรถไฟ Airport Rail Link Route จากสถานีต่างๆ ที่กำหนดให้ จงเขียนผังการทำงานและโปรแกรม การคำนวณอัตราค่าบริการ ตามตารางที่กำหนดให้

   <u>ข้อกำหนด</u>

   1. กำหนดสถานีปัจจุบัน กำหนดค่าคีย์บอร์ดจากผู้ใช้
   2. กำหนดสถานีปลายทาง กำหนดค่าคีย์บอร์ดจากผู้ใช้
   3. คำนวณอัตราค่าโดยสาร (บาท) ตามตารางอัตราค่าโดยสาร (บาท)

	พญาไท Phaya Thai	ราชปรารภ Ratchaprarop	มักกะสัน Makkasan	รามคำแหง Ramkamheang	หัวหมาก Hua Mark	บ้านทับช้าง Ban Thap Chang	ลาดกระบัง Lat Krabang	สุวรรณภูมิ Suvarnabhumi
พญาไท Phaya Thai		15	20	40	50	60	70	90
ราชปรารภ Ratchaprarop	15		15	30	40	50	60	80
มักกะสัน Makkasan	20	15		20	30	40	50	70
รามคำแหง Ramkamheang	40	30	20		20	30	40	50
หัวหมาก Hua Mark	50	40	30	20		20	30	40
บ้านทับช้าง Ban Thap Chang	60	50	40	30	20		20	30
ลาดกระบัง Lat Krabang	70	60	50	40	30	20		20
สุวรรณภูมิ Suvarnabhumi	90	80	70	50	40	30	20	

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 8 การบรรยายสรุป

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาทบทวนเนื้อหา โปรแกรมคอมพิวเตอร์สำหรับวิศวกร
- เพื่อให้นักศึกษาฝึกฝนการทำแบบฝึกหัด

เนื้อหา

- การบรรยายสรุป และทบทวนเนื้อหา

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายสรุปเนื้อหา
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท

การบรรยายสัปดาห์ที่ 8

การบรรยายสรุป

Conclusions

ภาษาซีเป็นการเปลี่ยนภาษาคอมพิวเตอร์เป็นภาษาเครื่อง หรือแมตซีนโค้ด (Machine code) ด้วยตัวคอมไพเลอร์ภาษาซี เพื่อให้คอมพิวเตอร์เข้าใจในโปรแกรมภาษาซีที่สร้างขึ้น ทำให้การประมวลผลคอมพิวเตอร์เข้าใจในรูปแบบการทำงาน โดยโครงสร้างของการเขียนโปรแกรมภาษาซี จะต้องประกอบด้วยส่วนสำคัญคือ Header file และ Main function หากไม่มีสองส่วนนี้โปรแกรมจะไม่สามารถทำงานได้ ในการพัฒนาโปรแกรมให้มีประสิทธิภาพขึ้นอยู่กับนำชุดฟังก์ชั่นหรือคำสั่งต่างๆ ไปประยุกต์ใช้งานในส่วนงานการคำนวณ วิเคราะห์ ประมวลผลสมการต่างๆ ทางด้านวิศวกรรมศาสตร์ ด้วยกระบวนการและภายใต้โครงสร้างของภาษาซี

การเขียนเพอร์ซูโด-โค้ด (Pseudo-code) ทำให้เข้าใจลำดับการทำงานให้ง่ายขึ้น โดยเขียนเป็นลำดับขั้นตอน และทำให้การสร้างผังงานได้สะดวก ตามเงื่อนไขรูปแบบการใช้สัญลักษณ์ผังงานต่างๆ หากเข้าใจหลักการเขียนเพอร์ซูโด-โค้ด ก็จะสามารถเขียนผังงานของโปรแกรมหรือระบบต่างๆ ได้ ทั้งนี้การเขียนผังงานสามารถเข้าใจถึงการออกแบบโปรแกรมหรือระบบที่ต้องการได้โดยใช้สัญลักษณ์ เพื่อลดความผิดพลาดในการสื่อสารระหว่างผู้ใช้งาน ซึ่งการเขียนผังงานนั้นมีความสำคัญอย่างยิ่ง ในการออกแบบโปรแกรมขนาดใหญ่และการทำงานร่วมกับคนจำนวนมาก ทั้งยังให้เข้าใจถึงภาพรวมของโปรแกรมและระบบ

ขั้นตอนการสร้างชื่อตัวแปรและกำหนดชนิดของตัวแปรให้ถูกต้องตามข้อกำหนดของภาษาซี โดยกำหนดชนิดตัวแปรสำหรับเก็บค่าข้อมูลชนิดจำนวนจริง จำนวนเต็ม และตัวอักษร ให้ตรงตามคุณสมบัติของตัวแปรชนิดนั้นๆ หากกำหนดผิดวัตถุประสงค์ จะส่งผลให้การจะทำให้โปรแกรมการทำงานมีความผิดพลาด ผลลัพธ์การคำนวณจะมีค่าผิดพลาดได้ ซึ่งตัวแปรที่สามารถระบุทศนิยมคือตัวแปรจำนวนจริงเท่านั้น รวมถึงการใช้เครื่องหมายทางคณิตศาสตร์ต้องคำนวณถึงรูปแบบการเขียนสัญลักษณ์เครื่องหมายทางภาษาซี เพื่อที่จะทำให้โปรแกรมสามารถทำงานได้อย่างถูกต้อง

การเรียกใช้ฟังก์ชั่นต่างๆหรือคำสั่ง ในภาษาซี ต้องประกาศส่วนหัวของโปรแกรม (header file) เพื่อสามารถเรียกใช้ฟังก์ชั่นต่างๆ ภายใน header file นั้นๆ หากไม่ได้ระบุฟังก์ชั่นที่เรียกใช้งาน จะไม่สามารถทำงานได้ โปรแกรมไม่สามารถทำงานได้ รวมถึงการกำหนดเครื่องหมายการคำนวณทางคณิตศาสตร์ให้เป็นไปตามรูปแบบภาษาซี และการเปลี่ยนรูปแบบสมการทางคณิตศาสตร์ ให้เป็นสมการสำหรับโปรแกรมภาษาซี

ตัวดำเนินการตัดสินใจหรือเงื่อนไข if, if-else เป็นการตรวจสอบภายใต้เงื่อนไขและขอบเขตที่กำหนดเพื่อให้โปรแกรมดำเนินการตามทางเลือกของผลลัพธ์เงื่อนไขเป็นจริง หรือ ผลลัพธ์เงื่อนไขเป็นเท็จ ซึ่งแสดงถึงทางเลือกของโปรแกรมเมื่อเงื่อนไขเป็นจริง จะทำให้ส่วนของกลุ่มคำสั่งต่างๆที่อยู่ภายใต้เครื่องหมายปีกกาเปิด

ปิด หรือถ้าเงื่อนไขเป็นเท็จ จะทำให้ส่วนของกลุ่มคำสั่งต่างๆที่อยู่ภายใต้เครื่องหมายปีกกาเปิดปิด โดยโปรแกรมจะเลือกดำเนินการตามเงื่อนไขที่กำหนด

ตัวดำเนินการตัดสินใจหลายๆเงื่อนไข if if-else,... else และ Switch-case เป็นการตรวจสอบภายใต้หลายเงื่อนไขที่ซับซ้อนขึ้น เพื่อให้โปรแกรมทำงานได้ทุกๆ กรณีตามเงื่อนไขที่ได้กำหนดไว้ หากเงื่อนไขไม่ครอบคลุมจะทำให้การเขียนโปรแกรมไม่ข้อผิดพลาดหรือไม่ตรงขอบเขตที่กำหนด

ตัวดำเนินการตัดสินใจแบบ Switch-case ออกแบบเพื่อใช้สำหรับระบุทางเลือกแบบเจาะจง จะไม่เหมาะสำหรับช่วงค่าข้อมูล โดยในแต่ละ case จะสิ้นสุดเมื่อใช้คำสั่ง break หากไม่มีคำสั่งดังกล่าวโปรแกรมจะดำเนินการต่อใน case ถัดไป ในกรณีที่นอกเหนือจาก case ที่กำหนด จะไปทำงานที่ default ของโปรแกรม ดังนั้นโดยการเปรียบเทียบการใช้งานระหว่าง if if-else และ Switch-case การทำงานของ Switch case จะทำงานได้รวดเร็วกว่าการตัดสินใจในแบบอื่นๆ

แบบฝึกหัดทบทวน

1. จงหาค่าของตัวแปร F เมื่อกำหนดให้

```
int A, B, C, D, F ;
A = 1 ; B = 2 ; C = 3 ; D = 4 ;
F = B*C%D;
```

   a. F = 2
   b. F = -2
   c. F = -4
   d. F = 4

2. จงหาค่าของตัวแปร F เมื่อกำหนดให้

```
int A, B, C, D, F ;
A = 5 ; B = 6 ; C = 7 ; D = 8 ;
F = (A*B+C*D)
```

   a. F = 2
   b. F = 86
   c. F = 99
   d. F = 85

3. จงเขียนสมการทางคอมพิวเตอร์ จากสมการทางคณิตศาสตร์ที่กำหนดมาให้

$$Y = \frac{AB}{CD} + \frac{B}{A*C} + \frac{ABC}{D}$$

   a. Y = A*B/C*D + B/A+C + A*B*C/D ;
   b. Y = A*B/C*D + B/(A+C) + A*B*C/D ;
   c. Y = A*B/(C*D) + B/(A*C) + A*B*C/D ;
   d. Y = A*B/C/D + B/A+C + A*B*C/D ;

4. ข้อใดต่อไปนี้ ให้ผลลัพธ์เท่ากับสมการ Y = (A+B/C-D)*E ;
    a. Y = ((A+B)/C-D)*E ;
    b. Y = (A+B)/C-D*E ;
    c. Y = A+B/C*E-D*E ;
    d. Y = (A+(B/C)-D)*E ;
5. กำหนดให้ A=1, B=10, C=100 นิพจน์ใดต่อไปนี้ให้ค่าตรรกะเป็นจริง
    a. !(C/B == B)
    b. B*A ==C/B
    c. !(A*C >= C/A)
    d. (B%C+A) < (C%B)+A
6. กำหนดให้ int A ; A = 7/ 2 -1+ 9*2 ; จงหาค่าในตัวแปร A ว่ามีค่าเท่ากับเท่าไหร่
    a. A = 20
    b. A = 20.5
    c. A = 13.5
    d. A= 13
7. ถ้ากำหนดให้ Y = 22 % X – 2 ; และผลลัพธ์ที่ได้จากสมการ Y จะมีค่าเท่ากับ 2 จงหาค่าของตัวแปร X
    a. X = 3
    b. X = 4
    c. X = 5
    d. X = 6
8. กำหนดให้ int A = 10, B = 5, C = 10; ข้อใดต่อไปนี้ให้ตรรกะที่เป็นจริง
    a. (C%B) > (A/C)
    b. (C*B) <= A*B%C
    c. (B-A) == (A-B-C)
    d. A/B > B%C
9. กำหนดให้ A มีค่าเท่ากับ 53.414 และให้ทำคำสั่งต่อไปนี้ printf(“ %.2f ”, A) ; ข้อใดคือผลลัพธ์ที่ได้
    a. 053.4
    b. 3.4
    c. 53.4
    d. 53.41

10. ข้อใดต่อไปนี้ใช้ฟังก์ชั่น scanf( ) ได้ถูกต้อง

a. scanf(“%d, x”);

b. scanf(“%d”, &x);

c. scanf(“%d”, x);

d. scanf(“%d, &x”);

11. ข้อใดต่อไปนี้ใช้ฟังก์ชั่น printf( ) ได้ถูกต้อง

a. printf(“%d, x”);

b. printf(“%d”,&x);

c. printf(“%d”,x);

d. printf(“%d,&x”);

12. จากโปรแกรมที่กำหนดให้ ถ้าต้องการให้ Y = 0 ต้องป้อนค่า X เท่ากับเท่าใด

```
int  X;
if ((X+2)>X || ((2*X)>X))
        Y = 1;
else
        Y = 0 ;
```

a. X = 2

b. X = 3

c. X = 4

d. X = 0

13. กำหนดตัวแปร A เป็น integer ถ้าต้องการเช็คว่าตัวแปร A เก็บเลขที่ลงท้ายด้วย 9 (เช่น 9, 19, 29, 39, ...) ต้องใช้คำสั่งอย่างไร

a. if ((A%9) == 0)

b. if ((A/9) == 0)

c. if ((A%10) == 9)

d. if ((A/10) == 9)

14. จากโปรแกรมที่กำหนดให้ เมื่อป้อนค่า A = 3, 4 ,5 จงหาผลลัพธ์ที่ได้หลังจากโปรแกรมทำงานเสร็จเรียบร้อย ในแต่ละครั้งที่ป้อนค่า A

```
int  A;
scanf("%d",&A);
if (A<=3)
{
      if (A%2==1)  printf("Case1");
      else   printf("Case2");
}
     printf("Case3");
```

a. Case1 Case2 Case3 Case3

b. Case2 Case2 Case3 Case3

c. Case2 Case3 Case3 Case3

d. Case1 Case3 Case3 Case3

15. จากโปรแกรมที่กำหนดให้ จงหาผลลัพธ์ที่ได้หลังจากโปรแกรมทำงานเสร็จเรียบร้อย

```
int  x = 6, y = 9 ;
if ((x>5) && (y<8))
   { y = 0; }
else if (x%3<0) || (y/3>3)
   {y = 1; }
else if ((x+y)<10) || (y-x)==3)
   {y = 2; }
else
   {y = 3 ;}
```

a. y = 3

b. y =2

c. y = 1

d. y = 0

16. จากโปรแกรมที่กำหนดให้ เมื่อมีการกำหนดค่าให้ตัวแปร x ข้อความใดเป็นจริง

```
int  x , y = 8 ;
scanf("%d",&x);
if (x>7)
     {y = y*2; }
if (x<3)
     { y = y-1; }
 else if (x>0)
     {y = y+1; }
 else if (x<0)
{y = 2; }
else
{y = 3;}
```

a. ถ้า x = 7 จะได้ y = 9 และถ้า x = 0 จะได้ y = 7

b. ถ้า x = 3 จะได้ y = 7 และถ้า x = 7 จะได้ y = 9

c. ถ้า x = 7 จะได้ y = 9 และถ้า x = 0 จะได้ y = 9

d. ถ้า x = 1 จะได้ y = 9 และถ้า x = 7 จะได้ y = 16

17. ถ้า int num[5] = { 2, 4, 3, 5, 6}; ข้อใดเป็นคำตอบของ x เมื่อ int x = num[0] * (num[4] % num[3]) ;

a. num[1]

b. num[2]

c. num[3]

d. num[0]

18. จากโปรแกรมที่กำหนดให้ เมื่อโปรแกรมทำงานเสร็จเรียบร้อย ค่า **x** จะมีค่าเท่าไหร่

```
int  x = 1  , y = 0  ;
do{
   x += y  ;
   y += 3 ;
 }while(y<3)  ;
```

a. x = 0

b. x = 6

c. x = 3

d. x = 9

19. จากโปรแกรมที่กำหนดให้ จงเลือกคำตอบที่ถูกที่สุด

```
int  i =0 , j =0  ;
for(i=2 ; i<=4 ; i++){
    if ((i-1)/2 == 0)
{
        printf("%d",i ) ;
        j = j+1 ;
    }
}
```

a. โปรแกรมนี้แสดงค่า i ทั้งหมด 3 ครั้ง

b. ค่า i ตัวสุดท้าย คือ 4

c. ค่า j สุดท้าย คือ 3

d. ไม่มีคำตอบใดถูกต้อง

20. printf (“%d”,15%3+3*5-7); จะได้คำตอบออกมาเป็น

    a. 6

    b. 7

    c. 8

    d. 9

21. ถ้าต้องการสร้างคำนวณพื้นที่ float area; สี่เหลี่ยมคางหมู โดยให้ float a, b; เป็นด้านของสีเหลี่ยมคู่ขนาน และ float h; เป็นความสูง จะต้องเขียน statement ว่า

    a. area = a+b*h*0.5;

    b. area = 1.0/2.0*a+b*h;

    c. area = 0.5*(a+b)/h;

    d. area = 0.5*(b+a)*h;

22. float pi=3.141592654; printf (“%.4f”,pi); จะได้

    a. 3.1415

    b. 3.1416

    c. 3.14159

    d. 3.141592

23. int i, j; for (i=0; i<5; ++i) j*=i; จะได้คำตอบ

    a. 0

    b. -1

    c. 1

    d. 15

24. ข้อใดคือตำตอบบนจอหลังรันโปรแกรมนี้

```
#include<stdio.h>
main( )
{
        int m, n, sum=0;
        for (m=0; m<5; m++)
                for(n=0; n<m; n++);
                        sum+=n;
        printf(“%d”,++sum;
}
```

    a. 10 b.11 c.20 d.21

25. Statement ใดเป็นการคำนวณสูตรนี้ $x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

a. x = -b + (sqrt(pow(b,2) - 4*a*c)/2*a;

b. x = (-b - (sqrt(pow(b,2) - 4*a*c))/2*a;

c. x = (-b + (sqrt(pow(b,2) - 4*a*c))/2/a;

d. x = -b + (sqrt(pow(b,2) - 4*a*c)) /2/a;

26. ข้อใดให้ค่าไม่ตรงกับข้ออื่นๆ เมื่อกำหนด int a=10; float b=21.0;

a. printf (“%d”, a++);

b. printf (“%d”, a+1);

c. printf (“%f”, b-10.0);

d. printf (“%d”, ++a);

27. Statement ใดเป็นการคำนวณสูตรนี้ $z = \sqrt{x^2 + y^2}$

a. z = sqrt(2*x+2*y);

b. z = sqrt(x*x+y*y);

c. z = sqrt(pow(x,x)+pow(y,y));

d. z = sqrt(x^2+y^2);

28. จงหาค่าของ z = x + y /2 เมื่อ x =6 และ y = 8

a. z = 7

b. z = 10

c. z = 14

d. z = 8

29. จงหาค่าของ z = pow(x,y) + pow(y,x) เมื่อ x =2 และ y = 3

a. z = 12

b. z = 17

c. z = 27

d. z = 56

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 9 การทำงานซ้ำ (Loop)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจกระบวนการทำงานซ้ำ
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและโปรแกรมของการทำงานซ้ำ

เนื้อหา

- การตรวจสอบเงื่อนไข ภายในลูปในการทำงานซ้ำ เป็นไปตามการเมื่อเงื่อนไขที่กำหนด ด้วยการทำงานซ้ำแบบ for loop, while loop และ do-while loop
- การเขียนโปรแกรมในการทำงานซ้ำๆ สำหรับ for loop, while loop และ do-while loop

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการทำงานซ้ำด้วยการตรวจสอบเงื่อนไข
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 9

การทำงานวนซ้ำ

Looping

บทเรียนนี้ การเขียนโปรแกรมการทำงานซ้ำๆ หรือทำงานในลักษณเดิมซ้ำ หรือการทำงานในส่วนนั้นซ้ำๆ วนกลับไปทำงานตามรูปแบบของลูป โดยผู้เขียนต้องการให้มีการคำนวณคำสั่งหรือกลุ่มของคำสั่งซ้ำๆ ตามจำนวนที่ต้องการ หรือจนกระทั่งพบภาวะที่ต้องการ หรือต้องการวนคำนวณกลุ่มคำสั่งชุดเดิมตลอดเวลา ทำให้ต้องมีชุดคำสั่งเฉพาะทางที่ทำหน้าที่ข้างต้น ในโครงสร้างของภาษาซี มีชุดคำสั่งเฉพาะทางดังกล่าวสำหรับการทำงานซ้ำๆ หรือการทำงานในลูป loop ดังนั้นคำสั่งในการใช้งานของรูปแบบการทำงานซ้ำหรือการวนลูปสามารถจำแนกได้ 3 ประเภท ดังนี้

1. For loop operation
2. While loop operation
3. Do-While loop operation

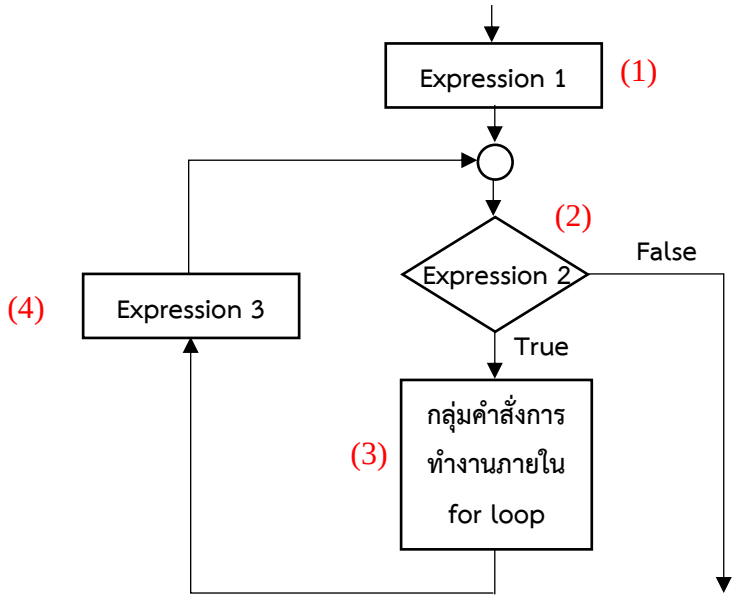
การทำงานซ้ำหรือการทำงานแบบลูป มี 2 ลักษณะพื้นฐานที่ต้องทำความเข้าใจ คือ

1. แบบกำหนดจำนวนลูปหรือระบุจำนวนครั้งที่แน่นอน เช่นกรณีที่ต้องการแสดงข้อความซ้ำๆ 100 ครั้งบนจอภาพ เป็นต้น การทำซ้ำลักษณะนี้จะใช้คำสั่ง for-loop
2. แบบทำงานซ้ำตามเงื่อนไข (เป็นตัวตัดสินว่าจะมีการทำซ้ำต่อไปหรือไม่) ใช้ในกรณีที่ต้องการให้โปรแกรมทำซ้ำๆ ตลอดเวลา ตราบใดที่เงื่อนไขที่กำหนดไว้ยังคงเป็นจริง การทำงานในลักษณะนี้จะเห็นว่าเป็นการทำซ้ำที่ไม่มีจำนวนรอบที่แน่นอน การทำงานซ้ำลักษณะนี้ สามารถใช้คำสั่ง do-while และ while

9.1 การทำงานซ้ำด้วย For Loop

การใช้คำสั่ง for สอดคล้องกับอนุกรมของเลขทางคณิตศาสตร์ $\sum_{i=1}^{n} x_i = i$, (กำหนดให้ *i* มีค่าเริ่มต้นเท่ากับ 1, 2, 3,..., n เพื่อกำหนดให้มีการทำงานซ้ำทั้งสิ้น n รอบ) ที่จะใช้กำหนดจำนวนรอบในการทำงาน ซึ่งในการวนด้วยคำสั่ง for และจะต้องสร้างตัวแปรขึ้นมาทำหน้าที่เป็น “ตัวนับ” (Counter) หมายถึงตัวแปร x ข้างต้น การเขียนรูปแบบการใช้คำสั่ง for loop แสดงผังงานประกอบ แสดงดังรูปที่ 9.1

รูปแบบผังงานของ For-loop



รูปที่ 9.1 ผังงานของรูปแบบ For-loop

เมื่อ expression 1 คือ กำหนดค่าเริ่มต้นของการทำงานซ้ำ โดยการระบุค่าคงที่หรือตัวแปรกำหนด

expression 2 คือ ตรวจสอบสภาวะการทำงาน ตามเงื่อนไขที่กำหนด โดยผลลัพธ์ของเงื่อนไขสามารถ ระบุได้ดังนี้ ผลลัพธ์ของเงื่อนไขเป็นจริง (True) จะทำงานซ้ำภายในเครื่องหมายปีกกาเปิดปิด { } และเมื่อผลลัพธ์ของเงื่อนไขเป็นเท็จ (False) จะออกจากการทำงานซ้ำ

expression 3 คือ การเปลี่ยนแปลงค่าการนับจำนวนของตัวแปรที่ทำงานในลูป

โดยกระบวนการขั้นตอนจะเริ่มต้นลำดับที่ (1) กำหนดค่าตัวแปรเริ่มต้นให้แก่ตัวนับ และจะทำการตรวจสอบเงื่อนไขในลำดับที่ (2) ทำการตรวจสอบเงื่อนไขที่กำหนด หากเงื่อนไขเป็นเท็จจะออกจากลูป หากเงื่อนไขเป็นจริงให้ทำลำดับที่ (3) ดังนั้นเมื่อเงื่อนไขเป็นจริง จะทำในกลุ่มคำสั่งภายใต้เครื่องหมายปีกกาเปิดและสิ้นสุดปีกกาปิด เมื่อทำกลุ่มคำสั่งเสร็จสิ้นจะทำในลำดับที่ (4) คือการเปลี่ยนแปลงค่าของตัวแปร เพิ่มขึ้นหรือลดลงค่าตัวแปรตัวนับ และกลับมาทำในลำดับที่ (2) ต่อไป และทำงานวนลูปจนกว่าเงื่อนไขจะเป็นเท็จจึงออกจากลูป และดำเนินงานในคำสั่งถัดไป

รูปแบบโปรแกรม For Loop

```
for ( Expression 1 ; Expression 2 ; Expression 3 )
{          (1)          (2)          (4)
    การทำงานกลุ่มคำสั่งต่างๆ ภายใน for loop ต้องอยู่ภายในเครื่องหมายปีกกา      (3)
}
```

ในการเขียนคำสั่งของ for loop ทั้ง 3 ส่วน จะต้องเป็นตัวแปรชื่อและชนิดตัวแปร เดียวกันทั้ง 3 ส่วน เพื่อให้โปรแกรมทำงานได้อย่างถูกต้อง หากกำหนดชื่อตัวแปรต่างกัน จะทำให้โปรแกรมไม่สามารถทำงานได้

เช่น **for (int j = 0 ; j < 10 ; j++)** หมายถึง ลูปทำงานซ้ำ จำนวน 10 รอบ

โดย Expression 1 กำหนดค่าเริ่มต้นตัวแปร j ชนิดจำนวนเต็ม ค่าเริ่มต้น j = 0

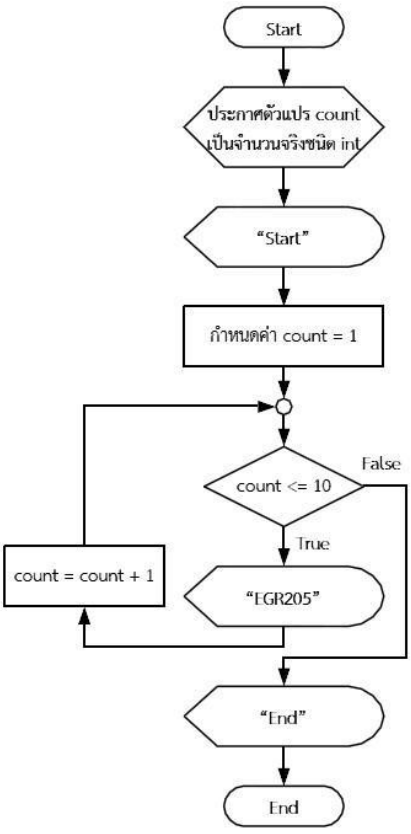
เมื่อ Expression 2 คือเงื่อนไขในการตรวจสอบ โดยกำหนดเงื่อนไข j < 10 โดยผลลัพธ์ในการตรวจสอบเงื่อนไขแบ่งเป็น 2 ผลลัพธ์ คือ เงื่อนไขเป็นจริง และเงื่อนไขเป็นเท็จ โดยหากเงื่อนไขเป็นจริงให้ทำคำสั่งภายในเครื่องหมายปีกกาและทำในส่วนของ Expression 3 ต่อไป โดยหากเงื่อนไขเป็นเท็จให้ออกจากลูปไปยังคำสั่งที่นอกเหนือจากลูปต่อไป และ Expression 3 คือการเปลี่ยนแปลงค่าของจำนวนตัวแปร โดยการเพิ่มค่า j++ หรือ j=j+1 และวนกลับไปทำในส่วนของ Expression 2 ต่อไป

for (int j = 0 ; k < 10 ; h++) หมายถึง ลูปไม่สามารถทำงานได้ เนื่องจากการระบุตัวแปรที่ไม่ถูกต้อง โดย Expression ทั้ง 3 ส่วน ใช้ตัวแปรที่แตกต่างกัน จึงทำให้การทำงานของลูปไม่สามารถทำงานได้อย่างถูกต้อง ควรแก้ไขตัวแปรให้เป็นตัวแปรเดียวกันใน Expression ทั้ง 3 ส่วน

for (; ;) หมายถึง การทำงานลูปแบบไม่จำกัด (Infinity Loop) เป็นการทำงานภายในลูปอย่างต่อเนื่องโดยไม่มีการตรวจสอบเงื่อนไขตามรูปแบบของ loop หากต้องการออกจากรูปประเภทนี้ จะใช้คำสั่ง break การทำงานภายในลูปจะสิ้นสุดทันที

ตัวอย่างที่ 9.1 จงเขียนผังการทำงานและโปรแกรมแสดงผลลัพธ์ข้อความ EGR205 จ้านวน 10 ครั้ง ทางเอาต์พุตหน้าจอคอมพิวเตอร์ โดยใช้คำสั่งการทำซ้ำแบบ for loop

ตัวอย่างผังงานที่ 9.1

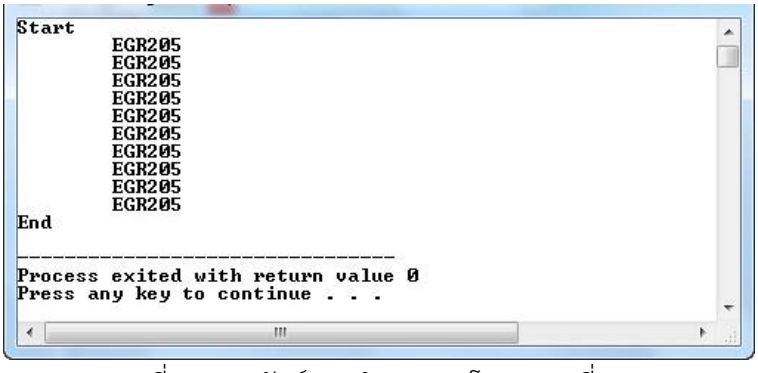


รูปที่ 9.2 ผังการทำงานของ for loop สำหรับตัวอย่างที่ 9.1

ตัวอย่างโปรแกรมที่ 9.1

```
1.  #include<stdio.h>
2.  main (void)
3.  {       int  count ;
4.          printf(“Start \n”) ;
5.          for( count = 1 ; count <=10 ; count++ )
6.          {
7.                  printf(“\tEGR205\n”) ;
8.          }
9.          printf(“End\n”) ;
10. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 9.1



รูปที่ 9.3 ผลลัพธ์การทำงานของโปรแกรมที่ 9.1

จากผลลัพธ์การทำงานของโปรแกรมที่ 9.1 จะแสดงข้อความ “EGR205” ทั้งหมด 10 ครั้ง โดยใช้ตัวแปร count เป็นตัวนับ โดยกำหนดค่าเริ่มต้นเท่ากับ 1 จากนั้นทำการตรวจสอบเงื่อนไข count น้อยกว่าเท่ากับ 10 หรือไม่ โดยใช้เครื่องหมายเปรียบเทียบ (< , > , <= , >= , ==) ถ้าเงื่อนไข count เป็นจริงจะทำในเครื่องหมายปีกกา { } เมื่อสิ้นสุดจะดำเนินการเพิ่มค่าตัวนับเท่ากับ count=count + 1 หรือ count++ และจะวนกลับไปตรวจสอบเงื่อนไขอีกครั้ง และทำซ้ำๆตราบใดที่ยังเป็นไปตามเงื่อนไขเป็นจริง หากการตรวจสอบไม่เป็นไปตามเงื่อนไขหรือเงื่อนไขเป็นเท็จ จะออกจากการทำงานซ้ำหรือออกจาก for loop นั้นเอง

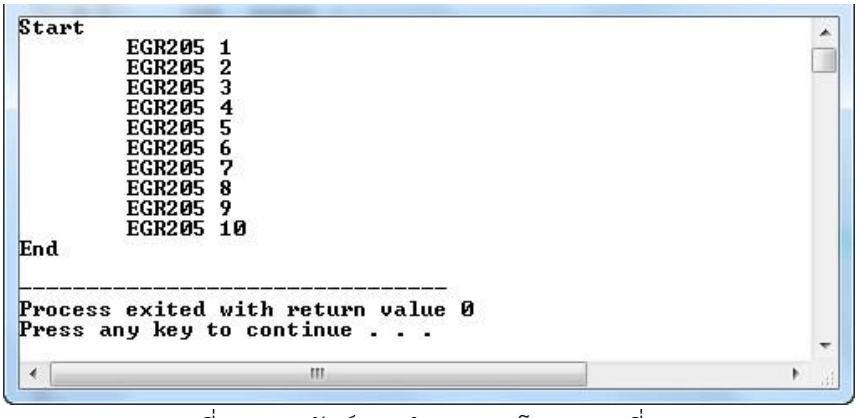
จากตัวอย่างผังงานที่ 9.2 จะเห็นได้ว่า ตัวแปร count เป็นตัวแปรที่ใช้ในการนับรอบของการทำซ้ำ หรือ วนซ้ำ โดยตัวแปร count จะเพิ่มขึ้นทีละ 1 ต่อการทำซ้ำทุกๆ 1 รอบ จนกว่า ตัวแปร count จะมีค่ามากกว่า 10 ก็จะหยุดการทำซ้ำ ดังนั้น เนื่องจากตัวแปร count มีการเปลี่ยนแปลงค่าเพิ่มขึ้นไปในแต่ละรอบที่มีการทำซ้ำ จึงทำให้สามารถนำค่าตัวนั้นมาคำนวณหรือแสดงค่าออกทางหน้าจอได้ ดังตัวอย่างโปรแกรมที่ 9.2

ตัวอย่างโปรแกรมที่ 9.2

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

```
2.  main ()
3.  {   int  count ;
4.      printf(“Start \n”) ;
5.      for( count = 1 ; count <=10 ; count++ )
6.      {
7.              printf(“\tEGR205 %d \n” , count) ;
8.      }
9.      printf(“End\n”) ;
10. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 9.2



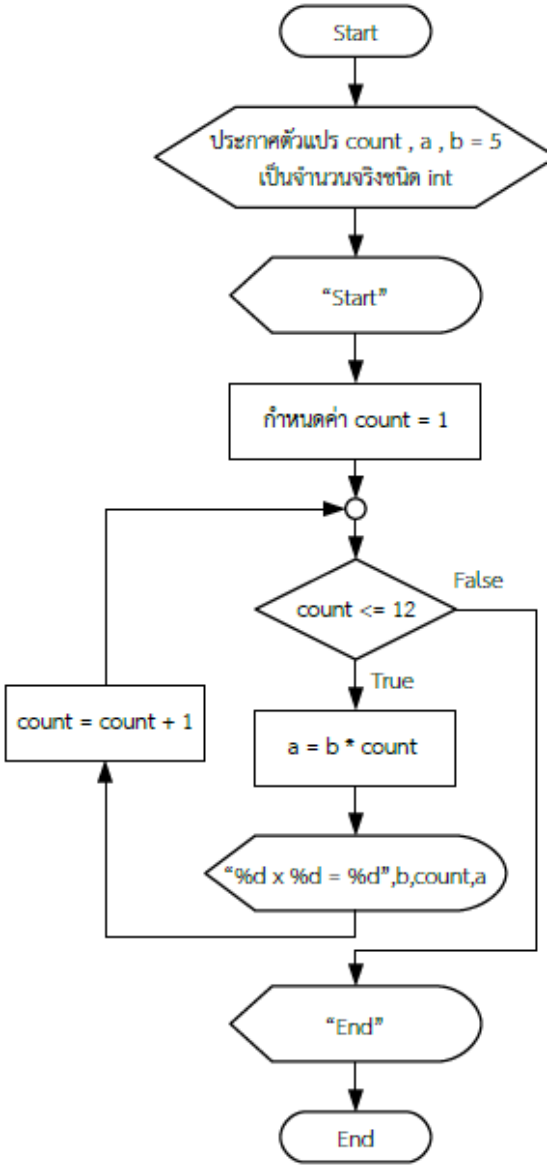
รูปที่ 9.4 ผลลัพธ์การทำงานของโปรแกรมที่ 9.2

จากผลลัพธ์การทำงานโปรแกรมที่ 9.2 แสดงให้เห็นผลลัพธ์การทำงานของโปรแกรม โดยแสดงค่าผลลัพธ์ของตัวแปร count เป็นจำนวนเต็ม ที่เพิ่มค่า count มีการเปลี่ยนแปลงจาก 1 ถึง 10 ซึ่งค่าของตัวแปร count เมื่อออกจากลูป มีค่าเท่ากับ 11 นั้นเอง ซึ่งสามารถนำไปใช้คำนวณในโปรแกรมได้อีกด้วย

ตัวอย่างโปรแกรมที่ 9.3 จงเขียนผังการทำงานและโปรแกรม สำหรับการคำนวณแม่สูตรคูณมาตรา 5 ด้วยการทำงานแบบ for loop

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
```

ตัวอย่างผังงานที่ 9.3



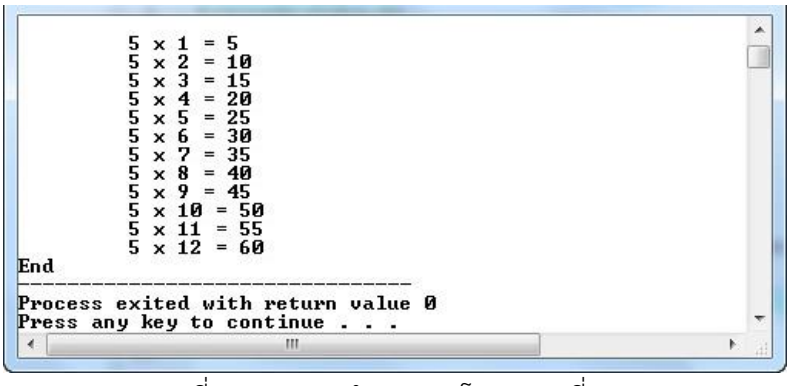
รูปที่ 9.5 ผังการทำงานของ for loop สำหรับตัวอย่างที่ 9.3

ตัวอย่างโปรแกรมที่ 9.3

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

```
2.  main ( )
3.   {   int count, b = 5 ;
4.       printf(“Start \n”) ;
5.       for ( count = 1 ; count <=12 ; count++ )
6.       {          a = b * count;
7.                  printf(“ \n\t %d x %d = %d ” , x , count  , a) ;
8.       }
9.       printf(“\nEnd”) ;
10.  }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 9.3



รูปที่ 9.6 ผลการทำงานของโปรแกรมที่ 9.3

จากโปรแกรมตัวอย่างที่ 9.3 แสดงให้เห็นถึงการทำซ้ำโดยมีการนำตัวแปร count มาคำนวณ โดยในระหว่างทำงานซ้ำในแต่ละรอบ ตัวแปร count จะคูณกับตัวแปร b และเก็บผลลัพธ์ไว้ในตัวแปร a จากนั้นจึงทำการแสดงค่าผลลัพธ์ที่คำนวณได้ออกทางหน้าจอคอมพิวเตอร์

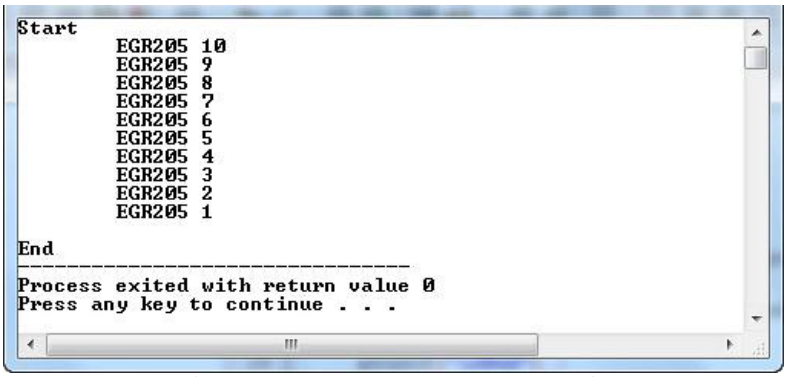
การทำงานซ้ำแบบสำหรับลดค่าของตัวนับ จากตัวอย่างที่ 9.3 นอกการใช้ for loop ในการทำงานซ้ำโดยการนับเพิ่มค่าของตัวแปร (ตัวนับเพิ่มขึ้น) และการนับลดค่าของตัวแปร (ตัวนับลดลง) ซึ่งผลลัพธ์การทำงานซ้ำหรือจำนวนรอบทำซ้ำยังเหมือนเดิม หากแตกต่างกันตรงที่ข้อมูลที่อยู่ในตัวแปร

ตัวอย่างโปรแกรมที่ 9.4 การแสดงวิธีการใช้คำสั่ง for loop แบบลดค่าของตัวนับ

```
1.  #include<stdio.h>
2.  main ( )
```

```
3.  {       int count ;
4.          printf(“Start \n”) ;
5.          for( count = 10 ; count > 0 ; count -- )
6.          {
7.                  printf(“\tEGR205 %d \n” , count) ;
8.          }
9.          printf(“\nEnd”) ;
10. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 9.4



รูปที่ 9.7 ผลลัพธ์ของโปรแกรมตัวอย่างที่ 9.4

จากผลลัพธ์โปรแกรมตัวอย่างที่ 9.4 ตัวแปร count จะลดค่าลงทีละหนึ่ง ด้วยคำสั่ง count - - หมายถึง count = count - 1 ดังนั้นเมื่อตัวแปร count มีค่าเท่ากับ 0 ทำให้เงื่อนไขเป็นเท็จ จึงออกจากการทำงานในลูป

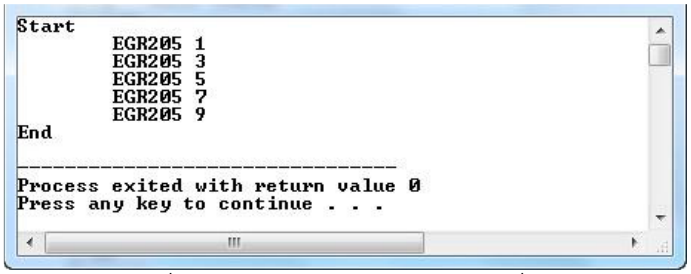
การทำงานซ้ำแบบสำหรับลดค่าใดๆของตัวนับ การทำงานซ้ำแบบ For loop สำหรับลดค่าใดๆ ของตัวนับ จากการทำคำสั่งวนซ้ำด้วยการเพิ่มค่า count++ หรือการลดค่า count - - โดยรูปแบบคำสั่งยังสามารถกำหนดให้ทำการเพิ่มค่าใดๆหรือการลดค่าใดๆ ของตัวนับ เช่น การบวกเพิ่มครั้งละ 2 สามารถเขียนได้ count+=2 หรือ count = count + 2 เป็นต้น ซึ่งสามารถกำหนดในรูปแบบอื่นๆ เช่น คูณครั้งละ 2 หมายถึง count *=2 เป็นต้น ดังนั้นสามารถทำการเพิ่มแบบจำนวนคงที่ได้ count+=x เมื่อ x คือค่าคงที่ รวมถึงการเพิ่มแบบทวีคูณ count *=x ซึ่งสามารถเขียนได้เป็น count = count * x

ตัวอย่างโปรแกรมที่ 9.5 จงเขียนโปรแกรมแสดงผลลัพธ์ EGR205 และให้แสดงเฉพาะเลขจำนวนเลขคี่ (Odd number) เท่านั้น จากลำดับตัวเลข 1 ถึง 10 ทางหน้าจอคอมพิวเตอร์

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

```
2.  main ( )
3.   {          int count ;
4.              printf(“Start \n”) ;
5.              for (count = 1 ; count <=10 ; count+=2)
6.              {
7.                      printf(“\tEGR205 %d \n” , count) ;
8.              }
9.              printf(“End\n”) ;
10.  }
```

ผลลัพธ์ตัวอย่างโปรแกรมตัวอย่างที่ 9.5



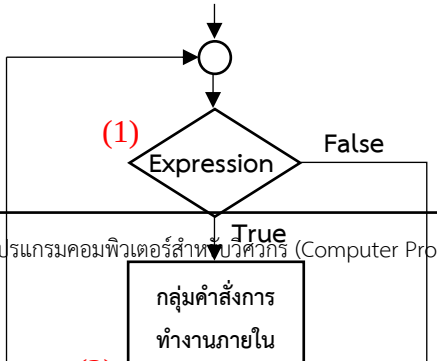
รูปที่ 9.8 ผลลัพธ์โปรแกรมตัวอย่างที่ 9.5

จากผลลัพธ์ตัวอย่างโปรแกรมที่ 9.5 ด้วยการเขียนคำสั่งให้ค่าของตัวแปร count เพิ่มค่าขึ้นครั้งละ 2 ด้วยคำสั่ง count +=2 (หมายถึง count = count + 2) โดยค่าตัวแปร count จะเปลี่ยนแปลงไปจากค่าเริ่มต้นจนถึงค่าสุดท้ายตามเงื่อนไขที่กำหนดแสดงค่าจำนวนรอบ ตัวแปร count = 1, 3, 5, 7, และ 9 ตามลำดับ จนกระทั่ง count มีค่ามากกว่า 10 ทำให้มีการทำซ้ำในโปรแกรมเพียง 5 ครั้ง ค่าสุดท้ายของตัวแปร count =11 ซึ่งการทำงานในรูปแบบนี้จะนำไปใช้กับการเขียนโปรแกรมที่ซับซ้อนยิ่งขึ้น

9.2 การทำงานซ้ำด้วย While Loop

การใช้คำสั่งการทำงานซ้ำแบบ while loop มีความแตกต่างจากคำสั่ง for โดยรูปแบบการใช้งานคำสั่ง while loop นั้นจะใช้เงื่อนไข (Logic Statement) ในการตรวจสอบการทำงานของ while loop เพื่อจะสิ้นสุดการทำงานหรือทำงานต่อไป โดยการตรวจสอบเงื่อนไข ถ้าตรงตามเงื่อนไข (เงื่อนไขเป็นจริง) คำสั่ง while ก็จะดำเนินการวนซ้ำในเครื่องหมายปีกกาเปิดปิด { } และหากไม่ตรงตามเงื่อนไขหรือเงื่อนไขเป็นเท็จจะออกจากการทำงานซ้ำ หรือจบการทำงานซ้ำของ while loop แสดงดังผังงานดังรูปที่ 9.9

ผังงานของ while Loop



รูปที่ 9.9 ผังงานของรูปแบบ While-loop

ผังงานของ while Loop ประกอบด้วย 2 ส่วน คือ expression และ กลุ่มคำสั่ง statements ภายในลูป ซึ่งการใช้สำหรับการเขียนคำสั่งของ while (expression) แสดงดังคำอธิบายดังนี้

expression คือ ตรวจสอบสภาวะการทำงานตามเงื่อนไขที่กำหนด โดยผลลัพธ์ของเงื่อนไขสามารถระบุได้ดังนี้ ผลลัพธ์ของเงื่อนไขเป็นจริง (True) จะทำงานซ้ำภายในเครื่องหมาย { } และเมื่อผลลัพธ์ของเงื่อนไขเป็นเท็จ (False) จะออกจากการทำงานซ้ำ

กลุ่มคำสั่ง statements คือ กลุ่มคำสั่งที่ทำงานภายใต้เครื่องหมายปีกกาเปิดและปีกกาปิด { } รวมถึงการเปลี่ยนแปลงค่าของตัวแปรสำหรับการนับจำนวนที่ทำงานใน while loop

โดยกระบวนการขั้นตอนจะเริ่มต้นลำดับที่ (1) ทำการตรวจสอบเงื่อนไขที่กำหนด หากเงื่อนไขเป็นเท็จจะออกจากลูป หากเงื่อนไขเป็นจริงให้ทำลำดับที่ (2) ดังนั้นเมื่อเงื่อนไขเป็นจริง จะทำในกลุ่มคำสั่งภายใต้เครื่องหมายปีกกาเปิดและสิ้นสุดปีกกาปิด รวมถึงการเปลี่ยนแปลงค่าของตัวแปรสำหรับตัวนับ เพิ่มขึ้นหรือลดลงค่าตัวแปรตัวนับ และกลับมาทำในลำดับที่ (1) ต่อไป และทำงานวนลูปจนกว่าเงื่อนไขจะเป็นเท็จ จึงออกจากลูปนั้นเอง

รูปแบบการเขียนโปรแกรม while Loop

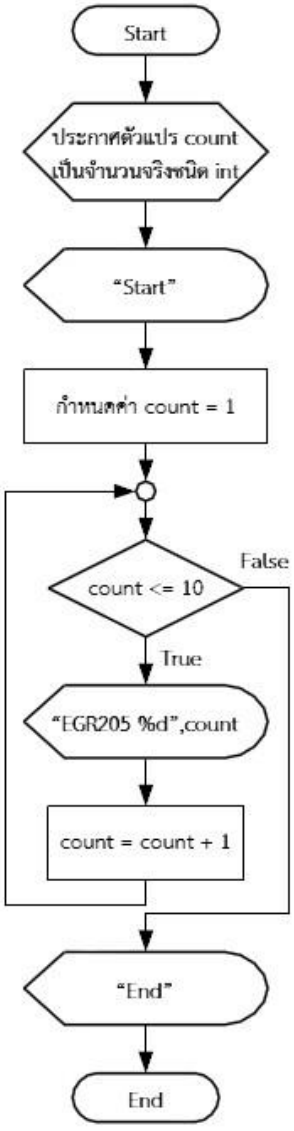
```
while ( expression )
{
    การทำงานกลุ่มคำสั่งต่างๆ ภายใน while loop ต้องอยู่ภายในเครื่องหมายปีกกา ;
    การเพิ่มจำนวนหรือลดจำนวนของตัวแปรสำหรับตัวนับ ;
}
```

จากรูปแบบการเขียนโปรแกรม while Loop มีการตรวจสอบเงื่อนไข ในส่วน expression ของคำสั่ง while ก่อนเสมอ ถ้าเงื่อนไขเป็นจริง ให้ทำงานภายในคำสั่ง หรือ กลุ่มคำสั่งที่ต้องการให้ทำซ้ำภายในเครื่องหมายปีกกา { } และจะทำซ้ำไปตราบใดที่เงื่อนไขเป็นจริง และหากไม่ตรงตามเงื่อนไขหรือเงื่อนไขเป็น

เท็จ จึงออกจากการทำงานซ้ำหรือออกจากลูปนั้นเอง ในกรณีเมื่อมีการตรวจสอบเงื่อนไขครั้งแรกเป็นเท็จ โปรแกรมก็จะไม่มีการทำงานภายในคำสั่งหรือกลุ่มคำสั่งใดๆ

ตัวอย่างที่ 9.6 จงเขียนผังการทำงานและโปรแกรม แสดงผลลัพธ์ข้อความ EGR205 ทางหน้าจอ จำนวน 10 ครั้ง โดยใช้คำสั่ง while loop

ผังงานตัวอย่างที่ 9.6



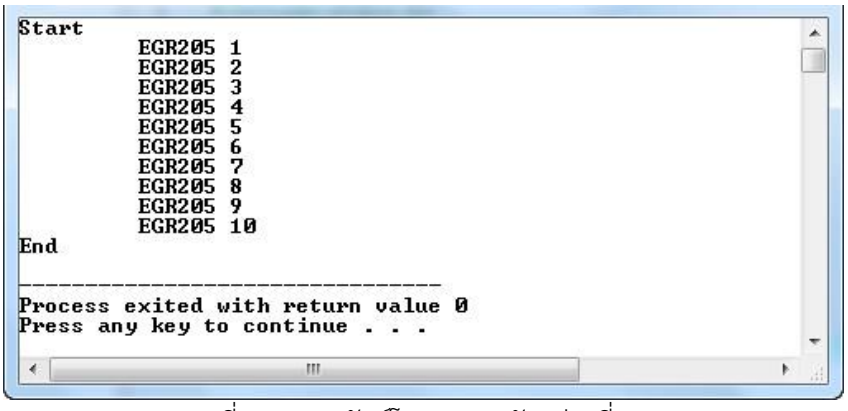
รูปที่ 9.10 ผังการทำงานของ ตัวอย่างที่ 9.6

โปรแกรมตัวอย่างที่ 9.6

```
1. #include<stdio.h>
2. main ( )
```

```
3.  {       int  count = 1 ;
4.          printf(“Start \n”) ;
5.          while ( count <= 10)
6.          {       printf(“ \t EGR205 %d \n” , count) ;
7.                  count++ ;
8.          }
9.          printf(“End\n”) ;
10.  }
```

ผลลัพธ์การทำงานของโปรแกรมตัวอย่างที่ 9.6



รูปที่ 9.11 ผลลัพธ์โปรแกรม ตัวอย่างที่ 9.6

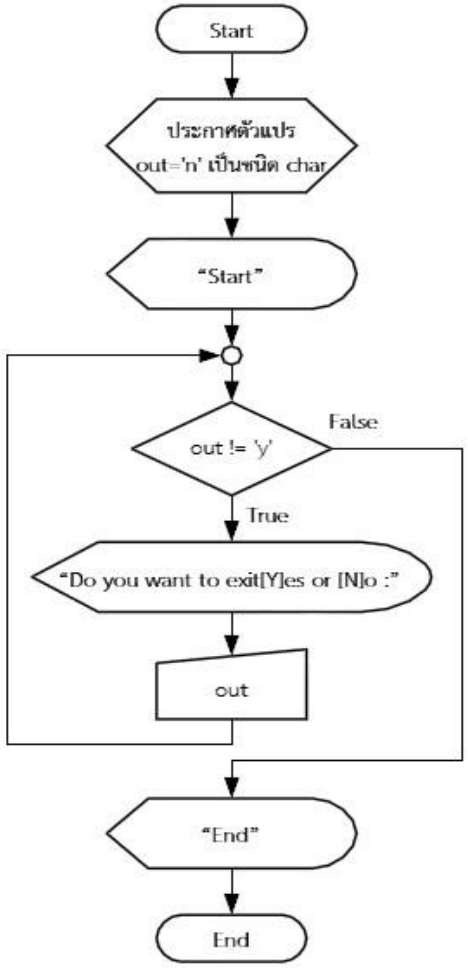
จากตัวอย่างได้กำหนดค่า count เท่ากับ 1 จากนั้นทำตามคำสั่งตามลำดับ จึงมาตรวจสอบเงื่อนไขของ while loop ตามเงื่อนไข count น้อยกว่าหรือเท่ากับ 10 หรือไม่ ถ้าตรงตามเงื่อนไข ให้ทำคำสั่งที่อยู่ภายในเครื่องหมายปีกกา ซึ่งมีการแสดงข้อความออกมาทางจอภาพ และทำการเพิ่มค่าของตัวนับไปอีก 1 จากนั้นก็จะกลับไปทำการตรวจสอบเงื่อนไข และทำซ้ำไปเรื่อยๆจนกว่า ค่าภายในตัวแปร count มีค่ามากกว่า 10 จึงจะหยุดการทำซ้ำ หรือออกจากการลูปนั้นเอง

สังเกตตัวอย่างนี้จะมีผลลัพธ์ที่ เหมือนกับตัวอย่างที่ 9.2 และตัวโปรแกรมจะสอดคล้องกับการใช้ for ซึ่งมีการวนลูปจำนวน 10 ครั้ง โดยใช้ตัวแปร count เป็นตัวนับ แต่มีข้อแตกต่างกันเล็กน้อย คือการเปลี่ยนแปลงค่า ตัวแปร count ต้องเขียนกำหนดคำสั่ง count++ หรือสามารถเขียนในรูปของ count = count +1 ภายในเครื่องหมายปีกกา **{ }**

นอกเนื่องจากนี้ การใช้คำสั่ง while แบบไม่จำกัดการทำซ้ำ สามารถทำได้โดยใช้การตรวจสอบเงื่อนไขอย่างเดียว เพื่อให้การตรวจสอบเงื่อนไขเป็นจริงเสมอ ดังตัวอย่างที่ 9.7

ตัวอย่างที่ 9.7 การใช้คำสั่ง while loop สำหรับการทำงานซ้ำๆ แบบไม่กำหนดการนับในลูป

ผังงานตัวอย่างที่ 9.7

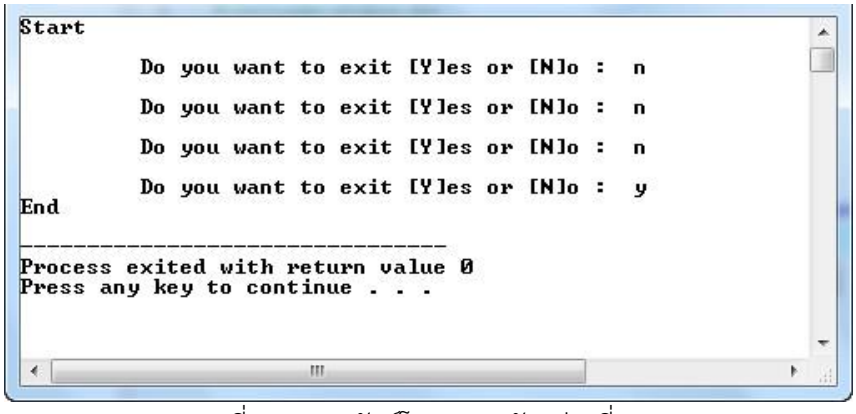


รูปที่ 9.12 แผนผังการทำงานของโปรแกรมตัวอย่างที่ 9.7

โปรแกรมตัวอย่างที่ 9.7

```
1.  #include<stdio.h>
2.  main ( )
3.  {           char  out = 'n' ;
4.              printf("Start \n") ;
5.              while ( out != 'y')
6.              {
7.                      printf(" \n \t Do you want to exit [Y]es or [N]o :  " ) ;
8.                      scanf("%c", &out) ;
9.              }
10.             printf("End\n") ;
11. }
```

ผลลัพธ์การทำงานของโปรแกรมตัวอย่างที่ 9.7



รูปที่ 9.13 ผลลัพธ์โปรแกรมตัวอย่างที่ 9.7

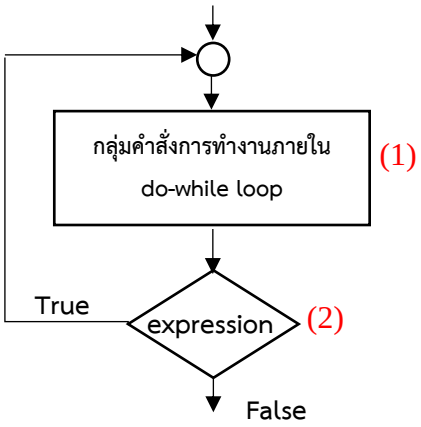
จากโปรแกรมตัวอย่างที่ 9.7 ด้วยการตรวจสอบเงื่อนไข out != 'y' หมายถึง ตัวแปร out ไม่เท่ากับอักษร y โดยกำหนดค่าเริ่มต้น out = 'n' เมื่อทำการตรวจสอบเงื่อนไข 'n' != 'y' และทำซ้ำในลูปตราบใดที่เงื่อนไขเป็นจริง และจะออกจากการทำซ้ำเมื่อเงื่อนไขเป็นเท็จ

จะสังเกตในคำสั่งที่การกำหนดค่าตัวแปร out มีค่าเท่ากับตัวอักษร n เมื่อโปรแกรมทำงานตามลำดับในส่วนของตรวจสอบเงื่อนไขของ while โดยตัวแปร out จะถูกทำการตรวจสอบตามเงื่อนไข มีค่าไม่เท่ากับอักษร y จริงหรือเท็จ ถ้าผลลัพธ์ตรงตามเงื่อนไขจริง จะทำกลุ่มคำสั่งที่อยู่ภายในเครื่องหมายปีกกา { } ซึ่งจะมีการแสดงผลลัพธ์ข้อความว่า " Do you want to exit [Y]es or [N]o : " ลำดับถัดมาคือ รอรับค่าจากแป้นพิมพ์ เพื่อเก็บไว้ในตัวแปร out โดยกำหนดตัวอักษรทางคีย์บอร์ด เมื่อป้อนอักษร n หรืออักษรใดๆ หรือตัวเลขจำนวนใดๆ แล้วเสร็จ โปรแกรมก็จะทำการวนกลับเพื่อตรวจสอบเงื่อนไขอีกครั้ง หากตรงตามเงื่อนไข ก็จะทำซ้ำๆ จนกระทั้งผู้ใช้งานจะพิมพ์ตัวอักษร y จึงจะออกจากการทำงานซ้ำ หรือจบการทำงานของโปรแกรม

9.3 การทำงานซ้ำด้วย do–while loop

คำสั่ง do–while มีการทำงานลักษณะเช่นเดียวกับคำสั่ง while แต่จะแตกต่างที่รูปแบบการทำงาน โดยคำสั่ง do...while จะทำกลุ่มคำสั่งที่อยู่ภายในเครื่องหมายปีกกา { } ก่อนเสมอ แล้วจึงจะทำการตรวจสอบเงื่อนไขใน while ถ้าผลการตรวจสอบเงื่อนไขเป็นจริง ให้กลับไปทำงานซ้ำภายในเครื่องหมายปีกกา { } ตราบใดผลการตรวจสอบเงื่อนไขเป็นเท็จ ให้ออกจากการทำงานของ do-while โดยผังการทำงานแสดงดังรูปที่ 9.14

ผังงานของ do-while Loop



รูปที่ 9.14 ผังงานของรูปแบบ do-while loop

ผังงานของ do-while Loop ประกอบด้วย 2 ส่วน คือ กลุ่มคำสั่ง statements และ expression ภายในลูป ซึ่งการใช้สำหรับการเขียนคำสั่งของ while (expression) โดย do-while จะทำให้กลุ่มคำสั่งก่อนถึงจะตรวจสอบเงื่อนไข แสดงดังคำอธิบายดังนี้

กลุ่มคำสั่ง statements คือ กลุ่มคำสั่งที่ทำงานภายใต้เครื่องหมายปีกกาเปิดและปีกกาปิด { } รวมถึงการเปลี่ยนแปลงค่าของตัวแปรสำหรับการนับจำนวนที่ทำงานใน while loop

expression คือ ตรวจสอบสภาวะการทำงานตามเงื่อนไขที่กำหนด โดยผลลัพธ์ของเงื่อนไขสามารถระบุได้ดังนี้ ผลลัพธ์ของเงื่อนไขเป็นจริง (True) จะทำงานซ้ำภายในเครื่องหมาย { } และเมื่อผลลัพธ์ของเงื่อนไขเป็นเท็จ (False) จะออกจากการทำงานซ้ำ

โดยกระบวนการขั้นตอนจะเริ่มต้นลำดับที่ (1) ทำในกลุ่มคำสั่งภายใต้เครื่องหมายปีกกาเปิดและสิ้นสุดปีกกาปิด รวมถึงการเปลี่ยนแปลงค่าของตัวแปรสำหรับตัวนับ เพิ่มขึ้นหรือลดลงค่าตัวแปรตัวนับ และทำการตรวจสอบเงื่อนไขที่กำหนดในลำดับที่ (2) หากเงื่อนไขเป็นเท็จจะออกจากลูป หากเงื่อนไขเป็นจริงให้ทำงานซ้ำในลำดับที่ (1) ต่อไปตราบที่เงื่อนไขยังเป็นจริง และหากเงื่อนไขจะเป็นเท็จ จึงออกจากลูปนั้นเอง แสดงรูปแบบของคำสั่ง do-while ดังนี้

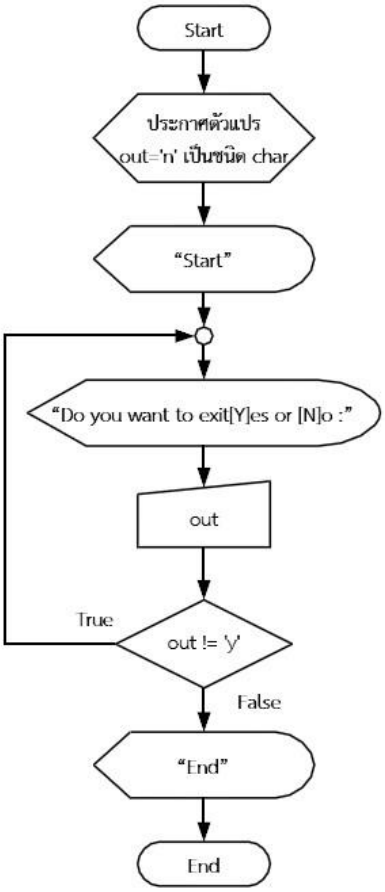
รูปแบบของคำสั่ง do-while

```
do
{
        คำสั่งหรือกลุ่มคำสั่งที่ต้องการให้ทำซ้ำ ;
}
while ( expression ) ;   //หมายเหตุ ต้องมีเครื่องหมาย ; ปิดท้ายคำสั่ง do-while นี้เสมอ
```

จากตัวอย่างที่ 9.7 การเขียนโปรแกรมด้วยคำสั่ง while นั้น ซึ่งมีรูปแบบผังการทำงานที่แตกต่างกัน โดยสามารถนำมาประยุกต์ใช้คำสั่งในเนื้อหาของ do-while ได้ดังตัวอย่างที่ 9.8

ตัวอย่างที่ 9.8 การใช้คำสั่ง do-while loop สำหรับการทำงานซ้ำๆ แบบไม่กำหนดการนับในลูป

ผังงานตัวอย่างที่ 9.8

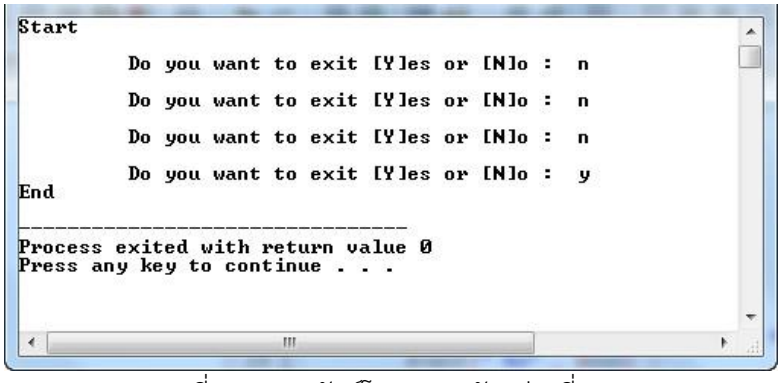


รูปที่ 9.15 ผังการทำงานของโปรแกรมตัวอย่างที่ 9.8

โปรแกรมตัวอย่างที่ 9.8

```
1.  #include<stdio.h>
2.  main ( )
3.  {           char  out = 'n' ;
4.              printf("Start \n") ;
5.              do
6.              {       printf(" \n \t Do you want to exit [Y]es or [N]o :  " ) ;
7.                      scanf(" %c" , &out) ;
8.              } while ( out != 'y') ;
9.              printf("End\n") ;
10. }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 9.8



รูปที่ 9.16 ผลลัพธ์โปรแกรมตัวอย่างที่ 9.8

ผลลัพธ์ของโปรแกรมตัวอย่างที่ 9.7 และผลลัพธ์ของโปรแกรมตัวอย่างที่ 9.8 จะมีผลลัพธ์เหมือนกัน แต่จะแตกต่างวิธีการใช้คำสั่ง เมื่อสังเกตผังการทำงานและโปรแกรม จะทำคำสั่งการแสดงข้อความ แล้วจึงทำการตรวจสอบเงื่อนไข ดังนั้นจึงทำให้ คำสั่ง do...while จะต้องทำคำสั่งหรือกลุ่มคำสั่งที่อยู่ภายในเครื่องหมายปีกกา { } ก่อนเสมอ แล้วจึงตรวจสอบเงื่อนไข และหากตรวจสอบแล้วเงื่อนไข ยังคงเป็นจริง ยังคงทำการวนซ้ำเพื่อทำคำสั่งที่อยู่ภายในเครื่องหมายปีกกา และเมื่อการตรวจสอบเงื่อนไขเป็นเท็จ จึงจะออกจากการทำซ้ำหรือออกจากลูปของโปรแกรม

<u>สรุปท้ายบท</u>

รูปแบบการทำงานของซ้ำของ for loop จะเป็นการทำงานซ้ำแบบมีจำนวนลูปนับได้ นิยมใช้ในการทำงานซ้ำๆที่สามารถกำหนดจำนวนลูปได้ จากเงื่อนไขทั้ง 3 ชุด และหากทำการระบุไม่ครบ ก็จะไม่สามารถทำงานได้ และตัวแปรที่ใช้ใน expression จะต้องเป็นตัวแปรเดียวกัน

โดยรูปแบบของ while loop และ do-while loop มีความใกล้เคียงกัน แต่มีความแตกต่างกันของการดำเนินการ ผังการทำงานและรูปแบบโครงสร้างของโปรแกรม โดย while loop จะตรวจสอบเงื่อนไขก่อนแล้วจึงจะทำคำสั่ง ซึ่งแตกต่างจาก do-while loop จะทำกลุ่มคำสั่งก่อนแล้วจึงตรวจสอบเงื่อนไข

ดังนั้นการเปลี่ยนรูปแบบจาก for loop ให้เป็น while loop หรือในรูปแบบของ do-while loop สามารถทำได้ในกรณีที่มีจำนวนลูปที่คงที่และนับจำนวนการทำซ้ำในแต่ละครั้งได้

คำถามท้ายบท

1. คำสั่งในการใช้งานของรูปแบบการทำงานซ้ำหรือการวนลูป สามารถจำแนกได้กี่ประเภท?
2. กำหนดให้ลูป for(int a=10 ; a<=100 ; a+=10) ลูปทำงานกี่ครั้ง?
3. กำหนดให้ตัวแปร count = count / A สามารถเขียนสมการในรูปแบบย่อได้หรือไม่?
4. ลูปประเภทใดสามารถนำไปสร้างรูปแบบการทำงานซ้ำแบบไม่จำกัด (Infinity loop) ?
5. การระบุค่าเริ่มต้นจำนวนนับ 100 และจำนวนนับลดลง ครั้งละ 5 จะเขียนได้วิธีใดบ้าง?
6. กำหนดให้เงื่อนไขของการทำงานซ้ำ while(z != 'z') ตัวแปรที่ทำงานซ้ำคือ?
7. กำหนดให้ for(int j=0 ; j<10 ; j++) ; { printf("Hello\n") ; } จะแสดงข้อความกี่ครั้ง?
8. การดำเนินการ z += a และ z++ กำหนดให้ z=1 และ a=1 มีผลลัพธ์ต่างกันอย่างไร?
9. ข้อแตกต่างระหว่าง while loop และ do-while loop คืออะไร?
10. เครื่องหมายปีกกาเปิดปิด { } สำหรับลูปทุกประเภทมีความสำคัญอย่างไร?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมและผังงาน สำหรับการหาผลบวกของเลขตั้งแต่ 1 จนถึงค่าที่เรารับมาจาก แป้นพิมพ์ โดยค่าที่รับจากแป้นพิมพ์มีค่าน้อยกว่าหรือเท่ากับ 0 ให้โปรแกรมทำการ แสดงข้อความว่า " **Error !! Please try again number more 0**" แล้วจบการทำงาน ถ้าค่าที่รับมาจากแป้นพิมพ์มีค่ามากกว่า 0 จึงทำการคำนวณหาค่าผลบวกและแสดงผลตามตัวอย่างแล้วค่อยจบการทำงาน

   ตัวอย่างผลลัพธ์ที่ 1 (ค่าที่รับมา น้อยกว่าหรือเท่ากับ 0)

   กำหนดค่าเท่ากับ -10

   ```
   Please Enter number = -10
   Error !! Please try again number more 0
   ```

   ตัวอย่างผลลัพธ์ที่ 2 (ค่าที่รับมา มากกว่า 0)

   กำหนดค่าเท่ากับ 10

   ```
   Please Enter number = 10
   Sum (1 2 3 4 5 6 7 8 9 10) = 55
   ```

2. จงเขียนโปรแกรมและผังงาน แสดงผล "Welcome To Sripatum University" จำนวน 10 ครั้ง โดยกำหนดให้ใช้การทำงานซ้ำ แบบ for-loop, while-loop และ do-while loop

3. จงเขียนโปรแกรมและผังงาน การทำงานซ้ำด้วยการแสดงผลลัพธ์การทำงานของการแสดงผลลัพธ์จำนวนเต็มบวก โดยกำหนดให้รับค่าอินพุตทางคีย์บอร์ดจากผู้ใช้งาน

ตัวอย่างแสดงผลลัพธ์ดังนี้

Enter number = 5

1

12

123

1234

12345

4. จงเขียนโปรแกรมและผังงาน ให้แสดง ชื่อ-นามสกุล รหัสนักศึกษา ออกทางเอาต์พุตหน้าจอภาพ โดยรับค่าจำนวนครั้งทางคีย์บอรด์

ตัวอย่างผลลัพธ์

Enter Number Count = 5

No. 1 Name: Engineering Surname: Sripatum ID 6099999

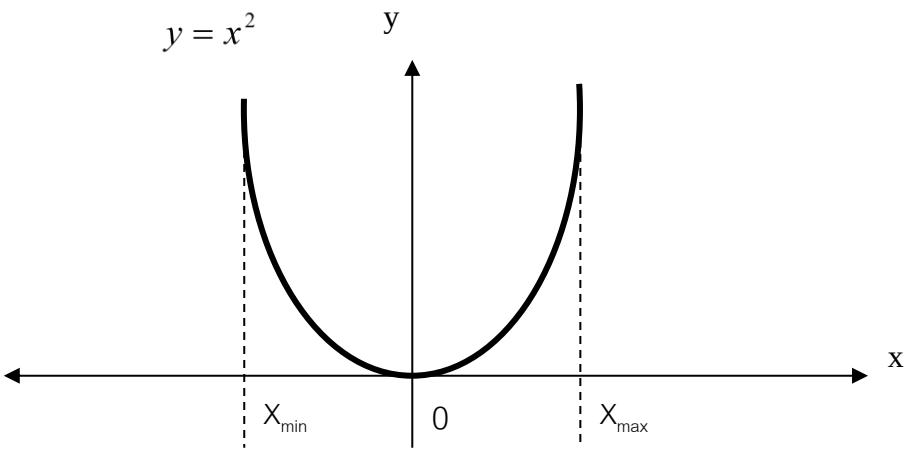
No. 2 Name: Engineering Surname: Sripatum ID 6099999

No. 3 Name: Engineering Surname: Sripatum ID 6099999

No. 4 Name: Engineering Surname: Sripatum ID 6099999

No. 5 Name: Engineering Surname: Sripatum ID 6099999

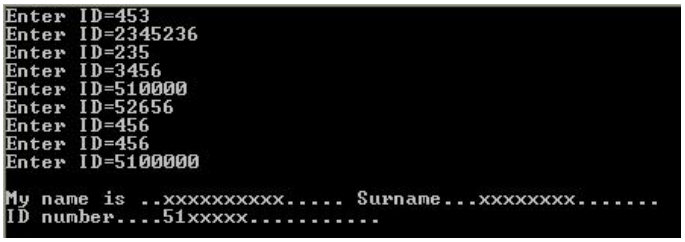
5. จงเขียนโปรแกรมและผังงาน สำหรับทำการคำนวณสมการพาราโบรา $y = x^2$ โดยกำหนดรับค่า x_{min} และ x_{max} ทางคีย์บอรด์ โดยกำหนดค่า x_{min} ต้องน้อยกว่า x_{max} โดยช่วงระหว่าง x_{min} และ x_{max} ทำการเพิ่มขึ้นครั้งละหนึ่ง



6. จงเขียนโปรแกรมและผังงาน สำหรับการตรวจสอบรหัสผ่าน Password โดยให้ลูปในการทำงาน ซึ่งทำการตรวจสอบ Password โดยกำหนดรหัสผ่านคือ **<u>รหัสนักศึกษาของตนเอง</u>** โดยทำการรับค่า password ทางคีย์บรอด์ รับค่าไปตรวจสอบ ถ้าค่าที่รับไม่ถูกต้อง ให้รับค่าไม่จำกัดจำนวนครั้งจนกว่ารหัสจะถูกต้อง เมื่อถูกต้องตามรหัสให้แสดงข้อความต่อไปนี้

My name is .....ชื่อ..... Surname .....นามสกุล.....
ID number .....xxxxxxx.....

ตัวอย่างผลลัพธ์การทำงาน



แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 10 ตัวแปรอาเรย์ (Array)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจกระบวนการใช้งานตัวแปรอาเรย์
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและโปรแกรมด้วยตัวแปรอาเรย์

เนื้อหา

- การสร้างตัวแปรอาเรย์ 1 มิติ, อาเรย์ 2 มิติ และเข้าใจรูปแบบอาเรย์ 3 มิติ
- การเข้าถึงตำแหน่งของสมาชิกในตัวแปรอาเรย์ 1 มิติ, อาเรย์ 2 มิติ และเข้าใจรูปแบบอาเรย์ 3 มิติ
- การให้การทำงานซ้ำหรือลูป ในการระบุตำแหน่ง การเก็บข้อมูล และอ่านข้อมูลของตำแหน่งอาเรย์

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการสร้างตัวแปรอาเรย์ กระบวนการเข้าถึงตำแหน่งหรือสมาชิกของอาเรย์
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 10

ตัวแปรอาเรย์

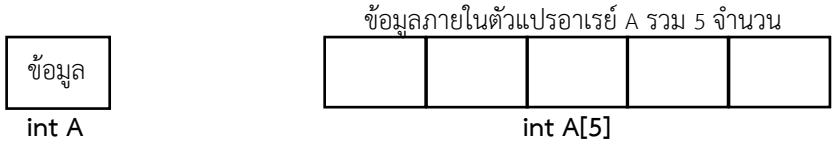
Array Variable

โดยตัวแปรปกติทั่วไปสามารถกำหนดค่าหรือเก็บค่าได้ตามข้อกำหนดของการประกาศตัวแปรนั้นๆ สำหรับการกำหนดตัวแปรของภาษาซี นอกเหนือจากตัวแปรปกติที่ใช้งานในเนื้อหาที่ผ่านมา ในบทเรียนนี้ศึกษาเรื่องตัวแปรที่สามารถเก็บค่าข้อมูลได้หลายๆจำนวน เรียกว่า ตัวแปรชนิดอาเรย์ (Array) ที่สามารถจัดเก็บค่าข้อมูล สำหรับตัวแปรต่างๆ ได้มากกว่า 1 จำนวน

สำหรับในเนื้อหานี้จะกล่าวถึงคุณลักษณะของตัวแปรอาเรย์ การใช้งานของอาเรย์แบบต่างๆ รวมถึงการนำอาเรย์ไปใช้ประโยชน์ในการเขียนโปรแกรม

10.1 ตัวแปรอาเรย์คืออะไร

เป็นการสร้างตัวแปร ที่สามารถเก็บค่าข้อมูลได้หลายๆ จำนวน โดยสามารถระบุจำนวนที่จัดเก็บรวมถึงขนาดของตัวแปรอาเรย์ สามารถกำหนดให้ตัวแปรแบบ int, char, float เป็นต้น (ตามข้อกำหนดการประกาศตัวแปร) โดยให้อยู่ในรูปแบบของตัวแปรอาเรย์ ในการเก็บค่าข้อมูลหลายๆ ข้อมูลไว้ในตัวแปร การประกาศตัวแปรในลักษณทั่วไปคือ int A นั้นสามารถเก็บค่าข้อมูลจำนวนเต็ม (integer) ได้ค่าเดียวเท่านั้น ซึ่งการสร้างเป็นตัวแปรอาเรย์ int A[5] แล้วจะสามารถเก็บเลขจำนวนเต็มได้ 5 จำนวน ภายในตัวแปร A[5] แสดงดังรูปที่ 10.1 ตัวแปรทั่วไปกับตัวแปรอาเรย์

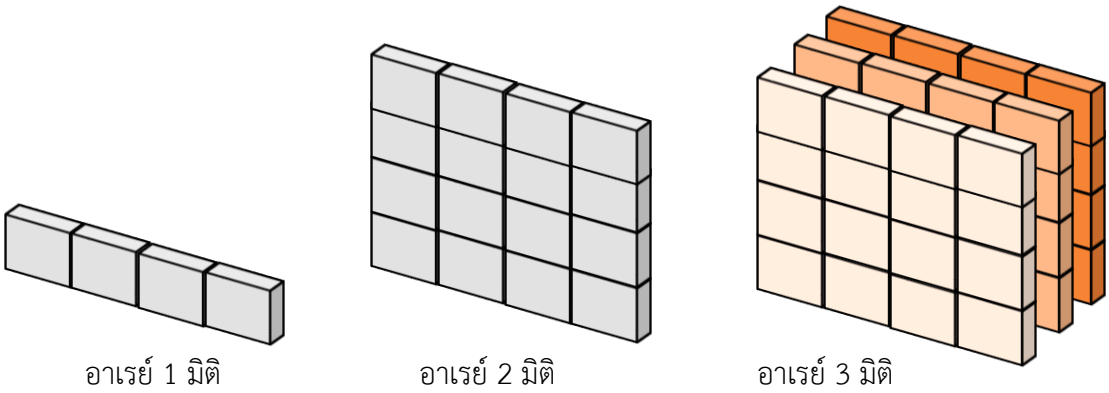


รูปที่ 10.1 ตัวแปรใช้งานทั่วไปชนิดจำนวนเต็มและตัวแปรอาเรย์

ตัวอย่างเช่น การสร้างตัวแปรอาเรย์เก็บข้อมูลหลายๆ ข้อมูลโดยไม่จำเป็นต้องสร้างตัวแปรหลายๆ ตัวแปร เมื่อต้องการเก็บข้อมูลอายุของเพื่อนทั้งหมด 20 คน จะต้องประกาศตัวแปรชนิดจำนวนเต็ม int age1, age2, age3, age4, ...,age20 รวม 20 ตัวแปร โดยหากนำตัวแปรอาเรย์มาประยุกต์ใช้งานสามารถประกาศตัวแปรชนิดจำนวนเต็ม int age[20] เป็นตัวแปรอาเรย์ที่สามารถเก็บค่าข้อมูลได้ 20 จำนวน ในแต่ละตำแหน่งของตัวแปรอาเรย์ เริ่มตำแหน่งของข้อมูลตัวแปร age[0] ถึง age[19] ตามลำดับ รวม 20 ตำแหน่ง ตัวแปรอาเรย์สามารถจำแนกลักษณะการใช้งาน ดังนี้

1. อาเรย์ 1 มิติ (One-Dimensional Array)
2. อาเรย์ 2 มิติ (Two-Dimensional Array)
3. อาเรย์ 3 มิติ (Three-Dimensional Array)

จากรูปที่ 10.2 เป็นการแสดงแบบจำลองให้เห็นความแตกต่างของอาเรย์ทั้ง 3 ลักษณะมีการใช้งานที่แตกต่างกันและโครงสร้างการประกาศตัวแปรอาเรย์ที่แตกต่างกันรวมถึงการเข้าถึงตำแหน่งของอาเรย์ด้วย รูปแบบการเก็บข้อมูลอาเรย์จะต่างกันตรงที่ลักษณะการจัดเก็บข้อมูล โดยอาเรย์ 1 มิติ จัดเก็บในแนว Single column ส่วนอาเรย์ 2 มิติ จะเก็บข้อมูลในแนว Row และ Column (รูปแบบของตาราง) และสำหรับอาเรย์ 3 มิติ จะเก็บข้อมูลในแนว Row, Column และ Depth



รูปที่ 10.2 รูปแบบจำลองของโครงสร้างอาเรย์ 1 มิติ อาเรย์ 2 มิติและอาเรย์ 3 มิติ

10.2 อาเรย์ 1 มิติ (One-Dimensional Array)

โครงสร้างตัวแปรอาเรย์ 1 มิติ ประกอบด้วย 3 ส่วน คือชนิดของตัวแปร ชื่อตัวแปร และจำนวนของอาเรย์ที่อยู่ภายในเครื่องหมาย [ ] อ่านว่า (Square brackets) การสร้างตัวแปรอาเรย์ 1 มิติ แสดงดังนี้

ชนิดของตัวแปร ชื่อตัวแปร [จำนวนอาเรย์]

ชนิดของตัวแปร คือ การระบุชนิดของตัวแปรเป็นไปตามข้อกำหนดของการประกาศชนิดของตัวแปร เช่น จำนวนเต็ม (int), จำนวนจริง (float), จำนวนอักษร (char) เป็นต้น

ชื่อตัวแปร คือ การระบุชื่อตัวแปรใช้งาน เป็นไปตามข้อกำหนดการประกาศชื่อตัวแปร ต้องไม่ซ้ำกับคำสงวนของภาษาซี

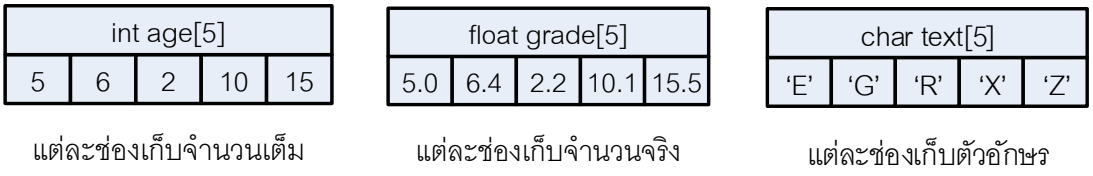
จำนวนอาเรย์ คือ กำหนดขนาดของตัวแปรอาเรย์เป็นจำนวนเต็มบวก

ตัวอย่างโปรแกรมที่ 10.1 รูปแบบการประกาศตัวแปรอาเรย์ 1 มิติ

```
1.  #include <stdio.h>          // ไลบรารี อินพุตและเอาต์พุต
2.  main ( )                    // ฟังก์ชั่นหลัก หรือ โปรแกรมหลัก
3.  {                           // เริ่มต้นการทำงานของโปรแกรม
4.   int age[5] ;               // ประกาศตัวแปรอาเรย์ age จำนวนเต็ม 5 จำนวน
5.   float grade[5] ;           // ประกาศตัวแปรอาเรย์ grade จำนวนจริง 5 จำนวน
6.   char text [5] ;            // ประกาศตัวแปรอาเรย์ text จำนวนอักษร  5 จำนวน
7.  }                           // สิ้นสุดการทำงานของโปรแกรม
```

จากตัวอย่างโปรแกรมที่ 1 เป็นการประกาศตัวแปรชื่อ age ให้เป็นอาเรย์ข้อมูลชนิดจำนวนเต็ม int มีขนาดเท่ากับ 5 จำนวน โดยตำแหน่งของอาเรย์ age[0], age[1],... ,age[4] สำหรับตัวแปร grade เป็นอาเรย์ข้อมูลชนิดจำนวนจริง float มีขนาดเท่ากับ 5 จำนวน โดยตำแหน่งของอาเรย์ grade[0], grade[1],... , grade[4] และการประกาศตัวแปร text ให้เป็นอาเรย์ข้อมูลชนิดตัวอักษร char มีขนาดเท่ากับ 5 จำนวน โดยตำแหน่งของอาเรย์ text[0], text[1],... , text[4]

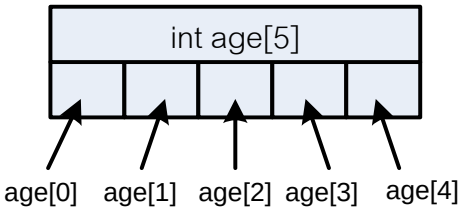
เมื่อประกาศ ตัวแปรอาเรย์ทั้งสามตัวคือ age, grade และ text แล้ว ในหน่วยความจำจะมีการจองพื้นที่เอาไว้ตามจำนวนที่กำหนด โดยตัวแปร age และ grade นั้นจะมีการเตรียมพื้นที่ว่างในหน่วยความจำสำหรับเก็บค่าตัวแปรละ 5 ค่า และตัวแปร test ก็มีการเตรียมพื้นที่เอาไว้เก็บตัวอักษร 5 ตัว ดังรูปที่ 10.3



รูปที่ 10.3 ตัวแปรชนิดอาเรย์ในการเก็บข้อมูล

การนำค่าใส่ลงในตัวแปรอาเรย์

จากที่ได้อธิบายเอาไว้แล้วว่าตัวแปรอาเรย์นั้นสามารถเก็บค่าได้หลาย ๆค่า โดยแต่ละค่าก็จะเหมือนกับเป็นตัวแปร 1 ตัว เช่น ถ้าประกาศตัวแปร int age[5] ก็จะเหมือนกับว่าเรามีตัวแปร age ถึง 5 ตัว ซึ่งแต่ละตัวนี้เราเรียกว่าสมาชิกของอาเรย์การอ้างถึงสมาชิกของอาเรย์จะต้องใช้หมายเลขลำดับ โดยเริ่มจาก 0, 1, 2, ... จนถึง “ขนาดของอาเรย์ลบด้วย 1” เช่นถ้าเราสร้างอาเรย์ int age[5] การอ้างถึงสมาชิกของอาเรย์จะใช้หมายเลข 0 ถึง 4 ดังรูปที่ 10.4

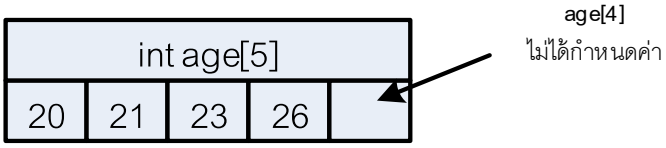


รูปที่ 10.4 การเข้าถึงสมาชิกของอาเรย์

จากรูปที่ 10.4 หากต้องการนำเอาค่า 20, 21, 23 และ 26 มากำหนดให้กับสมาชิกลำดับที่ 0, 1, 2 และ 3 ของอาเรย์ age ตามลำดับ สามารถเขียนโปรแกรมดังนี้

```
age[0] = 20;
age[1] = 21;
age[2] = 23;
age[3] = 26;
```

จะเห็นได้ว่าตัวแปร a เป็นอาเรย์แบบ int ซึ่งเก็บเลขจำนวนเต็มได้ 5 ค่า แต่จากตัวอย่างกำหนดค่าให้กับสมาชิกลำดับที่ 0 ถึง 3 โดยไม่ได้กำหนดค่าให้กับสมาชิก ลำดับที่ 4 เพราะว่าการใส่ข้อมูลลงในอาเรย์นั้นไม่จำเป็นจะต้องใส่ทุกๆ ช่องให้ครบจึงจะใช้งานได้ ช่องใดไม่ได้ใส่ค่าลงไป ซึ่งก็ไม่เก็บค่าอะไรไว้จะเป็นช่องว่างๆ ไปโดยอัตโนมัติ



รูปที่ 10.5 การระบุข้อมูลสมาชิกของอาเรย์

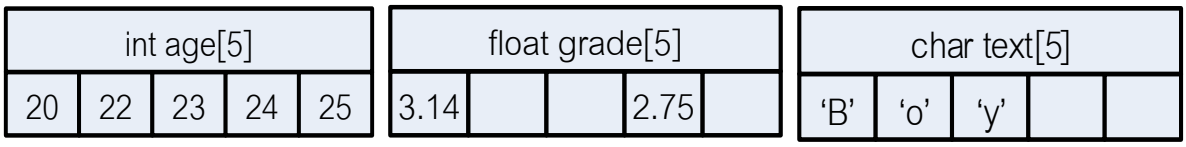
สำหรับการนำค่าใส่ในตัวแปรอาเรย์ของข้อมูลชนิดอื่นก็เช่นเดียวกัน จะต้องระบุด้วยว่าจะใส่ลงช่องที่เท่าไหร่ ดังตัวอย่างโปรแกรมที่ 10.2 ต่อไปนี้

ตัวอย่างโปรแกรมที่ 10.2

```
1.  #include<stdio.h>
2.  main ( )
3.  {       int age[5];
4.          float grade[5];
5.          char text [5];
6.          age[0] = 20;
7.          age[1] = 22;
8.          age[2] = 23;
```

```
9.          age[3] = 24;
10.         age[4] = 25;
11.         grade[0] = 3.14;
12.         grade[3] = 2.75;
13.         text[0] = 'B';
14.         text[1] = 'o';
15.         text[2] = 'y';
16. }
```

ผลการทำงานโปรแกรมตัวอย่างที่ 10.2 จะได้ดังนี้



รูปที่ 10.6 การระบุข้อมูลชนิดแตกต่างกับตัวแปรอาเรย์

จากโปรแกรมตัวอย่างที่ 10.2 มีการประกาศตัวแปรอาเรย์และกำหนดค่าให้กับสมาชิกของอาเรย์โดยจะกำหนดค่าให้กับสมาชิกทุกตัวของอาเรย์หรือกำหนดค่าให้กับสมาชิกเพียงบางตัวก็ได้ ขอเพียงให้ลำดับสมาชิกที่อ้างถึงนั้นมีอยู่จริงก็พอ ยกตัวอย่างเช่น ถ้า ประกาศว่า int age[5] จะสามารถกำหนดค่าได้ตั้งแต่ age[0] ถึง age[4] เท่านั้น ไม่สามารถกำหนดค่าให้กับ age[5], age[8] หรือ age[20] ได้ เพราะไม่มีสมาชิกที่มีหมายเลขลำดับเหล่านี้อยู่ และถ้าพยายามกำหนดก็จะก่อให้เกิดความผิดพลาด

10.3 การประยุกต์ใช้งานตัวแปรอาเรย์

ในแต่ละช่องหรือแต่ละสมาชิกของตัวแปรอาเรย์จะเหมือนเป็นหนึ่งตัวแปร เพราะฉะนั้นการนำเอาค่าในช่องใด ๆ ไปใช้ก็เหมือนกับการใช้งานตัวแปรธรรมดาตัวแปรหนึ่ง แสดงอาเรย์ a ดังนี้

```
int a[5];
        a[0] = 20;
        a[1] = 22;
        a[2] = 23;
        a[3] = 24;
        a[4] = 25;
```

ถ้าต้องการแสดงค่าในตัวแปรอาเรย์จะเขียนดังนี้

```
printf(" a 0 = %d\n" , a[0] );
printf(" a 1 = %d\n" , a[1] );
printf(" a 2 = %d\n" , a[2] );
printf(" a 3 = %d\n" , a[3] );
printf(" a 4 = %d\n" , a[4] );
```

หรือ อาจใช้ คำสั่งเกี่ยวกับ การวนลูปช่วยได้เช่น

```
int i ;
for (i=0 ; i<5 ; i++)
{
        printf(" a %d = %d\n",i, a[ i ] );
}
```

ซึ่งการเขียนทั้งสองแบบจะได้ผลลัพธ์เหมือนกันคือ

```
a 0 =  20
a 1 =  22
a 2 =  23
a 3 =  24
a 4 =  25
```

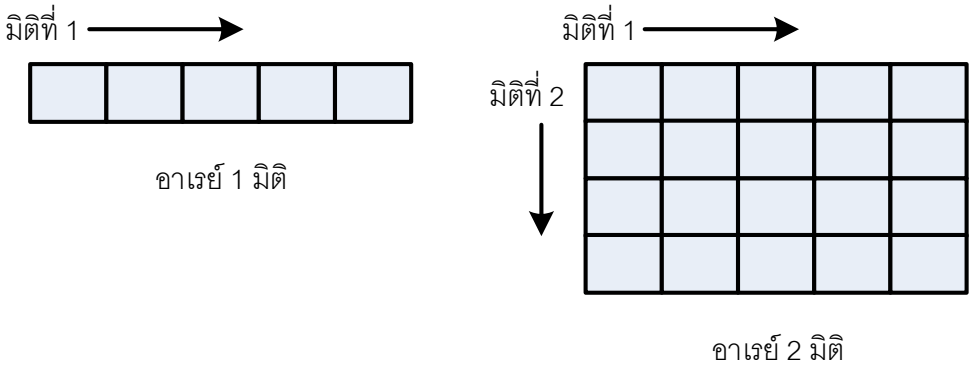
การจะนำค่า ของอาเรย์มาใช้นั้นจำเป็นต้อง อ้างอิงโดยชื่อและตำแหน่งของอาเรย์เสมอ จะไม่สามารถเรียกใช้ได้โดยการเรียกชื่ออย่างเดียวเช่น

```
printf(" a  = %d\n" , a );          // แบบนี้ไม่สามารถเรียกใช้ได้
printf(" a  = %d\n" , a[0] );       // สามารถเรียกใช้ได้
```

เนื่องจาก a เป็นตัวแปรอาเรย์ ดังนั้นเวลาใช้งาน ซึ่งจะต้องบอกด้วยว่าจะนำค่าในตำแหน่งใดไปใช้ โดยสามารถนำค่าในอาเรย์ไปคำนวณทางคณิตศาสตร์ เช่น บวก, ลบ, คูณ, หาร หรือประมวลผลใดๆ ก็ได้ในแบบเดียวกันกับตัวแปรทั่วๆ ไป

10.4 อาเรย์ 2 มิติ (Two-Dimensional Array)

การเก็บข้อมูลแบบอาเรย์1 มิตินั้นจะเก็บต่อกันเป็นอนุกรม และสามารถเก็บค่าหลาย ๆ ค่าได้พร้อมกัน และช่วยแก้ไขปัญหาในเรื่องของการประกาศตัวแปรมาก ๆ อีกด้วย แต่นอกเหนือจากอาเรย์1 มิติแล้ว ยังมีอาเรย์อีก 2 ประเภท นั่นคืออาเรย์ 2 มิติ และอาเรย์ 3 มิติ ซึ่งตัวแปรอาเรย์ 2 มิติจะเก็บข้อมูลไว้ในลักษณะของตาราง ดังรูปที่ 10.7

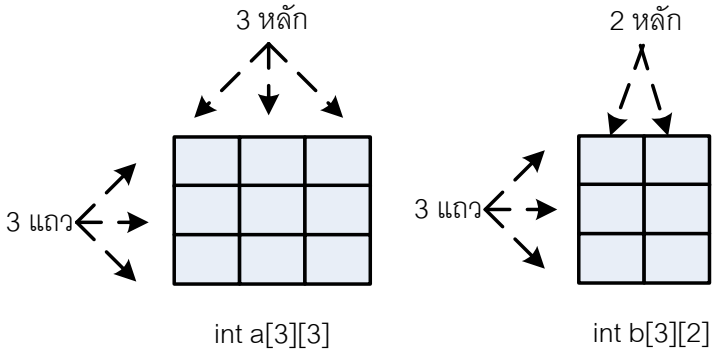


รูปที่ 10.7 ตัวแปรอาเรย์ 1 มิติ และตัวแปรอาเรย์ 2 มิติ

การสร้างอาเรย์ 2 มิตินั้นสามารถเขียนรูปแบบโค้ดภาษาซี ได้ดังนี้

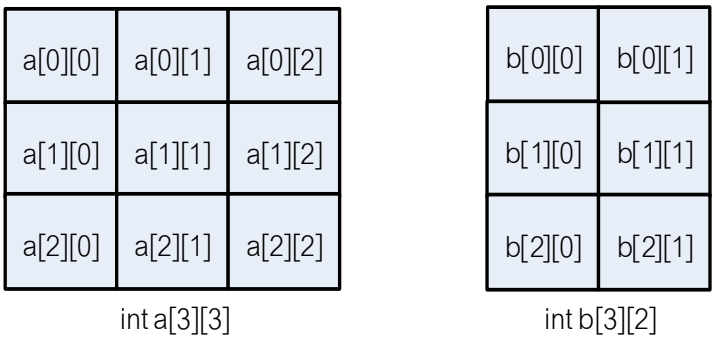
```
int a[3][3] ;
int b[2][3] ;
```

จากข้างต้นเป็นการประกาศตัวแปรอาเรย์ 2 มิติชื่อ a และ b โดยตัวแปรอาเรย์ a มีขนาด 3x3 และตัวแปรอาเรย์ b มีขนาด 2x3 แสดงดังรูปที่ 10.8



รูปที่ 10.8 จำนวนแถว (Row) และจำนวนหลัก (Column) ของตัวแปรอาเรย์ 2 มิติ

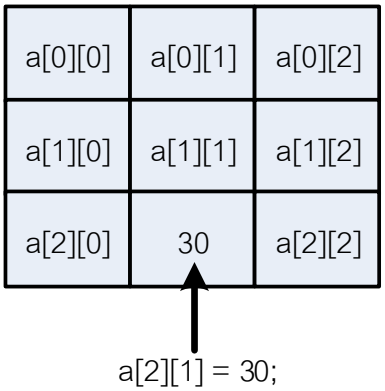
การนำค่าที่ต้องการเก็บในอาเรย์ จะต้องอ้างถึงลำดับของสมาชิกช่องนั้น ๆ ทั้งลำดับในแนวนอนและลำดับในแนวตั้ง หรือจะมองในลักษณะของคู่ลำดับก็ได้ดังรูปที่ 10.9 ต่อไปนี้



รูปที่ 10.9 ตำแหน่งสมาชิกของตัวแปรอาเรย์ 2 มิติ

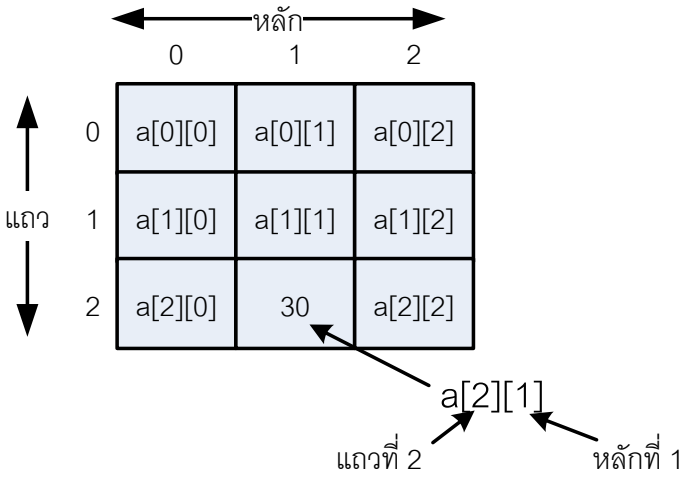
จะเห็นว่าหมายเลขลำดับของอาเรย์ในแต่ละแนวเริ่มต้นจาก 0 จนถึง “ขนาดในแนวนั้นลบด้วย 1” ถ้าประกาศอาเรย์ 2 มิติ ขนาด 3x3 ลำดับในแนวนอนก็จะเริ่มจาก 0 ถึง 2 รวมทั้งหมด 3 ช่อง และในแนวตั้งก็จะเริ่มจาก 0 ถึง 2 รวม 3 ช่อง ตามลำดับ สำหรับวิธีการนำเอาข้อมูลใส่ลงในตัวแปรอาเรย์ 2 มิติ ใช้หลักการเดียวกันกับอาเรย์ 1 มิติ โดยระบุช่องที่ต้องการใส่ค่าลงไป เช่น ถ้ากำหนดค่า 30 ลงในตำแหน่ง a[2][1] จะเขียนโปรแกรมดังนี้

```
int a[3][3] ;
a[2][1] = 30;
```



รูปที่ 10.10 การระบุค่าในตำแหน่งสมาชิกของตัวแปรอาเรย์ 2 มิติ

การอ้างถึงสมาชิกของอาเรย์ 2 มิติ จะใช้การระบุเลขลำดับสองตัวเรียงกัน คือ [x][y] โดย x เป็นเลขที่บอกว่าอยู่ช่องที่เท่าไหร่ในแนวนอน (Row) และ y บอกว่าอยู่ช่องที่เท่าไหร่ในแนวตั้ง (Column) แสดงดังรูปที่ 10.11



รูปที่ 10.11 การเข้าถึงตำแหน่งสมาชิกของตัวแปรอาเรย์ 2 มิติ

ตัวอย่างโปรแกรมที่ 10.3

```
1.  #include<stdio.h>
2.  main ( )
3.  {       // int a[3][3]={1,2,3,4,5,6,7,8,9}; หรือ  int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};  หรือ
4.          int a[3][3];
5.                  a[0][0] = 20;
6.                  a[0][1] = 25;
7.                  a[0][2] = 39;
8.                  a[1][0] = 24;
9.                  a[1][1] = 22;
10.                 a[1][2] = 23;
11.                 a[2][0] = 10;
12.                 a[2][1] = 3;
13.                 a[2][2] = 15;
14.                         printf(" %d %d %d \n" , a[0][0] , a[0][1], a[0][2] );
15.                         printf(" %d %d %d \n" , a[1][0] , a[1][1], a[1][2] );
16.                         printf(" %d %d %d \n" , a[2][0] , a[2][1], a[2][2] );
17. }
```

จากตัวอย่างโปรแกรมที่ 10.3 ประกาศตัวแปร a ให้เป็นอาเรย์ของ int ขนาด 3x3 และได้มีการกำหนดค่าให้กับสมาชิกในอาเรย์ทุกตำแหน่ง แต่ในหน่วยความจำจะมีการสร้างตัวแปรอาเรย์ a และเก็บค่าเอาไว้ ซึ่งเมื่อทำงานทดสอบโปรแกรม จะแสดงผลลัพธ์โปรแกรมที่ 10.3

ผลลัพธ์โปรแกรมที่ 10.3

```
20      25      39
24      22      23
10      3       15
```

จากตัวอย่างโปรแกรมที่ 10.3 เป็นการนำเอาค่าในแต่ละช่องของอาเรย์ออกมาแสดงที่จอภาพ โดยเขียนโค้ดให้แสดงออกมาทีละตัวๆ ตามลำดับ โดยใช้คำสั่ง printf( ) จำนวน 3 บรรทัด ซึ่งมีวิธีที่การที่สะดวกคือใช้การทำซ้ำด้วยคำสั่ง for loop เข้ามาช่วยเพื่อให้การเขียนโปรแกรมแสดงค่าข้อมูลจากตัวแปรอาเรย์ ดังนี้

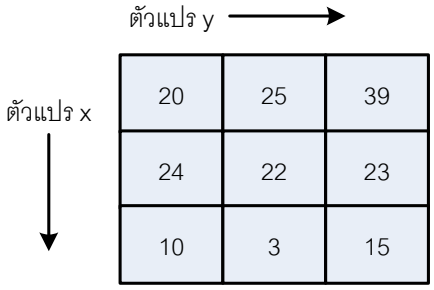
ตัวอย่างโปรแกรมที่ 10.4

```
1.  #include<stdio.h>
2.  main ( )
3.  {           int x , y;
4.              int a[3][3];
5.              a[0][0] = 20;   a[0][1] = 25;   a[0][2] = 39;
6.              a[1][0] = 24;   a[1][1] = 22;   a[1][2] = 23;
7.              a[2][0] = 10;   a[2][1] = 3;    a[2][2] = 15;
8.              for (x = 0 ; x<=2 ; x++ )
9.              {
10.                     for (y = 0 ; y<=2 ; y++ )
11.                     {
12.                             printf(" %d\t " , a[x][y] );
13.                     }
14.             printf("\n");
15.             }
16. }
```

ผลลัพธ์ตัวอย่างโปรแกรมที่ 10.4

```
20    25    39
24    22    23
10    3     15
```

จากโปรแกรมนี้จะเห็นได้ชัดว่าคำสั่ง for ช่วยให้การเขียนโปรแกรมในการทำซ้ำและแสดงผล โดยใช้ for loop ซ้อนกัน 2 ชั้น โดยคำสั่ง for ชั้นในจะใช้ตัวแปร y เป็นตัวนับ และค่าของ y นี้จะถูกใช้เป็นหมายเลขลำดับของอาเรย์ในแนวตั้ง ส่วนคำสั่ง for ชั้นนอกจะใช้ตัวแปร x เป็นตัวนับ และค่าของ x จะถูกใช้เป็นหมายเลขลำดับของอาเรย์ในแนวนอน ซึ่งค่าของ x และ y จะเพิ่มขึ้นทีละ 1 จาก 0 ถึง 2



รูปที่ 10.12 ตัวแปร x และตัวแปร y สำหรับตัวแปรของอาเรย์ 2 มิติ

ตัวอย่างโปรแกรมที่ 10.5 โจทย์กำหนดให้ ตัวแปร a มีข้อมูลเป็นเมตริกขนาด 3x3 และตัวแปร b มีข้อมูลเป็นเมตริกขนาด 3x3 ซึ่งมีชุดข้อมูลที่กำหนดให้ จงเขียนโปรแกรมหาผลลัพธ์การบวกของเมตริกทั้งสอง ดังข้อมูลที่กำหนดให้ต่อไปนี้

$$a=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \quad b=\begin{bmatrix}10 & 20 & 30\\ 40 & 50 & 60\\ 70 & 80 & 90\end{bmatrix}$$

ตัวอย่างโปรแกรมที่ 10.5

```
1.  #include <stdio.h>
2.  void main( )
3.  {
4.          int a[3][3] = {1,2,3,4,5,6,7,8,9 };
5.          int b[3][3] = {10,20,30,40,50,60,70,80,90} ;
6.          int c[3][3] ;
7.          int i , j ;
8.                  for( i =0 ; i<=2 ; i++)
9.                  {
10.                         for ( j = 0 ; j<=2 ; j++)
11.                         {
12.                                 c[ i ][ j ] = a[ i ][ j ] + b[ i ][ j ] ;
13.                                 printf( "C[%d][%d] = %d \n " ,  i,  j, c[ i ][ j ] ) ;
14.                         }
15.                 }
16. }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 10.5

C[3][3] =

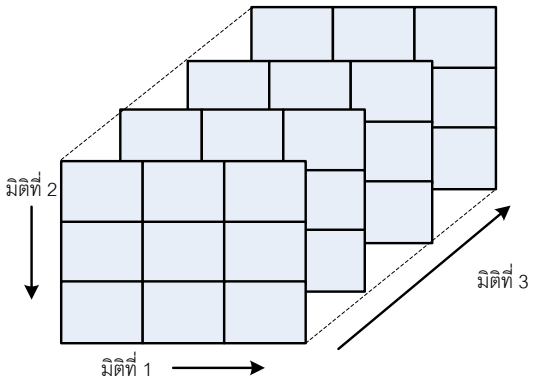
11	22	33
44	55	66
77	88	99

10.5 อาเรย์ 3 มิติ (Three-Dimensional Array)

ดังที่ได้อธิบายในตอนต้นว่า อาเรย์ 3 มิติจะเหมือนกับอาเรย์ขนาด 2 มิติมาซ้อนกันหลาย ๆ ชั้น ซึ่งก็เป็น ลักษณะของอาเรย์ 3 มิติ จะมีความลึกมาเกี่ยวข้องด้วย การประกาศอาเรย์ 3 มิติ ทำได้โดยเพิ่มมิติที่ 3 เข้าไป เช่น

```
int a[3][3][4] ;
```

จากการประกาศอาเรย์ a แบบอาเรย์ 3 มิติ ที่มีขนาด 3x3x4 โดยจำนวนเลข 2 ตัวแรก หมายถึง อาเรย์ 2 มิติ ขนาด 3x3 ช่อง หรือ 9 ตำแหน่ง และมิติที่ 3 เพิ่มเข้ามาซึ่งมีขนาดเท่ากับ 4 ซึ่งจะเหมือนกับมีอาเรย์ 2 มิติขนาด 3x3 เรียงซ้อนกันอยู่ 4 ชุดนั่นเอง แสดงดังรูปที่ 10.13



รูปที่ 10.13 โครงสร้างอาเรย์ 3 มิติ

การนำข้อมูลไปเก็บไว้ในอาเรย์ 3 มิติจะใช้เลขลำดับ 3 จำนวน เพื่ออ้างถึงตำแหน่งในแนวแนว (Row), แนวตั้ง (Column) และแนวลึก (Depth) ของอาเรย์ ดังตัวอย่างต่อไปนี้

```
a[0] [0] [0] = 20;
a[0] [0] [1] = 25;
a[0] [0] [2] = 39;
a[2] [0] [0] = 10;
a[2] [0] [1] = 3;
a[2] [0] [2] = 15;
```

โดยการใช้งานทั่วไปจะไม่ค่อยพบเห็นการใช้งานอาเรย์ 3 มิติ เพราะจะใช้ในการแก้ไขปัญหาเฉพาะด้าน เช่น การประมวณผลภาพขั้นสูง ระบบข้อมูลขนาดใหญ่ เป็นต้น ซึ่งส่วนใหญ่จะเห็นการใช้งานอาเรย์ 2 มิติ มากกว่า แต่อย่างไรก็ตาม หลักการในการกำหนดค่าและการเข้าถึงข้อมูลสามารถทำได้เช่นเดียวกัน

สรุปท้ายบท

ตัวแปรอาเรย์เป็นการเก็บข้อมูลที่มีลำดับข้อมูลหรือตำแหน่งของข้อมูล ที่ระบุได้อย่างชัดเจน ตามรูปแบบของอาเรย์ ซึ่งการสร้างตัวแปรอาเรย์สามารถเก็บข้อมูลได้มากกว่าหนึ่งจำนวน แต่ไม่สามารถเก็บชนิดข้อมูลที่แตกต่างกันได้ เช่น int a[10] ทุกสมาชิกหรือทุกตำแหน่งของอาเรย์จะเก็บข้อมูลชนิดจำนวนเต็มเท่านั้น หากต้องการเก็บข้อมูลที่มีความหลากหลาย ตัวแปรอาเรย์ไม่สามารถนำไปใช้งานได้ดีเท่าที่ควร การเข้าถึงตำแหน่งของอาเรย์หรือสมาชิกของอาเรย์ ตำแหน่งเริ่มต้นที่ 0 เสมอ เช่น int a[10] จะมีสมาชิกรวม 10 จำนวน เริ่มต้นตำแหน่งที่ 0 ถึง 9 คือ a[0], a[1] ,... ,a[9] รวมถึงอาเรย์ 2 มิติ โดยตำแหน่งเริ่มต้นที่แถวที่ 0 และหลักที่ 0 เช่นกัน ดังนั้นการระบุขนาดของอาเรย์สามารถระบุข้อมูลชนิดจำนวนเต็มบวกเท่านั้น

คำถามท้ายบท

1. ตัวแปรอาเรย์ในเนื้อหาที่เรียนมากี่ประเภท?
2. ข้อดีของตัวแปรอาเรย์ คืออะไร?
3. ข้อเสียของตัวแปรอาเรย์ คืออะไร?
4. การสร้างตัวแปรอาเรย์ 1 มิติ สำหรับเก็บข้อมูลอักษร 10 ตัวอักษร ทำอย่างไร?
5. ตำแหน่งแรกของตัวแปรอาเรย์ 1 มิติ int z[50] คือตำแหน่งใด?
6. การระบุจำนวนหรือขนาดของอาเรย์ ระบุชนิดข้อมูลแบบใด?
7. กำหนดให้ int a[5] = {1,2,3,4,5} ; หากต้องการข้อมูลในตำแหน่ง a[5] มีค่าเท่าใด?
8. กำหนดให้ int a[3][2] = {1,2,3,4,5,6} ; หากต้องการข้อมูลในตำแหน่ง a[1][1] มีค่าเท่าใด?
9. กำหนดให้ int b[5] = {1,2,3,4,5} ; หากต้องการข้อมูลในตำแหน่ง b[b[1]] มีค่าเท่าใด?
10. กำหนดให้ int b[2][2] = {1,0,3,2} ; หากต้องการข้อมูลในตำแหน่ง b[1][1]+b[1][2] มีค่าเท่าใด?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมและผังงาน สำหรับการเรียงลำดับจากน้อยไปหามาก จากจำนวนข้อมูลอาร์เรย์หนึ่งมิติ 2, 4, 1, 6, 8, 7, 0,-3,-8, 10 ต่อไปนี้

2	4	1	6	8	7	0	-3	-8	10

2. จงเขียนโปรแกรมและผังงาน สำหรับการคำนวณหา **ผลบวก** ระหว่างของเมตริก A และ B และหาผลลัพธ์ผลบวกของเมตริก โดยใช้ลูป (Loop) ในการคำนวณและแสดงผลลัพธ์การทำงานของโปรแกรม

กำหนดให้ $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$ $B=\begin{bmatrix}2 & -5 & 4\\ 7 & 6 & 9\\ -2 & 8 & 0\end{bmatrix}$

3. จงเขียนผังการทำงาน และโปรแกรมรับค่าจำนวนเลขทั้งหมด 5 ค่าจากคีย์บอร์ดเพื่อนำมา คำนวณหาค่า ผลรวม (Sum) และค่าเฉลี่ย (Average) ของจำนวนเลข 5 ตัวนั้น โดยกำหนดให้ใช้ตัวแปรแบบอาเรย์และการทำงานแบบ Loop ในการรับค่าและคำนวณผล

ตัวอย่างผลลัพธ์

```
Enter number 1 = 3
Enter number 2 = 7
Enter number 3 = 5
Enter number 4 = 2
Enter number 5 = 9
Sum (3 7 5 2 9 ) = 26.00
Average = 5.20
```

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 11 พอยน์เตอร์ (Pointer)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจกระบวนการใช้งานตัวแปรพอยน์เตอร์
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและโปรแกรมด้วยตัวแปรพอยน์เตอร์

เนื้อหา

- การสร้างตัวแปรพอยน์เตอร์ การระบุค่าของพอยน์เตอร์ และกำหนดตำแหน่งของพอยน์เตอร์ที่ต้องการ รวมถึงการเข้าถึงตำแหน่งของตัวแปร address
- การระบุตำแหน่ง การเก็บข้อมูล และอ่านข้อมูลของตำแหน่งอาเรย์

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการสร้างตัวแปรพอยน์เตอร์ กระบวนการเข้าถึงตำแหน่งของพอยน์เตอร์
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 11

พอยน์เตอร์

Pointer

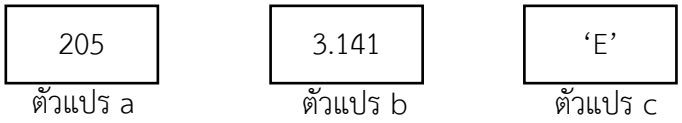
ในการศึกษาพื้นฐานในบทเรียนที่ผ่านมา สามารถใช้ตัวแปรชนิดต่างๆ มาใช้งานให้ถูกต้องและเหมาะสมกับข้อมูล เพื่อเข้าสู่การเขียนโปรแกรมที่มีความซับซ้อนเพิ่มขึ้น ตัวแปรชนิดพอยน์เตอร์เป็นการใช้งานที่มีรูปแบบชัดเจนและเข้าถึงตัวแปรโดยตรง หรือเรียกว่าข้อมูลในรีจีสเตอร์ ของตัวแปรนั้นๆ ทั้งนี้การใช้ตัวแปรชนิดพอยน์เตอร์อาจจะมีการใช้งานลดลง เนื่องจากภาษาคอมพิวเตอร์มีการพัฒนาอย่างต่อเนื่อง เพื่อผู้ใช้งานง่ายขึ้น และเป็นภาษาระดับสูงทั้งหมด เช่น c#, Visual Basic, Java เป็นต้น ซึ่งการเขียนโปรแกรมที่ไม่ต้องการความซับซ้อน โปรแกรมพื้นฐาน จึงไม่จำเป็นต้องใช้ตัวแปรชนิดพอยน์เตอร์ แต่การใช้ตัวแปรชนิดพอยน์เตอร์ยังมีข้อดีคือสามารถเข้าถึงตำแหน่งของตัวแปรหรือแอดเดรส (Address) ของตัวแปร เพื่อจัดการเกี่ยวกับข้อมูลได้โดยตรง ดังนั้นการใช้รูปแบบของพอยน์เตอร์จึงเหมือนกับตัวแปรที่ทำงานแบบอยู่เบื้องหลัง เพราะจะใช้พอยน์เตอร์ในการเข้าถึงตำแหน่งที่อยู่ของตัวแปรที่ต้องการ

การเขียนโปรแกรมที่จะใช้ตัวแปรพอยน์เตอร์ สามารถจัดการข้อมูลที่มีความซับซ้อน เช่น โครงสร้างข้อมูลพวก Stack, Queue หรือ Linked list เป็นต้น ซึ่งควรศึกษาถึงหลักการของพอยน์เตอร์เอาไว้ เพื่อให้เข้าใจแนวคิดและการทำงานพื้นฐานของพอยน์เตอร์

โดยตัวแปรทั่วๆ ไป จะใช้สำหรับเก็บข้อมูล การทำงานจะเป็นดังตัวอย่างต่อไปนี้

```
int a = 205 ;
float b = 3.141 ;
char c = ‘E’ ;
```

โดยตัวแปร a , b และ c ต่างก็เก็บข้อมูลเอาไว้ โดยเมื่อตัวแปรเหล่านี้ถูกประกาศขึ้นมา ก็จะมีการจองพื้นที่ในหน่วยความจำของเครื่องคอมพิวเตอร์เอาไว้สำหรับเก็บข้อมูลของตัวแปร



รูปที่ 11.1 การเก็บค่าข้อมูลของตัวแปร

เมื่อประกาศตัวแปร คอมพิวเตอร์จะจองพื้นที่ในหน่วยความจำเอาไว้สำหรับเก็บค่าของตัวแปร ซึ่งตำแหน่งที่ตัวแปรเหล่านี้อยู่ในหน่วยความจำนั้น เรียกว่า “แอดเดรส” (Address) ยกตัวอย่างเช่น หากสร้างตัว

แปร 10 ตัว ในหน่วยความจำก็แบ่งพื้นที่สำหรับสร้างตัวแปรนี้เอาไว้ให้ 10 ตัว ก็จะมีตำแหน่งเป็นของตนเองในหน่วยความจำทั้ง 10 ตัวแปร

11.1 การสร้างตัวแปรพอยน์เตอร์

ตัวแปรชนิดพอยน์เตอร์มีความแตกต่างจากตัวแปรทั่วไป คือจะเก็บค่าตำแหน่งหรือแอดเดรส ในหน่วยความจำของตัวแปร จะไม่ได้เก็บข้อมูลที่จะนำมาใช้งานโดยตรง โดยการสร้างตัวแปรพอยน์เตอร์ ก็เหมือนกับตัวแปรทั่วไป แต่จะใส่เครื่องหมาย * ไว้หน้าตัวแปร ดังนี้

```
datatype  *name
```

datatype คือการกำหนดชนิดของตัวแปร เช่น char int float double

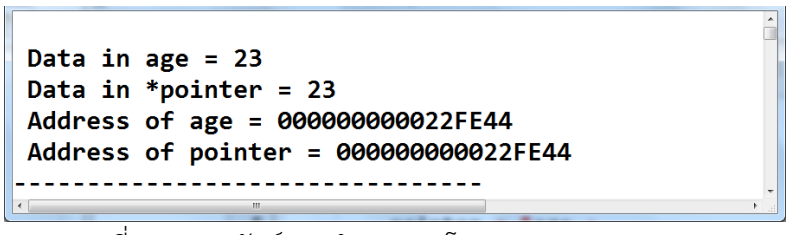
*name คือ เครื่องหมายดอกจันตามด้วยชื่อตัวแปรที่ต้องการสร้าง ตามข้อกำหนดของการประกาศตัวแปร

ตัวอย่างเช่น กำหนดตัวแปร age มีค่า 23 ถ้าให้ตัวแปรพอยน์เตอร์ชี้ไปที่ตัวแปร age ตัวแปรพอยน์เตอร์นั้นจะไม่ได้เก็บค่า 23 แต่จะเก็บค่าตำแหน่งในหน่วยความจำของตัวแปร age แทน ตัวแปรพอยน์เตอร์จะไม่สนใจเลยว่าตัวแปร age เก็บค่าอะไรไว้ แสดงดังตัวอย่างที่ 11.1

ตัวอย่างโปรแกรม 11.1

```
1.  #include<stdio.h>
2.  main ( )
3.  {       int age = 23 ;
4.          int *pointer ;
5.          pointer = &age ;
6.          printf (“ \n Data in age = %d ”, age) ;
7.          printf (“ \n Data in *pointer = %d ”, *pointer) ;
8.          printf (“ \n Address of age = %p ”, &age) ;
9.          printf (“ \n Address of pointer = %p ”, pointer) ;
10. }
```

ผลลัพธ์ของโปรแกรม 11.1



รูปที่ 11.2 ผลลัพธ์การทำงานของโปรแกรม 11.1

ตัวแปรชนิด pointer จะเก็บค่าแอดเดรสของตัวแปร age คือ 22FE44 (ฐานสิบหก) โดย สามารถใช้ค่าแอดเดรสนี้เพื่อเข้าถึงข้อมูลในตัวแปร age หรือจะใช้แสดงออกมาเพื่อตรวจสอบการทำงานของโปรแกรม แต่ในการเขียนโปรแกรมทั่วๆ ไปนั้น จะไม่เห็นการแสดงค่าของตัวแปรพอยน์เตอร์ออกมาทางจอภาพเพื่อสะดวกต่อการทำงานของโปรแกรม

11.2 การใช้งานพอยน์เตอร์

ตัวแปรพอยน์เตอร์จะใช้ในกรณีที่ต้องการเข้าถึงข้อมูลที่อยู่ในตัวแปรตัวที่เฉพาะเจาะจง การสร้างตัวแปรพอยน์เตอร์จะกำหนดด้วยเครื่องหมาย * นำหน้าชื่อตัวแปร เช่น int *test และกำหนดการชี้ตำแหน่งให้กับตัวพอยน์เตอร์ด้วยเครื่องหมาย & เช่น test = &ตัวแปรที่ต้องการ เมื่อกำหนดเสร็จสิ้น ตัวแปรพอยน์เตอร์จะสามารถเข้าไปเปลี่ยนแปลงค่าได้โดยที่ไม่ต้องเรียกใช้ตัวแปรตัวนั้นตรงๆ โดยดูตัวอย่างต่อไปนี้

ตัวอย่างโปรแกรมที่ 11.2

```
1.  #include<stdio.h>
2.  main ( )
3.  {       int age ;
4.          printf ( " How old are you ? " );
5.          scanf ( "%d", &age) ;
6.          printf(" You are %d year old. \n ", age);
7.  }
```

จากโปรแกรมที่ 11.2 โปรแกรมจะรับค่าอายุจากผู้ใช้ทางคีย์บอร์ด และแสดงค่าอายุโดยมีตัวแปร age เก็บค่าอายุเอาไว้และแสดงผลทางหน้าจอภาพ

ในกรณีนี้ ถ้าต้องการเปลี่ยนค่าที่อยู่ในตัวแปร age โดยที่ไม่ต้องเรียกใช้ตัวแปร age สามารถทำได้โดยใช้ตัวแปรพอยน์เตอร์ ดังนี้

ตัวอย่างโปรแกรมที่ 11.2 (ต่อ)

```
1.  #include<stdio.h>
2.  main ( )
3.  {           int age ;
4.              int *pointer ;
5.              printf ( " How old are you ? " ) ;
6.              scanf ( "%d", &age) ;
7.              pointer = &age ;
8.              *pointer = 50;
9.              printf(" You are %d year old. \n ", age) ;
10. }
```

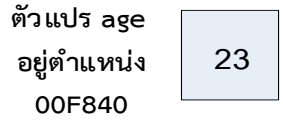
ตัวแปร *pointer เป็นพอยน์เตอร์สำหรับชี้ตำแหน่งไปยังตำแหน่งของตัวแปร age และเมื่อรับค่าจากผู้ใช้แล้ว ด้วยโปรแกรมระบุให้ตัวแปร pointer เก็บแอดเดรสของ (หรือชี้ตำแหน่งไปยัง) ตัวแปร age (คำสั่ง pointer = &age) ตอนนี้ตัวแปร pointer ก็จะชี้ตำแหน่งไปยังตัวแปร age จากนั้นสำหรับบรรทัดที่ 8 คือ

```
*pointer = 50 ;
```

บรรทัดนี้เป็นการเปลี่ยนแปลงข้อมูลในตำแหน่งที่ตัวแปร pointer ชี้ตำแหน่งอยู่ โดยกำหนดค่า 50 ให้กับตัวแปร ซึ่งสามารถอธิบายได้ว่าเมื่อโปรแกรมรับค่าจากผู้ใช้มาเก็บไว้ในตัวแปร age (ไม่ว่าจะเป็นค่าใดก็ตาม) เมื่อถึงบรรทัดนี้ตัวแปร age จะเปลี่ยนค่าเป็น 50 ทันที โดยที่ไม่ได้อ้างถึงชื่อตัวแปร age ทั้งนี้ตัวแปร age จะมีค่าเท่ากับ 50 และแสดงผลลัพธ์ทางหน้าจอ

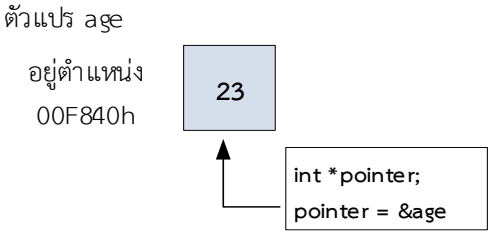
```
You are 50 year old.
```

แต่การใช้พอยน์เตอร์จะเป็นการเข้าไปเปลี่ยนค่าแบบทางอ้อม โดยไม่ต้องผ่านตัวแปร age การทำงานของพอยน์เตอร์จึงช่วยให้สามารถเปลี่ยนแปลงค่าหรือควบคุมข้อมูลที่อยู่ในตัวแปรที่ต้องการได้ เพื่อความเข้าใจให้ลองพิจารณารูปที่ 11.3 ต่อไปนี้



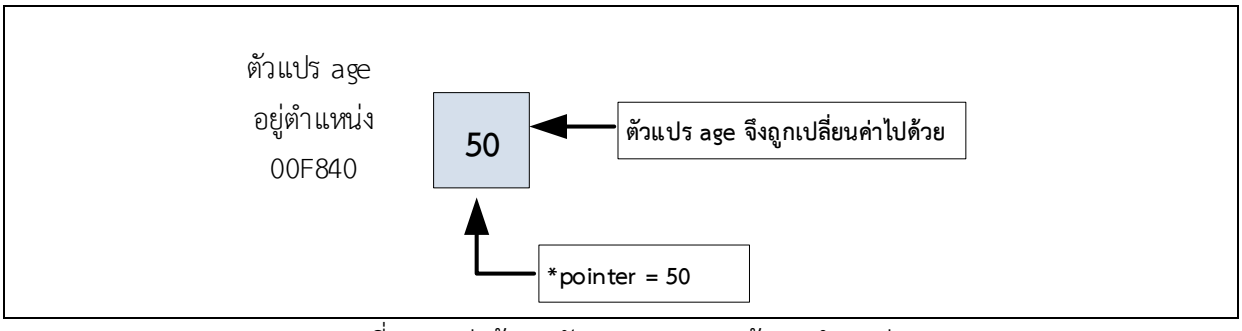
รูปที่ 11.3 ค่าข้อมูลตัวแปร age และข้อมูลตำแหน่ง

ในขั้นแรกกำหนดให้ตัวแปร age มีค่าเท่ากับ 23 อยู่ตำแหน่ง 00F840h ก็คือแอดเดรสในหน่วยความจำของตัวแปร age จากนั้นให้สร้างตัวแปรพอยน์เตอร์ และกำหนดให้ตัวแปรพอยน์เตอร์ ชี้ตำแหน่งไปยังตัวแปร age แสดงดังรูปที่ 11.4



รูปที่ 11.4 ค่าข้อมูลตัวแปร age และข้อมูลตำแหน่ง

ในขั้นตอนนี้ตัวแปรพอยน์เตอร์ จะชี้ตำแหน่งไปที่ตัวแปร age โดยเก็บแอดเดรสของตัวแปร age **(ในที่นี้คือ 00F840h)** เอาไว้ ดังนั้น หากกำหนดค่า *pointer = 50 เพื่อเปลี่ยนค่าที่อยู่ในตำแหน่ง 00F840h นี้ก็จะได้ ดังรูปที่ 11.3



รูปที่ 11.5 ค่าข้อมูลตัวแปร age และข้อมูลตำแหน่ง

ผลลัพธ์ที่เกิดขึ้นก็คือข้อมูลที่อยู่ในตำแหน่ง 00F840h จะถูกเปลี่ยนค่าไปเป็น 50 ซึ่งก็คือตัวแปร age โดยที่ไม่ได้เรียกใช้ตัวแปร age ในการเปลี่ยนแปลงค่า เมื่อโปรแกรมให้แสดงค่าผลลัพธ์ของตัวแปร age ก็จะแสดงค่า 50 ออกมาทางหน้าจอคอมพิวเตอร์

11.3 การใช้ตัวดำเนินการทางคณิตศาสตร์กับพอยน์เตอร์ (Pointer Arithmetic)

เนื่องจากตัวแปรพอยน์เตอร์ใช้ในการเก็บแอดเดรสของตัวแปรตัวอื่น ดังนั้นสามารถใช้ตัวดำเนินการทางคณิตศาสตร์ในการแก้ไขค่าของพอยน์เตอร์ได้ แต่ใช้ได้เฉพาะตัวดำเนินการบางตัวเท่านั้น คือ ตัวดำเนินการเพิ่มค่า (++) ตัวดำเนินการลดค่า (--) การเพิ่มค่าหรือลดค่าพอยน์เตอร์ด้วยจำนวนเต็มโดยใช้เครื่องหมาย +,-,+=,-=

```
1. int  a = 100 ;            // byte address  0x073C
2. int  *pa ;
3. pa = &a  ;
4. ( *pa )++ ;
5. pa++ ;
```

ตัวแปร a มีค่าเท่ากับ 100 อยู่ตำแหน่งที่ 073C และตัวแปรพอยน์เตอร์ pa เก็บตำแหน่งของตัวแปร a คือ 001FH

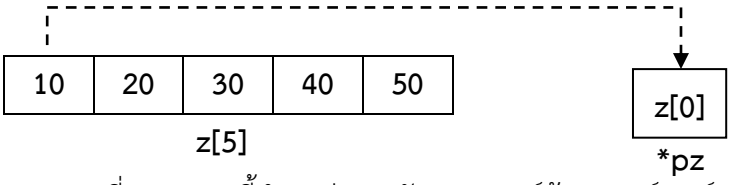
คำสั่ง (*pa)++ หมายถึงการนำค่าพอยน์เตอร์ที่ชี้ตำแหน่ง (*pa) บวกหนึ่ง ผลลัพธ์ที่ได้เท่ากับ 101

คำสั่ง pa++ จะมีค่าเท่ากับ pa = pa + 1 คือนำตำแหน่งเพิ่มขึ้นหนึ่ง โดยตัวแปรพอยน์เตอร์ pa เก็บตำแหน่ง 073C และทำการเพิ่มไปหนึ่งตำแหน่ง จะมีค่าเท่ากับ 0740 เนื่องจาก a เป็นตัวแปรชนิด int มีขนาด 4 ไบต์ (byte) ถ้าเป็นตัวแปรชนิด float จะมีค่าเท่ากับ 0023H เนื่องจากมีขนาด 4 ไบต์

11.4 พอยน์เตอร์กับอาร์เรย์

ตัวแปรพอยน์เตอร์กับตัวแปรอาร์เรย์มีคุณสมบัติเฉพาะการเรียกใช้ตัวแปรอาร์เรย์ นั้นคือ เขียนตัวแปรอาร์เรย์โดยไม่ต้องระบุเครื่องหมาย * นำหน้าตัวแปร โดยทำการระบุโดยตรง ดังตัวอย่างต่อไปนี้

```
int  z[5]  = { 10 , 20 , 30 , 40 , 50 } ;
int   *pz  ;
pz = z ;
```



รูปที่ 11.6 การชี้ตำแหน่งของตัวแปรอาเรย์ด้วยพอยน์เตอร์

ตัวแปรอาร์เรย์ z[5] มีการระบุค่าตัวแปรตั้งแต่ตำแหน่งของอาร์เรย์ z[0] z[1] , z[2] , z[3] และ z[4] มีจำนวนทั้งหมด 5 จำนวน และทำการประกาศตัวแปรพอยน์เตอร์ *pz ในการกำหนดตำแหน่งให้กับตัวแปรพอยน์เตอร์ โดยใช้คำสั่ง pz = z ซึ่งมีความหมายเดียวกับ pz = &z[0] ดังนั้นตัวแปรพอยน์เตอร์สามารถอ้างถึงสมาชิกของตัวแปรอาร์เรย์ได้

ตัวอย่างโปรแกรมที่ 11.3 การใช้พอยน์เตอร์กับตัวแปรอาร์เรย์

```
1.  #include <stdio.h>
2.  void main( )
3.  {       int boy[6] = { 1, 3, 5, 7, 9, 11 } ;
4.          int *pt  ;
5.          pt = boy ;
6.          boy[0] = *pt ;
7.          boy[2] = *( pt + 2 ) ;
8.          boy[4] = *pt + 2 ;
9.          *pt = boy[0] + boy[2] + boy[4] ;
10.         printf( "Data boy[0] = %d ,boy[2] = %d ,boy[4] = %d \n", boy[0], boy[2], boy[4]
    ) ;
11. }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 11.3

Data boy[0] = 9 , boy[2] = 5 , boy[4] = 3

11.5 อาร์เรย์ของพอยน์เตอร์

การประกาศตัวแปรพอยน์เตอร์ โดยทั่วไปสามารถระบุตำแหน่งได้เพิ่มหนึ่งตำแหน่งเท่านั้น นั้นอาจไม่เพียงพอสำหรับการใช้งาน ฉะนั้นการสร้างอาร์เรย์พอยน์เตอร์นั้น จึงได้นำมาใช้งาน การกำหนดพอยน์เตอร์เป็นอาร์เรย์สามารถนำไปใช้งานในการเก็บตำแหน่งของข้อมูลหลายตำแหน่งในตัวพอยน์เตอร์ ซึ่งระบุการทำงานได้ดังตัวอย่างต่อไปนี้

รูปแบบการสร้าง

ชนิดข้อมูล *** ชื่อพอยน์เตอร์ [** ขนาดของข้อมูล **]**

ตัวอย่างเช่น

int *a[5] ; float *boy[10] ; char * test [6] ;

ตัวอย่างการใช้อาร์เรย์ของพอยน์เตอร์

```
int  a[5] = {1, 3, 5, 7, 9 } ;
int *pa[5] ;
pa[0] = &a[0] ;
pa[1] = &a[1] ;
pa[2] = &a[2] ;
pa[3] = &a[3] ;
pa[4] = &a[4] ;
```

a[5] =

a[0]	a[1]	a[2]	a[3]	a[4]
1	**3**	**5**	**7**	**9**
01FFH	0201H	0203H	0205H	0207H

*pa[5] =

pa[0]	pa[1]	pa[2]	pa[3]	pa[4]
01FF	**0201**	**0203**	**0205**	**0207**
00EEH	00F0H	00F2H	00F4H	00F6H

ตัวอย่างโปรแกรมที่ 11.4 อาร์เรย์พอน์เตอร์ (Array-Pointer)

```
1.  #include <stdio.h>
2.  void main( )
3.  {       int  a[5] = { 1, 3, 5, 7, 9 } ;
4.          int  b[5] ;
5.          int  *pa[5] ;
6.                  pa[0] = a ;
7.                  pa[1] = a + 1 ;
8.                  pa[2] = a + 2 ;
9.                  pa[3] = a + 3 ;
10.                 pa[4] = a + 4
11.                         b[0] = *pa[4] ;
12.                         b[1] = *pa[3] ;
```

```
13.                     b[2] = *pa[2] ;
14.                     b[3] = *pa[1] ;
15.                     b[4] = *pa[0] ;
16.             printf( " Data b[0]=%d b[1]=%d b[2]=%d b[3]=%d b[4]=%d \n ", b[0],
    b[1], b[2] , b[3], b[4] ) ;
17.  }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 11.4

Data b[0] = 9 , b[1] = 7 , b[2] = 5 b[3] = 3 , b[4] = 1

ตัวอย่างโปรแกรมที่ 11.4 โจทย์กำหนดให้ ตัวแปร a มีข้อมูลเป็นเมตริกขนาด 1x5 และตัวแปร b มีข้อมูลเป็นเมตริกขนาด 1x5 ซึ่งมีชุดข้อมูลที่กำหนดให้ จงเขียนโปรแกรมหาผลลัพธ์การบวกของเมตริกทั้งสอง โดยใช้ตัวแปรพอยน์เตอร์ (Pointer) ในการคำนวณข้อมูล

กำหนดให้

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} 10 & 20 & 30 & 40 & 50 \end{bmatrix}$$

```
1.  #include <stdio.h>
2.  void main( )
3.  {       int a[5] = {1, 2, 3, 4, 5 };        int b[5] = { 10, 20, 30, 40, 50 } ;
4.          int c[5] ;      int  *pa[5] , *pb[5]  , *pc[5] ;          int i , j , k ;
5.          for( i = 0 ; i<=4 ; i++)
6.          {       pa[ i ] = &a[ i ] ;
7.                  pb[ i ] = &b[ i ] ;
8.                  pc[ i ] = &c[ i ] ;
9.          }
10.         for( j =0 ; j<=4 ; j++)
11.         {       *pc[ j ] = *pa[ j ] + *pb[ j ] ;
12.                 printf( "C[%d] = %d \n " , j, *pc[ j ] ) ;
13.         }
14.  }
```

ผลลัพธ์การทำงานของตัวอย่างโปรแกรมที่ 11.4

C[5] =

11	22	33	44	55

สรุปท้ายบท

ตัวแปรพอยน์เตอร์ คือการระบุตำแหน่งหรือชี้ตำแหน่ง โดยผ่านตัวแปรที่เป็นพอยน์เตอร์ ซึ่งตัวแปรแบบพอยน์เตอร์จะไม่เก็บค่าข้อมูลหรือผลลัพธ์การคำนวณทางคณิตศาสตร์ โดยจุดเด่นตัวแปรชนิดนี้จะเข้าถึงข้อมูลในระดับหน่วยความจำของโดยตรง ซึ่งสามารถนำไปประยุกต์ใช้งานในการจองพื้นที่ในหน่วยความจำเพื่อนำข้อมูลผ่านตำแหน่งของหน่วยความจำนั้นๆ ตัวแปรพอยน์เตอร์จะต้องมีการระบุตำแหน่งหรือชี้ตำแหน่งก่อนการใช้งานเสนอ หากไม่มีการระบุตำแหน่งให้กับตัวพอยน์เตอร์ จะทำตำแหน่งที่ชี้ข้อมูลไม่ถูกต้องนั้นเอง

คำถามท้ายบท

1. ข้อดีของตัวแปรพอยน์เตอร์คืออะไร?
2. สัญลักษณ์ใดที่ใช้ระบุว่าเป็นตัวแปรพอยน์เตอร์ คืออะไร?
3. สัญลักษณ์ที่ใช้ระบุตำแหน่งให้กับตัวแปรพอยน์เตอร์ คืออะไร?
4. ตัวแปรพอยน์เตอร์ ถูกออกแบบให้เก็บข้อมูลประเภทใด?
5. หากสร้างตัวแปร int *a[10] ตัวแปร a สามารถเก็บข้อมูลได้กี่จำนวน อะไรบ้าง?
6. จากตัวแปร int a = 10, *z ; กำหนดให้ z = &a ; ตัวแปร z ทำหน้าที่อะไร?
7. เมื่อ int z = 24 // address 017Fh และ int *pz =&z จงหา (*pz)++ มีค่าเท่าใด?
8. เมื่อ int z = 24 // address 017Fh และ int *pz =&z จงหา (pz)++ มีค่าเท่าใด?
9. ตัวแปรชนิดจำนวนจริง float มีขนาดกี่ไบต์ ?
10. ตัวแปรชนิดจำนวนจริง int มีขนาดกี่ไบต์ ?
11. ตัวแปรชนิดจำนวนจริง unsigned char มีขนาดกี่ไบต์ ?
12. หากตัวแปรพอยน์เตอร์ไม่ได้ชี้ตำแหน่งหรือระบุตำแหน่ง ตัวแปรพอยน์เตอร์ใช้งานได้หรือไม่?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมและผังงาน สำหรับการตรวจสอบหา **จำนวนเลขคู่** ของอาเรย์สองมิติ มีกี่จำนวนและมีค่าอะไรบ้าง โดยกำหนดให้อาเรย์ดังต่อไปนี้

   ข้อกำหนด **ใช้ตัวแปรพอยน์เตอร์ในการคำนวณทั้งหมด**

$$A[3][3] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2. จงเขียนโฟลว์ชาร์ตและโปรแกรม ทำการคำนวณหา **ผลบวก** ระหว่างของเมตริก A และ B และหาผลลัพธ์ผลบวกของเมตริก **(ใช้ตัวแปรแบบ pointer เท่านั้นในการคำนวณ)**

   กำหนดให้

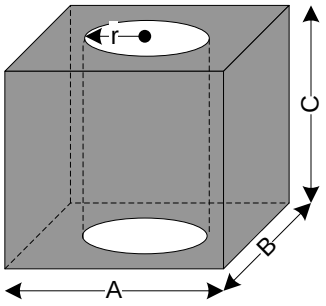
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -5 & 4 \\ 7 & 6 & 9 \\ -2 & 8 & 0 \end{bmatrix}$$

3. จงเขียนโฟลว์ชาร์ตและโปรแกรม ทำการคำนวณหา **ผลบวก** ระหว่างของเมตริก A และ B และหาผลลัพธ์ผลบวกของเมตริก **(ใช้ตัวแปรแบบ pointer เท่านั้นในการคำนวณ)**

   กำหนดให้

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -5 & 4 \\ 7 & 6 & 9 \\ -2 & 8 & 0 \end{bmatrix}$$

4. จงเขียนโปรแกรมและผังการทำงาน สำหรับการคำนวณหาค่าปริมาตรของรูปด้านล่าง **<u>โดยกำหนดให้ใช้ตัวแปรแบบ pointer ในการคำนวณเท่านั้น</u>**



โดยกำหนดให้ ปริมาตรทรงกระบอก = $\pi r^2 h$

ปริมาตรทรงสี่เหลี่ยม = กว้าง x ยาว x สูง

5. จงเขียนผังงานและหาผลลัพธ์ของโปรแกรมที่กำหนดให้

```
1.  #include <stdio.h>
2.  main( )
3.  {       int a[2][2] = { 1, 2, 3, 4 } ;      int b[2][2] = { 5, 6, 7, 8 } ;
4.          int *c[2][2] , *d[2][2] , *e[2][2] ,z[2][2], i , j ;
5.          for( i=0 ; i<=1 ; i++ )
6.          {       for( j=0 ; j<=1 ; j++ )
7.                  {       c[ i ][ j ] = &a[ i ][ j ] ;
8.                          d[ i ][ j ] = &b[ i ][ j ] ;
9.                          e[ i ][ j ] = &z[ i ][ j ] ;
10.                         *e[ i ][ j ] = (*c[ i ][ j ] ) * (*d[ i ][ j ] ) ;
11.                 }
12.         }
13.         for( i=0 ; i<=1 ; i++ )
14.         {       for( j=0 ; j<=1 ; j++ )
15.                 {       printf( " e[ %d][%d] =%d \n",  i , j , z[ i ][ j ] ) ;
16.                 }
17.         }
18. }
```

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 12 ฟังก์ชั่น (Function)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจกระบวนการสร้างฟังก์ชั่น และการใช้งานฟังก์ชั่นที่สร้างขึ้น
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและโปรแกรมฟังก์ชั่น

เนื้อหา

- การสร้างฟังก์ชั่นที่ไม่มีอินพุตและไม่มีเอาต์พุต การสร้างฟังก์ชั่นที่มีอินพุต และการสร้างฟังก์ชั่นที่มีเอาต์พุตออกจากฟังก์ชั่น
- การใช้งานรวมกับฟังก์ชั่นหลัก และการเรียกใช้งานฟังก์ชั่นย่อยหลายๆ ฟังก์ชั่น

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นหลักการสร้างฟังก์ชั่น กระบวนการเรียกใช้งานฟังก์ชั่น
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 12

ฟังก์ชั่น
Function

ฟังก์ชั่น (Function) หมายถึง ฟังก์ชั่นใดๆ ที่สามารถเรียกใช้งานได้ไม่จำกัด โดยเรียกใช้งานตามรูปแบบของฟังก์ชั่น การเรียกใช้ฟังก์ชั่นใดๆ ในโปรแกรมภาษาซี จะต้องนำเข้าไลบรารี่ ในส่วนหัวของโปรแกรม (Header File) เพื่อเรียกใช้งานฟังก์ชั่นใดๆภายใต้ไลบรารี่นั้นๆ เช่น ฟังก์ชั่นรากที่สอง sqrt() จะต้องเพิ่มในส่วน #include <math.h> เป็นต้น โดยทั่วไปฟังก์ชั่นในภาษาซี จะเป็นฟังก์ชั่นมาตรฐานที่มีใช้งานเฉพาะเจาะจง เช่น ฟังก์ชั่นทางคณิตศาสตร์ ฟังก์ชั่นเวลา ฟังก์ชั่นจัดการไฟล์ เป็นต้น

โดยในเนื้อหาเรื่องฟังก์ชั่น มุ่งเน้นให้สามารถสร้างฟังก์ชั่น ได้โดยเป็นฟังก์ชั่นที่ไม่ซ้ำซ้อนกับฟังก์ชั่นมาตรฐานของภาษาซี ฟังก์ชั่นที่สร้างขึ้นใหม่ขึ้นมาเองตามข้อกำหนดซึ่งสามารถเรียกใช้ฟังก์ชั่นมาตรฐานได้ และสามารถเรียกใช้งานได้อย่างถูกต้องตามรูปแบบของฟังก์ชั่น ด้วยภาษาซี สามารถแบ่งได้เป็น 2 ประเภทก็คือ

1. ฟังก์ชั่นมาตรฐาน หรือ ไลบรารีฟังก์ชัน (Library function)
2. ฟังก์ชั่นที่สร้างขึ้นเอง (User-defined function)

12.1 ฟังก์ชั่นมาตรฐาน (Standard function)

ฟังก์ชั่นมาตรฐาน เป็นฟังก์ชั่นที่มีการพัฒนามาพร้อมกับโปรแกรมภาษาซี เป็นฟังก์ชั่นที่มีรูปแบบการทำงานอย่างมีรูปแบบ เป็นลำดับขั้นตอน มีการนำข้อมูลอินพุต และข้อมูลเอาต์พุต หรือ การดำเนินการใดๆ ที่ได้ถูกกำหนดตามรูปแบบที่ได้กำหนดไว้แล้ว ยกตัวอย่างเช่น ฟังก์ชั่นทางคณิตศาสตร์ สามารถเรียกไลบรารี่ <math.h> เช่น sqrt คือฟังก์ชั่นการหารากที่สอง, pow คือฟังก์ชั่นเลขยกกำลัง, abs คือฟังก์ชั่นค่าสมบูรณ์ เป็นต้น

ฟังก์ชั่นมาตรฐานของภาษาซี ได้ถูกสร้างไว้เรียบร้อยแล้วซึ่งสามารถเรียกใช้ได้ทันที โดยในการเรียกใช้งานนั้น จะต้องทำการ include ไฟล์นามสกุล .h เข้ามาด้วย เช่น การเรียกใช้ฟังก์ชั่น printf เป็นคำสั่งที่ใช้แสดงข้อความออกทางจอภาพ หรือ scanf เป็นคำสั่งที่ใช้รับค่าทางคีย์บรอด์ จะต้อง include ไฟล์ชื่อ stdio.h เป็นต้น โดยคำสั่งในภาษาซี เช่น if, else, while, do..while ฯลฯ คือคำสั่งที่สามารถเรียกใช้ได้เลยโดยไม่ต้องทำการ include ไฟล์ใด ๆ เพราะคำสั่งเหล่านี้เป็นไวยากรณ์ของตัวภาษาซี (Syntax of C Language) ในการเรียกใช้งานไลบรารีฟังก์ชั่น จะต้องเขียนคำสั่ง #include <header file> ไว้ส่วนหัวของโปรแกรม เพื่อให้โปรแกรมภาษาซี เรียกใช้งานฟังก์ชั่นต่างๆ ที่อยู่ภายในไลบรารีที่เรียกใช้งานในโปรแกรม และจะต้องเรียกใช้งานให้ถูกต้องตามรูปแบบของฟังก์ชั่น ดังแสดงตัวอย่างต่อไปนี้

```
1.    #include <stdio.h>                    //นำเข้าไลบรารี่สำหรับอินพุตและเอาต์พุต
2.    #include <math.h>                     //นำเข้าไลบรารี่ทางคณิตศาสตร์
3.    float   data1, data2, value = 9.0 ;   //ประกาศตัวแปร
4.    data1 = sqrt( value ) ;               //คำนวณรากที่สองของตัวแปร value
5.    data2 = pow( value, 2) ;              //คำนวณเลขยกกำลัง 2 ของตัวแปร value
6.    printf("Data1=%f Data2=%f \n",data1,data2);  // แสดงผลลัพธ์ข้อมูล
```

ไลบรารี <stdio.h> ซึ่งเป็น header ของฟังก์ชั่นที่เกี่ยวข้องกับอินพุตละเอาต์พุต

ไลบรารี <math.h> ซึ่งเป็น header ของฟังก์ชั่นที่เกี่ยวข้องกับการคำนวณทางคณิตศาสตร์

12.2 การสร้างฟังก์ชั่นใหม่ (User-defined function)

ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมภาษาซี สามารถสร้างขึ้นใหม่ด้วยตนเอง ที่นอกเหนือจากฟังก์ชั่นมาตรฐาน ซึ่งการสร้างฟังก์ชั่นของภาษาซีขึ้นมาใหม่ เพื่ออำนวยความสะดวกในการเขียนโปรแกรมแกรมให้เป็นสัดส่วน การประมวลผลข้อมูลที่ต้องทำซ้ำๆ รวมถึงการสร้างฟังก์ชั่นเพื่อทำงานเฉพาะเจาะจง เช่น ฟังก์ชั่นคำนวณหาพื้นที่และปริมาณรูปทรงทางคณิตศาสตร์ ฟังก์ชั่นคำนวณแก้ปัญหาทางด้านวิศวกรรม เป็นต้น โครงสร้างการสร้างฟังก์ชั่นที่ประกอบด้วย ส่วนของอินพุตฟังก์ชั่น ชื่อฟังก์ชั่น และเอาต์พุตฟังก์ชั่น โดยฟังก์ชั่นจะรูปแบบของโครงสร้างฟังก์ชั่นมีดังนี้

```
Return-type   Function-name (Datatype Arg1, Datatype Arg2, ... , Datatype Argn)
 {
      Datatype variable     // ประกาศตัวแปรที่ใช้งานภายในฟังก์ชั่นเท่านั้น
      Statement_1 ;
      Statement_2 ;
            ...
      Statement_n ;
      return (value or variable) ;
 }
```

คำอธิบาย

Return-type คือชนิดของฟังก์ชั่นในส่วนเอาต์พุต โดยฟังก์ชั่นสามารถกำหนดรูปแบบของเอาต์พุตว่าต้องการชนิดใดเช่น int , float, char ,double เป็นต้น ซึ่งถ้าฟังก์ชั่นไม่มีเอาต์พุตสามารถกำหนดเป็น void (ไม่กำหนดเอาต์พุต)

Function-name คือชื่อฟังก์ชั่น การตั้งชื่อสามารถกำหนดได้จากรูปแบบของการตั้งชื่อของภาษาซี

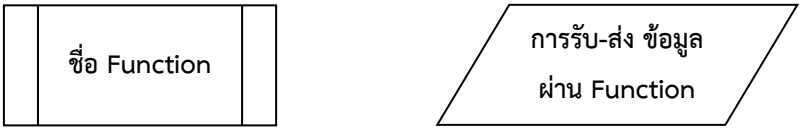
Datatype arg1, ... คือเป็นการกำหนดค่าชนิดของอินพุตที่ส่งเข้ามาให้กับฟังก์ชั่น ซึ่งต้องพิจารณาค่าของอินพุตของฟังก์ชั่นว่ามีจำนวนเท่าไรและชนิดของข้อมูล ถ้าฟังก์ชั่นไม่มีอินพุตสามารถกำหนดค่าเป็น (void) หรือใส่วงเล็บ ( ) ไม่ระบุค่าใดๆ

type variable คือการสร้างตัวแปรที่ใช้งานภายในฟังก์ชั่น สามารถประกาศตัวแปรได้ตามเงื่อนไขของภาษาซี

Statement คือกลุ่มคำสั่งที่สร้างขึ้นหรือคำสั่งพื้นฐาน โดยใช้งานภายในฟังก์ชั่นเท่านั้น

return(value) คือการส่งผลลัพธ์การทำงานของฟังก์ชั่น ไปยังตัวแปรเอาต์พุตฟังก์ชั่นที่เรียกใช้งาน ผลลัพธ์จะต้องชนิดเดียวกับ type ชนิดของฟังก์ชั่น ส่วนของ value นั่นคือผลลัพธ์ที่ต้องการส่งกลับ ถ้าฟังก์ชั่นไม่มีเอาต์พุต ในส่วนของคำสั่ง return จึงไม่จำเป็นต้องมีในโปรแกรม

โดยมีโฟลว์ชาร์ตที่ใช้งานการเรียกชื่อฟังก์ชั่น และการรับส่งข้อมูลผ่านฟังก์ชั่น



(ก) (ข)

รูปที่ 12.1 สัญลักษณ์ผังงานของฟังก์ชั่นย่อย (ก) และ (ข) สัญลักษณ์การรับส่งข้อมูลฟังก์ชั่น

main( ) เป็นฟังก์ชั่นหลักของโปรแกรมภาษาซี การทำงานของโปรแกรมจะเริ่มต้นการทำงานที่ฟังก์ชั่นหลัก (main function) โดยคำสั่งและฟังก์ชั่นต่างๆ ที่เขียนอยู่ภายในเครื่องหมายปีกกาเปิด **{** ตลอดจนสิ้นสุดปีกกาปิด **}** ของฟังก์ชั่นหลัก โปรแกรมจะทำงานไปตามลำดับบรรทัด ตั้งแต่บรรทัดแรกจนถึงบรรทัดสุดท้าย ถึงแม้ว่าจะคอมไพล์โปรแกรมเป็นไฟล์ .exe แต่การทำงานของโปรแกรมก็จะเริ่มต้นการทำงานจากโค้ดภายใน main( ) เสมอ

โดยประเภทของฟังก์ชั่นย่อยสามารถจำแนกออกเป็น 3 ลักษณะดังนี้

1. ฟังก์ชั่นย่อยที่ไม่มีการรับค่าอินพุตและไม่มีการส่งค่าเอาต์พุต
2. ฟังก์ชั่นย่อยที่มีการรับค่าทางอินพุต เข้าไปในฟังก์ชั่น
3. ฟังก์ชั่นย่อยที่มีการส่งค่าทางเอาต์พุต ออกจากฟังก์ชั่น

การสร้างฟังก์ชั่นย่อยสามารถเขียนรูปแบบได้ดังนี้

```
Output function_name ( input )
```

output เป็นการส่งค่ากลับ เพื่อนำมาใช้งานในฟังก์ชั่นหลัก ซึ่งควรระบุชนิดการส่ง ซึ่งมีหลายรูปแบบ เช่น char, int, float, double อื่นๆ เป็นต้น จะใช้ควบคู่กับคำสั่ง return ในกรณีที่ฟังก์ชั่นไม่มีเอาต์พุต ให้กำหนดเป็น void

function_name คือ ชื่อฟังก์ชั่นที่กำหนดขึ้นมาใหม่ ต้องไม่ซ้ำกับคำสงวนของภาษาซี

input เป็นการรับค่า เพื่อนำมาทำงานในฟังก์ชั่นนั้นๆซึ่งควรประกาศตัวแปรรองรับการรับค่าดังกล่าว สามารถระบุได้หลายรูปแบบเช่น char, int, float, double อื่นๆ เป็นต้น ในกรณีที่ฟังก์ชั่นไม่มีอินพุต ให้กำหนดเป็น void

void หมายถึงเป็นตัวระบุลักษณะที่ไม่มีค่าใดๆ หรือว่างเปล่า

ตัวอย่างการสร้างฟังก์ชั่น

void temp (void) ; หมายถึง ฟังก์ชั่นย่อยชื่อ temp เป็นฟังก์ชั่นไม่มีข้อมูลทางอินพุตและเอาต์พุต

void test (float x) ; หมายถึง ฟังก์ชั่นย่อยชื่อ test เป็นฟังก์ชั่นมีข้อมูลอินพุตเป็นจำนวนจริงเก็บข้อมูลในตัวแปร x และไม่มีการส่งข้อมูลทางเอาต์พุต

int answer(int a, int b, int c) ; หมายถึง ฟังก์ชั่นย่อยชื่อ answer เป็นฟังก์ชั่นมีข้อมูลอินพุตเป็นจำนวนเต็มเก็บข้อมูลในตัวแปร a, b, และ c จำนวน 3 ตัวแปร และมีการส่งข้อมูลทางเอาต์พุตเป็นจำนวนเต็ม

float calculate(int x, float y, char z) ; หมายถึง ฟังก์ชั่นย่อยชื่อ calculate เป็นฟังก์ชั่นมีข้อมูลอินพุตจำนวน 3 ตัวแปร โดยค่าจำนวนเต็มเก็บตัวแปร x, ค่าจำนวนจริงเก็บตัวแปร y, และตัวอักษรเก็บเก็บตัวแปร z และมีการส่งข้อมูลทางเอาต์พุตเป็นจำนวนจริง

12.3 การสร้างฟังก์ชั่นย่อยที่ไม่มีการรับค่าอินพุตและไม่มีการส่งค่าเอาต์พุต

การสร้างฟังก์ชั่นย่อยที่ไม่มีการรับค่าอินพุตและไม่มีการส่งค่าเอาต์พุต ด้วยการกำหนด void ลงไปในส่วนของเอาต์พุตฟังก์ชั่นและอินพุตฟังก์ชั่น โดยกำหนดชื่อฟังก์ชั่นย่อย ตามข้อกำหนดการตั้งชื่อฟังก์ชั่นก็มีกฎเกณฑ์เหมือนกับการตั้งชื่อตัวแปร เมื่อกำหนดรูปแบบดังกล่าว คำสั่งหรือกลุ่มคำสั่งต่างๆ ภายใต้เครื่องหมายปีกกาเปิด **{** สำหรับเริ่มการทำงานของฟังก์ชั่น และเครื่องหมายปีกกาปิด **}** สำหรับสิ้นสุดการทำงานของฟังก์ชั่นย่อย

รูปแบบโปรแกรมสำหรับฟังก์ชั่นย่อยที่ไม่มีการรับค่าอินพุตและไม่มีการส่งค่าเอาต์พุต สามารถเขียนได้ดังนี้

```
(1)      (2)      (3)
void ชื่อฟังก์ชั่นย่อย (void)
{
        คำสั่งหรือกลุ่มคำสั่งต่างๆ ; (4)
}
```

หมายเลข 1 คือส่วนของเอาต์พุตฟังก์ชั่น output โดยกำหนดให้เป็น void

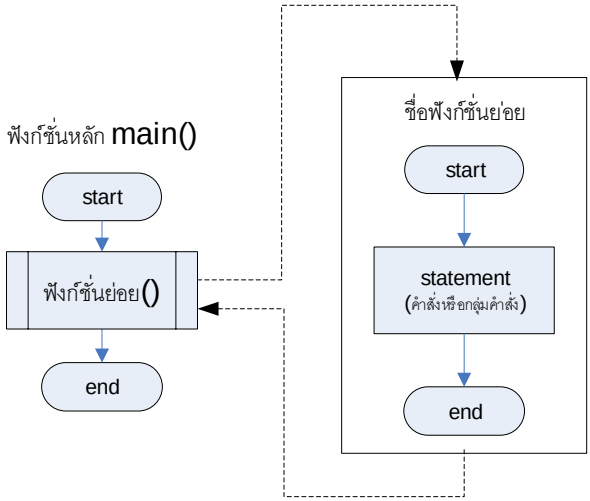
หมายเลข 2 คือกำหนดชื่อฟังก์ชั่นย่อย (โดยไม่ซ้ำกับคำสงวนของภาษาซี)

หมายเลข 3 คือส่วนของอินพุตฟังก์ชั่น input โดยกำหนดให้เป็น void

หมายเลข 4 คือ คำสั่งหรือกลุ่มคำสั่งของฟังก์ชั่นย่อย ภายใต้เครื่องหมายปีกกา { } ในฟังก์ชั่นย่อย

รูปแบบผังงานสำหรับฟังก์ชั่นย่อยที่ไม่มีการรับค่าอินพุตและไม่มีการส่งค่าเอาต์พุต

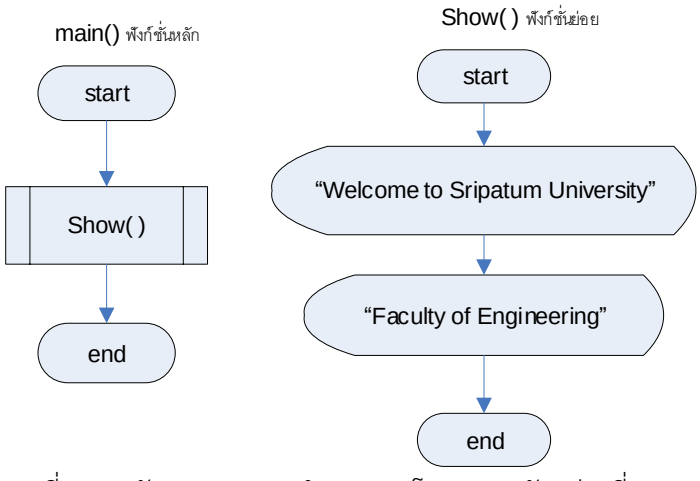
การเขียนผังงานของฟังก์ชั่นย่อย (เขียนเสมือนฟังก์ชั่นหลัก) จะเริ่มต้นด้วยสัญลักษณ์ เริ่มต้นการทำงาน (start) เปรียบเสมือนเครื่องหมายปีกกาเปิดเริ่มต้นการทำงานในโปรแกรม โดยลำดับถัดไปจะเป็นกลุ่มคำสั่ง หรือคำสั่งพื้นฐานของภาษาซี (statement) ในสัญลักษณ์ดำเนินการ หากฟังก์ชั่นสิ้นสุดการทำงาน ด้วยสัญลักษณ์สิ้นสุดการทำงานของฟังก์ชั่น (end) โดยฟังก์ชั่นหลัก (main function) จะทำหน้าที่เรียกใช้งานฟังก์ชั่นย่อย ด้วยสัญลักษณ์ฟังก์ชั่นย่อย โปรแกรมจะไปดำเนินการในส่วนของฟังกชั่นย่อยจนเสร็จสิ้น แล้วกลับมาตำแหน่งเดิมของฟังก์ชั่นหลัก และจบการทำงานของโปรแกรมหลัก แสดงดังรูปที่ 12.2



รูปที่ 12.2 ผังงานฟังก์ชั่นไม่มีอินพุตและเอาต์พุต

<u>ตัวอย่างที่ 12.1</u> การเขียนโปรแกรมฟังก์ชั่นย่อยแสดงข้อความอักษร โดยให้ฟังก์ชั่นย่อยเป็นแบบที่ไม่มีอินพุตและเอาต์พุต ทำการแสดงข้อความ **“Welcome to Sripatum University”** และ **“Faculty of Engineering”** โดยให้ฟังก์ชั่นหลักทำการเรียกใช้งาน

ผังการทำงานตัวอย่างที่ 12.1



รูปที่ 12.3 ผังงานของการทำงานของโปรแกรมตัวอย่างที่ 12.1

โปรแกรมตัวอย่างที่ 12.1

```
1.  #include<stdio.h>
2.  void show( )
3.  {
4.          printf ( " Welcome to Sripatum University \n " ) ;
5.          printf ( " Faculty of Engineering \n "  ) ;
6.  }
7.   void main( )
8.  {
9.          show ( ) ;
10. }
```

ฟังก์ชั่นย่อย

ฟังก์ชั่นหลัก

ผลลัพธ์การทดสอบโปรแกรมตัวอย่างที่ 12.1

```
Welcome to Sripatum University
Faculty of Engineering
```

คำอธิบาย เมื่อโปรแกรมเริ่มต้นการทำงาน คำสั่งที่อยู่ใน main( ) จะถูกเรียกให้ทำงานทีละบรรทัดๆ ตามลำดับ โดยภายใน main( ) มีการเรียกใช้ฟังก์ชั่น show( ) ซึ่งหลังจากทำคำสั่งในฟังก์ชั่น show( ) เรียบร้อยแล้ว โปรแกรมจะกลับมาทำคำสั่งถัดไปในฟังก์ชั่นหลัก main( ) การเรียกใช้ฟังก์ชั่นจึงเหมือนกับการ “กระโดดไปทำงานในฟังก์ชั่นแล้วกลับมาเมื่อเสร็จสิ้น” นั่นเอง

12.4 การสร้างฟังก์ชั่นย่อยที่มีการรับค่าทางอินพุต

การสร้างฟังก์ชั่นย่อยที่มีการรับค่าอินพุต ด้วยการกำหนดตัวแปรอินพุตเข้าไปยังฟังก์ชั่นย่อย โดยการระบุตัวแปรที่กำหนดในวงเล็บส่วนอินพุต ซึ่งตัวแปรของอินพุตสามารถระบุชนิดตัวแปรได้ ในส่วนของเอาต์พุต ฟังก์ชั่นกำหนดเป็น void และการกำหนดชื่อฟังก์ชั่นย่อย ตามข้อกำหนดการตั้งชื่อฟังก์ชั่นก็มีกฎเกณฑ์เหมือนกับการตั้งชื่อตัวแปร เมื่อกำหนดรูปแบบดังกล่าว คำสั่งหรือกลุ่มคำสั่งต่างๆ ภายใต้เครื่องหมายปีกกาเปิด **{** สำหรับเริ่มการทำงานของฟังก์ชั่น และเครื่องหมายปีกกาปิด **}** สำหรับสิ้นสุดการทำงานของฟังก์ชั่นย่อย รูปแบบโปรแกรมสำหรับฟังก์ชั่นย่อย มีการรับค่าอินพุต สามารถเขียนได้ดังนี้

```
(1)      (2)                                  (3)
void ชื่อฟังก์ชั่นย่อย ( datatype var-1, datatype var-2,... ,datatype var-n )
{
        คำสั่งหรือกลุ่มคำสั่งต่างๆ ; (4)
}
```

หมายเลข 1 คือส่วนของเอาต์พุตฟังก์ชั่น output โดยกำหนดให้เป็น void

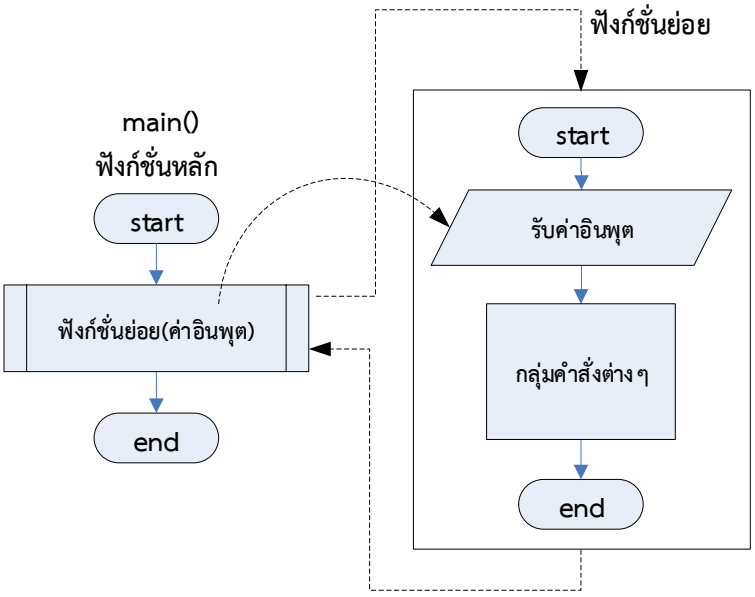
หมายเลข 2 คือกำหนดชื่อฟังก์ชั่นย่อย (โดยไม่ซ้ำกับคำสงวนของภาษาซี)

หมายเลข 3 คือส่วนของอินพุตฟังก์ชั่น input โดยกำหนดชนิดข้อมูล (datatype) พร้อมประกาศชื่อตัวแปร ที่ใช้ในฟังก์ชั่นย่อย

หมายเลข 4 คือ คำสั่งหรือกลุ่มคำสั่งของฟังก์ชั่นย่อย ภายใต้เครื่องหมายปีกกา { } ในฟังก์ชั่นย่อย

รูปแบบผังงานสำหรับฟังก์ชั่นย่อยมีการรับค่าอินพุต

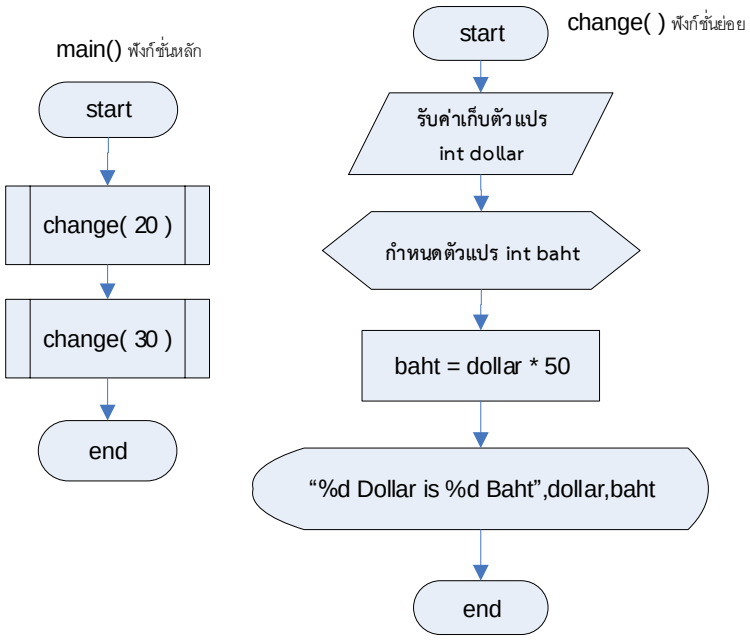
การเขียนผังงานของฟังก์ชั่นย่อย (เขียนเสมือนฟังก์ชั่นหลัก) จะเริ่มต้นด้วยสัญลักษณ์ เริ่มต้นการทำงาน (start) เปรียบเสมือนเครื่องหมายปีกกาเปิดเริ่มต้นการทำงานในโปรแกรม โดยลำดับถัดไปพิจารณาว่าฟังก์ชั่นมีข้อมูลทางอินพุตหรือไม่ เมื่อมีการรับข้อมูลทางอินพุต จะกำหนดสัญลักษณ์ผังงานสำหรับรับส่งข้อมูลทางอินพุต เก็บไว้ในตัวแปรที่กำหนดไว้ และดำเนินการทำในกลุ่มคำสั่งหรือคำสั่งพื้นฐานของภาษาซี (statement) ในสัญลักษณ์ดำเนินการ เมื่อฟังก์ชั่นสิ้นสุดการทำงาน จะเขียนด้วยสัญลักษณ์สิ้นสุดการทำงานของฟังก์ชั่น (end) โดยฟังก์ชั่นหลัก (main function) จะทำหน้าที่เรียกใช้งานฟังก์ชั่นย่อย ด้วยสัญลักษณ์ฟังก์ชั่นย่อย โปรแกรมจะไปดำเนินการในส่วนของฟังกชั่นย่อยจนเสร็จสิ้น แล้วกลับมาตำแหน่งเดิมของฟังก์ชั่นหลัก และจบการทำงานของโปรแกรมหลัก แสดงดังรูปที่ 12.4



รูปที่ 12.4 ผังงานฟังก์ชั่นย่อยมีการรับข้อมูลทางอินพุต

ตัวอย่างที่ 12.2 จงคำนวณการแลกเปลี่ยนอัตราเงินดอลล์ลาร์ (Dollar) เป็นเงินบาท (Baht) โดยการสร้างฟังก์ชั่นย่อย เพื่อการคำนวณอัตราแลกเปลี่ยนของเงินบาท กำหนดค่าเงินดอลล์ลาร์ เท่ากับ 20 ดอลล์ลาร์ (ข้อกำหนด 1 ดอลล์ลาร์ = 50 บาท)

ผังงานตัวอย่างที่ 12.2



รูปที่ 12.5 ผังงานของการทำงานของโปรแกรมตัวอย่างที่ 12.2

โปรแกรมตัวอย่างที่ 12.2

```
1.  #include <stdio.h>
2.  void change (int dollar)
3.  {       int baht  ;
4.          baht = dollar * 50 ;
5.          printf(“ %d Dollar is %d Baht \n”, dollar, baht) ;
6.  }
7.  main ( )
8.   {      change (20) ;
9.          change (30) ;
10. }
```

ฟังก์ชั่นย่อย

ฟังก์ชั่นหลัก

ผลลัพธ์การทดสอบโปรแกรมตัวอย่างที่ 12.2

```
20 Dollar is 1000 baht
30 Dollar is 1500 baht
```

คำอธิบาย เมื่อโปรแกรมเริ่มต้นการทำงาน คำสั่งต่างๆที่อยู่ใน main( ) จะถูกเรียกให้ทำงานทีละบรรทัด ตามลำดับ โดยภายใน main( ) มีการเรียกใช้ฟังก์ชั่น change(20) พร้อมระบุค่าคงที่ 20 ส่งไปยังอินพุตของฟังก์ชั่นย่อย change เก็บในตัวแปร dollar ที่อยู่ภายในฟังก์ชั่น และดำเนินการประกาศตัวแปร พร้อมคำนวณและแสดงผลลัพธ์ ซึ่งหลังจากทำคำสั่งในฟังก์ชั่น change เสร็จสิ้น โปรแกรมจะกลับมาทำคำสั่งถัดไปในฟังก์ชั่นหลัก main( ) การเรียกใช้ฟังก์ชั่น change(30) และทำงานเหมือนกับฟังก์ชั่น change(20) นั่นเอง เมื่อสิ้นสุดการทำงานของฟังก์ชั่นย่อยเสร็จสิ้นและจบการทำงานของโปรแกรม

ดังนั้นการรับค่าอินพุตของฟังก์ชั่น เรียกว่าค่าอินพุตที่รับ “พารามิเตอร์” (Parameter) หรือ “อาร์กิวเมนต์” (Argument) ของฟังก์ชั่นนั้น ๆ

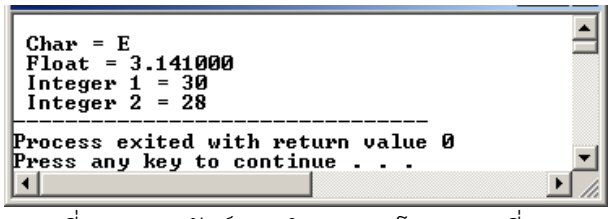
การรับค่าอินพุต หลายจำนวนที่แตกต่างกัน ของฟังก์ชั่นย่อย

การส่งค่าพารามิเตอร์ให้กับฟังก์ชั่น change() ในตัวอย่างโปรแกรม 9.3 นั้น เป็นการส่งค่าอินพุทเพียงหนึ่งจำนวน โดยการสร้างฟังก์ชั่นให้สามารถรับค่าอินพุทได้มากกว่าหนึ่งจำนวนได้ และสามารถกำหนดชนิดของข้อมูลตัวแปรได้หลากหลายๆ เช่นกัน ซึ่งไม่จำเป็นว่าจะต้องเป็นแบบจำนวนเต็มเท่านั้น แต่อาจเป็นค่าแบบสตริง, ตัวอักษร หรือจำนวนจริง (เลขทศนิยม) ได้เช่นกัน แสดงดังตัวอย่างโปรแกรมที่ 12.3 ดังนี้

ตัวอย่างโปรแกรมที่ 12.3

```
1.  #include<stdio.h>
2.  void function1 ( int a , int b , float c , char d )
3.  {       printf( " \n Char = %c  ", d ) ;
4.          printf( " \n Float = %f  ", c ) ;
5.          printf( " \n Integer 1 = %d  ", a ) ;
6.          printf( " \n Integer 2 = %d  ", b ) ;
7.  }
8.  main( )
9.  {       function1( 30 , 28 , 3.141 , 'E' );
10. }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 12.3



รูปที่ 12.6 ผลลัพธ์การทำงานของโปรแกรมที่ 12.3

การกำหนดอินพุตให้กับฟังก์ชั่นย่อยมากกว่าหนึ่งจำนวน จะต้องกำหนดอินพุตตามลำดับ จะสลับตำแหน่งกันไม่ได้และต้องกำหนดอินพุตให้ครบทุกตัว ดังเช่นตัวอย่างโปรแกรมที่ 12.3 สำหรับกรณีนี้พารามิเตอร์สองตัวแรกจะต้องเป็นค่าแบบเลขจำนวนเต็มเท่านั้น ตัวที่สามเป็นจำนวนจริง ตัวที่สี่เป็นตัวอักษร

12.5 การสร้างฟังก์ชั่นย่อยที่มีการส่งค่าทางเอาต์พุต

การสร้างฟังก์ชั่นย่อยที่มีการส่งข้อมูลหรือส่งค่าผลลัพธ์ทางเอาต์พุต ด้วยการระบุชนิดข้อมูลของเอาต์พุตออกจากฟังก์ชั่นย่อย สามารถระบุชนิดตัวแปรจำนวนเต็ม จำนวนจริง หรือตัวอักษร หากไม่มีข้อมูลส่งออกทางเอาต์พุตให้ระบุเป็น void ในส่วนของเอาต์พุตของฟังก์ชั่น การกำหนดชื่อฟังก์ชั่นย่อยตามข้อกำหนดการตั้งชื่อฟังก์ชั่นก็มีกฎเกณฑ์เหมือนกับการตั้งชื่อตัวแปร เมื่อกำหนดรูปแบบดังกล่าว คำสั่งหรือกลุ่มคำสั่งต่างๆ ภายใต้เครื่องหมายปีกกาเปิด **{** สำหรับเริ่มการทำงานของฟังก์ชั่นย่อย และเครื่องหมายปีกกาปิด **}** สำหรับสิ้นสุดการทำงานของฟังก์ชั่นย่อย

รูปแบบโปรแกรมสำหรับฟังก์ชั่นย่อย มีการส่งค่าทางเอาต์พุต สามารถเขียนได้ดังนี้

```
(1)          (2)                    (3)
Return-type ชื่อฟังก์ชั่นย่อย ( datatype var-1, datatype var-2,... ,datatype
var-n )
{
        คำสั่งหรือกลุ่มคำสั่งต่างๆ ;        (4)
        return (ค่าข้อมูลหรือตัวแปร)        (5)
}
```

หมายเลข 1 คือส่วนของเอาต์พุตฟังก์ชั่น output โดยกำหนดชนิดข้อมูล (datatype) พร้อมประกาศชื่อตัวแปร ที่ใช้ในฟังก์ชั่นย่อย ถ้าไม่มีการส่งค่าเอาต์พุตให้ระบุ void

หมายเลข 2 คือกำหนดชื่อฟังก์ชั่นย่อย (โดยไม่ซ้ำกับคำสงวนของภาษาซี)

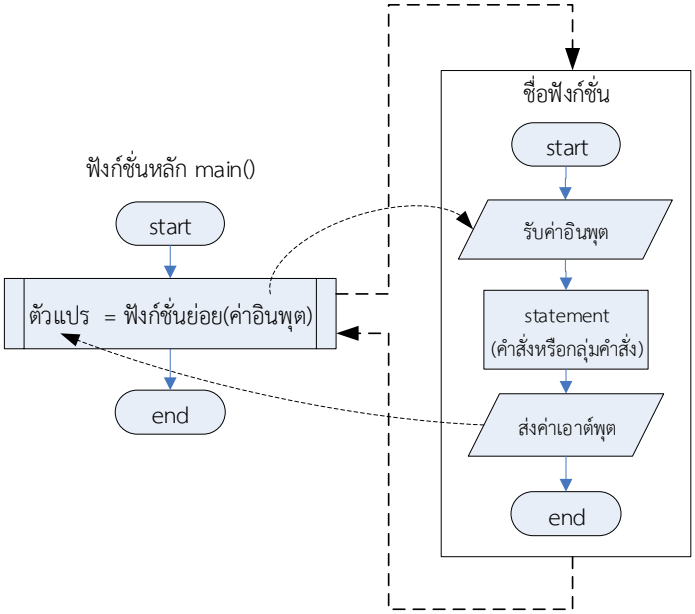
หมายเลข 3 คือส่วนของอินพุตฟังก์ชั่น input โดยกำหนดชนิดข้อมูล (datatype) พร้อมประกาศชื่อตัวแปร ที่ใช้ในฟังก์ชั่นย่อย ถ้าไม่มีการรับค่าอินพุตให้ระบุ void

หมายเลข 4 คือ คำสั่งหรือกลุ่มคำสั่งของฟังก์ชั่นย่อย ภายใต้เครื่องหมายปีกกา { } ในฟังก์ชั่นย่อย

หมายเลข 5 คือ คำสั่ง return() จะใช้เมื่อฟังก์ชั่นมีการส่งค่าออกจากฟังก์ชั่นย่อย และตามด้วยค่าข้อมูลหรือตัวแปร โดยการส่งข้อมูลจะต้องเป็นชนิดเดียวกับ Return-type ส่วนของเอาต์พุตฟังก์ชั่น

รูปแบบผังงานสำหรับฟังก์ชั่นย่อย มีการส่งค่าทางเอาต์พุต

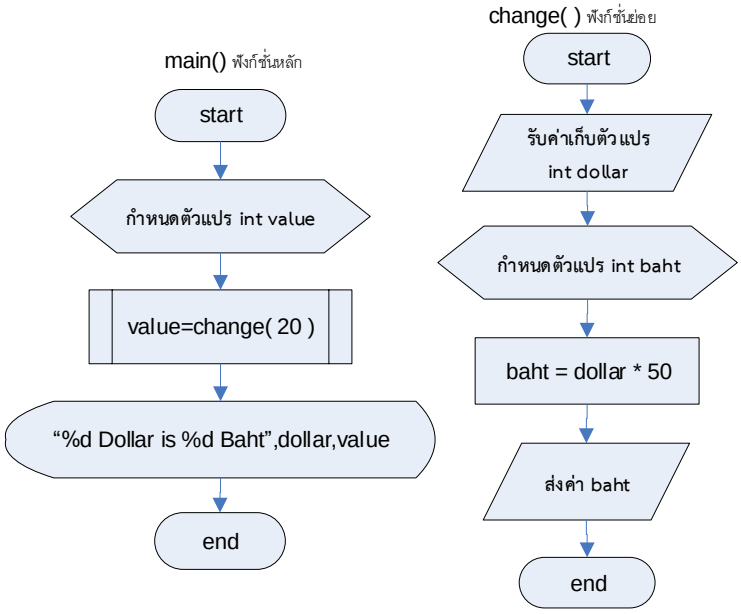
การเขียนผังงานของฟังก์ชั่นย่อย ที่มีการส่งค่าทางเอาต์พุตและรับค่าทางอินพุต จะเริ่มต้นด้วยสัญลักษณ์ เริ่มต้นการทำงาน (start) เปรียบเสมือนเครื่องหมายปีกกาเปิดเริ่มต้นการทำงานในโปรแกรม โดยลำดับถัดไปพิจารณาว่าฟังก์ชั่นมีข้อมูลทางอินพุต เมื่อมีการรับข้อมูลทางอินพุต จะกำหนดสัญลักษณ์ผังงานสำหรับรับส่งข้อมูลทางอินพุต เก็บไว้ในตัวแปรที่กำหนดไว้ และดำเนินการทำในกลุ่มคำสั่งหรือคำสั่งพื้นฐานของภาษาซี (statement) ในสัญลักษณ์ดำเนินการ ก่อนสิ้นสุดการทำงาน จะเขียนด้วยสัญลักษณ์การรับส่งข้อมูล เป็นการส่งข้อมูลทางเอาต์พุตไปเก็บไว้ในตัวแปรที่กำหนดไว้ไปยังตัวแปรที่รองรับในฟังก์ชั่นที่เรียกใช้งาน และสิ้นสุดการทำงานของฟังก์ชั่น (end) โดยฟังก์ชั่นหลัก (main function) จะทำหน้าที่เรียกใช้งานฟังก์ชั่นย่อย ด้วยสัญลักษณ์ฟังก์ชั่นย่อย โปรแกรมจะไปดำเนินการในส่วนของฟังกชั่นย่อยจนเสร็จสิ้น แล้วกลับมาตำแหน่งเดิมของฟังก์ชั่นหลัก พร้อมรับค่าจากเอาต์พุตของฟังก์ชั่นย่อย และจบการทำงานของโปรแกรมหลัก แสดงดังรูปที่ 12.7



รูปที่ 12.7 ผังงานฟังก์ชั่นย่อยมีการรับข้อมูลทางอินพุตและส่งข้อมูลทางเอาต์พุต

ตัวอย่างที่ 12.4 จงคำนวณการแลกเปลี่ยนอัตราเงินดอลล์ลาร์ (Dollar) เป็นเงินบาท (Baht) โดยการสร้างฟังก์ชั่นย่อย เพื่อการคำนวณอัตราแลกเปลี่ยนของเงินบาท กำหนดค่าเงินดอลล์ลาร์ เท่ากับ 20 ดอลล์ลาร์ (ข้อกำหนด 1 ดอลล์ลาร์ = 50 บาท)

ผังงานตัวอย่างที่ 12.4



รูปที่ 12.8 ผังงานของการทำงานของโปรแกรมตัวอย่างที่ 12.4

โปรแกรมตัวอย่างที่ 12.4

```
1.  #include <stdio.h>
2.  int change (int dollar)
3.  {       int baht  ;
4.          baht = dollar * 50 ;
5.          return (baht) ;
6.  }
7.  main ( )
8.  {        int value , x = 20;
9.          value = change ( x ) ;
10.         printf(" %d Dollar is %d Baht \n", x , value) ;
11. }
```

ฟังก์ชั่นย่อย

ฟังก์ชั่นหลัก

ผลลัพธ์การทดสอบโปรแกรมตัวอย่างที่ 12.4

20 Dollar is 1000 Baht

12.6 การประกาศตัวแปรภายในและภายนอก

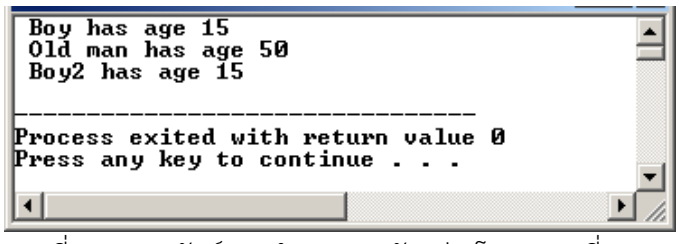
12.6.1 ตัวแปรภายใน หรือ ตัวแปรโลคอล" (Local Variable)

ในตัวอย่างที่ผ่านมา การประกาศตัวแปรเอาไว้ในฟังก์ชั่น main() และเรียกใช้จากฟังก์ชั่น main() นั่นเอง ตัวแปรที่ประกาศไว้ภายในขอบเขตของฟังก์ชั่นใดฟังก์ชั่นหนึ่งลักษณะนี้ เรียกว่า "ตัวแปรภายใน" หรือ "ตัวแปรโลคอล" (Local Variable) ซึ่งตัวแปรภายในหรือตัวแปรโลคอลนี้จะสามารถเรียกใช้ได้เฉพาะภายในฟังก์ชั่นที่มันถูกประกาศไว้เท่านั้น ลองดูตัวอย่างโปรแกรมที่ 12.5 ต่อไปนี้

ตัวอย่างโปรแกรมที่ 12.5

```
1.  #include<stdio.h>
2.  void test ( )
3.  {    int age ;
4.       age = 50;
5.       printf( "Old man has age %d \n" ,age) ;
6.  }
7.  main ( )
8.  {    int age ;
9.       age = 15;
10.      printf( " Boy has age %d \n " ,age) ;
11.      test( );
12.      printf( " Boy2 has age %d \n " ,age) ;
13. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 12.5



รูปที่ 12.9 ผลลัพธ์การทำงานของตัวอย่างโปรแกรมที่ 12.5

ภายใน main() มีการประกาศตัวแปรชื่อ age พร้อมทั้งกำหนดค่าเป็น 15 และในฟังก์ชั่น test() ก็มีการประกาศตัวแปร age ไว้เช่นกัน แต่เก็บค่า 50 เอาไว้ ตัวอย่างนี้สามารถ Build โปรแกรมได้โดยไม่เกิดความผิดพลาดใด ๆ เพราะตัวแปร age ทั้งสองนั้นอยู่คนละฟังก์ชั่นกันนั่นเอง ตัวหนึ่งอยู่ในฟังก์ชั่น test() อีกตัวหนึ่งอยู่ในฟังก์ชั่น main() ตัวแปร age ทั้งสองนี้ถือว่าเป็นตัวแปรภายในและจะเรียกใช้ได้เฉพาะภายในฟังก์ชั่นที่มันอยู่เท่านั้น จะเรียกข้ามมาจากฟังก์ชั่นอื่นไม่ได้

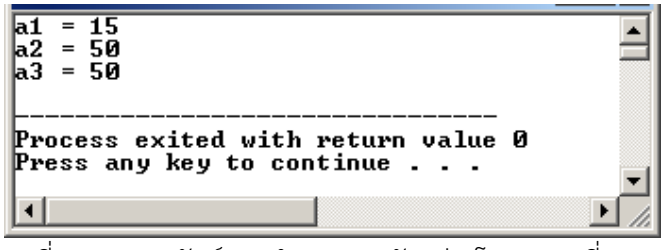
12.6.2 ตัวแปรภายนอก หรือ ตัวแปรโกลบอล (Global Variable)

ตัวแปรที่ถูกสร้างขึ้นนอกเหนือฟังก์ชั่นหลักและฟังก์ชั่นย่อย เป็นการสร้างตัวแปรตามรูปแบบการประกาศตัวแปรที่อยู่ในโปรแกรมและไม่อยู่ภายใต้ฟังก์ชั่นใดๆ เมื่อประกาศตัวแปรใดๆ ให้เป็นแบบภายนอกแล้ว ในการเรียกใช้ตัวแปรแบบภายนอกนั้นสามารถใช้งานได้ทั้งโปรแกรม

ตัวอย่างโปรแกรมที่ 12.6

```
1.  #include<stdio.h>
2.  int a ;
3.  void test( )
4.  {    a = 50;
5.       printf( "a2 = %d \n",a ) ;
6.  }
7.  main( )
8.  {    a = 15 ;
9.               printf( "a1 = %d \n",a ) ;
10.              test( );
11.              printf( "a3 = %d \n",a ) ;
12. }
```

ผลลัพธ์การทำงานของตัวอย่างโปรแกรมที่ 12.6



รูปที่ 12.10 ผลลัพธ์การทำงานของตัวอย่างโปรแกรมที่ 12.6

จากตัวอย่างโปรแกรมที่ 12.6 จะเห็นได้ว่าตัวแปร a ถูกประกาศไว้ด้านบนสุด โดยไม่ได้อยู่ภายในฟังก์ชั่นใดๆ เลย ซึ่งพื้นที่ส่วนนี้เองเป็นบริเวณที่จะใช้ประกาศตัวแปรแบบภายนอก ในที่นี้ตัวแปร a ที่ทุกๆ คำสั่งสามารถเรียกใช้ได้และค่าข้อมูลจะมีการปรับเปลี่ยนอยู่เสมอ

เมื่อผลการทำงานของโปรแกรม ซึ่งฟังก์ชั่นหลัก main() จะถูกเรียกก่อน ตัวแปร a จึงถูกกำหนดค่าให้เป็น 30 ต่อจากนั้นก็จะแสดงค่า a ออกมา และเรียกไปยังฟังก์ชั่น test() ซึ่งในฟังก์ชั่นนี้จะกำหนดค่า 50 ให้กับตัวแปร a เห็นว่าตัวแปร a ถูกเรียกใช้อีกครั้งหนึ่ง หลังจากนั้นเมื่อจบการทำงานของฟังก์ชั่น test() โปรแกรมจะกลับมาทำบรรทัดที่ 4 ของ main() ก็คือการแสดงค่า a ออกมาอีกครั้ง จะเห็นได้ว่า มีการแสดงข้อความว่าตัวแปร a มีค่าเป็น 50 ถึงสองครั้ง นั่นเพราะว่า a ถูกเปลี่ยนค่าไปในฟังก์ชั่น test() และบรรทัด

printf ในฟังก์ชั่น test() แสดงค่าของตัวแปร a ออกมาครั้งหนึ่ง จากนั้นเมื่อการทำงานกลับมาที่ main() ก็มีการแสดงค่าของ a ออกมาอีกครั้งหนึ่ง ซึ่งตัวแปร a ยังคงเก็บค่า 50 เอาไว้นั่นเอง

12.7 ฟังก์ชั่นมีความสำคัญอย่างไร

คำถาม *"ทำไมจะต้องสร้างฟังก์ชั่นเมื่อสามารถเขียนโค้ดทั้งหมดไว้ใน main() ได้?"*

เพื่อตอบคำถามนี้ ให้นักศึกษาลองพิจารณาถึงการเขียนโปรแกรมที่เอาโค้ดทั้งหมดไว้ใน main() สมมติว่าเขียนโปรแกรม 500 หรือ 1000 บรรทัด ถ้าจะหาจุดผิดพลาดของโปรแกรม หรือแก้ไขปรับปรุงโปรแกรม จะเสียเวลา ด้วยเหตุนี้การที่แยกโค้ดของโปรแกรมออกเป็นฟังก์ชั่น จึงช่วยให้โค้ดโปรแกรมดูง่ายขึ้นและเป็นสัดส่วนมากยิ่งขึ้น

การแยกโค้ดโปรแกรมออกมาเป็นฟังก์ชั่นมีข้อดี คือ การแก้ไขเพิ่มเติมสามารถทำได้ง่ายขึ้น และช่วยให้ไม่กระทบกับส่วนอื่น ๆ เพราะการทำงานของฟังก์ชั่นจะจบภายในตัวของมันเอง ยกตัวอย่างเช่น ถ้ามีโปรแกรมสัก 100 บรรทัด โดยภายใน 100 บรรทัดนี้มีส่วนที่ใช้ในการแสดงเมนู ส่วนที่ประมวลผลข้อมูล และส่วนที่แสดงผลลัพธ์ ก็สามารถแยกโค้ด 100 บรรทัดนี้ออกเป็น 3 ฟังก์ชั่นคือ

- ฟังก์ชั่น Menu() ใช้แสดงเมนูออกทางจอภาพ
- ฟังก์ชั่น Process() ใช้ในการคำนวณค่า
- ฟังก์ชั่น Display() ใช้แสดงผลลัพธ์

แต่ละฟังก์ชั่นจะมีการรับค่าพารามิเตอร์หรือส่งค่าข้อมูล (Return) ก็ขึ้นอยู่กับการออกแบบของผู้เขียนโปรแกรม ถ้าเกิดต้องการปรับปรุงการแสดงเมนู ก็เพียงเข้าไปแก้ไขที่ฟังก์ชั่น Menu() เท่านั้น ไม่ต้องไปยุ่งกับฟังก์ชั่น Process() หรือฟังก์ชั่น Display()

การแบ่งการทำงานของโปรแกรมออกเป็นส่วน ๆ นี้ ช่วยให้การพัฒนาโปรแกรมทำได้เร็วขึ้นด้วย เนื่องจาก ถ้ามีผู้พัฒนาหลายคน สามารถแยกกันพัฒนาโปรแกรมได้ โดยแยกเป็นฟังก์ชั่นๆ ไป เช่น

- นาย A สร้างฟังก์ชั่น Menu() สำหรับแสดงเมนูบนจอภาพ โดยกำหนดเมนูสีเหลือง ตัวอักษรสีแดง เป็นต้น
- นาย B สร้างฟังก์ชั่น Process() สำหรับประมวลผลข้อมูล โดยรับพารามิเตอร์ 3 ตัว เป็นเลขจำนวนเต็ม 2 ตัว และเลขทศนิยม 1 ตัว นำไปคำนวณโดยใช้สมการ XYZ (ยกตัวอย่าง) และคืนค่ากลับมายังจุดที่เรียกใช้เป็นเลขทศนิยม 3 ตำแหน่ง
- นาย C สร้างฟังก์ชั่น Display() สำหรับนำเลขทศนิยมที่คำนวณได้จากฟังก์ชั่น Process() มาแสดงออกทางจอภาพในรูปของกราฟ

จากตัวอย่างต้น จะเห็นว่านาย A นาย B และนาย C ได้รายละเอียดของฟังก์ชั่นไป โดยที่ทั้งสามคนสามารถแยกกันพัฒนาโปรแกรม เพียงรู้แค่ว่าต้องเขียนฟังก์ชั่นให้ได้ตามที่ต้องการ เมื่อเสร็จแล้วผู้ออกแบบโปรแกรมก็นำฟังก์ชั่นทั้งสามส่วนมารวมเป็นโปรแกรมที่สมบูรณ์ ตัวอย่างนี้แสดงถึงข้อดีของการใช้ฟังก์ชั่นที่ช่วยสนับสนุนการทำงานเป็นทีม

สรุปท้ายบท

การสร้างฟังก์ชั่นย่อย มี 3 รูปแบบ คือ รูปแบบที่ไม่มีการกำหนดค่าอินพุตและเอาต์พุต รูปแบบที่มีการกำหนดค่าอินพุตเข้าไปในฟังก์ชั่น และรูปแบบการส่งข้อมูลออกจากฟังก์ชั่น โดยแต่ละรูปแบบของฟังก์ชั่นสามารถในไปประยุกต์ใช้งานร่วมกับฟังก์ชั่นหลัก การเรียกใช้งานฟังก์ชั่นย่อยสามารถเรียกใช้งานได้ทุกๆส่วนของโปรแกรม โดยฟังก์ชั่นที่มีการรับค่าอินพุต สามารถระบุอินพุตที่มีความแตกต่างกันของชนิดตัวแปรได้ และในการระบุข้อมูลให้กับอินพุตต้องเป็นข้อมูลที่สอดคล้องกัน หากข้อมูลรับเป็นจำนวนเต็มแต่ทำการระบุจำนวนจริงหรือทศนิยม ค่าที่ได้รับไม่สอดคล้องกันทำให้ข้อมูลผิดพลาดได้ และการส่งข้อมูลออกจากฟังก์ชั่น ต้องเป็นชนิดเดียวกับข้อมูลที่ต้องการส่งออกจากฟังก์ชั่น โดยการส่งออกจากฟังก์ชั่นเอาต์พุตจะต้องใช้คำสั่ง return เสมอ และค่าข้อมูลต้องเป็นชนิดเดียวกับที่กำหนดในส่วนของเอาต์พุตฟังก์ชั่น

คำถามท้ายบท

1. สัญลักษณ์ผังงานแบบใดที่ใช้ในการรับส่งข้อมูล?
2. สัญลักษณ์ผังงานแบบใดที่ใช้สำหรับเรียกใช้งานฟังก์ชั่นย่อย?
3. Prototype function คืออะไร?
4. Local variable คืออะไร?
5. Global variable คืออะไร?
6. Arguments ของอินพุตหมายถึงอะไร?
7. void หมายถึงอะไร?
8. ฟังก์ชั่นหลักสามารถสร้างได้สูงสุดกี่ฟังก์ชั่นหลัก?
9. ฟังก์ชั่นย่อย ที่มีอินพุตเข้าไปในฟังก์ชั่น สามารถกำหนดชนิดของอินพุตแตกต่างกันได้หรือไม่?
10. การสร้างฟังก์ชั่น ชื่อ Main( ) สามารถสร้างได้หรือไม่?
11. คำสั่ง Return จะใช้ควบคู่กับฟังก์ชั่นประเภทใด?
12. กำหนดให้ float commands (int a, int b, float c, char d[10]) ข้อมูลอินพุตทั้งหมดมีกี่จำนวน?
13. จากข้อที่ 10 ข้อมูลเอาต์พุตของฟังก์ชั่นเป็นชนิดใด?
14. การสร้างตัวแปรชื่อเหมือนกันและชนิดเดียวกัน โดยใช้งานภายในฟังก์ชั่นย่อย และยังสร้างเหมือนกันภายในฟังก์ชั่นหลัก สามารถทำได้หรือไม่? เพราะอะไร?
15. การสร้างฟังก์ชั่นย่อยมีประโยชน์อย่างไร?

แบบฝึกหัดท้ายบท

1. จงเขียนผังการทำงานและโปรแกรม นับจำนวนเลขคู่ (Even number) และนับจำนวนเลขคี่ (Odd number) ของช่วงกลุ่มตัวเลข (จำนวนเต็ม) ของเลขคู่และเลขคี่ โดยกำหนดค่าอินพุตจากผู้ใช้งาน สาหรับค่าเริ่มต้นและค่าสิ้นสุด แสดงผลลัพธ์ดังตัวอย่างต่อไปนี้

   ข้อกำหนด

   ฟังก์ชั่นหลัก: รับข้อมูลจำนวนเต็มค่าเริ่มต้น และค่าสิ้นสุด ทางคีย์บอร์ดจากผู้ใช้ และแสดงผลลัพธ์

   ฟังก์ชั่นย่อย 1: ทำหน้าที่นับจำนวนเลขคู่ (Even number)

   ฟังก์ชั่นย่อย 2: ทำหน้าที่นับจำนวนเลขคี่ (Odd number)

   ตัวอย่างผลลัพธ์แสดงดังนี้

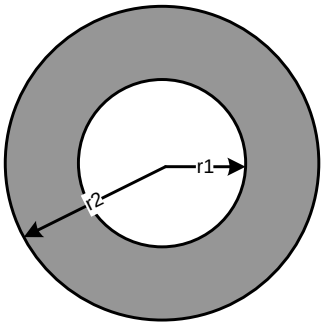
```
Enter number to start = -10
Enter number to end = 10
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
Even number = 11
Odd number = 10
```

2. จากรูปที่กำหนดให้ จงเขียนโฟล์ชาร์ตและโปรแกรมตามที่กำหนดให้ โดยสร้างฟังก์ชั่นย่อย สำหรับการคำนวณหาพื้นที่แรเงา โดยทำการรับค่าอินพุต รัศมีของวงกลมด้านในและด้านนอกจากคีย์บอร์ด ด้วยฟังก์ชั่นหลัก main()

   ข้อกำหนด

   ฟังก์ชั่นหลัก: ทำหน้าที่รับค่าอินพุททั้งหมดจากผู้ใช้ และแสดงผลลัพธ์การคำนวณทางหน้าจอ

   ฟังก์ชั่นย่อย: ทำหน้าที่คำนวณพื้นที่แรเงา



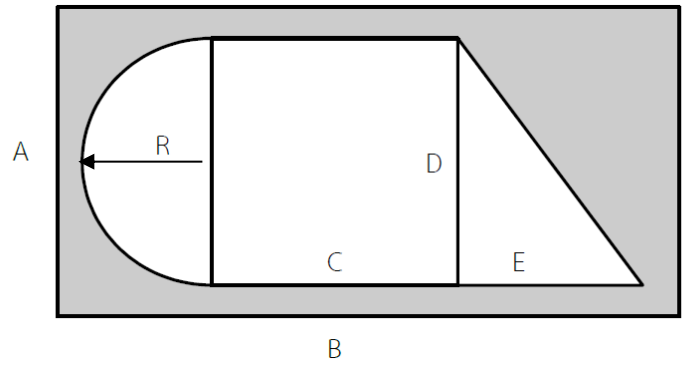
3. จงเขียนโปรแกรมและผังการทำงาน คำนวณหาพื้นที่แรเงาของรูปด้านล่างโดยกำหนดให้สร้างฟังก์ชั่นย่อยจำนวน 3 ฟังก์ชั่นย่อย เพื่อคำนวณหาค่าต่างๆต่อไปนี้

ฟังก์ชั่นย่อย 1: สำหรับคำนวณหา พื้นที่ครึ่งวงกลม

ฟังก์ชั่นย่อย 2: สำหรับคำนวณหา พื้นที่สี่เหลี่ยม

ฟังก์ชั่นย่อย 3: สำหรับคำนวณหา พื้นที่สามเหลี่ยมมุมฉาก

ฟังก์ชั่นหลัก: เรียกใช้งานโปรแกรมย่อยทั้งหมด เพื่อใช้คำนวณหาพื้นที่แรเงา ดังรูปและกำหนดให้ค่าอินพุตทั้งหมด ป้อนข้อมูลผ่านคีย์บอร์ด



แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 13 สตริง (String)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจแนวคิดเกี่ยวกับ string การใช้งานฟังก์ชั่น string
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและเขียนโปรแกรมที่เกี่ยวข้องกับ string
- เพื่อให้นักศึกษาสามารถใช้ไลบรารี่ที่เกี่ยวข้องกับ string

เนื้อหา

- การใช้งานฟังก์ชั่นของ string สำหรับข้อความอักษร การนับจำนวนข้อความอักษร การคัดลอกจำนวนข้อความอักษร รวมถึงการเชื่อมต่อข้อความเป็นประโยคข้อความอักษร การสร้างตัวแปรที่เกี่ยวข้องกับ string

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นและแนวคิดเกี่ยวกับ string การใช้งานฟังก์ชั่นใดๆของ string
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 13

สตริง

String

String คือสายของอักขระ (char) กลุ่มข้อความอักษรที่ประกอบขึ้นเป็นคำหรือประโยค โดยมีความยาวตามจำนวนที่สามารถกำหนดได้ รูปแปรของการเก็บข้อมูล string นั้นแตกต่างกันตามแต่ภาษาคอมพิวเตอร์ที่ต่างกัน เช่น รูปแบบที่มีการกำหนดจุดสิ้นสุดของ string โดยใช้อักขระใน ASCII code ตัวที่กำหนดไว้แล้ว ตามรูปแบบที่แสดงในรูปที่ 13.1

S	T	R	I	N	G	**\0**

รูปที่ 13.1 การแบ่งคั่นของตัวแปรสตริง (Delimiter String)

จากรูปที่ 13.1 จะเห็นว่าในช่องสี่เหลี่ยมสุดท้ายจะเป็นเครื่องหมาย **\0** (null) ที่ไม่เกี่ยวข้องกับคำ "STRING" เลย แต่อักขระนี้จะอยู่ร่วมกับ string และในภาษาซี ก็ใช้วิธีการ Delimiter ในการเก็บค่า string ในภาษาอื่นๆ อาจใช้วิธีการที่แตกต่างกันในการเก็บค่า string เช่น Length-Controlled ซึ่งใช้วิธีการเก็บค่าความยาวของคำหรือประโยคไว้ด้วย ตามรูปแบบที่แสดงในรูปที่ 13.2 ให้สังเกตุค่าแรกใน string คือขนาดของ string นั้น

6	**S**	**T**	**R**	**I**	**N**	**G**

รูปที่ 13.2 Delimiter String ในภาษาอื่นๆ นอกเหนือภาษาซี

13.1 String ในภาษาซี

การเขียน string ในภาษาซี ไม่ได้ทำการจำกัดความยาวของ string ล่วงหน้า แต่ใช้ \0 (null) character ในการกำหนดจุดสิ้นสุดของ string โดยอักขระที่ประกอบขึ้นเป็น string นั้นจะอยู่ร่วมกันในลักษณะของอาร์เรย์ที่มี **\0** เป็นสมาชิกตัวสุดท้าย

การสร้าง string ในโปรแกรมภาษาซีนั้นสามารถทำได้หลายวิธี เช่น

char str[9] = "Good Day";

ข้อมูลที่ถูกเก็บในอาร์เรย์ชื่อ str[ ] ประกอบด้วยดังนี้

G	**o**	**o**	**d**		**D**	**a**	**y**	**\0**

รูปที่ 13.3 ข้อมูลในตัวแปรอาเรย์ str

char month[ ] = "January";

ให้สังเกตุว่าเราไม่จำเป็นต้องระบุความยาวของ string ดังนี้

J	a	n	u	a	r	y	\0

รูปที่ 13.4 ข้อมูลในตัวแปรอาเรย์ month

ในทำนองเดียวกันกับตัวแปรชนิดอาร์เรย์ที่ได้ศึกษาในเนื้อหาที่ผ่านมา จะไม่สามารถใช้วิธีการกำหนดเครื่องหมายเท่ากับ (=) เพื่อทำการย้ายหรือทำสำเนา (Copy) ของข้อมูล string จากอาร์เรย์หนึ่งไปสู่อีกอาร์เรย์หนึ่งได้ เช่น

```
char str1[6] = "Hello";
char  str2;
str2 = str1;                // ไม่สามารถดำเนินการได้ ซึ่งเป็นวิธีการที่ ไม่ถูกต้อง
```

ดังนั้นเมื่อต้องการย้ายข้อมูลจากระหว่างตัวแปร string สามารถใช้วิธีการเดียวกับการย้ายข้อมูลในอาร์เรย์ คือการย้ายสมาชิกในอาร์เรย์ครั้งละตำแหน่งโดยใช้ loop เข้ามาร่วมด้วยนั่นเอง

13.2 ฟังก์ชั่นที่เกี่ยวข้องกับการรับข้อมูลและแสดงค่าข้อมูล

คำสั่ง scanf( )

การรับค่าโดยใช้คำสั่ง scanf นั้น ให้ใช้สัญลักษณ์ **%s** เป็น format specifier และสามารถทำการแทรกสัญลักษณ์อื่นๆระหว่าง เครื่องหมาย % และอักษร s

เช่น ตัวอย่างต่อไปนี้ การแทรกตัวเลข ซึ่งเป็นการระบุความยาวของ string ที่จะรับจากคีย์บอร์ด

```
char  month[10] ;
printf( "Please Enter a name of month" ) ;
scanf( "%9s" , month) ;
```

ให้สังเกตุว่าตัวแปร month นั้นมีขนาด 10 จำนวน แต่ระบุรับค่า %9s เท่ากับ 9 ตำแหน่งเท่านั้น เพราะโปรแกรมจะต่อท้าย string ด้วย \0 เสมอ จึงจำเป็นต้องกำหนดไว้ด้วยทุกครั้ง และยังสามารถกำหนดได้ว่าจะรับอักษรตัวใดบ้าง เช่น ถ้าต้องการที่จะรับ string ประกอบด้วยตัวเลข 0-9 เท่านั้น สามารถทำได้โดยใช้เครื่องหมาย Bracket [ ] ดังตัวอย่างต่อไปนี้

```
scanf ("%10[0 1 2 3 4 5 6 7 8 9]", str);
```

คำสั่ง printf( )

การแสดงผล string ทางจอภาพโดยใช้คำสั่ง printf ก็ต้องใช้เครื่องหมาย %s เช่นกัน เช่น

```
printf("Enter your string ");
scanf("%8s",str);
printf("Your string is %s\n",str);
```

คำสั่ง gets( ) คำสั่ง gets( ) เป็นอีกทางเลือกหนึ่งสำหรับการรับค่า string ผ่านทาง key board

```
char str[8];
printf("Enter your string ");
gets(str);
printf("Your string is %s\n",str);
```

คำสั่ง puts( ) คำสั่ง puts( ) สามารถใช้แสดงข้อมูลใน string ผ่านทาง key board

```
char str[ ]= "My string\n";
printf("Your string is");
puts(str);
```

13.3 ฟังก์ชั่นอื่นๆเกี่ยวกับ String

ในการเขียนโปรแกรมจะต้องทำงานร่วมกับข้อมูลที่เป็นคำหรือประโยคข้อความ รวมถึงข้อความอักษร จึงหลีกเลี่ยงไม่ได้ที่จะต้องใช้งานข้อมูลที่เป็น string และเนื่องจากรูปแบบของ string ที่ไม่ได้เป็น data type ของภาษาซี (แต่ดูเหมือนเป็นอาเรย์ของตัวอักษร) การใช้งานข้อมูลที่เป็น string จึงต้องมีคำสั่งต่างๆ เพื่ออำนวยความสะดวกในการทำงาน ซึ่งสามารถเรียกใช้งานได้โดยจะต้อง include ไฟล์ชื่อ string.h

โดยคำสั่งใดๆที่อยู่ภายใต้ไลบราลี่ string.h สามารถเรียกใช้งานได้อย่างอิสระ โดยในเนื้อหานี้จะใช้คำสั่งที่เกี่ยวข้องกับ string จำนวน 5 คำสั่ง ดังนี้

1. String Length: strlen
2. String copy: strcpy
3. String number copy: strncpy
4. String compare: strcmp
5. Concatenate String: strcat

โดยแต่ละรูปแบบของ string สามารถอธิบายได้ดังต่อไปนี้

1. String Length: strlen( ) คำสั่งนี้ใช้เพื่อตรวจสอบความยาวของข้อความที่ต้องการพิจารณา มีรูปแบบดังนี้

int strlen(const char* string)

ตัวอย่างโปรแกรมที่ 13.1

```
1.  #include <stdio.h>
2.  #include<string.h>
3.  main()
4.  {           char str[ ]= "Sripatum University";
5.              int length;
6.              length = strlen (str);
7.              printf("String length = %d", length);
8.  }
```

ผลลัพธ์โปรแกรมที่ 13.1

String length = 19

คำอธิบาย ข้อความอักษรที่อยู่ภายใต้เครื่องหมาย “ ” จะถูกนับทุกตัวอักษรหรือทั้งหมด ซึ่งช่องว่าง (space bar) ถือว่าเป็นข้อมูล 1 จำนวน โดยมีค่าเท่ากับ 0x20 ตรงกับ ascii code ของคีย์บอร์ด

2. String Copy: strcpy( ) คำสั่งนี้ใช้เพื่อคัดลอกข้อความต้นทาง (fromStr) ไปยังปลายทาง (toStr) ที่ต้องการพิจารณา มีรูปแบบดังนี้

char* strcpy(char* toStr , const char* fromStr)

ตัวอย่างโปรแกรมที่ 13.2

```
1.  #include<stdio.h>
2.  #include<string.h>
3.  main( )
4.  {   char str1[ ] = "Hello";
5.              char str2[7];
6.              strcpy(str2, str1);
7.              printf("Source word = %s\n" "destination word = %s\n",str1,str2);
8.  }
```

ผลลัพธ์โปรแกรมที่ 13.2

Source word = Hello

destination word = Hello

คำอธิบาย การใช้คำสั่งคัดลอก strcpy เป็นการคัดลอกข้อความอักษรจากต้นทางไปยังปลายทางตามรูปแบบการใช้งาน โดยตัวแปรปลายทางต้องมีขนาดที่มากกว่าหรือเท่ากับข้อมูลต้นทาง จึงจะสามารถคัดลอกข้อมูลได้อย่างถูกต้อง ซึ่งคำสั่ง strcpy มีปัญหาเกี่ยวกับความยาวของ string ต้นทางและปลายทาง ที่ผู้เขียนจะต้องคำนึงถึงตัวแปรในการรองรับข้อมูลให้มีขนาดหรือจำนวนที่เพียงพอต่อการดำเนินการ มิฉะนั้นจะทำให้ข้อมูลจากต้นทางเกินกว่าตัวแปรอาเรย์ปลายทาง ทำให้ไปซ้อนทับข้อมูลอื่นได้

3. String-number Copy คำสั่งนี้ใช้เพื่อคัดลอก ข้อความต้นทาง (fromStr) ไปยังปลายทาง (toStr) โดยระบุจำนวนอักษรที่ต้องการคัดลอก (size) มีรูปแบบดังนี้

char* strncpy(char* toStr, const char* fromStr, int size)

ตัวอย่างโปรแกรมที่ 13.3

```
1.   #include <stdio.h>
2.   #include <string.h>
3.  main ( )
4.  {  char str1[ ]= "To be or not to be";
5.     char str2[40];
6.     char str3[40];
7.     /* copy to sized buffer (overflow safe): */
8.     strncpy ( str2, str1, sizeof(str2) );
9.     /* partial copy (only 5 chars): */
10.    strncpy ( str3, str2, 5 );
11.    str3[5] = '\0';    /* null character manually added */
12.    puts (str1);
13.    puts (str2);
14.    puts (str3);
15. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 13.3

To be or not to be

To be or not to be

To be

คำอธิบาย การใช้คำสั่ง strncpy จะทำงานคล้ายกับ strcpy แต่ในรูปแบบนี้จะคัดลอกแบบระบุจำนวนหรือระบุขนาดที่ต้อง เช่น strncpy(str2, str1, 10) จำนวนที่ต้องการเท่ากับ 10 จำนวนอักษร ซึ่งเป็นการคัดลอกข้อความอักษรจากต้นทางไปยังปลายทางตามรูปแบบการใช้งาน โดยตัวแปรปลายทางต้องมีขนาดที่เท่ากันหรือมากกว่า

4. String Compare: strcmp( ) คำสั่งนี้ใช้เพื่อทำการเปรียบเทียบความเหมือนกันระหว่าง string สองชุดคือ str1 และ str2 โดยทำการเปรียบเทียบตามเงื่อนไขต่อไปนี้

ถ้า String ทั้งสองมีความยาวเท่ากันหรือไม่ และสมาชิกมีค่าตรงกันทุกตำแหน่งหรือไม่ ถ้าใช่ return 0

ถ้า String แรกมีความยาวกว่าหรือสมาชิกมีค่ามากกว่า String ที่สอง return ค่าบวก (มากกว่า 0)

ถ้า String แรกมีความยาวน้อยกว่าหรือสมาชิกมีค่าน้อยกว่า String ที่สอง return ค่าลบ (น้อยกว่า 0)

int* strcmp(const char* str1, const char* str2)

ตัวอย่างโปรแกรมที่ 13.4

```
1.  #include <stdio.h>
2.  #include <string.h>
3.   main ()
4.  {    char key[ ] = "apple";
5.       char buffer[80];
6.       do {
7.                printf ("Guess my favorite fruit? ");
8.                scanf ("%79s",buffer);
9.           } while (strcmp (key, buffer) != 0) ;
10.      puts ("Correct answer!");
11. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 13.4

Guess my favourite fruit? orange

Guess my favourite fruit? apple

Correct answer!

คำอธิบาย การใช้คำสั่ง strcmp จะเปรียบเทียบข้อความอักษรระหว่างต้นทาง str2 กับปลายทาง str1 ในการเปรียบเทียบตามข้อกำหนดมาตรฐาน ASCII Code (ตารางที่ 13.1) สำหรับตัวอักษรที่มีค่าแตกต่างกัน สามารถพิจารณาตัวอักษรที่มีค่าแตกต่างกัน เช่น ASCII code ของ 65 คือ A และ 97 คือ a เป็นต้น ดังนั้นการเปรียบเทียบจะเป็นไปตามเงื่อนไข srtcmp ที่กำหนดข้างต้น

ตารางที่ 13.1 ข้อมูล ASCII CODE 128 characters

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	**NUL** (null)	32	40	20	(space)	64	100	40	**@**	96	140	60	**`**
1	1	01	**SOH** (start of header)	33	41	21	**!**	65	101	41	**A**	97	141	61	**a**
2	2	02	**STX** (start of text)	34	42	22	**"**	66	102	42	**B**	98	142	62	**b**
3	3	03	**ETX** (end of text)	35	43	23	**#**	67	103	43	**C**	99	143	63	**c**
4	4	04	**EOT** (end of transmission)	36	44	24	**$**	68	104	44	**D**	100	144	64	**d**
5	5	05	**ENQ** (enquiry)	37	45	25	**%**	69	105	45	**E**	101	145	65	**e**
6	6	06	**ACK** (acknowledge)	38	46	26	**&**	70	106	46	**F**	102	146	66	**f**
7	7	07	**BEL** (bell)	39	47	27	**'**	71	107	47	**G**	103	147	67	**g**
8	10	08	**BS** (backspace)	40	50	28	**(**	72	110	48	**H**	104	150	68	**h**
9	11	09	**HT** (horizontal tab)	41	51	29	**)**	73	111	49	**I**	105	151	69	**i**
10	12	0a	**LF** (line feed - new line)	42	52	2a	*****	74	112	4a	**J**	106	152	6a	**j**
11	13	0b	**VT** (vertical tab)	43	53	2b	**+**	75	113	4b	**K**	107	153	6b	**k**
12	14	0c	**FF** (form feed - new page)	44	54	2c	**,**	76	114	4c	**L**	108	154	6c	**l**
13	15	0d	**CR** (carriage return)	45	55	2d	**-**	77	115	4d	**M**	109	155	6d	**m**
14	16	0e	**SO** (shift out)	46	56	2e	**.**	78	116	4e	**N**	110	156	6e	**n**
15	17	0f	**SI** (shift in)	47	57	2f	**/**	79	117	4f	**O**	111	157	6f	**o**
16	20	10	**DLE** (data link escape)	48	60	30	**0**	80	120	50	**P**	112	160	70	**p**
17	21	11	**DC1** (device control 1)	49	61	31	**1**	81	121	51	**Q**	113	161	71	**q**
18	22	12	**DC2** (device control 2)	50	62	32	**2**	82	122	52	**R**	114	162	72	**r**
19	23	13	**DC3** (device control 3)	51	63	33	**3**	83	123	53	**S**	115	163	73	**s**
20	24	14	**DC4** (device control 4)	52	64	34	**4**	84	124	54	**T**	116	164	74	**t**
21	25	15	**NAK** (negative acknowledge)	53	65	35	**5**	85	125	55	**U**	117	165	75	**u**
22	26	16	**SYN** (synchronous idle)	54	66	36	**6**	86	126	56	**V**	118	166	76	**v**
23	27	17	**ETB** (end of transmission block)	55	67	37	**7**	87	127	57	**W**	119	167	77	**w**
24	30	18	**CAN** (cancel)	56	70	38	**8**	88	130	58	**X**	120	170	78	**x**
25	31	19	**EM** (end of medium)	57	71	39	**9**	89	131	59	**Y**	121	171	79	**y**
26	32	1a	**SUB** (substitute)	58	72	3a	**:**	90	132	5a	**Z**	122	172	7a	**z**
27	33	1b	**ESC** (escape)	59	73	3b	**;**	91	133	5b	**[**	123	173	7b	**{**
28	34	1c	**FS** (file separator)	60	74	3c	**<**	92	134	5c	****	124	174	7c	**|**
29	35	1d	**GS** (group separator)	61	75	3d	**=**	93	135	5d	**]**	125	175	7d	**}**
30	36	1e	**RS** (record separator)	62	76	3e	**>**	94	136	5e	**^**	126	176	7e	**~**
31	37	1f	**US** (unit separator)	63	77	3f	**?**	95	137	5f	**_**	127	177	7f	**DEL** (delete)

ที่มา: https://anon117blog.files.wordpress.com/2014/08/ascii_tabla.png

5. Concatenate String: strcat คำสั่งนี้ใช้เพื่อการนำข้อความอักษรจากต้นทาง (source) นำไปต่อท้ายข้อความที่อยู่ปลายทาง (destination) พิจารณาจากรูปแบบดังนี้

char* strcat (char * destination, const char * source);

ตัวอย่างโปรแกรมที่ 13.5

```
1.  #include <stdio.h>
2.  #include <string.h>
3.   main ( )
4.  {    char str[80];
5.               strcpy (str,"these ");
6.               strcat (str,"strings ");
7.               strcat (str,"are ");
8.               strcat (str,"concatenated.");
9.               puts (str);
10. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 13.5

These string are concatenated.

คำอธิบาย การใช้คำสั่ง strcat เป็นการนำข้อความอักษรหรือตัวอักษรของต้นทาง source นำมาต่อท้ายข้อความที่มีอยู่ในปลายทาง destination โดยข้อมูลเดิมที่อยู่ในตัวแปรส่วนของ destination จะเพิ่มขึ้นข้อความอักษรจากเดิมที่มีอยู่ โดยไม่ลบข้อความเดิม

สรุปท้ายบท

คำสั่งที่ใช้ในรูปแบบของ String ที่อยู่ภายในไลบรารี่ string.h จะต้องมีการเรียกใช้งานไลบรารี่ จึงจะใช้งานคำสั่งใดๆในไลบรารี่ได้ โดยกำหนด #include <string.h> ในส่วนหัวของโปรแกรม หากไม่มีไลบรารี่ดังกล่าวคำสั่งที่เกี่ยวข้องกับ string จะไม่สามารถใช้งานได้ โดยตัวแปรที่เกี่ยวข้องกับ string จะใช้ตัวแปรอาเรย์ ที่ประกาศตัวแปรเป็น char เพื่อจัดเก็บตัวอักษรหรือข้อความอักษร ดังนั้น การกำหนดขนาดของอาเรย์จึงมีความสำคัญอย่างมาก ตัวแปร string จะต้องมี \0 สิ้นสุดข้อความอักษรเสมอ เสมือนจบประโยคข้อความอักษรนั้นเอง การเขียนโปรแกรมเพื่อตรวจสอบการสิ้นสุดของประโยคจึงใช้การตรวจสอบเมื่อมีค่าเท่ากับ \0

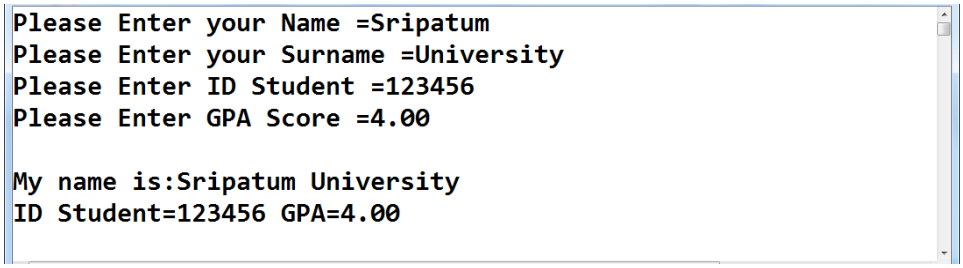
คำถามท้ายบท

1. ตัวแปรประเภทใดที่ใช้สำหรับการเก็บข้อมูลข้อความอักษร?
2. ข้อมูลตัวอักษรที่ใช้ในการตรวจสอบสิ้นสุดของประโยคข้อความอักษรคืออะไร?
3. หากต้องการนำข้อความ “Hello”และ“world” มาสร้างเป็นประโยค “Helloworld” ต้องใช้รูปแบบใด?
4. กำหนดให้ char a[x] = {“Sripatum University”} จงระบุค่าของ x ที่เหมาะสม?
5. กำหนดให้ char a[5] ={“1234”} จงหา a[5] มีค่าเท่าใด?
6. คำสั่งการคัดลอกข้อความแบบใดที่สามารถระบุจำนวนที่ต้องการคัดลอก?
7. ข้อความ Hello มีค่าเลขฐาน 16 เท่ากับเท่าใด?
8. ข้อแตกต่างระหว่างคำสั่ง strcpy และ strncpy คืออะไร?
9. กำหนดให้ int z = strlen(“We are Engineering”) ; ดังนั้น z มีค่าเท่าใด?
10. กำหนดให้ char a[ ] ={“Computer”} ; x = strlen(a)++ ; ดังนั้น x มีค่าเท่าใด?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมบันทึกข้อมูลของนักศึกษา **ชื่อ นามสกุล รหัสนักศึกษา และเกรดเฉลี่ย** ในรูปแบบของตัวแปร String โดยรับค่าอินพุตจากผู้ใช้และแสดงผลทางหน้าจอคอมพิวเตอร์

   ตัวอย่างผลลัพธ์การทำงานของโปรแกรม



2. จงเขียนโปรแกรมตรวจสอบจำนวนตัวอักษรของประโยคข้อความ โดยนับจำนวนทั้งหมด จำนวนตัวพิมพ์ใหญ่ (Uppercase) และจำนวนตัวพิมพ์เล็ก (Lowercase) ข้อความที่รับอินพุตจากผู้ใช้งาน

   ตัวอย่างผลลัพธ์การทำงานของโปรแกรม

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 14 เรียนรู้การใช้ตัวแปรแบบโครงสร้าง (Structure variable)

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจพื้นฐานของตัวแปรชนิดโครงสร้าง
- เพื่อให้นักศึกษาเข้าใจการเขียนผังงานและเขียนโปรแกรมที่เกี่ยวข้องกับตัวแปรชนิดโครงสร้าง

เนื้อหา

- การสร้างตัวแปรชนิดโครงสร้าง การประกาศตัวแปรโครงสร้าง
- การเขียนโปรแกรมใช้งานตัวแปรชนิดโครงสร้าง

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายความรู้เบื้องต้นและแนวคิดเกี่ยวกับตัวแปรชนิดโครงสร้าง
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน
- ให้ทำแบบฝึกหัดท้ายบท
- ให้นักศึกษาเตรียมอ่านเนื้อหาล่วงหน้าในสัปดาห์ถัดไป

การบรรยายสัปดาห์ที่ 14

ตัวแปรแบบโครงสร้าง
Structure variable

บทเรียนนี้ กล่าวถึงการสร้างตัวแปรชนิดโครงสร้างข้อมูล เป็นการกำหนดชนิดข้อมูลใหม่ขึ้นมาใช้งาน โดยเป็นการรวบรวมกันของชนิดข้อมูลพื้นฐานที่เป็นชนิดเดียวกันหรือแตกต่างกันเข้ามาไว้ในตัวแปรชนิดโครงสร้าง ซึ่งสามารถจัดเก็บข้อมูลหรือสร้างชุดข้อมูลภายในตัวแปรโครงสร้างเดียวกัน

ตัวแปรโครงสร้าง (structure) สามารถจัดการข้อมูลที่มีจำนวนมากและมีความหลากหลายของแต่ละชนิดข้อมูล เช่น การจัดเก็บข้อมูลของนักศึกษา การจัดเก็บข้อมูลเมนูอาหาร การจัดเก็บตารางสำคัญต่างๆ เป็นต้น โดยบทเรียนนี้มุ่งเน้นให้สามารถนำไปประยุกต์ใช้งานในงานด้านวิศวกรรมศาสตร์ และอื่นๆที่เกี่ยวข้อง รูปแบบการเขียนโค้ดของภาษาซี แสดงดังนี้

14.1 ตัวแปรโครงสร้าง

ตัวแปรโครงสร้าง (structure) เป็นการสร้างตัวแปรใหม่ขึ้นโดยเป็นการนำตัวแปรชนิดพื้นฐานของภาษาซีมาเป็นโครงสร้างของตัวแปร โดยชนิดของตัวแปรที่อยู่ในโครงสร้างมีความแตกต่างกันได้

```
struct structure_name {
        datatype variable1 ;
        datatype variable2 ;
        datatype variable3 ;
        ...
        datatype variableN ;
        } ;
```

คำอธิบายดังนี้

struct คือ การกำหนดโครงสร้างข้อมูล structure

structure_name คือ กำหนดชื่อตัวแปรโครงสร้าง ต้องไม่ซ้ำกับคำสงวนของภาษาซี

datatype คือการระบุชนิดของข้อมูล ตามข้อกำหนดการประกาศตัวแปร

variable คือชื่อตัวแปรที่สร้างขึ้น ตามข้อกำหนดการตั้งชื่อตัวแปร

การเรียกใช้งานของตัวแปรชนิดโครงสร้าง สามารถระบุได้ดังนี้

struct structure_name structure_variable*1*,... , structure_variable*N* ;

ตัวอย่างเช่น การสร้างตัวแปรโครงสร้างข้อมูลนักศึกษา โดยกำหนดให้มีการระบุข้อมูล รหัสนักศึกษา (ID) เป็นจำนวนเต็ม ข้อมูลอักษรชื่อนักศึกษา (Name) จำนวน 20 ตัวอักษร ข้อมูลนามสกุลนักศึกษา (Surname) จำนวน 20 ตัวอักษร และ ข้อมูลเกรดเฉลี่ย (GPA) เลขจำนวนทศนิยม แสดงดังนี้

```
1.  struct students {
2.          int ID ;
3.          char Name[20] ;
4.          char Surname[20] ;
5.          float GPA ;
6.  }
```

เมื่อกำหนดตัวแปรชนิดโครงสร้างชื่อ students โดยสร้างชนิดข้อมูลของการประกาศตัวแปร ID, Name[20], Surname[20] และ GPA โดยจะไม่กำหนดค่าคงที่หรือระบุค่าคงที่ ให้กับตัวแปรที่สร้างขึ้นภายในข้อมูลโครงสร้าง จะทำการระบุข้อมูลทางอินพุตเมื่อมีการเรียกใช้งาน

ในการสร้างตัวแปรชนิดโครงสร้าง student สามารถระบุตัวแปรต่อจากการเรียกใช้งานตัวแปรโครงสร้าง จะสามารถทำได้โดยการเรียกใช้งานดังนี้

struct student somsak, paitoon, anusorn ;

ดังนั้นตัวแปรทั้ง 3 ตัวแปร ได้แก่ somsak, paitoon, และ anusorn จะเป็นตัวแปรชนิดโครงสร้าง มีข้อมูลเสมือนสืบทอดจากโครงสร้าง student นั้นเอง แสดงดังโปรแกรมที่ 14.1

ตัวอย่างโปรแกรมที่ 14.1

```
1.  struct students {
2.          int ID ;  float GPA ;
3.          char Name[20] ;
4.          char Surname[20] ;
5.   }
6.  main ( )
7.  {       struct student somsak, paitoon, anusorn ;
8.  }
```

ในการกำหนดค่าข้อมูลให้ตัวแปรโครงสร้างสามารถทำได้โดย

somsak.ID = 12345 ; เป็นการระบุข้อมูลไปยังตัวแปร ID เก็บค่า 12345

somsak.GPA = 3.54 ; เป็นการระบุข้อมูลไปยังตัวแปร GPA เก็บค่า 3.54

srtcpy (somsak.Name, “Somsak”) ; เป็นการระบุข้อมูลไปยังตัวแปร Name เก็บข้อความอักษร Somsak

srtcpy (somsak.Surname, “Sangraksa”) ; เป็นการระบุข้อมูลไปยังตัวแปร Name เก็บข้อความอักษร Sangraksa

การกำหนดข้อความอักษรจะไม่สามารถกำหนดได้โดยตรง เช่น somsak.Name = “Somsak” ; จึงไม่สามารถทำได้ เนื่องจากเป็นการระบุข้อความ string จะต้องใช้คำสั่งของ strcpy เท่านั้น

ตัวอย่างโปรแกรมที่ 14.2

```
1.  #include <stdio.h>
2.  struct income {
3.                  float salary;
4.                  float bonus ;
5.                  float extra;
6.  }
7.  main( )
8.  {   struct income somsak ;
9.          somsak.bonus = 10000;
10.         somsak.salary = 25000;
11.         somsak.extra = 5000;
12.         printf("Bonus =%.2f\n",somsak.bonus);
13.         printf("Salary =%.2f\n",somsak.salary);
14.         printf("Extra = %.2f\n",somsak.extra);
15.         printf("Total Salary
    =%.2f\n",somsak.salary+somsak.extra+somsak.bonus);
16. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 14.2

Bonus = 10000.00

Salary = 25000.00

Extra = 5000.00

Total Salary = 40000.00

ตัวอย่างโปรแกรมที่ 14.3

```
1.  #include <stdio.h>
2.  #include <math.h>
3.  struct DATA {
4.          float x[10] ;
5.  } value ;
6.  main( )
7.  {       for( int i=0; i<9 ;i++)
8.          {       value.x[i] = pow(2,i) ;
9.          }
10.         for( int j=0; j<9 ;j++)
11.         {       printf("Output =%.2f\n",value.x[j] );
12.         }
13.  }
```

ผลลัพธ์การทำงานโปรแกรมที่ 14.3

Output = 1.00

Output = 2.00

Output = 4.00

Output = 8.00

Output = 16.00

Output = 32.00

Output = 64.00

Output = 128.00

Output = 256.00

ตัวอย่างโปรแกรมที่ 14.4

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  main( )
4.  {       struct DATA {
5.                  char data [20] ;
6.                  int x ;
7.                  } name[10] ;
8.          strcpy(name[2].data, "Computer“ );
9.          strcpy(name[0].data, "for" );
10.         strcpy(name[5].data, "Engineer" );
11.         printf("%s %s %s \n", name[2].data, name[0].data, name[5].data) ;
12. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 14.4

Computer for Engineer

จากตัวอย่างโปรแกรมที่ 14.4 ตัวแปรโครงสร้าง DATA จะมีตัวแปรภายในโครงสร้างที่ประกาศตัวแปรเอาไว้ใช้งาน และการประกาศตัวแปรชนิดโครงสร้างจะมีตัวแปร name[10] หมายถึงมีอาเรย์ name[0] ถึง name[9] ที่สามารถเก็บข้อมูลได้เหมือนกัน โดยการเข้าถึงตำแหน่งของอาเรย์จะต้องระบุตำแหน่งของอาเรย์และตามด้วยข้อมูลของโครงสร้างที่ต้องการเก็บค่าข้อมูล ในการแสดงผลหรือการนำไปใช้งานก็ทำเช่นเดียวกับการเก็บข้อมูล

สรุปท้ายบท

ตัวแปรชนิดโครงสร้าง structure เป็นการสร้างตัวแปรที่สามารถจัดเก็บกลุ่มข้อมูล ที่มีความหลากหลายของตัวแปร ภายในตัวแปรชนิดโครงสร้างนั้นเอง ดังนั้นเมื่อมีการแก้ไขโครงสร้างข้อมูลหลัก จะทำให้ข้อมูลมีประแก้ไขหรือปรับเปลี่ยนทั้งหมด จึงเป็นข้อดีเมื่อมีการเพิ่มเติมกลุ่มข้อมูล หรือลบกลุ่มข้อมูล สามารถทำได้ในโครงสร้างข้อมูลแหล่งเดียว เพื่อง่ายต่อการแก้ไขและเปลี่ยนแปลงเพิ่มเติมข้อมูล โดยทั่วไปแล้วตัวแปรโครงสร้างจะมีการใช้งานอย่างแพร่หลาย สำหรับการสร้างฐานข้อมูล การเก็บข้อมูลให้เป็นสัดส่วน และการจัดกลุ่มข้อมูลให้สอดคล้องกันอีกด้วย ด้วยความยืดหยุ่นของการสร้างตัวแปรชนิดนี้ ทำให้ง่ายต่อการใช้งานในการข้อมูล แสดงดังรูปแบบโครงสร้างดังนี้

```
struct ชื่อโครงสร้าง {
                การประกาศตัวแปร ภายในชนิดโครงสร้าง ตามเงื่อนไขของการประกาศตัวแปร
} ตัวแปรโครงสร้าง ;
```

คำถามท้ายบท

1. ตัวแปรชนิดโครงสร้างสามารถสร้างกลุ่มข้อมูลชนิดใดบ้าง?
2. ประโยชน์ของการสร้างตัวแปรชนิดโครงสร้าง มีอย่างไรบ้าง?
3. การเข้าถึงตัวแปรภายในโครงสร้างต้องใช้สัญลักษณ์ใด?
4. การสร้างกลุ่มตัวแปรภายในชนิดโครงสร้างข้อมูลสามารถกำหนดค่าคงที่ได้หรือไม่?
5. ชื่อโครงสร้างและชื่อตัวแปรชนิดโครงสร้าง แตกต่างกันอย่างไร?
6. กำหนดให้ struct student{ char gender[10] , float tall, float weight } มีข้อมูลภายในโครงสร้างทั้งหมดกี่จำนวน?
7. กำหนดให้ struct Data{ char a[10] } z[5] ; ตัวแปร z เก็บข้อมูลได้สูงสุดกี่จำนวน?
8. หากต้องการกำหนดข้อความอักษรให้กับตัวแปร char food[10] ภายใต้ตัวแปรโครงสร้าง Dinnerfood จะต้องใช้รูปแบบใด?
9. กำหนดให้ Data.x = 10 และ Data.y = 5 หากต้องการผลรวมของ x และ y ต้องเขียนคำสั่งอย่างไร?
10. กำหนดให้ Temp1.dataX = 27.3 ; Temp2.dataY = 28.3 ; Temp3.dataZ = 26.3 ; ตัวแปรชนิดโครงสร้างมีกี่จำนวน? และตัวแปรภายในโครงสร้างมีกี่จำนวน?

แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรม สำหรับเก็บข้อมูลหนังสือจำนวน 5 เล่ม โดยให้ผู้ใช้กรอกข้อมูลของหนังสือ ทั้ง 5 เล่ม มีรายละเอียดดังนี้
    - ชื่อหนังสือ (Book)
    - ชื่อผู้แต่ง (Author)
    - ปีที่พิมพ์ (Year)
    - จำนวนหน้า (Page)
    - ราคา (Price)

2. จงเขียนโปรแกรม สำหรับเก็บข้อมูลเกรดเฉลี่ยของนักศึกษา (GPA) จำนวน 20 คน โดยให้เก็บข้อมูลของนักเรียนเป็นแบบอาเรย์ โดยใช้ For loop ในการเก็บข้อมูลเกรดเฉลี่ย

3. จงเขียนโปรแกรม สำหรับเก็บข้อมูลรถยนต์ SuperCAR (รถยนต์ส่วนบุคคล) จำนวน 10 คัน โดยกำหนดข้อมูลดังต่อไปนี้
    - ยี่ห้อ (Brand)
    - ปีที่ผลิต (Year)
    - แรงม้าเครื่องยนต์ (Horsepower)
    - สีรถ (Color)
    - ความเร็วสูงสุด (Speed)

แผนการสอน (Lesson Plan)

วิชา EGR205 โปรแกรมคอมพิวเตอร์สำหรับวิศวกร

สัปดาห์ที่ 15 สรุปและทบทวนเนื้อหา

วัตถุประสงค์เชิงพฤติกรรม

- เพื่อให้นักศึกษาเข้าใจหลักการทำงานของภาษาซีและรูปแบบคำสั่งต่างๆ ในการแก้ไขปัญหาด้วยคอมพิวเตอร์
- เพื่อพัฒนาแนวคิด การประมวณผล การตัดสินใจ ให้มีระบบและแบบแผน

เนื้อหา

- ทบทวนเนื้อหา อาเรย์ ฟังก์ชั่น พอยน์เตอร์ สตริง และตัวแปรโครงสร้าง

กิจกรรมการสอน/การดำเนินการและกิจกรรม

- อธิบายสรุป เนื้อหา อาเรย์ ฟังก์ชั่น พอยน์เตอร์ สตริง และตัวแปรโครงสร้าง
- บรรยายเนื้อหา พร้อมยกตัวอย่างประกอบ
- ทำแบบฝึกหัด

สื่อการสอน

- เอกสารประกอบการสอนในรูปแบบสื่ออิเล็กทรอนิกส์ และสื่อออนไลน์
- เอกสารประกอบการสอน
- ระบบ e-learning
- อธิบายประกอบบนกระดานดำ

งานที่มอบหมาย

- ให้นักศึกษาทบทวนบทเรียน ก่อนสอบวัดผล
- ให้ทำแบบฝึกหัดท้ายบท ก่อนสอบวัดผล

การบรรยายสัปดาห์ที่ 15

สรุปและทบทวนเนื้อหา

Conclusions and Discussions

ในการแก้ไขปัญหาทางด้านวิศวกรรม โดยใช้โปรแกรมวิเคราะห์และหาผลลัพธ์การทำงานของระบบนั้นๆ ทำให้มีประสิทธิ์ภาพการทำงาน รวดเร็ว และถูกต้องแม่นยำ ซึ่งโปรแกรมที่ใช้งานในปัจจุบันมีมากมายหลากหลายประเภท ขีดความสามารถการทำงานแต่ต่างกัน ใช้งานจะทำงานได้แตกต่างกันไป ขึ้นอยู่กับว่าจะใช้โปรแกรมไปแก้ไขในงานชนิดใด

การเขียนโปรแกรมภาษาซี มีการพัฒนามาอย่างต่อเนื่อง เป็นพื้นฐานการพัฒนาไปสู่อีกหลายๆ โปรแกรมที่มีใช้งานในปัจจุบัน โดยคำสั่งพื้นฐานที่ได้ศึกษาใช้งานทั้งหมดนั้นสามารถนำมาประยุกต์ใช้งานได้หลากหลายรูปแบบ ขึ้นอยู่กับการใช้งานประเภทใด เช่น ในการวิเคราะห์หาผลลัพธ์การทำงานของระบบเบรกรถยนต์ระบบ ABS ที่ถูกติดตั้งในรถยนต์ ระบบควบคุมการทำงานของหุ่นยนต์ในอุตสาหกรรม ระบบป้องกันภัย เป็นต้น

การใช้โปรแกรมภาษาซีเข้ามาแก้ไขปัญหา สามารถนำคำสั่งต่างที่ได้ศึกษามาประยุกต์ใช้งาน สามารถนำมาใช้แก้ปัญหาได้ไม่มาก ขึ้นอยู่กับผู้ใช้งานหมั่นฝึกฝนคำสั่งต่างๆ และแสวงหาความรู้ใหม่ๆ จะสามารถเขียนโปรแกรมภาษาซี ได้อย่างคล่องแคล่ว และจะสนุกกับการเขียนโปรแกรม ที่มีความซับซ้อนมากยิ่งขึ้น การเขียนโปรแกรมได้ดีและถูกต้อง จึงต้องอาศัยการฝึกฝนอย่างสม่ำเสมอ อย่างต่อเนื่อง จึงจะประสบผลสำเร็จในการใช้งานอย่างดีเยี่ยม

ในเนื้อหาที่ได้ศึกษามาทั้งหมด เป็นเพียงคำสั่งพื้นฐานทั่วไปที่ควรจะศึกษาในระดับเริ่มต้น ในการเขียนภาษาซี นั้นยังมี คำสั่งอีกมากมายที่เราจะต้องศึกษาเพิ่มเติม ยกตัวอย่างเช่น การทำงานระดับ Hardware ติดต่อกับ Input/output port ของคอมพิวเตอร์ เพื่อทำการควบคุมอุปกรณ์ต่างๆ, การทำงานติดต่อข้อมูลสื่อสารผ่านระบบ Network, การทำงานเก็บฐานข้อมูล Database ฯลฯ เป็นต้น ซึ่งภาษาซีนั้นมาความยืดหยุ่นในการทำงานร่วมกับโปรแกรมอื่นๆ เช่น word, excel, Access, SQL เป็นต้น การทำงานของภาษาซีนั้นสามารถทำงานได้หลากหลายขึ้นอยู่ผู้ใช้จะนำไปใช้งานให้มีประโยชน์ไม่มากก็น้อย

ในสรุปของบทนี้จะทำการยกตัวอย่าง การเขียนภาษาซี ในการแก้ไขปัญหา เพื่อช่วยในการทำงานในแบบต่างๆ ให้มีความถูกต้องแม่นยำและรวดเร็ว เพื่อลดความผิดพลาดที่อาจจะเกิดขึ้น ดังตัวอย่างดังต่อไปนี้

ตัวอย่างที่ 15.1 จงเขียนโปรแกรม ให้เปลี่ยนมุมองศาให้เป็นมุมเรเดียนของ sin ,cos, tan โดยกำหนดให้องศาเริ่มต้นที่ $0°$ จนถึง $360°$ โดยเพิ่มที่ละ $5°$

โปรแกรมตัวอย่างที่ 15.1

```
1.  #include <stdio.h>
2.  void main( )
3.  {       float  a , b, c , pi = 3.141 ; int i ;
4.          for (i=0 ; i<=360 ; i+=5)
5.          {       a = sin (i * pi / 180) ;
6.                  b = cos (i * pi / 180)  ;
7.                  c = tan (i * pi / 180)  ;
8.                  printf("\tsin(%d)= %f",i,a);
9.                  printf("\tcos(%d)= %f",i,b);
10.                 printf("\ttan(%d)= %f\n",i,c);
11.         }
12. }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 15.1

```
sin(0)= 0.000000    cos(0)= 1.000000    tan(0)= 0.000000
sin(5)= 0.087139    cos(5)= 0.996196    tan(5)= 0.087472
sin(10)= 0.173616   cos(10)= 0.984813    tan(10)= 0.176293
sin(15)= 0.258771   cos(15)= 0.965939    tan(15)= 0.267896
sin(20)= 0.341958   cos(20)= 0.939715    tan(20)= 0.363896
sin(25)= 0.422544   cos(25)= 0.906343    tan(25)= 0.466207
sin(30)= 0.499914   cos(30)= 0.866075    tan(30)= 0.577219
sin(35)= 0.573482   cos(35)= 0.819218    tan(35)= 0.700036
sin(40)= 0.642687   cos(40)= 0.766129    tan(40)= 0.838875
sin(45)= 0.707002   cos(45)= 0.707212    tan(45)= 0.999704
sin(50)= 0.765939   cos(50)= 0.642914    tan(50)= 1.191355
sin(55)= 0.819048   cos(55)= 0.573725    tan(55)= 1.427598
sin(60)= 0.865927   cos(60)= 0.500171    tan(60)= 1.731261
sin(65)= 0.906217   cos(65)= 0.422812    tan(65)= 2.143309
sin(70)= 0.939614   cos(70)= 0.342237    tan(70)= 2.745508
```

sin(75)= 0.965862	cos(75)= 0.259058	tan(75)= 3.728368
sin(80)= 0.984762	cos(80)= 0.173908	tan(80)= 5.662560
sin(85)= 0.996170	cos(85)= 0.087435	tan(85)= 11.393329
sin(90)= 1.000000	cos(90)= 0.000296	tan(90)= 3374.837158
sin(95)= 0.996222	cos(95)= -0.086844	tan(95)= -11.471375
sin(100)= 0.984865	cos(100)= -0.173324	tan(100)= -5.682221
sin(105)= 0.966015	cos(105)= -0.258485	tan(105)= -3.737218
sin(110)= 0.939816	cos(110)= -0.341680	tan(110)= -2.750576
sin(115)= 0.906468	cos(115)= -0.422275	tan(115)= -2.146628
sin(120)= 0.866223	cos(120)= -0.499658	tan(120)= -1.733632
sin(125)= 0.819388	cos(125)= -0.573239	tan(125)= -1.429400
sin(130)= 0.766320	cos(130)= -0.642460	tan(130)= -1.192790
sin(135)= 0.707421	cos(135)= -0.706792	tan(135)= -1.000889
sin(140)= 0.643141	cos(140)= -0.765748	tan(140)= -0.839885
sin(145)= 0.573967	cos(145)= -0.818878	tan(145)= -0.700919
sin(150)= 0.500428	cos(150)= -0.865778	tan(150)= -0.578009
sin(155)= 0.423081	cos(155)= -0.906092	tan(155)= -0.466929
sin(160)= 0.342515	cos(160)= -0.939512	tan(160)= -0.364567
sin(165)= 0.259344	cos(165)= -0.965785	tan(165)= -0.268532
sin(170)= 0.174199	cos(170)= -0.984710	tan(170)= -0.176904
sin(175)= 0.087730	cos(175)= -0.996144	tan(175)= -0.088069
sin(180)= 0.000593	cos(180)= -1.000000	tan(180)= -0.000593
sin(185)= -0.086549	cos(185)= -0.996248	tan(185)= 0.086875
sin(190)= -0.173032	cos(190)= -0.984916	tan(190)= 0.175682
sin(195)= -0.258199	cos(195)= -0.966092	tan(195)= 0.267261
sin(200)= -0.341401	cos(200)= -0.939918	tan(200)= 0.363225
sin(205)= -0.422006	cos(205)= -0.906593	tan(205)= 0.465486
sin(210)= -0.499401	cos(210)= -0.866371	tan(210)= 0.576429
sin(215)= -0.572996	cos(215)= -0.819558	tan(215)= 0.699153
sin(220)= -0.642233	cos(220)= -0.766510	tan(220)= 0.837866
sin(225)= -0.706583	cos(225)= -0.707630	tan(225)= 0.998520
sin(230)= -0.765557	cos(230)= -0.643368	tan(230)= 1.189923

sin(235)= -0.818708	cos(235)= -0.574210	tan(235)= 1.425799
sin(240)= -0.865630	cos(240)= -0.500684	tan(240)= 1.728894
sin(245)= -0.905967	cos(245)= -0.423349	tan(245)= 2.139998
sin(250)= -0.939411	cos(250)= -0.342793	tan(250)= 2.740457
sin(255)= -0.965708	cos(255)= -0.259630	tan(255)= 3.719557
sin(260)= -0.984659	cos(260)= -0.174491	tan(260)= 5.643031
sin(265)= -0.996118	cos(265)= -0.088025	tan(265)= 11.316329
sin(270)= -1.000000	cos(270)= -0.000889	tan(270)= 11.249454
sin(275)= -0.996273	cos(275)= 0.086254	tan(275)= -11.550490
sin(280)= -0.984967	cos(280)= 0.172740	tan(280)= -5.702014
sin(285)= -0.966168	cos(285)= 0.257913	tan(285)= -3.746107
sin(290)= -0.940019	cos(290)= 0.341123	tan(290)= -2.755661
sin(295)= -0.906718	cos(295)= 0.421738	tan(295)= -2.149956
sin(300)= -0.866519	cos(300)= 0.499144	tan(300)= -1.736008
sin(305)= -0.819728	cos(305)= 0.572754	tan(305)= -1.431205
sin(310)= -0.766700	cos(310)= 0.642005	tan(310)= -1.194227
sin(315)= -0.707840	cos(315)= 0.706373	tan(315)= -1.002076
sin(320)= -0.643594	cos(320)= 0.765367	tan(320)= -0.840897
sin(325)= -0.574453	cos(325)= 0.818538	tan(325)= -0.701803
sin(330)= -0.500941	cos(330)= 0.865482	tan(330)= -0.578800
sin(335)= -0.423618	cos(335)= 0.905841	tan(335)= -0.467651
sin(340)= -0.343072	cos(340)= 0.939309	tan(340)= -0.365238
sin(345)= -0.259916	cos(345)= 0.965631	tan(345)= -0.269167
sin(350)= -0.174783	cos(350)= 0.984607	tan(350)= -0.177515
sin(355)= -0.088320	cos(355)= 0.996092	tan(355)= -0.088667
sin(360)= -0.001185	cos(360)= 0.999999	tan(360)= -0.001185

ตัวอย่างที่ 15.2 จงหาผลลัพธ์ Cross product $u \times v$ ของเวกเตอร์ u = 2i + j + k และ v = - 4i + 3j + k

ถ้า $u = u_1 i + u_2 j + u_3 k$ และ $v = v_1 i + v_2 j + v_3 k$

$$u \times v = \begin{bmatrix} i & j & k \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix}$$

โปรแกรมตัวอย่างที่ 15.2

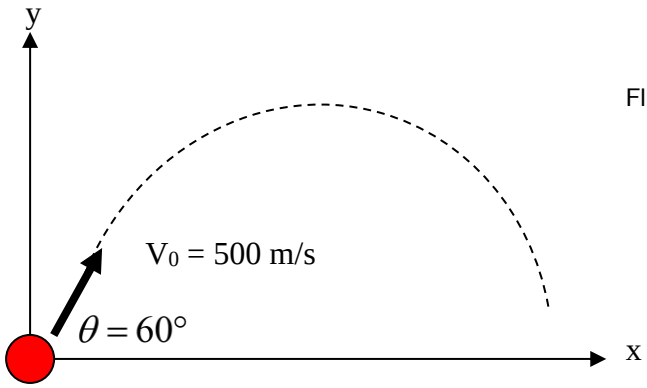
```
1.  #include <stdio.h>
2.  void main( )
3.  {       int u[3] = { 2, 1, 1 };
4.          int v[3] = { -4, 3, 1 } ;
5.          int y[3] ;
6.          int i = 0 , j = 1, k = 2 ;
7.          y[i] =(u[1] * v[2])- (v[1] * u[2]) ;
8.          y[j] =-((u[0] * v[2])- (v[0] * u[2])) ;
9.          y[k] =(u[0] * v[1])- (v[0] * u[1]) ;
10.         printf("i=%d\tj=%d\tk=%d\n",y[i],y[j],y[k]);
11. }
```

ผลลัพธ์จากทำงานของโปรแกรมที่ 15.2

i = -2 j = 6 k = 10 ดังนั้น $u \times v = -2i - 6j + 10k$

ตัวอย่างที่ 15.3 จากรูปการเคลื่อนที่โปรเจกต์ไทป์ (Projectile) วัตถุอยู่ที่ตำแหน่งเริ่มต้น มีความเร็วต้นเท่ากับ 500 m/sec ทำมุม $\theta = 60°$ **จงเขียนฟังก์ชั่นย่อย**ของโปรแกรมคำนวณหา จุดสูงสุด (y_{max}) , เวลาที่ใช้ทั้งหมด (t) และระยะทางตกกระทบถึงพื้น (R)

กำหนดให้



Maximum height $y_{max} = \frac{(v_0 \sin\theta)^2}{2g}$

Flight time $t = \frac{2v_0 \sin\theta}{g}$

Range $R = \frac{v_0^{\ 2}}{g} \sin 2\theta$

โปรแกรมตัวอย่างที่ 15.3

```
1.  #include <stdio.h>
2.  #include <math.h>
3.  float  g = 9.81 ;
4.  float  pi = 3.141 ;
5.  float ymax(int v , int theta)
6.  {   float z ;
7.          z =pow( (v * sin(theta * pi /180) ), 2) / (2*g) ;
8.          return(z);
9.  }
10. float time(int v , int theta)
11. {       float z ;
12.         z = ( 2*v*sin(theta * pi /180) )/ g ;
13.         return(z);
14. }
15. float rang(int v , int theta)
16. {       float  z ;
17.         z = (pow(v,2) / g) *sin(2*theta*pi/180) ;
18.         return(z) ;
```

```
19.  }
20.  void main( )
21.  {       int v = 500 , theta = 60 ;
22.          float Z1 , Z2 , Z3 ;
23.          Z1 = ymax( v , theta ) ;
24.          Z2 = time( v , theta ) ;
25.          Z3 = rang( v , theta ) ;
26.                  printf("ymax = %f \n time = %f \n range = %f  \n" , Z1, Z2, Z3) ;
27.  }
```

ผลลัพธ์การทำงานของโปรแกรมที่ 15.3

ymax = 9554.394531

time = 88.269783

range = 2204.995910

ตัวอย่างที่ 15.4 จงเขียนโปรแกรมและผังงาน สำหรับฟังก์ชั่นหลักและฟังก์ชั่นย่อย ตามข้อกำหนดต่อไปนี้

ข้อกำหนดให้

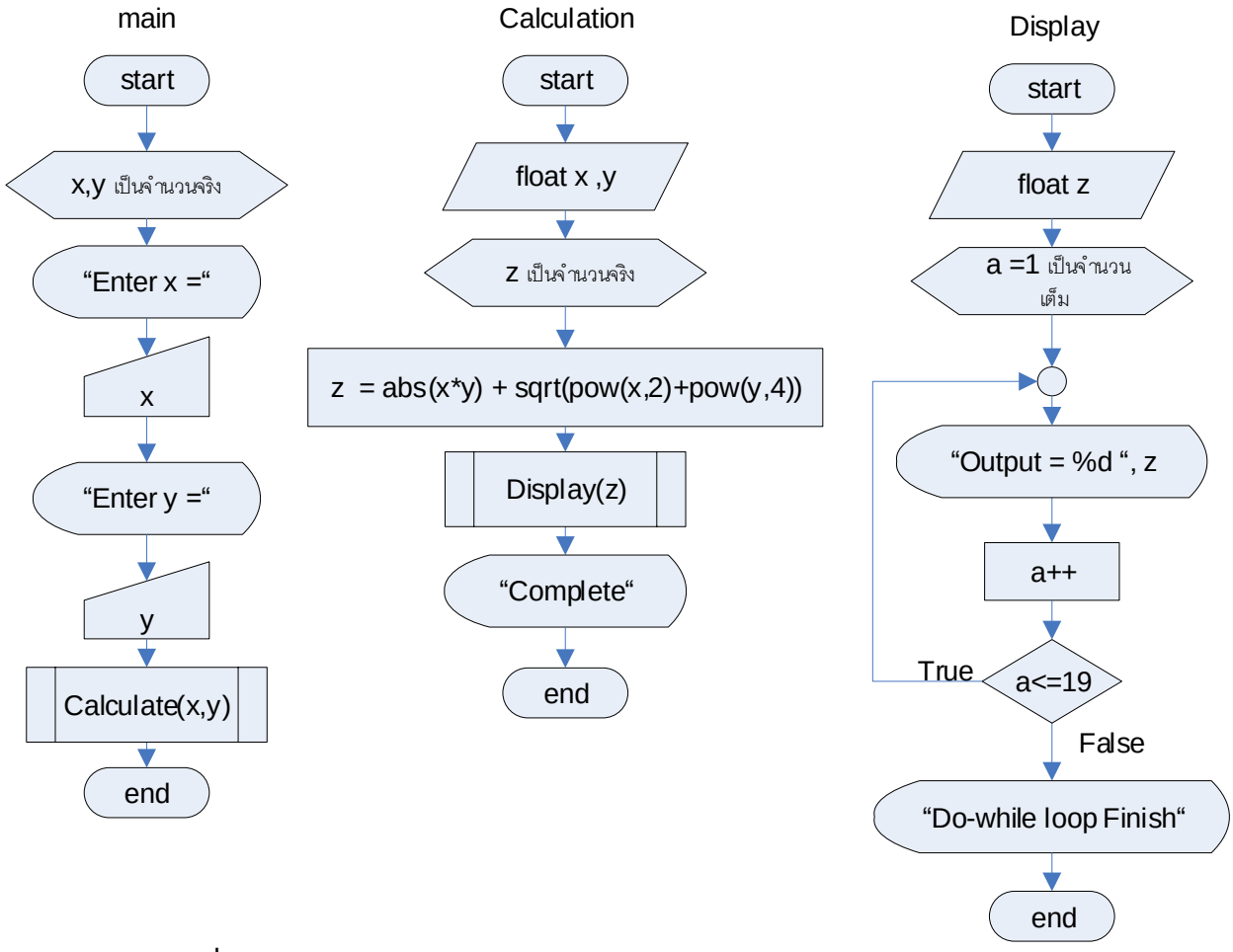
a) ฟังก์ชั่นย่อย **Calculate** ทำหน้าที่รับค่าอินพุทสองจำนวน จากฟังก์ชั่นหลัก เพื่อทำการคำนวณหาผลลัพธ์ของ z เรียกใช้งานฟังก์ชั่นย่อย **Display** ทำการแสดงผลลัพธ์ และแสดง ข้อความ **"Complete"** เมื่อฟังก์ชั่นย่อย Display ทำงานเสร็จสิ้น

สมการ $z = |xy| + \sqrt{x^2 + y^4}$

b) ฟังก์ชั่นย่อย **Display** ทำหน้าที่รับค่าอินพุท จากผลลัพธ์ของฟังก์ชั่น Calculate เพื่อทำการแสดงผลลัพธ์ โดยแสดงผลลัพธ์ทั้งหมดจำนวน **19 ครั้ง** โดยใช้วิธี **Do-while loop** และแสดงข้อความ **"Do-while Loop Finish"** เมื่อเสร็จสิ้นการทำงานของลูป

c) ฟังก์ชั่นหลัก ทำหน้าที่รับค่าทางคีย์บอร์ดสองจำนวน โดยเรียกใช้งานฟังก์ชั่นย่อย **Calculate**

ผังงานโปรแกรมตัวอย่างที่ 15.4

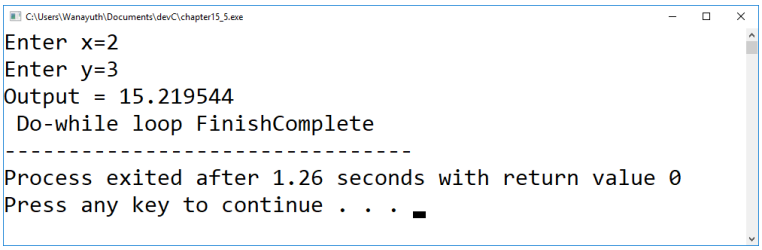


โปรแกรมตัวอย่างที่ 15.4

```
1.  #include <stdio.h>
2.  #include <math.h>
3.  void calculate (float x, float y)
4.  {  float z ;
5.                  z = abs(x*y) + sqrt(pow(x,2)+pow(y,4));
6.                  display(z);
7.                  printf(“Complete”);
8.  }
9.  void display (float z)
10. {    int a = 1 ;
11.      do
12.      {
13.                  printf(“Output = %f\n”,z);
```

```
14.             a++ ;
15.     } while(a<=19) ;
16.     printf(“ Do-while loop Finish”);
17. }
18.  main( )
19. {   float x ,y ;
20.             printf(“Enter x=”) ;
21.             scanf(“%f”,&x) ;
22.             printf(“Enter y=”) ;
23.             scanf(“%f”,&y) ;
24.             calculate(x,y) ;
25. }
```

ผลลัพธ์การทำงานโปรแกรมที่ 15.4



C/C++ Open Source Software

- Software DEV C++ สามารถดาวน์โหลดได้ที่ https://sourceforge.net/projects/orwelldevcpp/

C/C++ Library

- http://www.cplusplus.com/

แบบฝึกหัดทบทวน (จำนวน 50 ข้อ)

1. Array[i][j][k] มีกี่มิติ

   a. 1 มิติ b. 2 มิติ c. 3 มิติ d. 4 มิติ

2. จาก int Array[3]={0,2,4}; จงหาค่าขอ y = Array[Array[1]]

   a. y = 0 b. y = 4 c. y = 2 d. y = 3

3. การสร้างตัวแปรอาเรย์ ข้อใดถูกต้อง

   a. int abs[10] ; b. char xyz[-1]; c. float abc[2.3][5]; d. int ijk[2][2] ;

4. ตัวแปรอาเรย์ char text[20] = "HELLO WORLD" ; จงหา text[6] มีค่าเท่าใด

   a. O b. W c. Space bar d. ไม่มีค่าใดๆ

5. จากโปรแกรมที่กำหนดให้ จงหาผลลัพธ์ตัวแปรอาเรย์ b

```c
#include <stdio.h>
main()
{
    int a[5] = {1,3,5,7,9};
    int b[5] ;
    for ( int i=0 ;i<5; i++)
    {
        b[i] = a[i] +1 ;
        printf("b[%d] = %d\n",i,b[i]);
    }

}
```

   a. [1 3 5 7 9] b. [9 7 5 3 1] c. [10 8 6 4 2] d. [2 4 6 8 10]

6. กำหนดให้ a[2][2] = {1,2,3,4} และ b[2][2] = {5,6,7,8} ; จงหาค่าผลลัพธ์ z = a[1][1] + b[1][1] / 2 ;

   a. z = 3 b. z = 8 c. z = 4 d. z = 6

7. กำหนดให้ a[5] = {1,2,3,4,5} และ b[5] = {5,4,3,2,1} ; จงหาค่าผลลัพธ์ x = a[b[4]] + b[a[1]] ;

   a. x = 5 b. x = 7 c. x = 6 d. x = 8

8. จากโปรแกรมที่กำหนดให้ จงหาผลลัพธ์ตัวแปรอาเรย์ c

```c
#include <stdio.h>
main()
{   int c[2][3];
    int a[2][3] = {1,2,3,4,5,6};
    int b[2][3] = {6,5,4,3,2,1};
    for ( int i=0 ;i<2; i++)
    {   for ( int j=0 ;j<3; j++)
        {
            c[i][j] =  b[i][j] - a[i][j] ;
            printf("c[%d][%d] = %d\n",i,j,c[i][j]);
        }
    }
}
```

a. [-5 3 1 -1 3 5] b. [-5 -3 -1 1 3 5] c. [5 3 1 -1 -3 -5] d. [-5 -3 -1 -1 -3 -5]

9. ตัวแปรอาเรย์ในข้อใดสามารถเข้าถึงข้อมูลได้ในตำแหน่ง Array[43][52]

a. int Array[60][10] b. int Array[100] c. int Array[43][52] d. int Array[50][60]

10. Array[a][b] ข้อใดอธิบายได้ถูกต้อง

a. ตัวแปรอาเรย์ 2 มิติ สามารถเก็บข้อมูลได้สูงสุด axb จำนวน

b. ตัวแปรอาเรย์ 2 มิติ สามารถเก็บข้อมูลได้สูงสุด a จำนวน และสำรอง b จำนวน

c. ตัวแปรอาเรย์ 2 มิติ สามารถเก็บข้อมูล a จำนวนหลัก b จำนวนแถว

d. ตัวแปรอาเรย์ 2 มิติ สามารถเก็บข้อมูลได้สูงสุด a^b จำนวน

11. ข้อใดถูกต้อง

a. void หมายถึงการประกาศตัวแปรชนิดพิเศษ ascii ของฟังก์ชั่น

b. void หมายถึงการระบุไม่มีค่าใดๆในส่วนของอินพุตและเอาท์พุตของฟังก์ชั่น

c. void หมายถึงการกำหนดค่าให้กับส่วนของอินพุตและเอาท์พุตของฟังก์ชั่น

d. void หมายถึงการสร้างตัวแปรแบบฟังก์ชั่นชนิด void

12. ข้อใดถูกต้อง

a. return ในฟังก์ชั่นทำหน้าที่การส่งข้อมูลออกจากฟังก์ชั่น

b. return ในฟังก์ชั่นทำหน้าที่การรับส่งข้อมูลและแจ้งเตือน

c. return ในฟังก์ชั่นทำหน้าที่รับค่าจากอินพุตจากฟังก์ชั่น

d. return ในฟังก์ชั่นทำหน้าที่คำนวณข้อมูลจากฟังก์ชั่น

13. ข้อใดถูกต้อง

a. voids calculate(int a, int b)

b. return calculate(int a, int b)

c. int calculate(int a, int b)

d. void calculate(int a, b)

14. หากประกาศฟังก์ชั่นย่อย ในส่วนหัวของโปรแกรม (Header File) เรียกว่าอะไร

a. Include file

b. Global parameter

c. Prototype function

d. Local parameter

15. เมื่อฟังก์ชั่นย่อยใดๆ ที่มีการส่งค่าออกจากฟังก์ชั่นทางเอาท์พุต ต้องข้อใดถูกต้อง

a. void b. return c. argument d. parameter

16. ข้อใดไม่ถูกต้อง

a. void calculate(int a, float b, float c)

b. void calculate(int a[0] , int b , int c)

c. void calculate(float a, float b, int c)

d. void calculate(char a, void, float c)

17. ตัวแปรที่ประกาศภายในฟังก์ชั่นเรียกว่าอะไร

a. Local Variable b. Global Variable c. Sub-Function d. Argument

18. ข้อใดถูกต้อง

a. ฟังก์ชั่นหลักสามารถสร้างได้มากกว่า 1 ฟังก์ชั่น โดยเรียกใช้งานฟังก์ชั่นย่อยใดๆ

b. ฟังก์ชั่นหลักสามารถสร้างได้ 1 ฟังก์ชั่นเท่านั้น โดยเรียกใช้งานฟังก์ชั่นย่อยใดๆ

c. ฟังก์ชั่นหลักสามารถสร้างได้มากกว่า 1 ฟังก์ชั่นเท่านั้น โดยฟังก์ชั่นย่อยใดๆ สามารถเรียกใช้งานได้

d. ฟังก์ชั่นหลักสามารถสร้างได้ 1 ฟังก์ชั่นเท่านั้น โดยเรียกใช้งานได้ทั้งฟังก์ชั่นหลักและฟังก์ชั่นย่อยใดๆ

19. จงหาผลลัพธ์ของ z เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
int z = 5 ;
int sum(int x,int y)
{   int z = x + y ;
    return (z);
}
main()
{ int a=5, b=6 ;
    int z = sum(a,b);
    printf("z=%d\n",z);
}
```

a. z=5 b. z=6 c. z=11 d. z=30

20. จงหาผลลัพธ์เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
#include <math.h>
float calculate(int a,int b,int c)
{   float z = 0.5 * (a + b) * c ;
    return (z);
}
main()
{ int a=4, b=5, c=6 ;
    float z = calculate(a,b,c);
    printf("Output=%.2f\n",z);
}
```

a. Output= 0.00 b. Output=10.00 c. Output=27.00 d. Output=60.00

21. ข้อใดถูกต้อง

a. ตัวแปร pointer ทำหน้าที่เก็บข้อความอักษรของตำแหน่งที่กำหนด

b. ตัวแปร pointer ทำหน้าที่ระบุตำแหน่งของตัวแปรที่กำหนด

c. ตัวแปร pointer ทำหน้าที่เก็บค่าตัวแปรของตำแหน่งที่กำหนด

d. ตัวแปร pointer ทำหน้าที่แสดงเลขฐานสิบหกของตำแหน่งที่กำหนด

22. ข้อใดประกาศตัวแปรถูกต้อง

a. int &pointer, &abc ;

b. int *pointer, *abc ;

c. int **pointer, *abc ;

d. int *pointer *abc ;

23. ข้อใดเป็นการกำหนดที่ถูกต้อง กำหนดให้ตัวแปร int z, *pz ;

a. int *pz = $z ;

b. int *pz = %z ;

c. int *pz = &z ;

d. int *pz = z ;

24. จงหาผลลัพธ์ x = *(pa+=2) เมื่อกำหนดให้ int a[5] = {1,2,3,4,5} ; *pa = a ;

a. x = 2

b. x = 3

c. x = 4

d. x = 5

25. จงหาผลลัพธ์ x = (*pb)++ เมื่อกำหนดให้ int b[3][3] = {1,2,3,4,5,6,7,8,9} ; *pb = &b[2][2] ;

a. x = 7

b. x = 8

c. x = 9

d. x = 10

26. ข้อใดถูกต้อง สำหรับการแสดงข้อมูลเลขฐาน 16 ของตัวแปรชนิดพอยเตอร์ *out

a. printf(" %s\n", &out) ;

b. printf(" %e\n", &out) ;

c. printf(" %h\n", &out) ;

d. printf(" %p\n", &out) ;

27. ข้อใดเป็นการแสดงผลลัพธ์ถูกต้อง เมื่อ int *px =&x , float *py =&y และ char *pz = &z

a. printf("%d %f %c\n", *px, *py, *pz) ;

b. printf("%d %f %c\n", &px, &py, &pz) ;

c. printf("%d %f %c\n", $px, $py, $pz) ;

d. printf("%e %e %e\n", px, py, pz) ;

28. ข้อใดกล่าวผิด ของตัวแปรชนิดพอยเตอร์

a. ลดปริมาณการใช้หน่วยความจำของคอมพิวเตอร์

b. สามารถเข้าถึงตัวแปรการคำนวณได้อย่างรวดเร็ว

c. ตัวแปรมีขนาดเล็กและประหยัดหน่วยความจำคอมพิวเตอร์

d. ตัวแปรมีความโดดเด่น ด้วยเครื่องหมาย *

29. จงหาผลลัพธ์ของ y เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
main ( )
{    int a[10] ={1,2,3,4,5,6,7,8,9,10} ;
     int *pa[10] , y;
     for ( int i=0 ; i<10 ; i++ )
     {    pa[i] = &a[i] ;
     }
          y =  ++(*(pa[4]+3) ) + ++(*pa[1]+=2) ;
          printf("y= %d\n",y);
}
```

a. y=12 b. y=13 c. y=14 d. y=16

30. จงหาผลลัพธ์ Q เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
main ( )
{    int a[5] ={1,2,3,4,5} ;
     int b[5] ={1,2,3,4,5} ;
     int *pa = a, *pb = b ;
     int  Q ;

     Q =  ++(*(pa+2) ) + ++*(pb+4) ;

     printf("Q= %d\n",Q);
}
```

a. Q=3 b. Q=5 c. Q=9 d. Q=10

31. ข้อใดถูกต้อง

a. ตัวแปร string ทำหน้าที่เก็บข้อความอักษรหรือประโยคอักษรที่กำหนด

b. ตัวแปร string ทำหน้าที่เก็บตัวอักษรและค่าจำนวนเต็มของตัวแปรที่กำหนด

c. ตัวแปร string ทำหน้าที่เก็บค่า ASCII ของตัวแปรที่กำหนด

d. ตัวแปร string ทำหน้าที่เก็บค่าอักษรและเลขฐานสิบหกตัวแปรที่กำหนด

32. ข้อใดประกาศตัวแปรถูกต้อง

a. char a[10] ="Hello World" ;

b. char a[11] = 'Hello World' ;

c. char a[11] = Hello World ;

d. char a[12] ="Hello World" ;

33. ข้อใดคือการบ่งบอกตำแหน่งสุดท้ายของการเก็บข้อมูลของตัวแปร string

a. \0 b. \n c. \p d. \s

34. จงหาผลลัพธ์ int z = strlen(x) เมื่อ char x[ ] = "Engnieering\n" ;

a. z = 10 b. z = 11 c. z = 12 d. z = 13

35. จงหาผลลัพธ์ x = strlen(a)-1 + strlen(b)-1 เมื่อ char a[ ] = "University" ; char b[ ] ="Sripatum" ;

a. x = 18 b. x = 17 c. x = 16 d. x = 15

36. ข้อใดถูกต้อง กำหนดให้ char a[50] ;

a. strcpy(a , "printf(Hello\n)") ;

b. strcpy(a , 'Engineering') ;

c. strcpy(a = "Sripatum") ;

d. strcpy(a = "Welcome"\n) ;

37. ข้อใดเป็นการแสดงผลลัพธ์ถูกต้อง กำหนดให้ char a[ ]="EGR205" ; char b[ ]="Hello"; char c[ ]="World" ;

a. printf("%c %c %c\n", a[ ], b[ ], c[ ]) ;

b. printf("%s %s %s\n", &a, &b, &c) ;

c. printf("%s %s %s\n", a, b, c) ;

d. printf("%c %c %c\n", 'a', 'b', 'c') ;

38. คำสั่งข้อใดถูกต้องในการกำหนดค่า b = a โดยกำหนดให้ ตัวแปร a เก็บข้อความ "Program Computer for Engineering"

a. b = a ; b. strcpy(a, b) ; c. b = strcpy(a,b) ; d. strcpy(b, a) ;

39. จงหาผลลัพธ์ของตัวแปร z เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
#include <string.h>
main()
{   char x[]="Programming" ;
    char y[]="Engineering" ;
    char z[50] ;
    strcpy(z,y);
    strcpy(z,x);
    strcat(z," for ");
    strcpy(z,y);
    puts(z);
}

```

a. Programming for Engineering b. Engineering c. Programming d. Engineer

40. จงหาผลลัพธ์ของตัวแปร c เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
#include <string.h>
main()
{   char a[]="Sripatum" ;
    char c[50] ;
    for(int i=0;i<strlen(a);i++)
    {
        c[strlen(a)-i]= a[i];
    }

    puts(c);

}

```

a. Sripatum b. Sritumpa c. mutapirS d. mutSirpa

41. ข้อใดถูกต้อง

a. ตัวแปร structure ทำหน้าที่เก็บตัวแปรฐานข้อมูลในการคำนวณขนาดใหญ่

b. ตัวแปร structure ทำหน้าที่สร้างกลุ่มตัวแปรพื้นฐานให้เป็นโครงสร้างตัวแปรใหม่

c. ตัวแปร structure ทำหน้าที่เก็บค่าตัวแปรและระบุค่าตัวแปรที่ต้องการ

d. ตัวแปร structure ทำหน้าที่เป็นฐานข้อมูลตัวแปรชนิดข้อความอักษร

42. ข้อใดประกาศตัวแปรถูกต้อง

a. structure student data[10] ;

b. student structure data[10] ;

c. struct student data[10] ;

d. structs student data[ ] ;

43. ข้อใดถูกต้องสำหรับแสดงผลข้อมูล กำหนดให้ตัวแปรชนิดโครงสร้างชื่อ stu1 ประกอบด้วยตัวแปร int a[10], char b[10], และ int c[10]

a. printf("%d %s %d\n", stu1.a[10], stu1.b[10], stu1.c[10]) ;

b. printf("%s %s %s\n", stu1.a, stu1.b, stu1.c) ;

c. printf("%d %s %d\n", stu1.a[1], stu1.b, stu1.c[1]) ;

d. printf("%d %d %d\n", stu1.a[1], stu1.b[1], stu1.c[1]) ;

44. ข้อใดถูกต้อง ในการกำหนดข้อความอักษร "Hello World" ให้กับ char text[50] ของตัวแปรชนิดโครงสร้าง word

a. text.world = {"Hello World"};

b. word.text."Hello World" ;

c. strcpy(word.text , "Hello World") ;

d. strncpy(word.text ="Hello World" , 11) ;

45. ข้อใดประกาศตัวแปรชนิดโครงสร้างได้ถูกต้อง

a. struct struct{ int a[10] char b[10] char c[10] } ;

b. struct structures{ int a[10] char b[10] char c[10] } ;

c. structure struct { int a[10], char b[10] , char c[10] } ;

d. struct structures{ int a[10] ; char b[10] ; char c[10] ; } ;

46. ข้อใดถูกต้อง โดยกำหนดให้ int variable ; เป็นตัวแปรโครงสร้างของ name

a. sturcture.name = variable ;

b. name.variable = 100 ;

c. name.structure = variable ;

d. struct.name = 100 ;

47. ข้อใดถูกต้อง เมื่อกำหนดให้ตัวแปรโครงสร้าง data.var1 = 'Y' และ data.var2 = 'N'

a. ตัวแปร var1 และ var2 ประกาศตัวแปรชนิด char

b. ตัวแปร var1 และ var2 ประกาศตัวแปรชนิด int

c. ตัวแปร var1 และ var2 ประกาศตัวแปรชนิด float

d. ตัวแปร var1 และ var2 ประกาศตัวแปรชนิด structure

48. จงหาผลลัพธ์ z = data.var1 + data.var2 โดยกำหนดค่า strcpy(data.var1, "150") และ strcpy(data.var2, "250")

a. 400 b. 150 c. 250 d. ไม่มีข้อใดถูก

49. จงหาผลลัพธ์ เมื่อสิ้นสุดการทำงานของโปรแกรม

```c
#include <stdio.h>
struct DATA{
    int x ; int y ; int z ;
    char var1[20] ;
    char var2[10] ;
}data[5];
main()
{
    data[1].x = 5 ;
    data[1].y = 3 ;
    float z = 0.5 * ++ data[1].x * ++ data[1].y ;
    printf("Output = %f \n",z);
}

```

a. Output = 7.500000

b. Output = 12.000000

c. Output = 4.000000

d. Output = 8.000000

50. ตัวแปรชนิดโครงสร้าง data สามารถรองรับการเก็บข้อมูลได้สูงสุดกี่จำนวน

```c
#include <stdio.h>
struct DATA{
    int x ; int y ; int z ;
    char var1[20] ;
    char var2[10] ;
}data[5];

```

a. 165 b. 33 c. 30 d. ไม่มีข้อใดถูก

บรรณานุกรม

[1] ประภาพร ช่างไม้, (2545) “คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น”, นนทบุรี, อินโฟเพรส, พิมพ์ครั้งที่ 1, สิงหาคม 2545, 342 หน้า, ISBN 974-90566-7-1

[2] Brian W. Kernighan, Dennis M. Ritchie, (1988), “The C programming language second editon,” Prentice Hall, ISBN 0-13-110362-8

[3] Tim Bailey, (2005) “An Introduction to the C Programming Language and Software Design” Online: www.acfr.usyd.edu.au/homepages/academic/tbailey/index.html.

[4] Yukel Uckan, (1999), “Problem Solving Using C: Structured Programming Techniques 2nd Edition,” McGraw-Hill, ISBN 0-256-26377-9