คำนำ

เอกสารประกอบการเรียนการสอนเล่มนี้ใช้ประกอบการสอนใน รายวิชา ED-032-109 ขั้นตอนวิธีและการเขียนโปรแกรม เนื้อหาเหมาะสำหรับผู้เริ่มต้นเขียนโปรแกรม ซึ่งยังไม่มีพื้นฐานด้านนี้มาก่อนเนื้อหาถูกออกแบบให้สอดคล้องกับหลักสูตรครุศาสตรบัณฑิต สาขาวิชาคอมพิวเตอร์ ในกลุ่มรายวิชาเอกบังคับ เนื้อหาของหนังสือถูกแบ่งออกเป็น 11 บท ประกอบด้วย บทที่ 1 ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรม บทที่ 2 ขั้นตอนการทำงานของอัลกอริทึม บทที่ 3 โครงสร้างภาษา C บทที่ 4 ประเภทของข้อมูล และตัวดำเนินการ บทที่ 5 การรับและแสดงผลข้อมูล บทที่ 6 คำสั่งควบคุม บทที่ 7 คำสั่งทำซ้ำ(Loops) บทที่ 8 อาร์เรย์ (Array) บทที่ 9 พอยเตอร์ (Pointers) บทที่ 10 การจัดการข้อมูลชนิดสตริง และบทที่ 11 ข้อมูลชนิดโครงสร้าง

เมื่อนักศึกษาได้ศึกษาเอกสารประกอบการสอนเล่มนี้แล้ว จะสามารถเขียนโปรแกรมแก้โจทย์ปัญหาทางคอมพิวเตอร์ และสามารถนำความรู้ไปประยุกต์ใช้กับภาษาคอมพิวเตอร์อื่นได้ ในส่วนท้ายของแต่ละบทเรียน เป็นส่วนของการทดลอง และแบบฝึกหัดที่เป็นประโยชน์ในการทบทวนทำความเข้าใจในเนื้อหาแต่ละบท ปัญหาในแบบฝึกหัดครอบคลุมลักษณะต่างๆ ที่สำคัญที่ได้อธิบายในบทเรียน เหมาะที่นักศึกษาจะใช้เป็นโจทย์ที่จะเรียนรู้เพิ่มเติมด้วยตนอง

ลาวัณย์ ดุลยชาติ

ตุลาคม 2563

สารบัญ

หน้า

บทที่ 1
ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรม

เครื่องคอมพิวเตอร์เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ สิ่งที่คอมพิวเตอร์เข้าใจคือสัญญาณทางไฟฟ้า แต่ในการเขียนโปรแกรมให้คอมพิวเตอร์ทำงานนั้นมีภาษาให้เลือกใช้หลายภาษา นักศึกษาจะต้องเข้าใจว่าเหตุใดคอมพิวเตอร์จึงสามารถประมวลภาษาโปรแกรมได้ และโปรแกรมที่ได้ถูกสร้างขึ้นก็มีอยู่หลายประเภทขึ้นอยู่กับการประยุกต์มาใช้งานกับเครื่องคอมพิวเตอร์ สำหรับโปรแกรมภาษาที่นำมาใช้ในการศึกษาของเอกสารประกอบการสอนเล่มนี้ คือ โปรแกรม Dev C++ ซึ่งนักศึกษาจะต้องเข้าใจขั้นตอนในการพัฒนาโปรแกรมด้วยภาษาซีให้สามารถประมวลผลตามต้องการได้

เครื่องคอมพิวเตอร์เป็นอุปกรณ์อิเล็กทรอนิกส์อย่างหนึ่ง โดยการให้เครื่องคอมพิวเตอร์ทำงานจะต้องป้อนคำสั่งให้กับมันและต้องเป็นคำสั่งที่เครื่องคอมพิวเตอร์เข้าใจ การนำคำสั่งมาเรียงต่อกันให้ทำงานอย่างดอย่างหนึ่งเรียกว่า โปรแกรม เมื่อโปรแกรมถูกป้อนเข้าไปในเครื่องคอมพิวเตอร์ตัวเครื่องจะทำงานทีล่ะคำสั่ง สำหรับการใช้คำสั่งสั่งงานในคอมพิวเตอร์ทำงานนั้นจะต้องใช้ภาษาที่คอมพิวเตอร์เข้าใจ ภาษาที่คอมพิวเตอร์เข้าใจนั้นเรียนว่า ภาษาเครื่อง (Machine Language) ซึ่งเป็นรหัสเลขฐานสอง เมื่อมีการป้อนภาษานี้เข้าไปในเครื่องคอมพิวเตอร์ รหัสฐานสองจะถูกเปลี่ยนเป็นสัญญาณทางไฟฟ้าที่คอมพิวเตอร์เข้าใจ

แต่ถ้ามนุษย์ต้องการป้อนโปรแกรมให้กับคอมพิวเตอร์เป็นฐานสองนั้นจะทำได้ยากมากเพราะเป็นภาษาที่มนุษย์เข้าใจยากมาก จึงได้ออกแบบตัวอักษรภาษาอังกฤษให้แทนรหัสตัวเลขฐานสองเหล่านั้น ซึ่งเรียกว่า รหัสนีโมนิก (mnemonic) ภาษาคอมพิวเตอร์ที่ใช้รหัสนีโมนิกในการเขียนเรียกว่า ภาษาแอสเซมบลี (Assembly Language) ต่อมาได้มีการพัฒนาชุดคำสั่งภาษาต่างๆใกล้เคียงกับภาษาที่มนุษย์เข้าใจเรียกว่า ภาษาระดับสูง (High level Language) ซึ่งมีอยู่หลายภาษาได้แก่ ภาษาเบสิก ปาสคาล ภาษาซี เป็นต้น สำหรับภาษาแอสเซมบลีเป็นภาษาที่ทำงานได้เร็วเพราะเข้าถึงหน่อยประมวลผลได้เร็วที่สุด เราเรียนภาษานี้ว่า ภาษาระดับต่ำ (low level Language)

สำหรับภาษาซีถือว่าเป็นภาษาที่อยู่ในระดับสูง แต่ความสามารถของคำสั่งภาษาซีบางคำสั่งจะทำงานได้ดี ใกล้เคียงกับภาษาระดับต่ำ แถมเขียนได้ง่ายกว่าภาษาแอสแซมบลี และสามารถติดต่อกับฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ ตารางที่ 1.1 แสดงการแบ่งประเภทของภาษาโปรแกรม

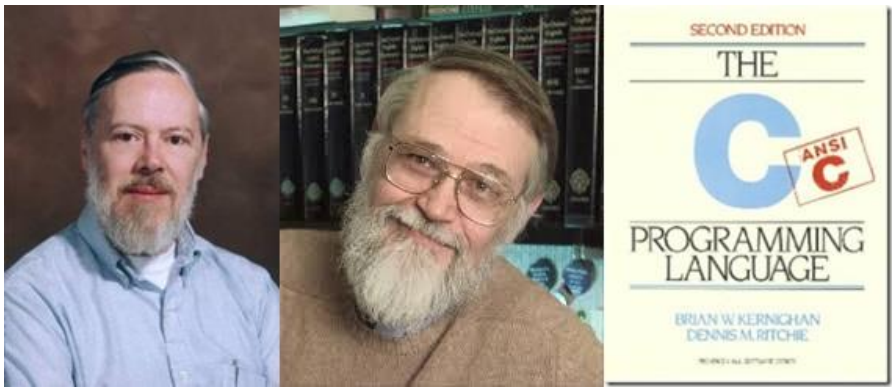
ระดับสูง (Highest level)	ระดับต่ำ (Lowest level)
C Modula-2 Pascal Cobol Fortran Basic	Assembly language

ตารางที่ 1.1 แสดงประเภทของภาษาโปรแกรม

■ ประวัติความเป็นมาของภาษาซี

ในปี ค.ศ. 1972 Dennis Ritchie เป็นผู้คิดค้นสร้างภาซีขึ้นเป็นครั้งแรก โดยพัฒนามาจากภาษา B ละภาษา BCPL แต่ในขณะนั้นยังไม่มีการใช้งานภาษาซีอย่างกว้างขวางนัก จนกระทั่งในปี ค.ศ.1978 Brian Kernighan ได้ร่วมกับ Dennis Ritchie พัฒนามาตรฐานภาษาซีขึ้นเรียกว่า K&R (Kernighan & Ritchie) และเขียนหนังสือชื่อ “The C Programming Language” ออกมาเป็นเล่มแรก ทำให้มีผู้เริ่มสนใจภาษาซีเพิ่มมากขึ้น และด้วยความยืดหยุ่นของโปรแกรมภาษาซีที่สามารถปรับการใช้งานกับคอมพิวเตอร์ชนิดต่าง ๆ ได้ทำให้ภาษาซีไดรับความนิยมเพิ่มมากขึ้นเรื่อย ๆ จนแพร่หลายทั่วโลก จนมีบริษัทต่าง ๆ สร้างและผลิตภาษาซีออกมาเป็นจำนวนมาก เกิดเป็นภาษาอีกหลายรูปแบบ เนื่องจากนั้นยังไม่มีการกำหนดมาตรฐานการสร้างภาษาซี ดังนั้นในปี ค.ศ. 1988 Ritchie และ Kernighan จึงได้ร่วมกับ ANSI (American National Standards Institute) สร้างมาตรฐานของภาษาซีขึ้น เรียกว่า ANSI C เพื่อใช้เป็นตัวกำหนดมาตรฐานในการสร้างภาษาซีรุ่นต่อ ๆ ไป

ปัจจุบันภาษาซียังคงได้รับความนิยมและใช้งานอย่างกว้างขวาง เนื่องจากเป็นภาษาที่เหมาะกับการเขียนโปรแกรมแบบโครงสร้าง (structured programming) และเป็นภาษาที่มีความยืดหยุ่นมาก คือ ใช้กับเครื่องต่าง ๆ ได้ และที่สำคัญในปัจจุบันภาษาโปแกรมรุ่นใหม่ เช่น C++, Perl , Java , C# , Objective-C ฯลฯ ยังใช้หลักการภาษาซีเป็นพื้นฐานด้วย กล่าวคือ หากมีพื้นฐานของภาษาซีมาก่อน ก็สามารถศึกษาโปรแกรมรุ่นใหม่เหล่านี้ได้ง่ายขึ้น



รูปที่ 1.1 รูปซ้ายมือคือ Mr.Dennis Ritchie, รูปตรงกลางคือ Mr. Brain Kernighan และรูปขวามือคือหนังสือที่ทั้งคู่แต่งร่วมกัน

■ ลักษณะเด่นของภาษา C

หากพิจารณาในราบละเอียดแล้ว ภาษา C มีลักษณะเด่นกว่าภาษาระดับสูงทั่วไปในหลายๆ ด้านด้วยกัน ดังรายละเอียดต่อไปนี้

1. ความสามารถในการใช้งานบนสภาพแวดล้อมที่แตกต่างกัน (Portability)

ลักษณะเด่นนี้ถือเป็นจุดเด่นของภาษา C เลยทีเดียว กล่าวคือ ภาษา C สามารถโปรแกรมหรือรันอยู่คอมพิวเตอร์ได้หลายระดับ ตั้งแต่เมนเฟรมคอมพิวเตอร์ จนถึงไมโครคอมพิวเตอร์ซึ่งเมื่อ

เทียบกับภาษาอื่นๆแล้ว ยากจะที่จะหาภาษาใดเทียบเคียงได้เท่า ดังนั้น ซอร์สโค้ดภาษา C ที่เขียนในคอมพิวเตอร์ระดับหนึ่งสามารถนำมาใช้งานบนคอมพิวเตอร์อีกระดับหนึ่งได้โดยไม่ต้องเปลี่ยนชุดคำสั่งเลย หรืออาจต้องเปลี่ยนแปลงบางชุดคำสั่งเพียงเล็กน้อย อีกทั้งยังสามารถนำโปรแกรมภาษา C ไปใช้งานบนระบบปฏิบัติการที่หลาหลาย ถึงแม้ว่าจะมีแพล็ตฟอร์มที่แตกต่างกันก็ตาม

2. มีประสิทธิภาพสูง (Efficiency)

ประสิทธิภาพที่นำมาใช้วัดกับภาษา C สามารถวัดได้จาก 2 แนวทางนี้คือ

- ชุดคำสั่งที่มีความกะทัดรัด และกระชับมาก
- การจัดการหน่วยความจำบนภาษา C มีประสิทธิภาพที่สูงมาก
- มีการทำงานที่รวดเร็ว เทียบเท่ากับภาษาระดับต่ำ ทั้งนี้เนื่องจากภาษา C มีความใกล้ชิดกับฮาร์ดแวร์มากกว่าภาษาระดับสูงอื่นๆ โดยสามารถติดต่อกับรีจิสเตอร์ และหน่วยความจำได้โดยตรงเช่นเดียวกับภาษาแอสเซมบลี

3. ความสามารถในการโปรแกรมแบบโมดูล (Modularity)

ภาษา C อนุญาตให้มีการแบ่งโมดูล เพื่อคอมไพล์ได้ ซึ่งสามารถลิงค์เชื่อมโยงเข้าด้วยกันได้อีกรูปแบบโปรแกรมสามารถเขียนขึ้นได้ตามแบบแผนการโปรแกรมเชิงโครงสร้างได้อย่างดีเยี่ยม ภาษา C คือภาษาที่ประกอบด้วยฟังก์ชั่น ทั้งนี้โมดูลต่างๆ จะเขียนอยู่ในรูปแบบของฟังก์ชั่นทั้งสิ้น

4. พอยน์เตอร์ (Pointer Operations)

ภาษา C มีความสามารถในการทำงานแบบพอยน์เตอร์เป็นอย่างมาก ยากที่จะพบได้ในภาษาระดับสูงทั่วไป โดยพอยน์เตอร์หรือตัวชี้สามารถกำหนดได้จากชนิดข้อมูล (Data Types) หลายชนิดด้วยกัน เช่นเดียวกับฟังก์ชั่น หรือโครงสร้าง รวมถึงตัวแปรแบบอาร์เรย์ ก็ยังสามารถถูกจัดการด้วยการนำพอยน์เตอร์เข้ามาช่วยก็ยังได้

5. มีความยืดหยุ่นสูง (Flexible Level)

ถึงแม้ว่าภาษา C จะจัดอยู่ในภาษาคอมพิวเตอร์ระดับสูงก็ตาม แต่ภาษา C ก็ยังสามารถเขียนใช้งานร่วมกับภาษาระดับต่ำอย่างภาษาแอสเซมบลีได้ ดังนั้นในบางครั้งจึงมีการกล่าวว่า ภาษา C เป็นภาษาที่อยู่กึ่งกลางระหว่างภาษาระดับสูงและภาษาระดับต่ำ

6. ตัวอักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ แตกต่างกัน (Case Sensitivity)

ตามปกติภาษาระดับสูงทั่วไปตัวแปรที่ตั้งขึ้นด้วยตัวอักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่สามารถนำมาใช้งานร่วมกันได้ แต่ในภาษา C จะถือว่ามีความหมายแตกต่างกัน ดังนั้น อักษรตัวพิมพ์ใหญ่ (Upper Case) และอักษรตัวพิมพ์เล็ก (Lower Case) จะมีความแตกต่างกันอย่างสิ้นเชิง เช่น NUM ไม่เท่ากับ num ในขณะเดียวกัน Num ก็จะไม่เท่ากับ num เช่นกัน ดังนั้นการอ้างอิงชื่อตัวแปรและชื่อฟังก์ชั่น จึงต้องอ้างอิงบนชื่อที่ถูกต้องด้วย

■ กฎเกณฑ์การเขียนโปรแกรมภาษา C

สำหรับกฎเกณฑ์การเขียนโปรแกรมภาษา C สามารถสรุปได้ดังนี้

1. จะต้องกำหนด พรีโปรเซสเซอร์ที่ต้นโปรแกรมก่อน เช่น #include <stdio.h>
2. คำสั่งต่างๆ จะต้องใช้ตัวพิมพ์เล็ก
3. ตัวแปรที่ใช้งานในโปรแกรม จะต้องถูกประกาศไว้เสมอ
4. ภายในโปรแกรมจะต้องมีอย่างน้อยหนึ่งฟังก์ชัน ซึ่งก็คือ ฟังก์ชัน main ()

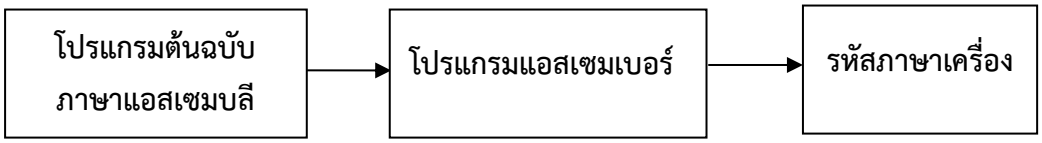
5. ใช้เครื่องหมาย { เพื่อบอกจุดเริ่มต้นของชุดคำสั่ง และเครื่องหมาย } เพื่อบอกจุดสิ้นสุดของชุดคำสั่ง โดยสามารถมีเครื่องหมาย {} ซ้อนอยู่ภายในได้

6. สิ้นสุดของโปรแกรมแต่ละประโยคคำสั่ง จะต้องจบด้วยเครื่องหมาย ; สามารถใช้เครื่องหมาย /* comment */ หรือ //comment เพื่อระบุหมายเหตุภายในโปรแกรม โดยข้อความอธิบายภายเครื่องหมายดังกล่าว จะไม่ถูกนำมาประมวลผล

■ ตัวแปลภาษา

ในการเขียนโปรแกรมคอมพิวเตอร์ไม่ว่าจะเขียนด้วยโปรแกรมระดับสูงหรือโปรแกรมระดับต่ำเราจะต้องแปลงภาษาเหล่านั้นให้เป็นรหัสภาษาเครื่องที่คอมพิวเตอร์เข้าใจเสียก่อน คอมพิวเตอร์จึงจะทำงานได้ ตามที่ได้กล่าวมาแล้วภาษาคอมพิวเตอร์เป็นการนำชุดคำสั่งแต่ละคำสั่งมาต่อกันให้คอมพิวเตอร์ทำงาน การเขียนชุดคำสั่งนี้ไม่ว่าจะเขียนด้วยภาษาอะไรจะเรียกว่าโปรแกรมต้นฉบับ (Source Program) หรือรหัสต้นฉบับ (Source Code) จากนั้นเราจะต้องแปลงให้เป็นภาษาเครื่องที่คอมพิวเตอร์ทำงานได้เรียกว่า Executable Program

ในการเขียนโปรแกรมด้วยภาษาแอสเซมบลี จะใช้ตัวแปลงภาษาให้เป็นภาษาเครื่องที่เรียนว่าแอสเซมบลี (Assembler) ขั้นตอนการแปลสามารถเขียนได้ดังรูปที่ 1.2



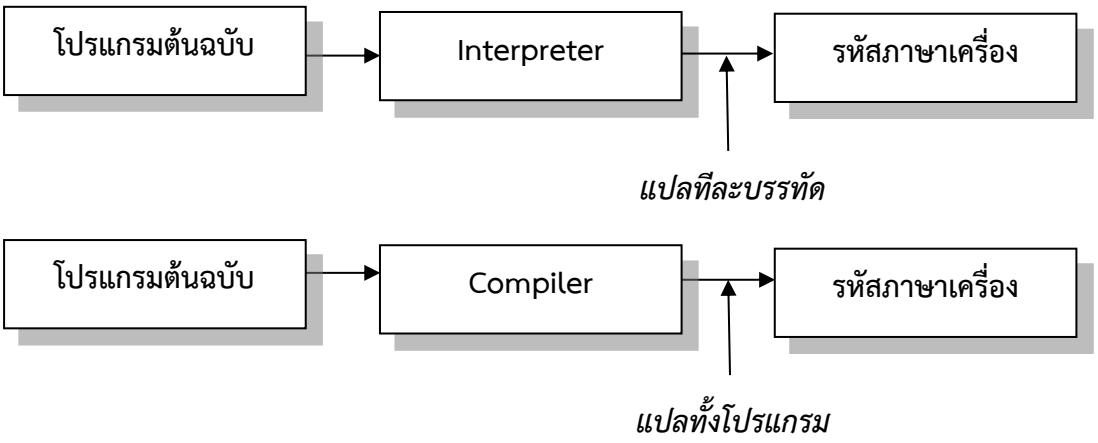
รูปที่ 1.2 ขั้นตอนการแปลภาษาแอสเซมบลีเป็นภาษาเครื่อง

สำหรับการเขียนโปรแกรมด้วยภาษาระดับสูงจะมีวิธีในการแปลงสองประเภทคือ

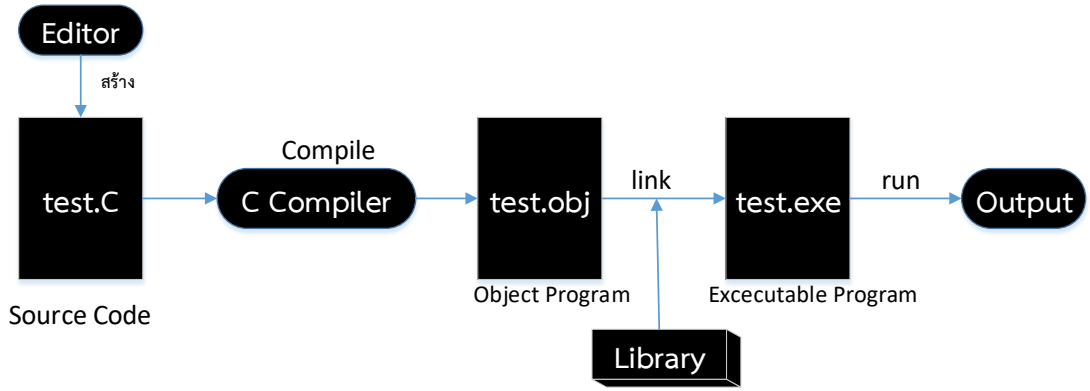
1. **อินเทอร์พรีเตอร์ (Interpreter)** คือ การแปลงคำสั่งทีละคำสั่งให้เครื่องทำงานที่ละคำสั่ง จากนั้นจึงแปลคำสั่งในบรรทัดต่อไปเช่นการเขียนด้วยภาษาเบสิก ตัวที่แปลภาษาประเภทนี้เรียกว่า การทำงานของตัวอินเทอร์พรีเตอร์นี้จะแปลความหมายของคำสั่งทีละคำสั่ง ถ้าไม่พบข้อผิดพลาดเครื่องจะทำคำสั่งที่แปลได้ แต่ถ้าพบข้อผิดพลาดจะหยุดการทำงานและแจ้งข้อผิดพลาดออกมา

2. **คอมไพเลอร์ (Compiler)** คือ แปลโปรแกรมต้นฉบับทั้งหมด และแปลให้เป็นรหัสภาษาเครื่องถ้าพบข้อผิดพลาดก็จะแจ้งออกมา ทำให้โปรแกรมทำงานได้เร็ว เพราะเครื่องไม่ต้องแปลอีกเมื่อจะทำคำสั่งถัดไป สำหรับตัวแปรภาษาเบสิกรุ่นใหม่ๆ จะทำการแปลแบบคอมไพเลอร์ เช่น เทอร์โบเบสิก หรือวิชวลเบสิก เป็นต้น

สำหรับขั้นตอนการแปลภาษาทั้งสองประเภทนี้แสดงได้ดังรูปที่ 1.3



รูปที่ 1.3 ขั้นตอนการแปลภาษาโปรแกรม



รูปที่ 1.4 ขั้นตอนการสร้างโปรแกรมด้วยภาษา C

■ ชนิดของข้อผิดพลาด (Type of Errors)

สำหรับข้อผิดพลาดที่เกิดขึ้นจากการเขียนโปรแกรม สามารถแบ่งออกเป็น 3 ชนิดด้วยกัน คือ

1. ข้อผิดพลาดที่เกิดจากไวยากรณ์ (Syntax Errors)

ข้อผิดพลาดชนิดนี้ เกิดจากการใช้ไวยากรณ์หรือรูปแบบภาษาที่ผิด เช่น สะกดคำสั่งผิด แทนที่จะต้องพิมพ์คำสั่งว่า scanf ก็พิมพ์เป็น scant เป็นต้น ซึ่งเมื่อผ่านการแปลแล้ว ตัวแปลภาษาก็จะไม่รู้จักคำสั่งดังกล่าว อย่างไรก็ตาม ข้อผิดพลาดที่เกิดจากไวยากรณ์นั้น คอมไพเลอร์สามารถตรวจพบ และแจ้งข้อผิดพลาดที่เกิดขึ้นให้ทราบได้

2. ข้อผิดพลาดที่เกิดจากตรรกะโปรแกรม (Logic Errors)

ข้อผิดพลาดชนิดนี้ จัดเป็นข้อผิดพลาดที่เกิดจากตัวโปรแกรมเมอร์เอง เช่น การใช้ตรรกะในการสร้างเงื่อนไขที่ผิดพลาดได้ หรือการสร้างสูตรคำนวณที่ผิด ส่งผลให้ผลลัพธ์ผิดพลาดไม่ตรงกับความเป็นจริง เช่น เงื่อนไขที่ต้องการสร้างคือ ถ้าคะแนน >= 80 จะได้เกรด A แต่กลับสร้างเงื่อนไขเป็น

ถ้าคะแนนรวม >80 จะได้เกรด A จึงส่งผลให้นักศึกษาที่ได้คะแนน 80 ไม่ได้เกรด A เป็นต้น ซึ่งข้อผิดพลาดชนิดนี้ คอมไพเลอร์จะไม่สามารถตรวจพบได้ ดังนั้น ตรรกะในการสร้างเงื่อนไข และสูตรการคำนวณต่างๆ จึงจำเป็นต้องได้รับการทดสอบความถูกต้องทุกครั้งก่อนนำไปใช้งานจริงเสมอ

3. ข้อผิดพลาดในขณะรันโปรแกรม (Runtime Errors)

กรณีพบข้อผิดพลาดชนิดนี้ในขณะที่รันโปรแกรมอยู่ โปรแกรมจะมีข้อความแจ้งให้ทราบ และจะหยุดทำงานไป ตัวอย่างชุดคำสั่งที่ก่อให้เกิด Runtime Errors เช่น ได้เผลอกำหนดตัวหารมีค่าเป็นศูนย์ โดยสมมติ ว่ากำหนดให้ x = 8 และ y = 0 จากนั้นมีการสั่งให้คำนวณค่าของ x หารด้วย y ครั้นเมื่อนำโปรแกรมนี้ไปคอมไพล์ ก็จะไม่พบข้อผิดพลาดใดๆ แต่เมื่อมีการสั่งรันโปรแกรมเมื่อใด ก็จะมีข้อผิดพลาดแจ้งให้ทราบดังรูป สำหรับหนทางแก้ไขก็คือ จะต้องกลับไปแก้ไขซอร์สโค้ดในโปรแกรมให้ถูกต้องเสียก่อน

■ ขั้นตอนการพัฒนาโปรแกรม

การเขียนโปรแกรมคอมพิวเตอร์ให้ทำงานได้ตามที่เราต้องการนั้น ผู้เขียนโปรแกรมจะต้องรู้ว่าจะให้โปรแกรมทำอะไร มีข้อมูลอะไรที่ต้องการให้กับโปรแกรมบ้าง และต้องการอะไรจากโปรแกรม รวมทั้งรูปแบบการแสดงผลด้วย โดยทั่วไปแล้วขั้นตอนการพัฒนาโปรแกรมแบ่งได้ดังนี้

1. กำหนดและวิเคราะห์ปัญหา (Problem Definition and Problem Analysis)
2. เขียนผังงานและซูโดโค้ด (Pseudocoding)
3. เขียนโปรแกรม (Programming)
4. ทดสอบและแก้ไขโปรแกรม (Program Testing and Debugging)
5. ทำเอกสารและบำรุงรักษาโปรแกรม (Program Documentation and Maintenance)

1. กำหนดและวิเคราะห์ปัญหา

ขั้นตอนนี้เป็นขั้นตอนแรกสุดที่นักเขียนโปรแกรมจะต้องทำ การให้คอมพิวเตอร์แก้ไขปัญหาต่างๆ ให้เรานั้น เราจะต้องมีแนวทางที่แก้ไขปัญหาที่เหมาะสมให้กับคอมพิวเตอร์ เพื่อให้การทำงานเป็นไปอย่างมีประสิทธิภาพ โดยมีขั้นตอนย่อยๆดังนี้

1) กำหนดขอบเขตของปัญหา โดยกำหนดรายละเอียดให้ชัดเจนว่าจะให้คอมพิวเตอร์ทำอะไรตัวแปรค่าคงที่ที่ต้องใช้เป็นลักษณะใด ถ้าหากเราไม่กำหนดขอบเขตของปัญหาจะทำให้คอมพิวเตอร์ตัดสินใจได้ยากว่าข้อมูลต่างๆที่เกิดขึ้นนั้นถูกหรือผิด

2) กำหนดลักษณะของข้อมูลเข้าและออกจากระบบ (Input/Output Specification) โดยต้องรู้ว่าข้อมูลที่จะส่งเข้าไปเป็นอย่างไร มีอะไรบ้าง เพื่อให้โปรแกรมทำการประมวลผลแสดงผลลัพธ์ เช่น การรับค่าจากคีย์บอร์ด การใช้เมาส์ การกำหนดปุ่มต่างๆ ลักษณะการแสดงผลทางหน้าจอว่าจะให้มีรูปร่างอย่างไรโดยคำนึงถึงผู้ใช้เป็นหลักในการออกแบบโปรแกรม

3) กำหนดวิธีการประมวลผล (Process Specification) โดยต้องรู้ว่าจะให้คอมพิวเตอร์ทำการประมวลผลอย่างไร จึงได้ผลลัพธ์ตามต้องการ

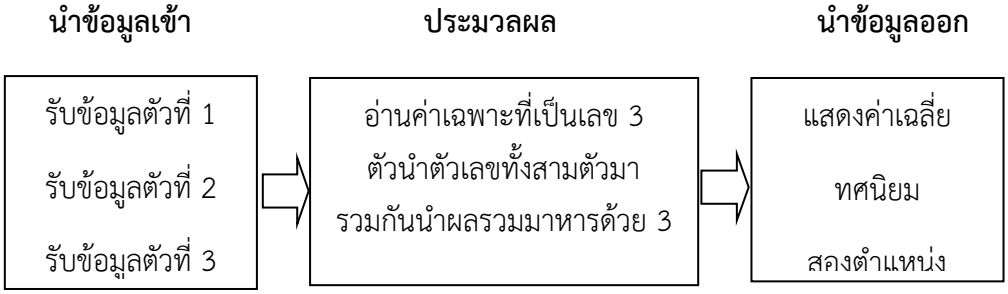
ตัวอย่างที่ 1.1 ถ้าหากต้องการออกแบบโปรแกรมให้คอมพิวเตอร์รับค่ารับค่าข้อมูล 3 ค่า และแสดงค่าเฉลี่ยทางจอภาพ เราอาจกำหนดและวิเคราะห์ปัญหาได้ดังนี้

1. รับข้อมูลจากคีย์บอร์ด

1.1 รับข้อมูลเฉพาะที่เป็นที่เป็นตัวเลขมาเก็บในตัวแปร
1.2 ถ้าข้อมูลเท่ากับ 0 ให้รับใหม่

2. หาค่าเฉลี่ย
   2.1 รวมค่าทุกค่าที่รับมาเข้าด้วยกัน
   2.2 นำค่าผลรวมที่ได้หารด้วย 3
   2.3 ค่าผลลัพธ์ทางจอภาพ
3. แสดงผลลัพธ์โดยมีทศนิยมสองตำแหน่ง
   3.1 แสดงคำว่าค่าเฉลี่ยเท่ากับ
   3.2 แสดงผลลัพธ์โดยมีทศนิยมสองตำแหน่ง

จะเห็นว่าเราจะนำปัญหามาแจกแจงย่อยว่าจะต้องทำอะไรบ้าง โดยข้อมูลที่รับเข้าไปคือตัวเลขสามตัว การประมวลผลคือการหาค่าเฉลี่ย ส่วนเอาต์พุตคือการพิมพ์ผลลัพธ์ เราสามารถเขียนการทำงานของระบบดังแผนภาพในรูปที่ 1.5



รูปที่ 1.5 ขั้นตอนการทำงานของระบบ

2. การเขียนผังงานและซูโดโค้ด

หลังจากที่ได้วิเคราะห์ปัญหาแล้ว ขั้นตอนต่อไปจะต้องใช้เครื่องมือช่วยในการออกแบบโปรแกรม ซึ่งยังไม่ได้เขียนโปรแกรมจริงๆ แต่จะช่วยให้เขียนโปรแกรมได้ง่ายขึ้น และทำให้ผู้อื่นนำโปรแกรมของเราไปพัฒนาต่อได้ง่ายขึ้น โดยเขียนเป็นลำดับขั้นตอนการทำงานของโปรแกรมที่เรียกว่า **อัลกอริทึม** (Algorithm) ซึ่งจะแสดงขั้นตอนการแก้ไขปัญหา โดยใช้ประโยคที่ชัดเจนไม่คลุมเครือ และมีรายละเอียดการทำงานพอสมควรเพียงพอที่จะนำไปเขียนเป็นโปรแกรมให้ทำงานจริงๆ โดยอัลกอริทึมนั้นอาจเขียนให้อยู่ในรูปแบบเฉพาะตัว โดยแต่ละส่วนจะเป็นแนวทางในการเขียนโปรแกรม ซึ่งทำให้เขียนเป็นโปรแกรมภาต่างๆ ได้ง่ายขึ้น ส่วนผังงานจะใช้สัญลักษณ์ต่างๆ แทนการทำงานและทิศทางของโปรแกรม

3. การเขียนโปรแกรม

หลังจากที่ผ่านขั้นตอนทั้งสองแล้ว ขั้นตอนต่อไปจะต้องเขียนเป็นโปรแกรมที่สามารถประมวลผลได้ โดยเปลี่ยนขั้นตอนการทำงานให้อยู่ในรหัสภาษาคอมพิวเตอร์ การเขียนโปรแกรมจะต้องเขียนภาษาที่คอมพิวเตอร์เข้าใจโดยอาจใช้ภาษาระดับสูง หรือภาษาระดับต่ำซึ่งสามารถเลือก

ได้หลายภาษา การเขียนโปรแกรมแต่ละภาษาจะต้องทำตามหลักไวยากรณ์ (syntax) ที่กำหนดไว้ในภาษานั้น นอกจากนี้การเลือกใช้ภาษาจะต้องพิจารณาถึงความถนัดของผู้เขียนโปรแกรมด้วย

4. การทดสอบและแก้ไขโปรแกรม

หลังจากเขียนโปรแกรมจะต้องทดสอบความถูกต้องของโปรแกรมที่เขียนขึ้น หาจุดผิดพลาดของโปรแกรมว่ามีหรือไม่ และตรวจสอบจนไม่พบที่ผิดอีก จุดผิดพลาดของโปรแกรมนี้เรียนว่า บัก (**Bug**) ส่วนการแก้ไขข้อผิดพลาดให้ถูกต้องเรียนว่า **ดีบัก** (debug) โดยทั่วไปแล้วข้อผิดพลาดจากการเขียนโปรแกรมจะมีสองประเภทคือ

1) การเขียนคำสั่งไม่ถูกต้องตามหลักการเขียนโปรแกรมภาษานั้นๆ ซึ่งเรียกว่าSyntax Errorsหรือ Coding Error ข้อผิดพลาดประเภทนี้เรามักพบตอนแปลภาษาโปรแกรมเป็นรหัสภาษาเครื่อง

2) ข้อผิดพลาดทางตรรกะ หรือ Logic Error เป็นข้อผิดพลาดที่โปรแกรมทำงานได้ แต่ผลลัพธ์ออกมาไม่ถูกต้อง

5. ทำเอกสารและบำรุงรักษาโปรแกรม

ขั้นตอนนี้จะทำให้ผู้ใช้สามารถใช้งานโปรแกรมได้อย่างมีประสิทธิภาพ และสะดวกในการตรวจสอบข้อผิดพลาดโดยเขียนเป็นเอกสารประกอบโปรแกรมขึ้นมา โดยทั่วไปแล้วแบ่งออกเป็น 2 ประเภท คือ

1) คู่มือการใช้หรือ User Document หรือ User Guide ซึ่งจะอธิบายการใช้โปรแกรม

2) คู่มือโปรแกรมเมอร์ หรือ Program Document หรือ Technical Reference ซึ่งจะอำนวยความสะดวกในการแก้ไขโปรแกรม และพัฒนาโปรแกรมในอนาคต โดยจะมีรายล่ะเอียดต่างๆเกี่ยวกับโปรแกรมแกรม เช่น ชื่อโปรแกรม การรับข้อมูล การพิมพ์ผลลัพธ์ขั้นตอนต่างๆ ในโปรแกรมเป็นต้น

ส่วนการบำรุงรักษาโปรแกรม (Maintenance) ที่ผู้เขียนโปรแกรมจะต้องคอยตรวจสอบการใช้โปรแกรมจริง เพื่อแก้ไขข้อผิดพลาดซึ่งอาจเกิดขึ้นในภายหลัง รวมทั้งพัฒนาโปรแกรม ให้ทันสมัยอยู่เสมอเมื่อเวลาผ่านไป

■ การพัฒนาโปรแกรมด้วย Dev-C++

การเขียนโปรแกรมคอมพิวเตอร์ภาษาต่าง ๆ ปกติเขียนคำสั่งต่าง ๆ ด้วยโปรแกรมที่เรียกว่า editor ซึ่งอาจใช้โปรแกรมพวก word Processor เขียนโดยไม่ใช้รูปแบบพิเศษต่าง ๆ หรือใช้โปรแกรมที่มีรูปแบบพิเศษน้อย ๆ เช่น notepad เขียน แล้วเปลี่ยนภาษาที่เขียนนั้นเป็นภาษาเครื่องโดยใช้ compiler ของภาษาคอมพิวเตอร์ที่ใช้ เช่น คอมไพล์เลอร์ของภาษาซี ซึ่งก็มีผู้ผลิตหลายรายซึ่งต้องเป็นไปตามมาตรฐาน ANSI C และมีรายละเอียดเพิ่มเติมแตกต่างกันไป

ในการเรียนการสอนวิชานี้จะใช้ editor และ compiler ที่รวมกันไว้แล้ว ใน ชุดพัฒนาหรือเครื่องมือที่ช่วยในการพัฒนาโปรแกรม ภาษาอังกฤษเรียกว่า IDE (Integral Development Environment) ซึ่งเป็นโปรแกรมที่ออกแบบมาเพื่อช่วยในการสร้างโปรแกรมของผู้ที่จะทำการเขียนโปรแกรมทำงานได้ง่ายขึ้น ไม่ต้องแยกใช้ editor เขียนโปรแกรม แล้วเรียกใช้ compiler ทำการ

คอมไพล์โปรแกรมอีก โดยโปรแกรม Dev-C++ ในหัวข้อนี้จะอธิบายเนื้อหาเกี่ยวกับวิธีการดาวโหลด ติดตั้ง และการใช้งานโปรแกรม Dev-C++ ให้ได้ทราบกัน

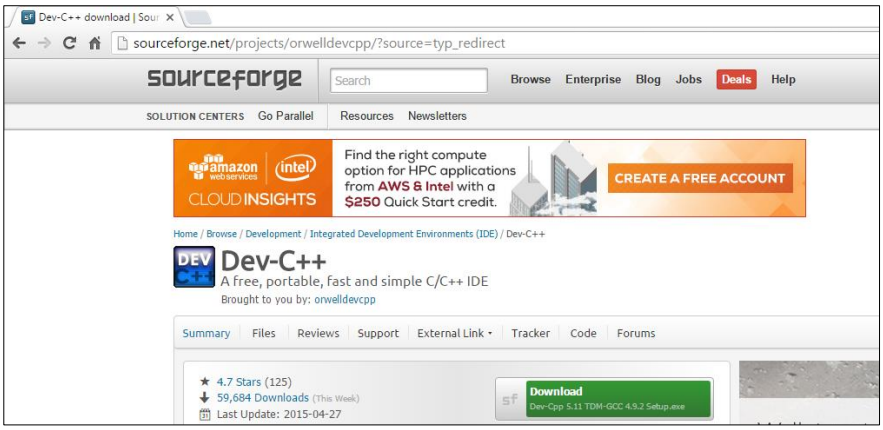
1. ตรวจสอบเครื่องก่อนติดตั้งโปรแกรม Dev-C++

ก่อนดาวโหลดและติดตั้งโปรแกรม Dev-C++ ผู้อ่านควรตรวจสอบว่าคอมพิวเตอร์ที่จะติดตั้งโปรแกรมDev-C++ สามารถใช้งานโปรแกรม Dev-C++ ได้หรือไม่ โดยบริษัท Bloodshed ผู้ซึ่งพัฒนาโปรแกรมภาษา Dev-C++ ได้กำหนดความต้องการ (Requirement) ในด้านต่างๆของเครื่องคอมพิวเตอร์ (computer Specification) ที่จำเป็นต้องมี เพื่อให้สามารถติดตั้งโปรแกรมและทำงานได้อย่างมีประสิทธ์ภาพไว้ดังนี้

Specification	Requirement
ซีพียู(CPU)	400MHz
หน่วยความจำ(Memory)	32 MB ขึ้นไป
ฮาร์ดดิสก์(Harddisk)	ต้องมีพื้นที่ 200 MB ขึ้นไป
ระบบปฏิบัติการ(Operating Systems)	Microsoft windows 2000 หรือ windows xp เป็นต้นไป

2. วิธีการดาวน์โหลดโปรแกรม Dev-C++

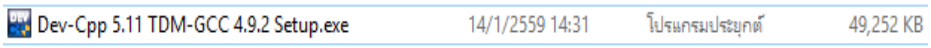
ในหัวข้อนี้จะอธิบายวิธีการดาวน์โหลดโปรแกรม Dev-C++ สำหรับมาใช้ในการพัฒนาโปรแกรมภาษาซี ซึ่งโปรแกรมนี้สามารถดาวน์โหลดมาติดตั้งใช้งานได้ฟรี ขั้นตอนการดาวน์โหลดโปรแกรมให้ไปที่ http://www.bloodshed.net/dev/devcpp.html หรือ http://sourceforge.net/projects/orwelldevcpp/postdownload?source=dlp คลิก ดังนี้



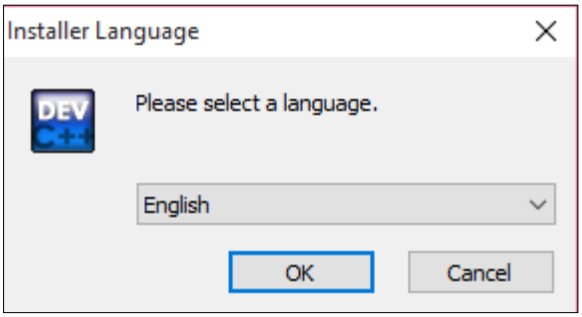
3. วิธีการติดตั้งโปรแกรม Dev-C++

ในหัวข้อนี้อธิบายการติดตั้งโปรแกรม Dev-C++โดยขั้นตอนการติดตั้งโปรแกรมดังนี้

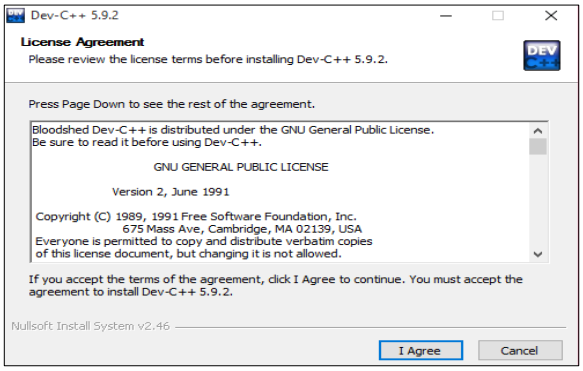
1) ไปยังไดเร็คทอรีที่จัดเก็บโปรแกรมที่ดาวน์โหลดมาเมื่อสักครู่นี้ และดับเบิลคลิก Download ดังรูป



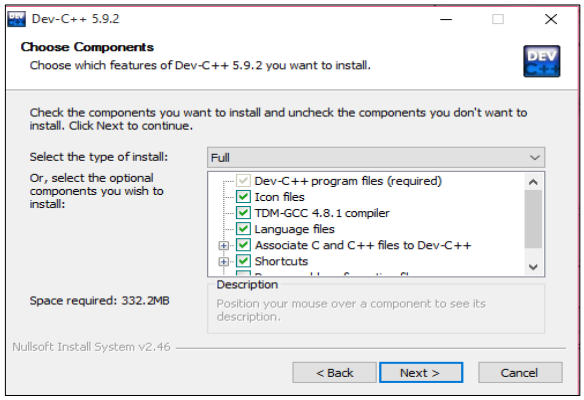
2) หลังดับเบิ้ลคลิกเปิดไฟล์ จะมีหน้าจอติดตั้ง โปรแกรมปรากฏขึ้นดังรูป ให้คลิกปุ่ม ok เพื่อไปยังหน้าจอถัดไป เลือกภาษาที่ใช้แสดงขั้นตอนการติดตั้ง ซึ่งค่าตั้งต้นจะเป็นภาษาอังกฤษ จากนั้น คลิก ok



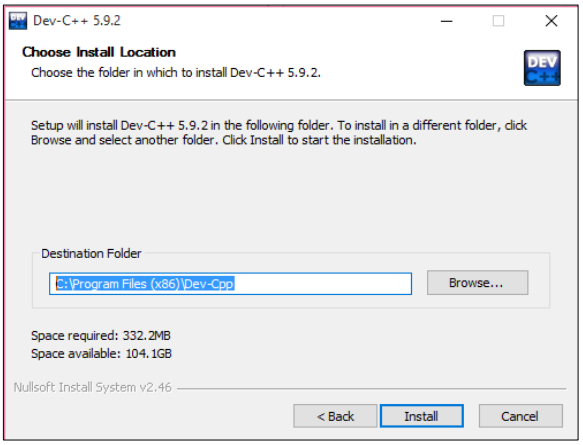
3) โปรกรมจะแสดงหน้าจอ License Agreement ให้คลิกปุ่ม I Agree



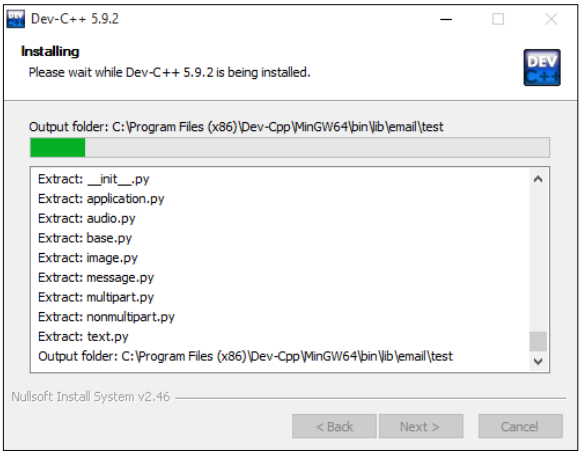
4) หน้าจอต่อไปจะแสดงผลคุณสมบัติของโปรแกรม Dev-C++ ที่จะติดตั้งซึ่งสามารถเลือกติดตั้งเพียงบางคุณสมบัติได้ จากรูปเป็นการเลือกเลือกติดตั้งคุณสมบัติทั้งหมดจากนั้นคลิกปุ่ม Next



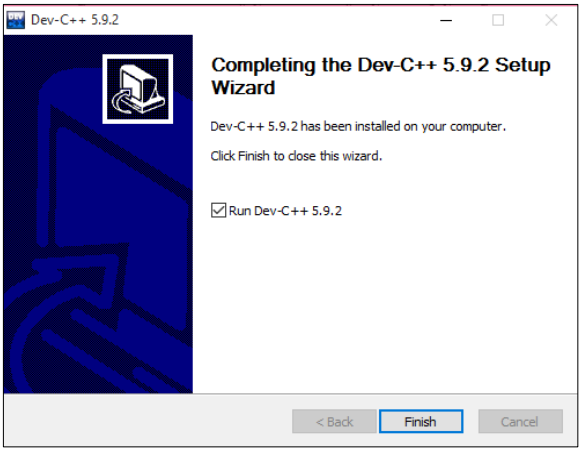
5) ขั้นตอนต่อมาเป็นการระบุว่าจะติดตั้งโปรแกรม Dev-C++ ลงที่ไดเร็ททอร์ใด ซึ่งดีฟอลต์ไดเร็คทอรีของโปรแกรม Dev-C++ คือ C:\Program Files (x86) \Dev-Cpp ผู้อ่านสามารถเลือกติดตั้งโปรแกรมลงในไดเร็คทอรีอื่นได้โดยคลิกปุ่ม Browse แล้วเลือกได้เร็คทอรีที่ต้องการ จากนั้นคลิกปุ่ม Install เพื่อติดตั้ง



6) โปรแกรมที่จะเริ่มทำการติดตั้ง ดังรูป



7) หน้าจอจะปรากฏดังรูปเพื่อแสดงรายละเอียดให้ทราบว่าโปรแกรมได้ถูกติดตั้งอย่างสมบูรณ์ และคลิกปุ่ม Finish

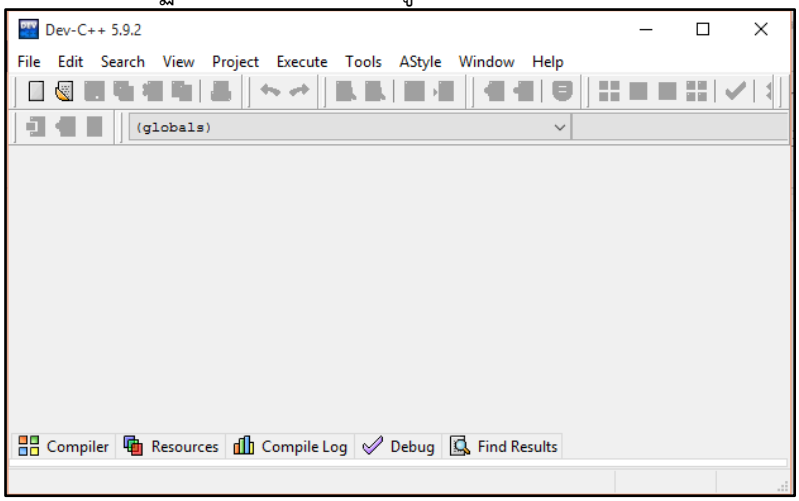


4. วิธีการใช้งานโปรแกรม Dev-C++

เมื่อมาถึงหัวข้อนี้ คิดว่านักศึกษาทุกคนคงสามารถดำเนินการดาวน์โหลดและโปรแกรม Dev-C++ เรียบร้อยแล้ว ดังนั้นต่อไปเราจะมาศึกษาวิธีการใช้งานโปรแกรมกัน

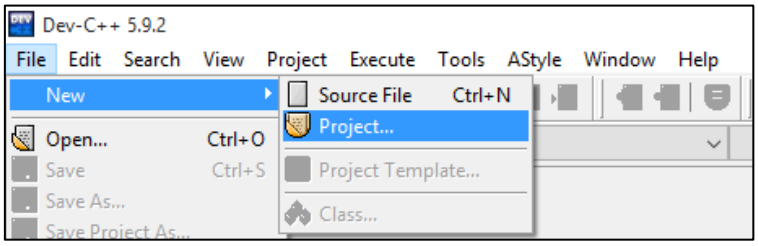
1) ให้เปิดโปรแกรม Dev-C++

- สำหรับ Windows 7 ไปที่ start >All programs > Bloodshed Dev-C++ >Dev-C++ จะปรากฏ หน้าจอโปรแกรมดังรูป

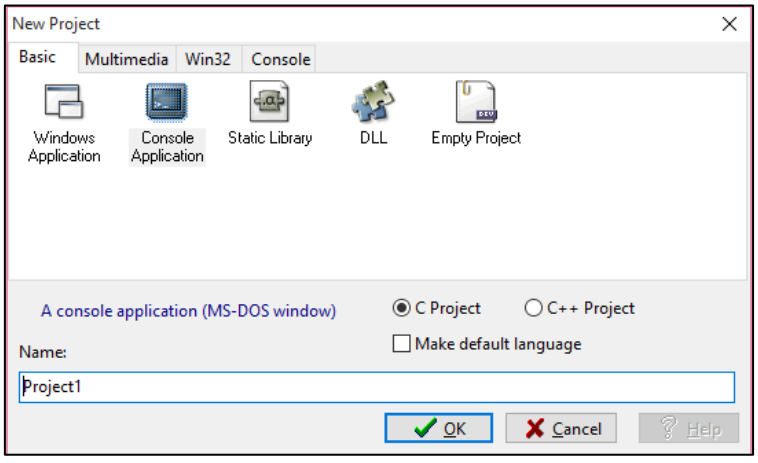


2) ไปที่เมนู File >New>Project จะมีหน้าต่างปรากฏขึ้นดังรูป ให้เลือกหัวข้อ Console Application จากนั้นกรอกข้อมูลสำหรับโปรเจ็คใหม่ตามรายละเอียดดังนี้

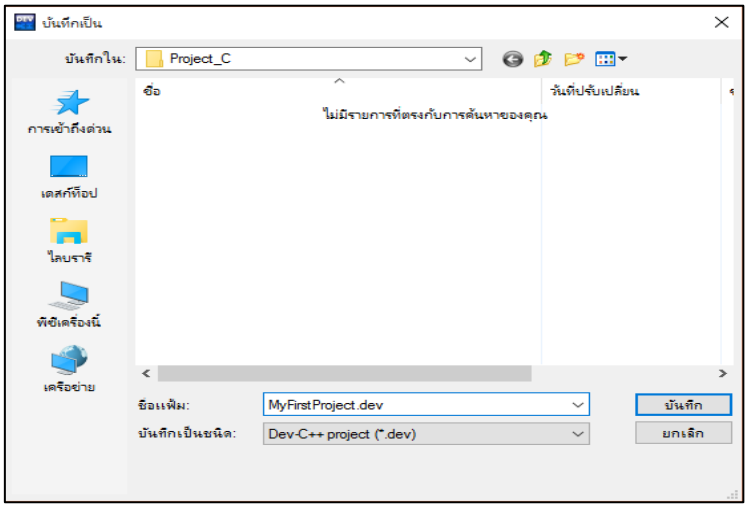
- Name : ให้ระบุชื่อโปรเจ็ค ในที่นี้ชื่อโปรเจ็คว่า MyFirstProject



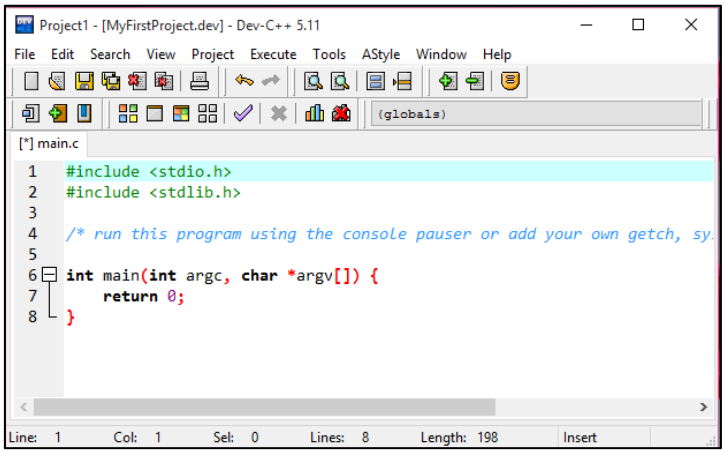
- เลือก C Project จากนั้นคลิกปุ่ม OK



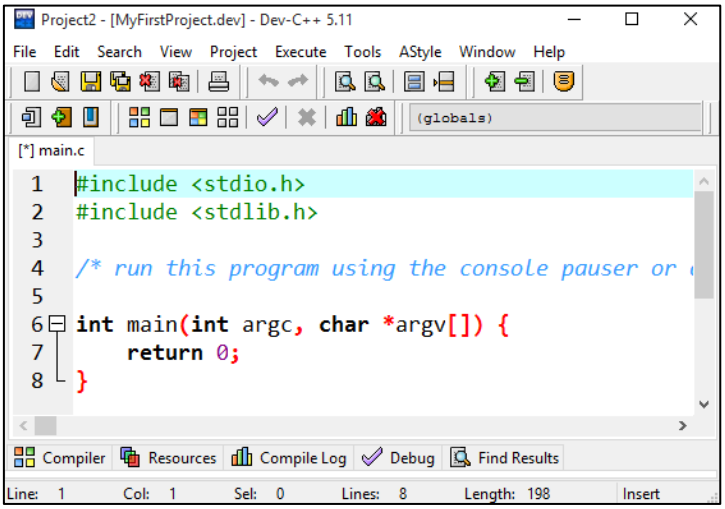
3) โปรแกรมจะแสดงหน้าต่างให้เลือกไดเร็คทอรีที่ต้องการจัดเก็บไฟล์ของโปรเจ็คนี้ ในที่กำหนดให้นำโปรเจ็คไปจัดเก็บไว้ที่ไดเร็คทอรี D:\Project_C จากนั้นคลิกปุ่ม save ดังรูป



4) หลังจากสร้างโปรเจ็คเสร็จสมบูรณ์แล้ว จะมีโปรเจ็คชื่อ MyFirstProject ปรากฏขึ้นโดยโปรแกรมจะสร้างโค้ดตั้งต้นให้อัตโนมัติชื่อว่า main.c ดังรูป

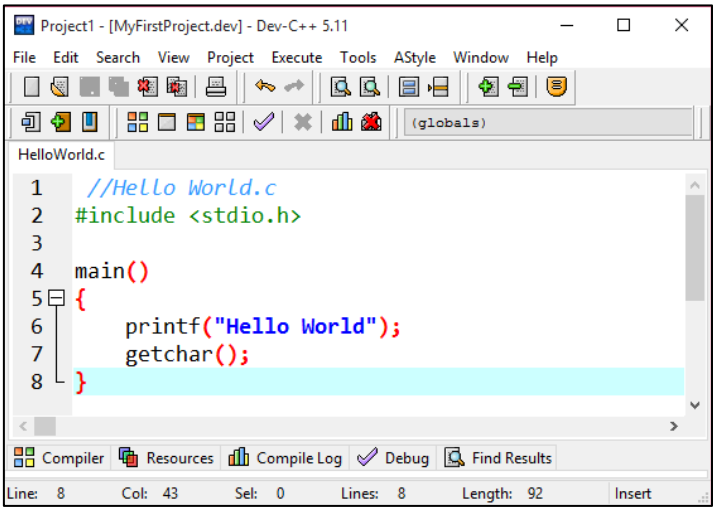


5) แก้ไขโค้ดโปรแกรมไฟล์ main.c ดังรูป

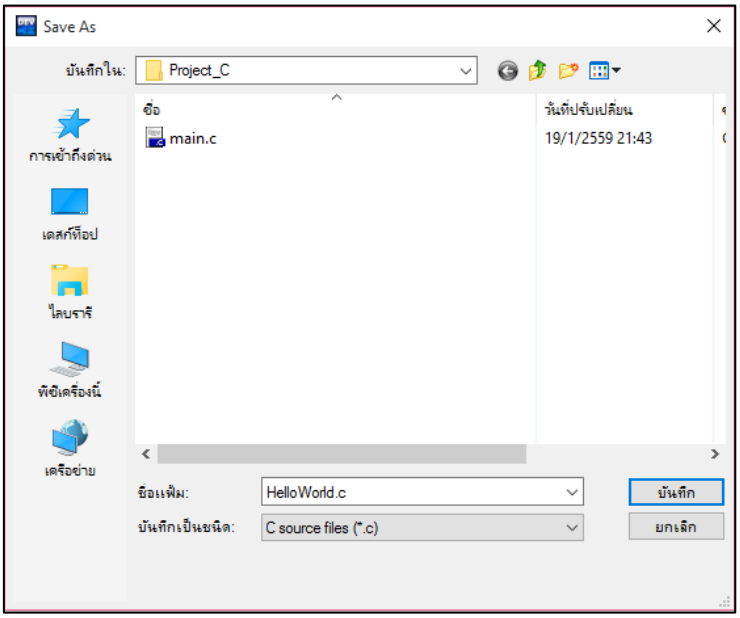


Ctrl S

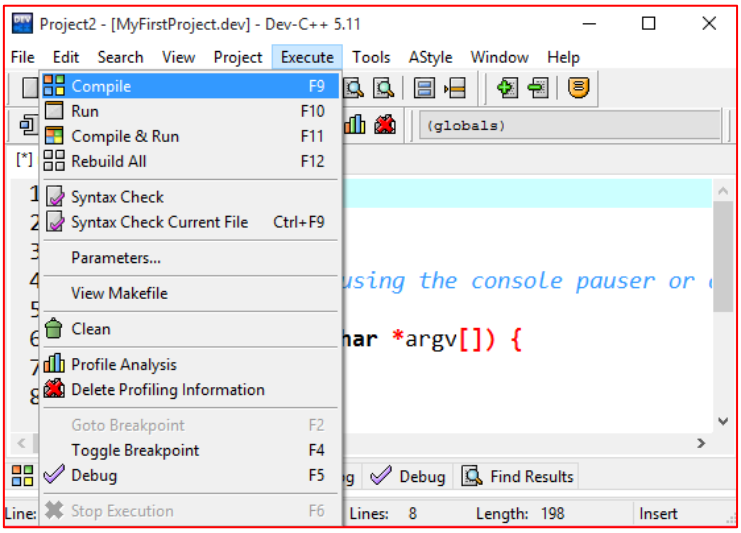
6) บันทึกไฟล์ main.c โดยคลิกปุ่ม  บันทึกโปรแกรมดังรูป หรือกดคีย์ +



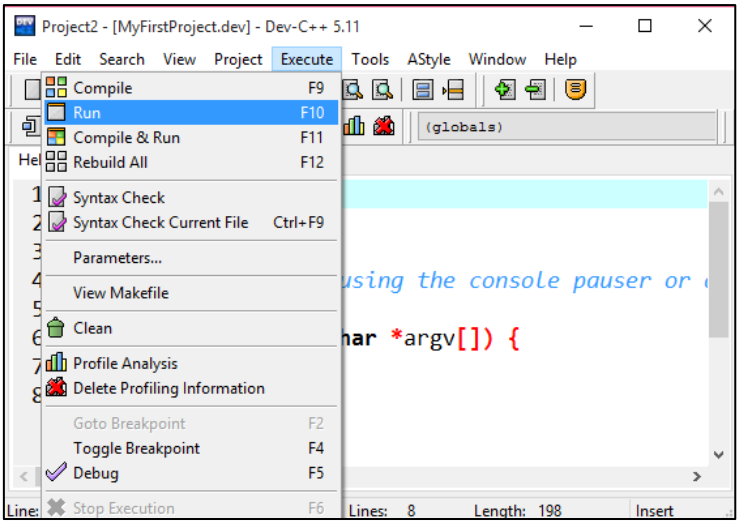
7) โปรแกรมจะแสดงหน้าต่างบันทึกไฟล์ ดังรูป ให้ระบุชื่อโปรแกรมเป็น HelloWorld.c จากนั้นคลิกปุ่ม Save ดังรูปซึ่งจะเป็นการบันทึกไฟล์ HelloWorld.c และโปรเจ็ค MyFirstProject.dev จัดเก็บไว้ที่ไดเร็กทอรี D:\Project_C



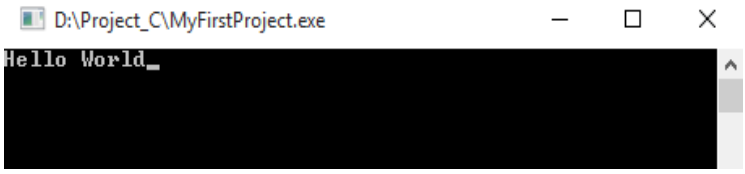
8) ขั้นตอนต่อมาให้ทำการคอมไพล์โปรแกรม โดยไปที่เมนู **Execute** และเลือก **Compile** หรือกด F9 ดังรูป



9) ต่อไปเราจะมาสั่งรันโปรแกรมเพื่อดูผลลัพธ์ของโปรแกรมกัน โดยให้ไปที่เมนู Execute และเลือกที่ **Run** หรือ F10 ทั้งนี้สามารถเลือกคอมไพล์ และรันโปรแกรมพร้อมกันได้โดยเลือกเมนู Compile & Run ก็ได้

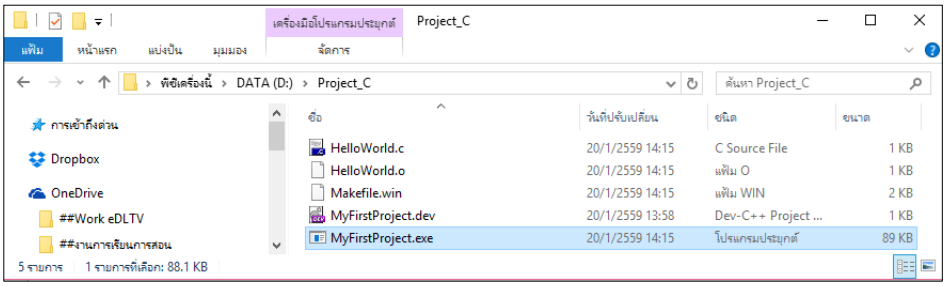


10) ผลลัพธ์ของโปรแกรมก็จะปรากฏก็ดังรูป คือ จะมีข้อความ HelloWorld ปรากฏขึ้นที่หน้าต่าง Command Prompt โดยหลังจากที่กดคีย์ใดๆแล้วหน้าต่าง Command prompt นี้ก็จะหายไป



11) เมื่อโปรแกรม HelloWorld.c ถูกคอมไพล์และรันแล้ว จะปรากฏไฟล์โปรแกรม MyFirstProject.exe ที่ไดเร็คทอรี D:\Project_C โดยโปรแกรม Dev-C++ จะสร้างไฟล์นี้ขึ้นมาให้

อัตโนมัติ โดยหากดับเบิลคบิกที่ไฟล์MyFirstProject.exe ก็จะได้ผลลัพธ์ของโปรแกรมเหมือนขั้นที่ 10



มาถึงตรงนี้นักศึกษาจะพบว่าเราสามารถเขียนโปรแกรมภาษาซี รวมถึงสั่งคอมไพล์และรันโปรแกรมเพื่อดูผลลัพธ์ของโปรแกรมด้วย Dev-C++ ได้แล้ว

■ สรุปท้ายบท

ภาษาคอมพิวเตอร์ที่ใช้สำหรับเขียนโปรแกรม มีทั้งภาษาระดับต่ำและภาษาดับสูง ภาษาระดับต่ำ ได้แก่ ภาษาแอสเซมบลี ส่วนภาษาระดับสูงเป็นภาษาใกล้เคียงกับภาษามนุษย์ ได้แก่ ภาษาปาสคาส ภาษาซี เป็นต้น ในการเขียนโปรแกรมด้วยภาษาซีต้องจะใช้โปรแกรมคอมไพเลอร์ในการแปลภาษาให้เป็นภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้ ตัวแปลภาษาซีใช้งานง่าย ได้แก่ โปรแกรม Dev C++ ในการเขียนโปรแกรมนั้นผู้พัฒนาโปรแกรมในภายหลัง ขั้นตอนการพัฒนาโปรแกรม ได้แก่ การกำหนดและวิเคราะห์ปัญหา การเขียนผังงานและซูโดโค้ด การเขียนโปรแกรม การทดสอบและแก้ไขโปรแกรม และการทำงานเอกสารและบำรุงรักษาโปรแกรม

■ การทดลอง

ให้นักศึกษาดาวน์โหลด และติดตั้งโปรแกรม Dev-C++ ลงในเครื่องคอมพิวเตอร์

■ แบบฝึกหัดท้ายบทที่ 1

1. จงอธิบายลักษณะเด่นของภาษา C มีอะไรบ้าง
2. จงอธิบายกฎเกณฑ์การเขียนโปรแกรมภาษา C มาโดยสรุป
3. ภาษา C ใช้ตัวแปลภาษาชนิดใด
4. ข้อผิดพลาดที่เกิดขึ้นจากการเขียนโปรแกรม มีกี่ประเภท อะไรบ้าง
5. จงอธิบายขั้นตอนการพัฒนาโปรแกรมมาพอสังเขป
6. จงสรุปขั้นตอนการเขียนโปรแกรมภาษา C จนกระทั่งได้เอ็กซีคิวไฟล์

บทที่ 2
ขั้นตอนการทำงานของอัลกอริทึม

ในการเขียนโปรแกรมให้คอมพิวเตอร์ทำงานนั้น จำเป็นต้องมีหลักในการเขียนโปรแกรมโดยอาจเขียนเป็นคำอธิบายง่ายๆ ขึ้นมาก่อน หรือเขียนเป็นผังงาน เพื่อแสดงขั้นตอนการทำงานของโปรแกรม ก่อนที่จะนำไปเขียนเป็นโปรแกรมคอมพิวเตอร์ต่อไป นอกจากนี้การเขียนคำอธิบายการทำงาน หรือการเขียนผังงานยังทำให้การกลับศึกษาโปรแกรมในภายหลังทำได้ง่ายขึ้นอีกด้วย

ตามที่ได้กล่าวมาแล้วว่าก่อนที่จะลงมือเขียนโปรแกรมเราจะต้องออกแบบขั้นตอนการทำงานหรืออัลกอริทึม (Algorithm) ก่อน ซึ่งจะเป็นเครื่องมือในการแสดงขั้นตอนการทำงานระบบงานใดๆ เพื่อให้การเขียนเป็นไปได้รวดเร็วและง่ายขึ้น โดยเราอาจเขียนอัลกอริทึมในลักษณะผังงาน (Flowchart) หรือรหัสเทียมที่เรียกว่าซูโดโค้ด (Pseudo-codes) ก็ได้

■ ซูโดโค้ด (Pseudo-codes)

รหัสเทียมหรือหรือรหัสจำลองหรือซูโดโค้ดเป็นคำอธิบายขั้นตอนการทำงานของโปรแกรม โดยใช้ถ้อยคำผสมระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้าง หรืออาจใช้ภาษาไทยก็ได้แต่ควรเขียนภาษาอังกฤษ โดยช่วยให้ผู้เขียนโปรแกรมสามารถพัฒนาขั้นตอนต่าง ให้เป็นโปรแกรมได้ง่ายขึ้น แต่ส่วนใหญ่แล้ว คำที่ใช้มักเป็นคำเฉพาะ (Reserve Word) ที่มีอยู่ในการเขียนโปรแกรมและมักจะเขียนด้วยตัวอักษรใหญ่ ซูโดโค้ดที่ดีจะต้องมีความชัดเจน สั้น และได้ใจความ ข้อมูลต่างๆที่ใช้จะถูกเขียนอยู่ในรูปของตัวแปร ซูโดโค้ดนี้บางครั้งจะเรียกว่าอัลกอริทึม รูปแบบทั่วไปจะเป็นดังนี้

รูปแบบ

```
Algorithm <ชื่อของอัลกอริทึม>
  1.    .....
  2.    .....
        .....
        .....
END
```

ตัวอย่างเช่น ในการเขียนซูโดโค้ดสำหรับให้คอมพิวเตอร์หาค่าเฉลี่ยจากข้อมูลที่เข้ารับทางแป้นพิมพ์ ถ้าใส่ค่าศูนย์จะหยุดป้อนข้อมูล อาจเขียนได้ดังนี้

รูปแบบ

Algorithm การหาค่าเฉลี่ย

1. ตัวนับ = 0
2. ผลรวม = 0
3. รับค่าทางแป้นพิมพ์เก็บไว้ไหนตัวแปร (ข้อมูล)
4. ถ้าข้อมูลมากกว่า 0
   เพิ่มค่าตัวนับขึ้นหนึ่งค่า
   ผลรวม = ผลรวม + ค่าข้อมูล
   ย้อนกลับไปทำขั้นตอนที่ 3
   ถ้าไม่มากกว่าไปทำขั้นตนที่ 5
5. ค่าเฉลี่ย = ตัวรวมหารด้วยตัวนับ
6. แสดงค่าเฉลี่ยทางจอภาพ โดยมีทศนิยมสองตำแหน่ง
7. จบ

จะเห็นได้ว่าขั้นตอนการหาค่าเฉลี่ยได้เขียนไว้อย่างเข้าใจ เราสามารถได้ว่าการทำงานต่างๆ จะต้องใช้ตัวแปรใดบ้าง แต่ละขั้นตอนมีการประมวลผลอย่างไร แต่โดยทั่วไปแล้วซูโดโค้ดจะถูกเขียนด้วยภาษาอังกฤษดังต่อไปนี้

Algorithm Avarage_Sum

1. count = 0
2. sum = 0
3. INPUT (value)
4. IF value > 0 THEN
   count = count + 1
   sum = sum + value
   GOTO 3
   ELSE GOTO 5
5. average = sum / count
6. OUTPUT (average)
7. END

ในการคำนวณหาพื้นที่สามเหลี่ยม เราอาจเขียนซูโดโค้ดได้ดังต่อไปนี้

รูปแบบ
ซูโดโค้ดหาพื้นที่สามเหลี่ยม เริ่มต้น 1. รับค่าความยาวของด้านที่เป็นฐานมาเก็บในตัวแปร x 2. รับค่าความยาวของส่วนสูงมาเก็บในตัวแปร Y 3. คำนวณพื้นที่โดย ARRAY = (X*Y)/2 จบ

หรืออาจเขียนภาษาอังกฤษได้เป็น

START 1. READ X 2. READY 3. Compute ARRAY = (x*y)/2 4. Print ARRAY END

แม้ว่าการเขียนซูโดโค้ด จะไม่มีรูปแบบแน่นอน แต่โดยทั่วไปแล้วมักจะทำกันดังลักษณะต่อไปนี้

1. การรับข้อมูลเข้าและการแสดงผลข้อมูล

ในการรับข้อมูลจะนิยมใช้คำว่าREAD หรือ INPUT ตามด้วยตัวแปลที่ต้องการใช้เก็บข้อมูล ถ้าหากมีตัวแปลหลายตัวจะใช้เครื่องหมายคอมมา (“,”) คั่น ส่วนการแสดงผลมักใช้คำว่า PRINT

2. การคำนวณ

ในการประมวลผลแบบคำนวณจะขึ้นต้นด้วยคำว่า Compute ตามด้วยตัวแปลที่ต้องการเก็บค่าจาการคำนวณ เครื่องหมายเท่ากับและนิพจน์การคำนวณ ตัวอย่างเช่น

```
Compute  ARRAY  = (X * Y) / 2
```

3. การตัดสินใจและทดสอบทางเลือก

การตัดสินใจเพื่อเลือกทำระหว่างสองทางจะใช้คำว่า IF หรือ IF-THEN-ELSE และ WNDIF โดยจะเปรียบเทียบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำกลุ่มคำสั่ง (Statement) กลุ่มหนึ่งถ้าเป็นเท็จจะทำกลุ่มคำสั่งอีกกลุ่มหนึ่ง ตัวอย่างเช่น

```
IF number > 0 THEN
        PRINT POSITVE NUMBER
ELSE
        PRINT NEGATIVE NUMBER
ENDIF
```

จากตัวอย่าง หมายความว่า ถ้าหากค่า number มีค่ามากกว่า 0 ให้คอมพิวเตอร์พิมพ์คำว่า POSITIVE NUMBER ถ้าหาไม่มากกว่า 0 จะพิมพ์คำว่า NEGATIVE NUMBER

สำหรับกรณีที่มีทางเลือกมากกว่าสองทางจะใช้คำว่า CASE และ ENDCASE โดยจะทำกลุ่มคำสั่งที่มีคำนำหน้ากลุ่มเท่ากับค่าในตัวแปรที่อยู่หลัง CASE ตัวอย่างเช่น

```
CASE num OF
  1.  :PRINT 11111
  2.  :PRINT 22222
  3.  :PRINT 33333
ENDCASE
```

จากตัวอย่างถ้าค่าในตัวแปร num เป็น 1 จะให้พิมพ์คำว่า 11111 ถ้าตัวแปร num มีค่าเป็น 2 จะให้พิมพ์คำว่า 22222

4. การทำแบบวนซ้ำ

ในการทำซ้ำหมายความว่าให้ระบบทำงานซ้ำๆ ตามเงื่อนไขที่กำหนด โดยจะมีการเปรียบเทียบเงื่อนไขในการทำซ้ำ แบ่งออกได้เป็นสามรูปแบบดังนี้

1) การทำซ้ำที่มีการเพิ่มค่าในแต่ละรอบ จะใช้คำว่า FOR และ ENDFOR โดยมีคำว่า IN STEPS OF เป็นการบอกค่าที่เพิ่มในแต่ละรอบ ถ้าไม่มีคำว่า IN STEP OF หมายความว่าเพิ่มค่ารอบละหนึ่ง

2) การทำซ้ำจนระบบมีเงื่อนไขอย่างหนึ่งจึงหยุดทำ จะใช้คำว่า REPEAT – UNTIL ดังรูปแบบต่อไปนี้

รูปแบบ

```
REPEAT
        Statement_1
        ……..
UNTIL (Condition)
```

3) ถ้าเงื่อนไขเป็นจริงจะทำคำสั่งภายใน จะใช้คำว่า DO – WHILE โดยจะตรวจสอบเงื่อนไขก่อนที่ทำชุดคำสั่งภายใน ดังรูปแบบต่อไปนี้

รูปแบบ
DO (Condition) WHLE Statement_1 ..... ENDDO

5. การกระโดดข้าม

การกระโดดข้ามไปทำชุดคำสั่งใดๆ จะใช้คำว่า LABEL กำหนดตำแหน่งที่จะกระโดดมาและใช้คำว่า GOTO ในตำแหน่งที่จะกระโดด ตัวอย่าง เช่น

รูปแบบ
START : Statement_1 ..... AB1 : .... GOTO AB1 END

ตัวอย่าง 2.1 ถ้าหากต้องการเขียนซูโดโค้ดในการบวกเลข 1 + 2 + 3 +.....+100 และพิมพ์ผลลัพธ์ออกมา อาจเขียนได้ดังนี้

```
START
    I = 0
    SUM = 0
    DO (I<= 100) WHILE
        Compute SUM = SUM +I
        Compute I = I +1
```

■ การเขียนผังงาน (Flowchart)

ผังงานหรือเรียกกันว่า **โฟลวชาร์ต** เป็นแผนภาพที่ใช้ออกแบบและอธิบายการทำงานของโปรแกรมโดยอาศัยรูปทรงต่างๆ ควบคู่ไปกับลูกศร แต่ละรูปในแผนภาพจะหมายถึงการทำงานหนึ่งขั้นตอนส่วนลูกศรจะแทนลำดับการทำงานขั้นตอนต่างๆ รวมทิศทางการไหลของข้อมูลตั้งแต่เริ่มต้นจนได้ผลลัพธ์ตามต้องการ ระบบงานทุกชนิดที่ผ่านการวิเคราะห์เป็นลำดับขั้นตอนแล้วจะสามารถเขียนเป็นผังงาน

1. ประโยชน์ของผังงาน

1. ช่วยอธิบายลำดับขั้นตอนการทำงานของโปรแกรม
2. ทำให้ตรวจสอบข้อผิดพลาดของโปรแกรมได้ง่าย
3. ทำให้ผู้อื่นสามารถศึกษาการทำงานของโปรแกรมและแก้ไขโปรแกรมได้ง่าย

2. การเขียนผังงานที่ดี

1. เขียนตามสัญลักษณ์ที่กำหนด
2. ใช้ลูกศรแสดงทิศทางการทำงานจากบนลงล่าง
3. อธิบายสั้น ให้เข้าใจง่าย
4. ทุกแผนภาพต้องมีทิศทางเข้าออก
5. ไม่ควรโยงลูกศรไปที่ไกลมากๆ ถ้าต้องทำให้สัญลักษณ์การเชื่อมต่อแทน

3. ประเภทของผังงาน

1) ผังงานระบบ (System Flowchart)

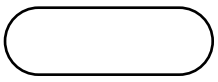
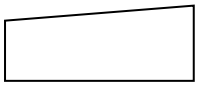
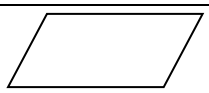
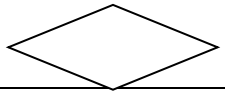
ใช้แสดงขั้นตอนการทำงานภายในระบบงานหนึ่งๆ โดยกล่าวถึงข้อมูลต่างๆ ที่เกี่ยวข้องทั้งหมด เช่น เอกสารเบื้องต้นคืออะไร วัสดุที่ใช้คืออะไร ใช้หน่วยความจำประเภทใด จะต้องส่งผ่านไปยังหน่วยงานใด วิธีการประมวลผลและการแสดงผลลัพธ์ โดยจะกล่าวอย่างกว้างๆ ไม่สามารถนำมาเขียนเป็นโปรแกรมได้

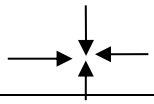
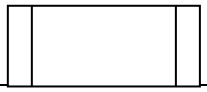
2) ผังงานโปรแกรม (Program Flowchart)

ผังงานประเภทนี้ จะแสดงถึงขั้นตอนของคำสั่งที่ใช้ในโปรแกรม การรับข้อมูล การประมวลผล การแสดงข้อมูล บางครั้งจะเรียกว่าผังการเขียนโปรแกรม

4. สัญลักษณ์ที่ใช้ในการเขียนผังงาน

การเขียนผังงานจะต้องใช้ภาพสัญลักษณ์ต่างๆ นำมาเรียงต่อกันเพื่อแสดงลำดับการทำงาน สัญลักษณ์มาตรฐานที่เรียกว่า สัญลักษณ์ ANSI (American National Standards Institute) ที่ควรทราบมี ดังนี้

สัญลักษณ์รูปภาพ	ความหมาย
	การเริ่มต้นหรือสิ้นสุดการทำงานของโปรแกรม
	รับข้อมูลหรือนำเข้าจากคีย์บอร์ด
	รับข้อมูลหรือแสดงผลข้อมูล
	การตัดสินใจ หรือการเปรียบเทียบ

สัญลักษณ์รูปภาพ	ความหมาย
	ทิศทาง
	การประมวลผลการคำนวณต่างๆ
	การทำงานย่อย
	จุดเชื่อมต่อ(connection)
	แสดงผลข้อมูลออกทางเครื่องพิมพ์

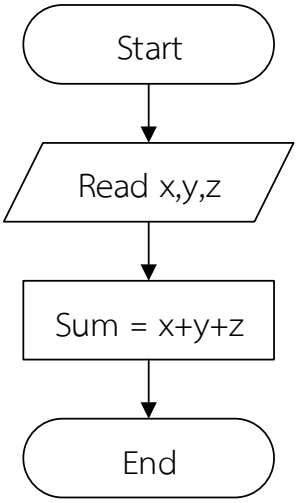
■ รูปแบบการจัดภาพของผังงาน

ต่อไปจะกล่าวถึงรูปแบบของผังงาน ซึ่งนิยมใช้เป็นมาตรฐานในการเขียนโปรแกรมแบบโครงสร้าง (Structure Programming) โดยโปรแกรมทุกโปรแกรมจะมีโครงสร้างการควบคุมเพื่อกำหนดทิศทางการทำงานของโปแกรม โครงสร้างโดยทั่วไปมีอยู่ 3 รูปแบบ ดังต่อไปนี้

1. โครงสร้างการทำงานแบบลำดับ (Sequence) จะแสดงขั้นตอนการทำงานที่เรียงลำดับกลับไป ไม่มีการข้ามขั้น หรือย้อนกลับไปทำคำสั่งที่ได้ทำไปแล้ว ดังตัวอย่างในรูปที่ 2.3

ตัวอย่าง 2.3 การเขียนผังงานแสดงการหาค่าเฉลี่ย และผลคูณของเลขสามจำนวน สมมติเลขสามจำนวนคือ x, y, z ต้องการหาผลบวก ขั้นตอนวิธี คือ หาผลบวกของเลขทั้งสามจำนวน ผังงานเป็นดังนี้

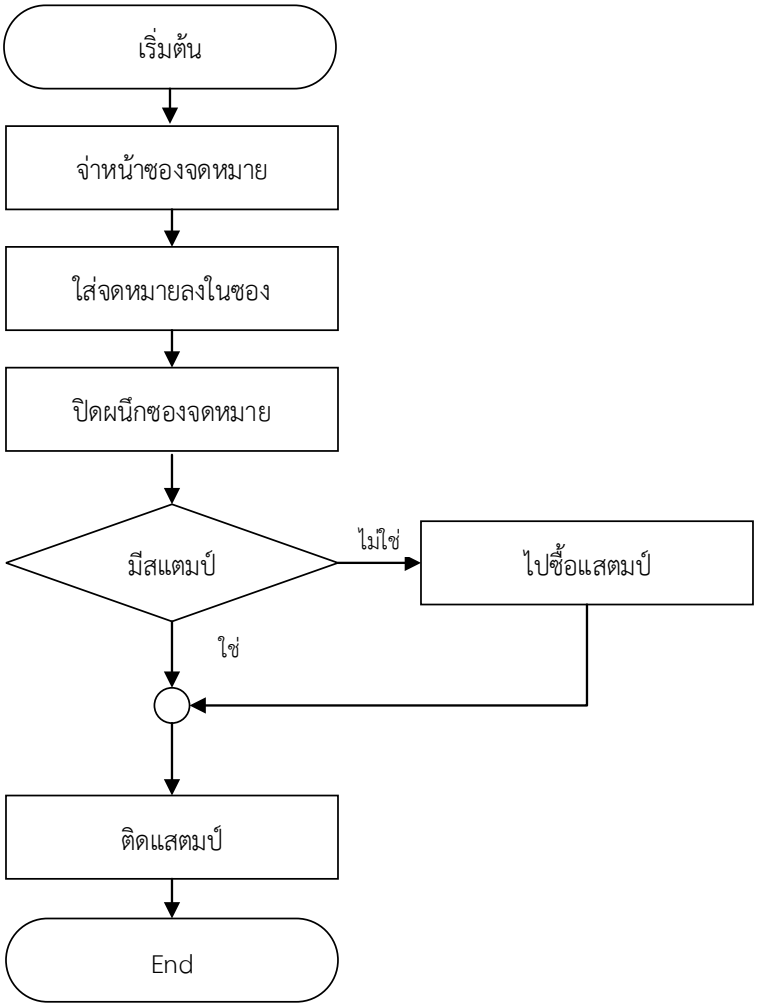
1) อ่านค่าตัวเลข 3 จำนวน
2) นำเลข 3 จำนวนมาบวกกัน
3) แสดงผลลัพธ์ที่ได้จากการบวก
4) จบการทำงาน



2. ผังโปรแกรมแบบมีการเลือก (Selection) เป็นโครงสร้างที่ตรวจสอบเงื่อนไข(Condition) ให้โปรแกรมเลือกทำอย่างใดอย่างหนึ่ง ซึ่งมีอยู่สามกรณี ดังต่อไปนี้

1) การเลือกแบบหนึ่งเส้นทาง จำทำงานเฉพาะเมื่อเงื่อนไขเป็นจริงเท่านั้น ผังงานแสดงได้ดังรูปที่ 2.4

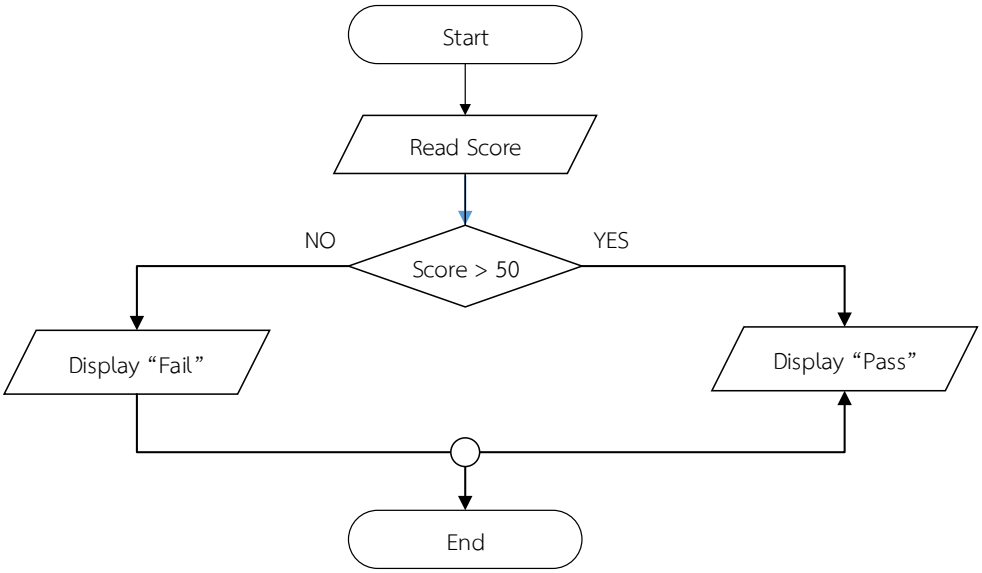
ตัวอย่าง 2.4 การเขียนผังงานแสดง



2) การเลือกทำแบบสองเส้นทาง จะพิจารณาเงื่อนไขที่เป็นจริงและเป็นเท็จ โดยถ้าเป็นจริงจะทำอย่างหนึ่ง ถ้าเป็นเท็จจะทำอีกอย่างหนึ่งผังงานแสดงได้ดังรูปที่ 2.5

ตัวอย่าง 2.5 การเขียนผังงานแสดง

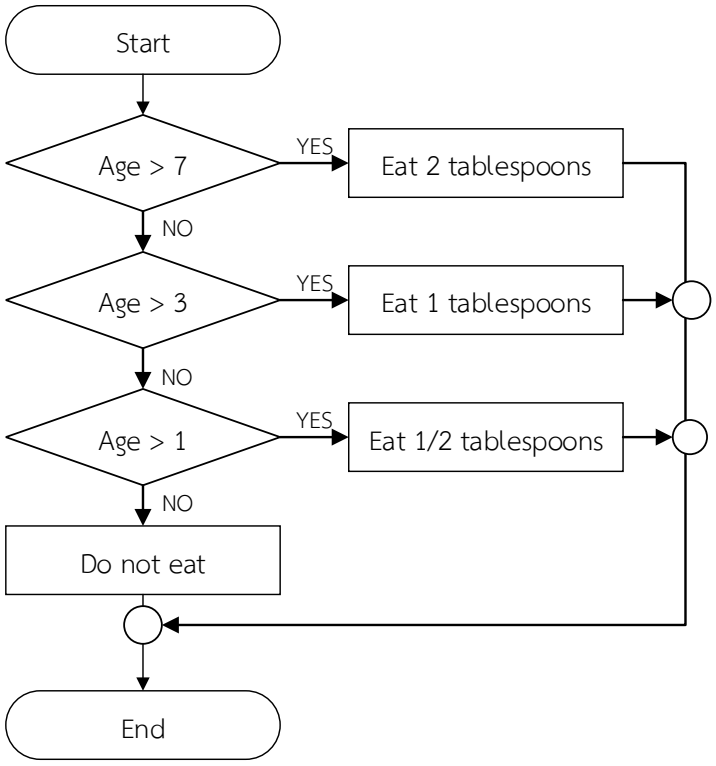
1. รับค่าคะแนนนักศึกษา
2. ตรวจสอบค่าคะแนนนักศึกษาว่ามากกว่า 50 หรือไม่
3. ถ้าคะแนนมากกว่า 50 ให้แสดง Pass ถ้าคะแนนน้อยกว่า 50 แสดง Fail
4. จบการทำงาน



3) การเลือกทำแบบหลายเส้นทาง จะพิจารณาเงื่อนไขต่างๆ ที่เกิดขึ้น ถ้าเท่ากับทางเลือกใดจะให้ไปทำงานตามทางเลือกนั้นผังงานแบบเลือกหลายทางสามารถเขียนได้ดังรูปที่ 2.6

ตัวอย่าง 2.6 การเขียนผังงานแสดงวิธีการรับประทานยา ที่แบ่งตามขนาดของอายุของผู้ทางดังนี้

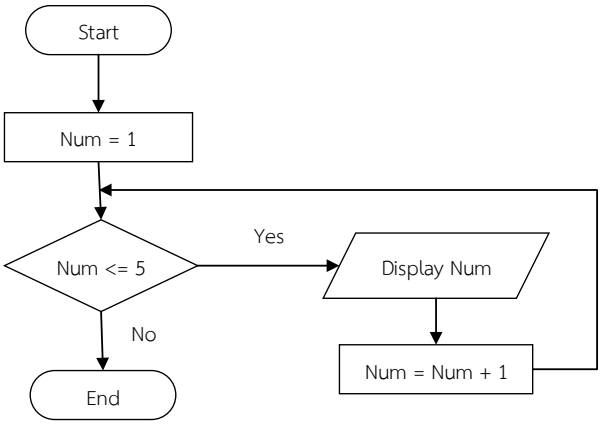
1. อายุมากกว่า 7 ปีรับประทานครั้งละ 2 ช้อนชา
2. อายุมากกว่า 3 ปีถึง 7 ปีรับประทานครั้งละ 1 ช้อนชา
3. อายุมากกว่า 1 ปีถึง 3 ปีรับประทานครั้งละ 1/2 ช้อนชา
4. แรกเกิด ถึง 1 ปีห้ามรับประทาน



3. ผังโปรแกรมแบบทำงานวนซ้ำ (Looping)

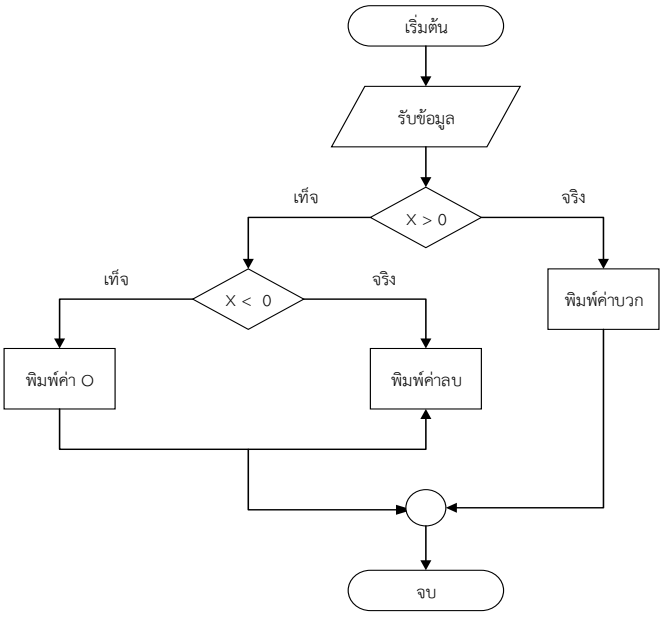
ตัวอย่างที่ 2.7 การเขียนขั้นตอนการทำงานแบบวนซ้ำ(Looping) เพื่อแสดงตัวเลข 1

1. กำหนดตัวแปร Num กำหนดให้มีค่าเริ่มต้นเท่ากับ 1
2. ตรวจสอบตัวแปรนำว่ามีค่าน้อยกว่าหรือเท่ากับ 5 หรือไม่
3. ถ้าตัวแปร Num น้อยกว่าหรือเท่ากับ 5 ให้แสดงค่า Num แล้วเพิ่มค่า Num ครั้งละ 1
4. ตรวจสอบตัวแปร Num และทำซ้ำข้อ 3 จนกระทั่งตัวแปรนำมีค่ามากกว่า 5 จึงให้จบการทำงาน



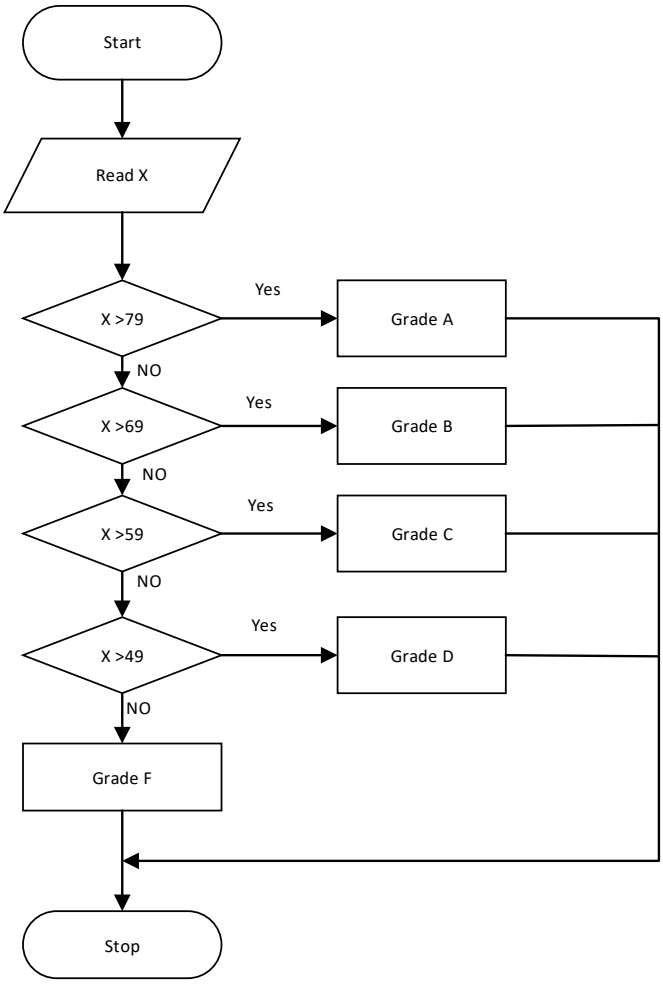
ตัวอย่าง 2.8 จงเขียนผังงานให้รับข้อมูลในเลขเข้าไปหนึ่งตัว เก็บในตัวแปร X จากนั้นให้พิมพ์ตามเงื่อนไขต่อไปนี้

ถ้า X > 0 “เป็นเลขบวก”
ถ้า X < 0 “เป็นเลขลบ”
ถ้า X = 0 “เป็นเลขศูนย์”



ตัวอย่าง 2.9 ถ้าหากต้องการนำคะแนนของนักศึกษามาตัดเกรดตามเงื่อนไขดังต่อไปนี้

คะแนน	80 - 100	ได้เกรด A
คะแนน	70 – 79	ได้เกรด B
คะแนน	60 – 69	ได้เกรด C
คะแนน	50 – 59	ได้เกรด D
คะแนน	50 – 59	ได้เกรด F



ตัวอย่าง 2.10 จากการรับข้อมูลตัวเลขใดๆ หนึ่งตัว จงหาว่าตัวเลขที่รับเข้ามาเมื่อพิจารณาแยกตามประเภทแล้วได้เลขศูนย์ เลขจำนวนบวก และเลขลบ ประเภทละกี่ตัว โดยเขียนซูโดโค้ดและผังงาน เพื่อแสดงรายละเอียดของการทำงาน

วิธีทำ

การวิเคราะห์ปัญหา

ข้อมูลที่รับเข้า	ตัวเลขใดๆ หนึ่งตัว
ผลลัพธ์ที่ต้องการ	ตัวเลขแสดงการนับของจำนวนที่เป็นเลขศูนย์
	ตัวเลขแสดงการนับของจำนวนที่เป็นเลขบวก
	ตัวเลขแสดงการนับของจำนวนที่เป็นเลขลบ

อัลกอริทึม

1. การกำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆ
2. รับค่าข้อมูลเป็นตัวเลข
3. ตรวจสอบข้อมูล

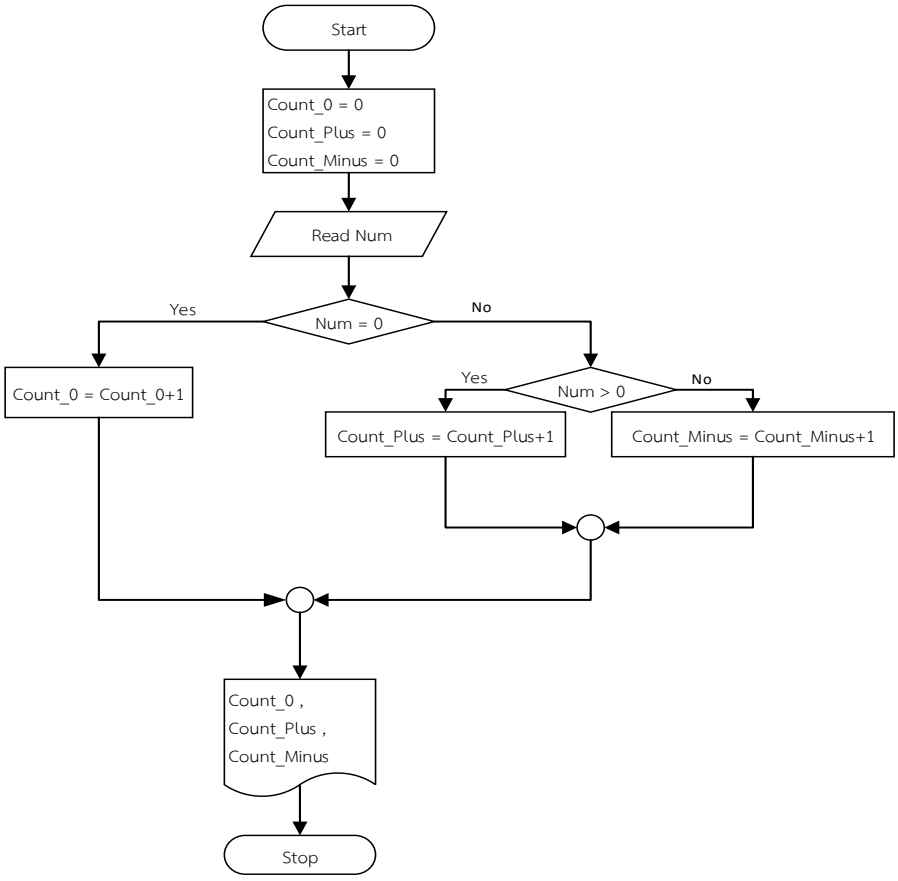
ถ้าเป็นศูนย์	ให้ตัวนับเลขศูนย์เพิ่มค่าขึ้นหนึ่ง
ถ้าไม่ใช่ศูนย์	ให้ตรวจสอบว่าเป็นเลขบวกหรือไม่
ถ้าเป็น	ให้ตัวนับเลขบวกเพิ่มค่าขึ้นหนึ่ง
ถ้าไม่เป็น	ให้ตัวนับเลขลบเพิ่มค่าขึ้นหนึ่ง

4. แสดงผลลัพธ์ที่ต้องการทั้ง 3 ค่า
5. จบการทำงาน

รายละเอียดของตัวแปรที่ใช้เป็นดังนี้

Num	แทนข้อมูลตัวเลขใดๆ
Count_0	แทนค่าการนับจำนวนที่เป็นศูนย์
Count_Plus	แทนค่าการนับจำนวนที่เป็นบวก
Count_Minus	แทนค่าการนับจำนวนที่เป็นลบ

สำหรับซูโดโค้ดสามารถเขียนได้ดังต่อไปนี้



■ สรุปท้ายบท

ในการเขียนโปรแกรมให้ประมวลผลต่างๆ ผู้เขียนโปรแกรมจะต้องออกแบบขั้นตอนการทำงานขึ้นมาก่อน เพื่อใช้ช่วยในการเขียนโปรแกรมว่าจะให้โปรแกรมทำงานในลักษณะใด และใช้สำหรับศึกษาโปรแกรมในภายหลัง โดยจะแสดงว่าข้อมูลที่จะเข้าสู่ระบบเป็นข้อมูลแบบใด มีอะไรบ้าง ต้องการให้โปรแกรมประมวลผลอย่างไร และโปรแกรมให้เอาต์พุตอะไรออกมา การเขียนขั้นตอนการทำงานนี้ อาจใช้คำในภาษาอังกฤษที่เข้าใจง่ายมาเขียน เรียกว่า ซูโดโค้ด (Pseudocodes) หรือรหัสเทียม การเขียนอีกวิธีหนึ่งจะใช้แผนภาพสำหรับอธิบายขั้นตอนการทำงาน แผนภาพนี้เรียกว่า ผังงาน (Flowchart)

■ การทดลอง

1. จงเขียนโฟลวชาร์ตแสดงขั้นตอนการทอดไข่เจียว
2. จงเขียนโฟลวชาร์ตแสดงการตรวจสอบตัวเลขว่าเป็นเลขคู่หรือเลขคี่
3. จงเขียนโฟลวชาร์ตแสดงการหาผลบวกของเลขจำนวนเต็มตั้งแต่ 1 ถึง 20
4. จงเขียนโฟลวชาร์ตของซูโดโค้ดดังต่อไปนี้

```
set total to 0
set number to 0
read amount
     IF EOF
       SKIP
     ELSE
       DO  process  and read UNTIL EOF
            IF amount > 3,000
            add 1 to number
            read amount
            END DO
        END IF
        IF number > 0
          print  number , “Sale over 3,000 Baht”
        ELSE
          print “Sale not over 3,000 Baht”
        END IF
     print total
```

5. จงเขียนซูโดโค้ดของการรับข้อมูลจำนวนสิบค่าและหาค่าเฉลี่ย
6. จงเขียนซูโดโค้ดของการรับข้อมูลจำนวนสิบค่าแล้วแสดงค่าสูงสุดต่ำสุดทางเครื่องพิมพ์

■ แบบฝึกหัดท้ายบทที่ 2

1. จงเขียนขั้นตอนการทำงานและผังงาน ในการถอนเงินจากตู้ ATM
2. จงเขียนขั้นตอนการทำงานและผังงาน เพื่อหาปริมาตรรูปทรงกระบอก (สูตร: $PI*R^2*H$)
3. จงเขียนขั้นตอนการทำงานและผังงาน เพื่อรับข้อมูลน้ำหนักหน่วยเป็นปอนด์ แล้วแปลงให้เป็นกรัม แล้วพิมพ์แสดงผลลัพธ์ (1 ปอนด์ = 454 กรัม)
4. จงเขียนขั้นตอนการทำงานและผังงาน เพื่อรับข้อมูล 10 ค่า แล้วหาค่าเฉลี่ย
5. จงเขียนขั้นตอนการทำงานและผังงาน เพื่อรับข้อมูล 10 ค่า แล้วแสดงค่าสูงสุดต่ำสุด

บทที่ 3
โครงสร้างภาษา C

การเขียนโปรแกรมคอมพิวเตอร์นั้นผู้เขียนโปรแกรมจะต้องทราบหลักการและรูปแบบของการเขียนโปรแกรมภาษานั้นๆ ภาษาซีก็เช่นเดียวกันรูปแบบของภาษานี้ประกอบด้วยส่วนต่างๆ 5 ส่วน ได้แก่ ส่วนที่เป็นพรีโพรเซสเซอร์ไดเร็กทีฟ ส่วนกำหนดค่า ส่วนฟังก์ชันหลัก (main) การสร้างฟังก์ชัน และส่วนอธิบายโปรแกรม โดยผู้พัฒนาโปรแกรมจะต้องทราบว่าแต่ละส่วนนั้นมีรูปแบบการเขียนอย่างไร

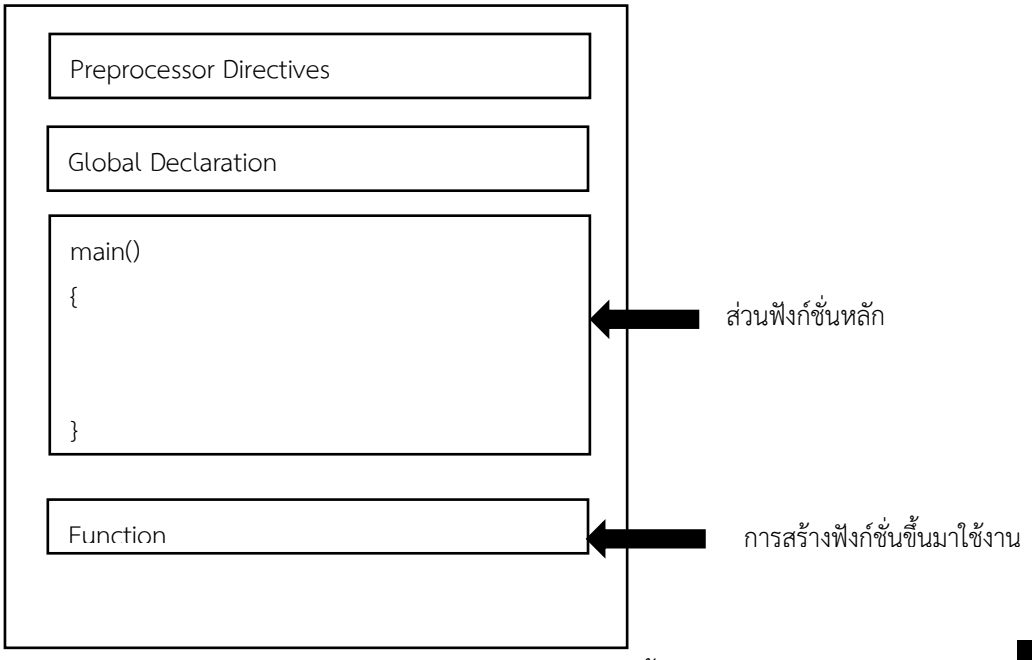
สำหรับในบทนี้จะกล่าวถึงขั้นตอนการเขียนโปรแกรมอย่างง่าย โดยโปรแกรมต่างๆ ที่จะกล่าวถึงตลอดทั้งเล่มนี้จะเขียนขึ้นสำหรับรันบน Dev C++ เป็นหลัก

▪ โครงสร้างโปรแกรม

การเขียนโปรแกรมแต่ละภาษานั้นโครงสร้างของโปรแกรมจะต่างกัน ในบทนี้จะกล่าวถึงโครงสร้างของการเขียนโปรแกรมด้วยภาษาซี (C/C++) รวมทั้งการเขียนโปรแกรมอย่างง่าย ลักษณะโครงสร้างของภาษาซีแบ่งออกได้เป็น 5 ส่วนดังต่อไปนี้

1. พรีโพรเซสเซอร์ไดเร็กทีฟ (Preprocessor directives)
2. ส่วนการกำหนดค่า (Global declarations)
3. ส่วนฟังก์ชันหลัก (The main() function)
4. การสร้างฟังก์ชันและการใช้ฟังก์ชัน (Uses-defined function)
5. ส่วนอธิบายโปรแกรม (Program comments)

โครงสร้างของโปรแกรมประกอบด้วยหลายส่วน แต่ในการเขียนโปรแกรมนั้นเราไม่จำเป็นต้องเขียนหมดทุกส่วน ส่วนใดไม่ใช้ก็สามารถตัดทิ้งได้ แต่ทุกโปรแกรมต้องมีส่วนพรีโพรเซสเซอร์ไดเร็กทีฟ และส่วนฟังก์ชันหลัก รายละเอียดของส่วนต่างๆ เป็นดังต่อไปนี้



1. พรีโพรเซสเซอร์ไดเร็กทีฟ (Preprocessor directives)

ส่วนนี้ทุกโปรแกรมต้องมี จะใช้สำหรับเรียกไฟล์ที่โปรแกรมต้องการในการทำงาน และกำหนดค่าต่างๆ โดยคอมไพเลอร์จะกระทำตามคำสั่งก่อนที่จะคอมไพล์โปรแกรม ซึ่งจะต้องเริ่มต้นด้วยเครื่องหมาย **ไดเร็กทีฟ (directive)** # และตามด้วยชื่อโปรแกรมหรือชื่อตัวแปรที่ต้องการกำหนดค่า ส่วนนี้อาจเรียกอีกชื่อหนึ่งว่าส่วนหัวโปรแกรม (Header Part) ซึ่งจะมีพรีโพเซสเซอร์อยู่หลายคำสั่งดังตารางดังนี้

ไดเร็กทีฟ	ความหมาย
#define	กำหนดค่าคงที่ในโปรแกรม
#include	คำสั่งที่ใช้ผนวกแฟ้มข้อมูลเข้าไปในตัวโปรแกรม
#undef	ไม่ได้กำหนดแมโคร preprocessor
#ifdef	ส่งกลับ true ถ้าแมโครนี้ถูกกำหนดไว้
#ifndef	ส่งกลับ true ถ้าแมโครนี้ไม่ได้กำหนดไว้
#if	ทดสอบเงื่อนไขถ้าเวลาประมวลผลแล้วเงื่อนไขเป็นจริง
#else	ทางเลือกสำหรับ #if
#elif	#else และ #if ในหนึ่ง statement
#endif	เงื่อนไขจบก่อนการประมวลผล
#error	พิมพ์ error message บน stderr
#pragma	ปัญหาคำสั่งพิเศษในการคอมไพเลอร์โดยใช้วิธีการที่ได้มาตรฐาน

ในที่นี้ขอกล่าวเพียง 2 ตัวแรก สำหรับไดเร็กทีฟที่ใช้กันบ่อยๆ ได้แก่

1.1 คำสั่ง #include

เป็นการแจ้งให้คอมไพเลอร์อ่านไฟล์อื่นเข้ามาคอมไพล์ร่วมด้วย รูปแบบการใช้จะทำโดยเขียน #include แล้วตามด้วยชื่อไฟล์ ดังนี้

รูปแบบ

```
#include <ชื่อไฟล์>
หรือ
#include “ชื่อไฟล์”
```

ตัวอย่าง

```
#include <stdio.h>
#include "myheader.h”
```

#include <stdio.h>	หมายความว่า อ่านไฟล์ stdio.h เข้ามาด้วย
#include “Pro1.c”	หมายความว่า อ่านไฟล์ Pro1.c เข้ามาด้วย

การกำหนดชื่อไฟล์ตามหลัง #include นั้น อาจใช้เครื่องหมาย < > คร่อมชื่อไฟล์ก็ได้ซึ่งจะเป็นการอ่านไฟล์จากไดเร็กทอรีที่กำหนดไว้ก่อน แต่ถ้าใช้ “ ” เป็นการอ่านไฟล์จากไดเร็กทอรีปัจจุบันที่กำลังติดต่ออยู่ และไฟล์ที่จะ include เข้ามานี้จะต้องไม่มีฟังก์ชัน main() โดยมากแล้วจะประกอบด้วยโปรแกรมย่อย ค่าคงที่ หรือข้อกำหนดต่างๆ

1.2 คำสั่ง #define

เป็นการกำหนดค่านิพจน์ต่างๆ ให้กับชื่อของตัวแปร โดยมีรูปแบบดังนี้

รูปแบบ

```
#define NAME VALUE
```

ตัวอย่าง

```
#define MAX_ARRAY_LENGTH 20
```

#define END 20 ;	กำหนด END มีค่าเท่ากับ 20
#define A 5*6+3 ;	กำหนด A มีค่าเป็น 5*6+3

มาโครที่กำหนดไว้ล่วงหน้า (Predefined Macros)

อย่างไรก็ตามในภาษา C ยังมีมาโครมาตรฐานที่ถูกนิยามไว้โดยคอมไพเลอร์ ซึ่งจะขึ้นต้นด้วยเครื่องหมาย _ จำนวน 2 ตัวเขียนติดกัน และลงท้ายด้วยเครื่องหมาย_ จำนวน 2 ตัวติดกัน ดังตารางนี้

ชื่อมาโคร	ความหมาย
__LINE__	หมายเลขบรรทัดในโปรแกรมที่ใช้เรียกมาโครนั้นๆ(เป็นค่าคงที่ชนิดเลขจำนวนเต็ม)
__FILE__	ชื่อโปรแกรมของซอร์สโค้ด (เป็นค่าคงที่ชนิดข้อความสตริง)
__DATE__	วันที่ที่คอมไพล์โปรแกรม (เป็นค่าคงที่ชนิดข้อความสตริงโดยมีรูปแบบเป็นวันที่ "MMM DD YYYY")
__TIME__	เวลาที่คอมไพล์โปรแกรม(เป็นค่าคงที่ชนิดข้อความสตริงโดยมีรูปแบบเป็นวันที่ "HH:MM:SS")
__STDC__	ถ้ามีค่าเท่ากับ 1 หมายความว่า คอมไพเลอร์ที่ใช้งานสนับสนุนมาตรฐานภาษา C

ตัวอย่าง

```
#include <stdio.h>

main() {

   printf("File :%s\n", __FILE__ );
   printf("Date :%s\n", __DATE__ );
   printf("Time :%s\n", __TIME__ );
   printf("Line :%d\n", __LINE__ );
   printf("ANSI :%d\n", __STDC__ );
}
```

ผลลัพธ์โปรแกรม

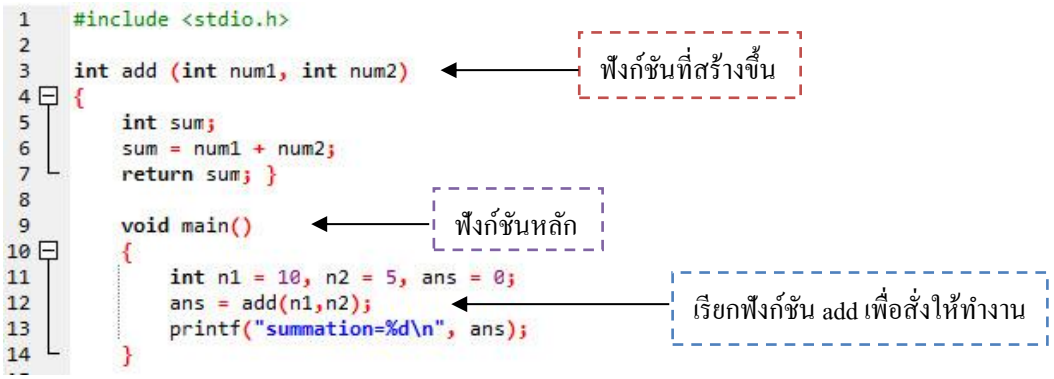
```
File :test.c
Date :Jun 2 2012
Time :03:36:24
Line :8
ANSI :1
```

2. ส่วนประกาศ (Global declarations)

ส่วนนี้จะใช้ในการประกาศตัวแปรหรือฟังก์ชันที่ต้องใช้ในโปรแกรม โดยทุกๆ ส่วนของโปรแกรมสามารถจะเรียกใช้ข้อมูลที่ประกาศไว้ในส่วนนี้ได้ ส่วนนี้บางโปรแกรมอาจไม่มีก็ได้ สำหรับรายละเอียดต่างๆ จะได้กล่าวต่อไป

3. ส่วนฟังก์ชันหลัก (main() function)

ส่วนนี้ทุกโปรแกรมจะต้องมี ซึ่งจะประกอบไปด้วยประโยคคำสั่งต่างๆ ที่จะให้โปรแกรมทำงานโดยนำคำสั่งต่างๆ มาต่อเรียงกัน และแต่ละประโยคคำสั่งจะจบด้วยเครื่องหมายเซมิโคลอน (Semicolon ;) โดยโปรแกรมหลักนี้จะเริ่มต้นด้วย **main()** ตามด้วยเครื่องหมายปีกกาเปิด { และจบด้วยเครื่องหมายปีกกาปิด }



4. ส่วนกำหนดฟังก์ชันขึ้นใช้เอง (Uses-defined function)

เป็นการเขียนคำสั่งและฟังก์ชันต่างๆ ขึ้นใช้ในโปรแกรม โดยต้องอยู่ในเครื่องหมาย { } และต้องสร้างฟังก์ชันหรือคำใหม่ที่ให้ทำงานตามที่เราต้องการให้กับโปรแกรมและสามารถเรียกใช้ได้ภายในโปรแกรม ตัวอย่างเช่น

```
#include <stdio.h>
main()
{
        function();       /*เรียกใช้ฟังก์ชันที่สร้างขึ้น*/
}
function()                 /*สร้างฟังก์ชันใหม่ โดยให้ชื่อว่า function*/
{
        Return ;           /*คืนค่าที่เกิดจากการทำฟังก์ชัน*/
}
```

5. ส่วนอธิบายโปรแกรม (Program comments)

ส่วนนี้ใช้เขียนคอมเมนต์โปรแกรม เพื่ออธิบายการทำงานต่างๆ ทำให้ผู้ศึกษาโปรแกรมในภายหลังทำความเข้าใจโปรแกรมได้ง่ายขึ้น เมื่อคอมไพล์โปรแกรมส่วนนี้จะถูกข้ามไป

```
/* Comment*/ หรือ //Comment
```

ตัวอย่างที่ 3.1

```
#include <stdio.h>

int main() {
   /* my first program in C */
   printf("Hello, World! \n");

   return 0;
}
```

โปรแกรมนี้เมื่อทำงาน (เลือกเมนู Excecute>Run หรือกดคีย์ < F9>) คอมพิวเตอร์จะพิมพ์คำว่า Hello, World! ออกทางจอภาพ พิจารณาจากโปรแกรมจะเห็นได้ว่าในฟังก์ชันหลักมีการเรียกใช้ฟังก์ชัน printf() ซึ่งจะทำหน้าที่พิมพ์ข้อความหรือสตริง (string) ที่อยู่ในเครื่องหมายคำพูดออกทางหน้าจอ และจบฟังก์ชันด้วยเครื่องหมายเซมิโคลอน โดยจะเก็บฟังก์ชันนี้ไว้ใน **stdio** (ย่อมา

จาก standard input output) ซึ่งจะเก็บชุดคำสั่งเกี่ยวกับการส่งข้อมูลเข้าออกเอาไว้ เราจึงต้องเรียก stdio.h ขึ้นมา

จากตัวอย่างโปรแกรมที่ 3.1 จะพิมพ์คำว่า Hello, World! และมีเครื่องหมายแบ็คสแลช (Backslash) และตามด้วยตัว n (\n) เพิ่มขึ้นมา ซึ่งจะเป็นตัวบอกว่าถ้าพิมพ์คำว่า Hello, World! จบแล้วให้ขึ้นบรรทัดใหม่ ซึ่งรายละเอียดของฟังก์ชัน printf() ยังมีอีกมากซึ่งจะได้ศึกษาต่อไป

■ สรุปท้ายบท

โครงสร้างของภาษาซีที่ทุกโปรแกรมจะต้องมีคือ ส่วนที่เป็นพรีโพรเซสเซอร์ไดเร็กทีฟ และส่วนฟังก์ชันหลัก พรีโพรเซสเซอร์ไดเร็กทีฟจะเป็นตัวบอกให้คอมไพเลอร์อ่านไฟล์ขึ้นมาคอมไพล์ร่วมด้วย ส่วนฟังก์ชันหลักจะใช้คำว่า main() และมีเครื่องหมายปีกกาเปิดและปีกกาปิด ฟังก์ชันต่างๆที่จะให้โปรแกรมทำงานจะอยู่ภายในเครื่องหมายปีกกานี้

■ การทดลอง

1. จงบอกส่วนประกอบของซอร์สโค้ดโปรแกรมแต่ละบรรทัดต่อไปนี้

#include <stdio.h>
#include <conio.h>
 	
main()
{
float fMoney, fTax;
 	
printf("Enter the income is yearly : ");
scanf("%f", &fMoney);
fTax = fMoney * 0.07;
printf("\nThe income is yearly : %.2f", fTax);
printf("\n123456 Dr.Lawan Dulyachart ");
 	
getch();
}
	1. .....
	2. .....
	
	3. .....
	4. .....
	5. .....
	
	6. .....
	7. .....
	8. .....
	9. .....
	10. .....
	
	11. .....
	12. .....

2. จงเขียนโปรแกรมเพื่อทดสอบผลลัพธ์ที่ได้

```
#include <stdio.h>

int main() {
   /* print message */
   printf("Kalasin University. \n"); // \n is newline
   printf("Faculty of Education. \n");
   return 0;
}
```

3. จงเขียนโปรแกรมเพื่อทดสอบผลลัพธ์ที่ได้

```
#include <stdio.h>

main() {
        /* print message box */
   printf("****************************** \n");
   printf("**                            ** \n");
   printf("**        Hello ComEd         ** \n");
   printf("**                            ** \n");
   printf("****************************** \n");
}
```

■ แบบฝึกหัดท้ายบทที่ 3

1. จากโค้ด การทดลองข้อ 2 จงเขียนโปรแกรมเพิ่มเพื่อสั่งพิมพ์ข้อความเพิ่มเติมอีก 2 บรรทัด ดังนี้

```
Hello, Computer Education.
This is my first C program.
```

2. จงเขียนโปรแกรมเพื่อหาผลลัพธ์

```
#include <stdio.h>

 int main()
{
  char string[] = "Hello World";
   printf("%s\n", string);
```

```
   return 0;
}
```

3. จงเขียนโปรแกรม เพื่อหาผลลัพธ์

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

int main()
{
  while (TRUE)
  {
    printf("Hello World\n");
  }
   return 0;
}
```

4. จงเขียนโปรแกรม เพื่อหาข้อผิดพลาดของโปรแกรม

```
#include <stdio.h>

int Main() {
   /* print message */
   printf("Please verify again.\n"):
   printf("This is my program.\n")
   return 0;
```

เมื่อนำไปคอมไพน์แล้วเกิดข้อผิดพลาด ดังนี้

```
D:\Example\C\10struct\3-โครงสร้าง\ex-04.cpp: In function 'int Main()':
D:\Example\C\10struct\3-โครงสร้าง\ex-04.cpp:5:36: error: expected ';' before ':' token
    printf("Please verify again.\n"):
                                    ^
D:\Example\C\10struct\3-โครงสร้าง\ex-04.cpp:7:12: error: expected '}' at end of input
    return 0;
           ^
```

จงแก้ไขให้ถูกต้องพร้อมทั้งอธิบายเหตุผล

บทที่ 4
ประเภทของข้อมูล และตัวดำเนินการ

การเขียนโปรแกรมคอมพิวเตอร์นั้นจะต้องมีการประมวลผลกับข้อมูล โดยข้อมูลจะถูกเก็บอยู่ในหน่วยความจำของคอมพิวเตอร์ในรูปแบบของตัวแปร การประกาศตัวแปรต่างๆ จะใช้หน่วยความจำไม่เท่ากัน และมีช่วงของการเก็บข้อมูลไม่เท่ากัน ผู้เขียนโปรแกรมจะต้องทราบว่าข้อมูลที่ต้องการประมวลผลนั้นเป็นข้อมูลประเภทใด และในการประมวลผลจะต้องมีการกระทำกับตัวแปรต่างๆตัวที่นำมากระทำเรียกว่าตัวดำเนินการ ซึ่งมีการดำเนินการทางคณิตศาสตร์และทางลอจิก ดังนั้นผู้เขียนโปรแกรมจะต้องทำความเข้าใจกับประเภทของข้อมูล และการใช้ตัวดำเนินการ จึงสามารถเขียนโปรแกรมให้ทำงานตามที่ต้องการได้

จากตัวอย่างโปรแกรมอย่างง่ายที่ผ่านาในบทที่3 จะเห็นว่าถ้าจะให้คอมพิวเตอร์ประมวลผลจะต้องทำการเขียนชุดคำสั่งที่เรียกว่าโปรแกรมให้กับคอมพิวเตอร์ และการทำงานของโปรแกรมอาจต้องมีการรับข้อมูลเข้าไปเพื่อทำการประมวลผล ข้อมูลที่รับเข้าไปอาจอยู่ในรูปของค่าคงที่ หรือตัวแปร โดยค่าคงที่เป็นค่าที่มีค่าคงที่ตลอดโปรแกรม ส่วนตัวแปรจะเป็นค่าในหน่วยความจำที่สามารถเปลี่ยนแปลงได้ในการทำโปรแกรม ตัวอย่างในโปรแกรมที่ผ่านมาจะเห็นว่ามีการประกาศตัวแปรเป็น integer เพื่อเก็บค่าที่เป็นเลขจำนวนเต็ม และค่าในตัวแปรนั้นสามารถนำมาประมวลผลและมีการเปลี่ยนแปลงได้

■ ประเภทของข้อมูล

การประกาศข้อมูลในการเขียนโปรแกรมจะเป็นการกำหนดชื่อของข้อมูล หรือกำหนดประเภทของข้อมูลขึ้นมาใหม่โดยข้อมูลในภาษาซีอาจแบ่งออกได้เป็น 4 กลุ่มดังต่อไปนี้

- ข้อมูลชนิดซิมเปิล (simple type)
- ข้อมูลประเภทสตริง (string type)
- ข้อมูลประเภทโครงสร้าง (structure type)
- ข้อมูลประเภทพอยน์เตอร์ (pointer)

ในหัวข้อนี้กล่าวถึงข้อมูลชนิดซิมเปิลและข้อมูลชนิดสตริงก่อน ส่วนข้อมูลประเภทอื่นๆจะกล่าวต่อไปในภายหลัง

1. ข้อมูลชนิดซิมเปิล

ข้อมูลชนิดซิมเปิลแบ่งได้เป็นข้อมูลประเภทลำดับ(Ordinal Type) และข้อมูลประเภทจำนวนจริง (Real Data Type)

1) ข้อมูลประเภทลำดับ (Ordinal Type)

ข้อมูลแบบลำดับเป็นข้อมูลที่มีค่าเป็นลำดับแน่นอน เช่น ตัวเลขที่ใช้ในการนับ ลำดับตัวอักษรเป็นต้น ในภาษาซียังแบ่งข้อมูลชนิดลำดับออกได้หลายประเภท ในที่นี้จะกล่าวถึงข้อมูลประเภทจำนวนเต็ม ข้อมูลอักขระ และข้อมูลตรรกะ

- **ข้อมูลชนิดจำนวนเต็ม (Integer Data Type) :** ข้อมูลประเภทนี้จะใช้เก็บตัวเลขที่เป็นจำนวนเต็ม ในคอมพิวเตอร์จะใช้หน่วยความจำในการเก็บข้อมูล ถ้าหากคอมพิวเตอร์ใช้หน่วยความจำ 8 บิตหรือ 1 ไบต์ในการเก็บข้อมูล จะทำให้เก็บข้อมูลที่เป็นเลขฐานสิบได้ในช่วง 0 ถึง 255 แต่ถ้าใช้หน่วยความจำมากกว่านั้นในการเก็บข้อมูล ก็จะสามารถเก็บตัวเลขช่วงที่กว้างขึ้นได้

ข้อมูลชนิดจำนวนเต็มนี้ยังแบ่งได้หลายประเภทขึ้นกับขนาดของหน่วยความจำที่คอมพิวเตอร์ใช้เก็บ โดยข้อมูลประเภทต่างๆ แสดงได้ดังตารางต่อไปนี้

ตารางที่ 4.1 แสดงประเภทของข้อมูลชนิดจำนวนเต็ม

ประเภท	ความหมาย	ขนาด (ไบต์)	ช่วงของข้อมูลที่เก็บได้
char	ตัวอักขระ	1	-128 to 127 or 0 to 255
unsigned char	ตัวอักขระ ไม่รวมเครื่องหมาย	1	0 to 255
signed char	ตัวอักขระรวมเครื่องหมาย	1	-128 to 127
int	จำนวนเต็ม	2 or 4	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	ไม่รวมเครื่องหมาย	2 or 4	0 to 65,535 or 0 to 4,294,967,295
short	จำนวนเต็มแบบสั้น	2	-32,768 to 32,767
unsigned short	จำนวนเต็มแบบสั้น ไม่รวมเครื่องหมาย	2	0 to 65,535
long	จำนวนเต็มแบบยาว	4	-2,147,483,648 to 2,147,483,647
unsigned long	จำนวนเต็มแบบยาว ไม่รวมเครื่องหมาย	4	0 to 4,294,967,295

(ขนาดข้อมูลอาจไม่ตรงได้ ขึ้นอยู่กับของคอมไพเลอร์ของภาษา C ที่ใช้งาน)

สามารถใช้โอเปเรเตอร์ sizeof(type) เพื่อนับจำนวนของตัวแปรได้ตัวอย่างต่อไปนี้

ตัวอย่าง 4.1

```c
#include <stdio.h>
#include <limits.h>
int main() {
   printf("Storage size for int : %d \n", sizeof(int));
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Storage size for int : 4
```

- **ข้อมูลประเภทตัวอักขระ (Character Data Type) :** ข้อมูลประเภทนี้จะเป็นตัวอักขระหนึ่งตัว ซึ่งเป็นไปตามตารางรหัส ASCII ประกอบด้วยข้อมูลที่เป็นตัวอักษร ตัวเลข และอักขระพิเศษ ข้อมูลประเภทนี้จะเป็นข้อมูลแบบลำดับได้ เนื่องจากเรียงลำดับรหัส ASCII ข้อมูลประเภทนี้จะใช้เนื้อที่ในการเก็บหนึ่งไบต์ เช่น

'A','B','C'

นอกจากนี้ยังมีอักขระพิเศษ เช่น

'\n' รหัสขึ้นบรรทัดใหม่ '\t' รหัสเว้นวรรค 1 tab '\a' เสียง Beep

- **ข้อมูลประเภทตรรกะ (Boolean Data Type) :** จะเป็นค่าทางลอจิก ได้แก่ จริง (True) กับ เท็จ (False) จะใช้ในคำสั่งควบคุมเพื่อตัดสินใจการทำงาน ในการเรียงลำดับจะให้ค่าที่เป็นเท็จลำดับก่อนค่าที่เป็นจริง ในการเขียนโปรแกรมบางครั้งจะแทนค่าจริงด้วยเลขจำนวนเต็ม 1 หรือค่าที่มากกว่า 1 และแทนค่าเท็จด้วยเลข 0

2) ข้อมูลประเภทจำนวนจริง (Real Data Type)

ข้อมูลประเภทนี้จะเป็นจำนวนจริงหรือเลขทศนิยม ข้อมูลประเภทนี้จะจัดลำดับก่อนหลังได้ยาก จึงไม่เป็นข้อมูลชนิดลำดับเนื่องจากทศนิยมมีได้หลายตำแหน่ง ข้อมูลจำนวนจริงนี้ยังแบ่งออกได้หลายประเภท โดยแต่ละประเภทจะใช้หน่วยความจำในการเก็บแตกต่างกัน ทำให้เก็บข้อมูลได้แตกต่างกัน ดังตารางต่อไปนี้

ตารางที่ 4.2 แสดงประเภทของข้อมูลชนิดจำนวนจริง

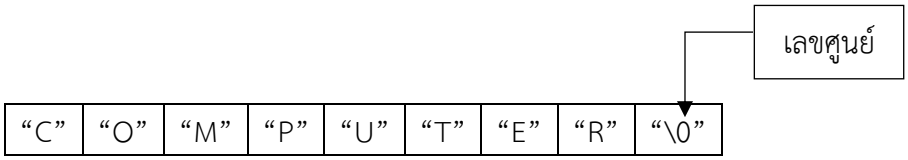
ประเภท	ความหมาย	ขนาด (ไบต์)	ช่วงของข้อมูลที่เก็บได้
float	เลขทศนิยม	4	1.2E-38 to 3.4E+38(6 decimal places)
double	เลขทศนิยม 2 เท่า	8	2.3E-308 to 1.7E+308 (15 decimal places)
long double	เลขทศนิยม 2 เท่า แบบยาว	10	3.4E-4932 to 1.1E+4932 (19 decimal places)

(ขนาดข้อมูลอาจไม่ตรงได้ ขึ้นอยู่กับของคอมไพเลอร์ของภาษา C ที่ใช้งาน)

3) ข้อมูลประเภทสตริง (string type)

นอกจากข้อมูลแบบตัวเลขและตัวอักษรแล้ว ในภาษาซียังมีข้อมูลอีกประเภทหนึ่งเรียกว่า **สตริง** ข้อมูลประเภทนี้จะเป็นการนำตัวอักขระมาต่อเรียงกันเป็นข้อความตั้งแต่หนึ่งตัวขึ้นไป โดย

สามารถเก็บตัวอักขระได้ 255 ตัวโดยตัวอักขระต้องอยู่ในเครื่องหมาย “ ” ในการเขียนโปรแกรมด้วยภาษาซี จะมีการเติมตัวอักษรว่าง NULL (\0) เป็นตัวสุดท้าย อย่างเช่นการเก็บสตริงคำว่า “COMPUTER” จะใช้เนื้อที่ในการเก็บ 9 ไบต์ โดยแต่ละไบต์เป็นดังนี้



■ การประกาศตัวแปรและค่าคงที่

ในการเขียนโปรแกรมถ้าหากต้องการรับข้อมูลจากภายนอกมาเก็บไว้ หรือต้องการเก็บผลลัพธ์ระหว่างการประมวลผลจะต้องมีตัวแปรสำหรับเก็บ จากโครงสร้างของโปรแกรมในบทที่ผ่านมาจะเห็นว่าในโปรแกรมจะมีส่วนประกาศ ซึ่งอยู่ต่อจากส่วนชื่อโปรแกรม โดยส่วนประกาศนี้ในโปรแกรมบางโปรแกรมอาจไม่มี แต่ถ้าหากโปรแกรมต้องการใช้ค่าคงที่ หรือใช้ตัวแปรต่างๆ ผู้เขียนโปรแกรมจะต้องมีส่วนนี้สำหรับประกาศ

1. การประกาศตัวแปร

การสร้างตัวแปรขึ้นมาใช้งานเรียกว่าการประกาศตัวแปร ในการเขียนโปรแกรมคอมพิวเตอร์ส่วนใหญ่จะต้องมีการประกาศตัวแปรเสมอ ตัวอย่างเช่น ในโปรแกรมในบทที่ผ่านมามีการรับข้อมูลจากแป้นพิมพ์เข้าสู่คอมพิวเตอร์ โดยจะมีการประกาศตัวแปรเอาไว้ ข้อมูลที่รับเข้ามาจะถูกนำไปเก็บในตัวแปรที่ประกาศเอาไว้ การประกาศตัวแปรสามารถทำได้ดังนี้

1) การประกาศตัวแปรที่ละตัวหรือกลุ่มตัวแปร

รูปแบบ

```
type variable_list;
```

โดยที่ type หมายถึง ชนิดข้อมูลของตัวแปร variable หมายถึงชื่อของตัวแปรในการประกาศตัวแปร สามารถประกาศครั้งละหลายตัวได้ถ้าหากเป็นตัวแปรประเภทเดียวกันจะใช้เครื่องหมาย , คั่น ตัวอย่างเช่น ถ้าหากจะประกาศตัวแปรชื่อ Data1 และ Data2 สำหรับเก็บจำนวนเต็มสามารถทำได้ดังนี้

ตัวอย่าง การประกาศตัวแปร

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

2) การประกาศตัวแปร พร้อมกับกำหนดค่าเริ่มต้น

เป็นการประกาศตัวแปรสามารถประกาศพร้อมกับกำหนดค่าเริ่มต้นให้กับตัวแปรได้ดังนี้

รูปแบบ

```
type variable_name = value;
```

ตัวอย่าง

```
extern int d = 3, f = 5;    // declaration of d and f.
int d = 3, f = 5;           // definition and initializing d and f.
byte z = 22;                // definition and initializes z.
char x = 'x';               // the variable x has the value 'x'.
```

3) การประกาศตัวแปรชนิดค่าคงที่

เป็นการประกาศตัวแปรพร้อมประกาศพร้อมกำหนดค่าเริ่มต้นให้กับตัวแปรนั้น จะไม่สามารถเปลี่ยนค่าที่หลังได้อีก

ตัวอย่าง

```
const double e = 2.71889898989;
const char msg[ ] = “Warning....”;
```

2. การประกาศค่าคงที่

ค่าคงที่(Constant) เป็นค่าในหน่วยความจำที่มีค่าคงที่ตลอดโปรแกรม ในการประกาศค่าคงที่จะเป็นการกำหนดชื่อให้ค่าคงที่ ถ้าในโปรแกรมส่วนใดเรียกชื่อที่ประกาศไว้ก็จะได้ข้อมูลตามที่กำหนด การประกาศค่าคงที่จะใช้คำว่า const นำหน้า ซึ่งทำได้สองลักษณะดังต่อไปนี้

1) ค่าคงที่ที่นิยามด้วย #define (#define Preprocessor)

ในภาษา C ได้เตรียมพรีโพรเซสเซอร์ไดเร็กทีฟ ชื่อ #define เพื่อนำมาใช้สำหรับกำหนดค่าคงที่ในโปรแกรม ซึ่งจะกำหนดไว้ต้นโปรแกรม โดยมีรูปแบบการเขียนดังนี้

รูปแบบ

```
#define identifier value
```

โดยที่ define คือ พรีโพรเซสเซอร์ไดเร็กทีฟ
identifier คือ ชื่อค่าคงที่ ซึ่งมักกำหนดเป็นตัวพิมพ์ใหญ่
value คือ ค่าของค่าคงที่

ตัวอย่าง

```
#include <stdio.h>

#define LENGTH 10
#define WIDTH  5
```

```
#define NEWLINE '\n'

int main() {

   int area;

   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
value of area : 50
```

2) ค่าคงที่ที่ระบุค่าโดยตรงลงไป (const Keyword)

รูปแบบ

```
const type variable = value;
```

ตัวอย่าง

```
#include <stdio.h>

int main() {

   const int  LENGTH = 10;
   const int  WIDTH = 5;
   const char NEWLINE = '\n';
   int area;

   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);

   return 0;
```

ผลลัพธ์โปรแกรม

value of area : 50

■ กฎการตั้งชื่อตัวแปร

การประกาศตัวแปรจะต้องมีการกำหนดชื่อให้กับตัวแปร เพื่อให้โปรแกรมทำงาน กฎการตั้งชื่อในภาษาซียังใช้กับชื่อต่างๆ ในโปรแกรมได้อีกด้วย ตัวอย่างเช่น การกำหนดชื่อโปรแกรม ชื่อของตัวแปรต่าง เป็นต้น ในตัวอย่างที่ผ่านมาเราได้เห็นการตั้งชื่อตัวแปรและค่าคงที่มาบ้างแล้ว การตั้งชื่อในภาษา C รูปแบบดังนี้

1. ชื่อจะต้องไม่ซ้ำกับคำสงวน(Reserved word) และคำมาตรฐานที่คอมไพเลอร์รู้จัก
2. จะต้องขึ้นต้นด้วยตัวอักษร (A-Z,a-z) หรือเครื่องหมาย_ (Underscore) เท่านั้น
3. ตัวต่อไปต้องเป็นตัวอักษรหรือตัวเลขหรือเครื่องหมาย_
4. การตั้งชื่อจะต้องไม่มีช่องว่าง
5. ตัวอักษรตัวเล็กและตัวอักษรตัวใหญ่จะมีความหมายแตกต่างกัน

คำสงวน (Keywords) เป็นคำที่มีความหมายที่โปรแกรมรู้จัก โดยมีรูปแบบการใช้งานที่แน่นอน ส่วนคำมาตรฐานเป็นคำที่มีความหมายอยู่แล้ว โปรแกรมสามารถเรียกใช้งานได้เลย คำสงวนในภาษาซี ได้แก่

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

ตัวอย่าง การตั้งชื่อที่ถูกต้อง

a	month	MONTH	_var1
salary	stdCode	num1	day_of_week
EmpID	NAME	FLOAT	topofWindow

ตัวอย่าง การตั้งชื่อที่ไม่ถูกต้อง

222_cpu	7-eleven	do	total$
*point	Emp no	Num1,2	@mails

■ ตัวดำเนินการ (Operator)

ในการเขียนโปรแกรมตัวดำเนินการจะเป็นตัวที่ทำหน้าที่รวมค่าต่างๆ และกระทำกับค่าต่างๆ ให้เป็นค่าเดียวกัน อย่างเช่นโปรแกรมในบทที่ผ่านมามีการนำข้อมูลที่เป็นตัวแปรมาคูณกับค่าคงที่ซึ่งจะต้องใช้ตัวดำเนินการทางคณิตศาสตร์เพื่อทำการคูณ ตัวดำเนินการมีหลายประเภทดังต่อไปนี้

1. ตัวดำเนินการทางคณิตศาสตร์
2. ตัวดำเนินการเปรียบเทียบ
3. ตัวดำเนินการทางตรรกะ
4. ตัวดำเนินการระดับบิต
5. ตัวดำเนินการกำหนดค่า
6. ตัวดำเนินการอื่นๆ (Misc Operators ↦ sizeof & ternary)

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

ใช้สำหรับกระทำการคำนวณทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร โดยจะนำข้อมูลตัวหนึ่งไปกระทำอีกตัวหนึ่ง โดยให้ผลลัพธ์เป็นตัวเลขทางคณิตศาสตร์ สมมุติว่า A = 10, B = 20 ดังนี้

ตัวดำเนินการ	กระบวนการ	ตัวอย่าง
+	บวก(Addition)	A + B = 30
-	ลบ(Subtraction)	A – B = -10
*	คูณ(Multiplication)	A * B = 200
/	หาร(Real number Division)	B / A = 2
%	การหารแบบเอาเศษ(Modulus)	B % A = 0
++	การเพิ่มค่าขึ้นหนึ่ง(Increment)	A++ = 11
--	การลดค่าลงหนึ่ง(Decrement)	A-- = 9

ถ้านำเลขจำนวนจริงไปกระทำกับเลขใด ผลลัพธ์ที่ออกมาจะเป็นตัวเลขจำนวนจริง

2. ตัวดำเนินการเปรียบเทียบ (Relational Operators)

ตัวดำเนินการเปรียบเทียบ (Relational Operators)จะนำข้อมูลสองค่ามาเปรียบเทียบกัน โดยข้อมูลสองค่าจะต้องเป็นข้อมูลประเภทเดียวกัน ผลลัพธ์ที่ได้จะเป็นค่าทางลอจิกคือจริงหรือเท็จ จากตัวอย่างในตาราง สมมุติว่า ตัวแปร A = 10, ตัวแปร B = 20 ดังนี้

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
==	เท่ากับ	(A == B) is not true.
!=	ไม่เท่ากับ	(A != B) is true.
>	มากกว่า	(A > B) is not true.
<	น้อยกว่า	(A < B) is true.
>=	มากกว่าหรือเท่ากับ	(A >= B) is not true.
<=	น้อยกว่าหรือเท่ากับ	(A <= B) is true.

3. ตัวดำเนินการทางตรรกะ (Logical Operator)

ตัวดำเนินการทางตรรกะ (Logical Operator) ประกอบด้วย การทำ AND , OR และ NOT เมื่อกระทำกับค่าใด ผลลัพธ์ที่ออกมาจะเป็นจริงหรือเท็จ สมมุติว่า ตัวแปร A = 10, ตัวแปร B = 20 ดังนี้ ตัวดำเนินการทางตรรกะแสดงได้ดังตารางต่อไปนี้

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&&	AND ค่าสองค่า ถ้าค่าทั้งสองเป็นจริงผลลัพธ์จะเป็นจริง	(A && B) is false.
||	OR ค่าสองค่า ถ้าค่าทั้งสองเป็นเท็จผลลัพธ์จะเป็นเท็จ	(A || B) is true.
!	NOT เปลี่ยนค่าเป็นจริงเป็นเท็จ จากเท็จเป็นจริง	!(A && B) is true.

ซึ่งผลที่ได้จากตารางดังกล่าวจะเป็นไปตามตารางค่าความจริงดังนี้

ตัวดำเนินการ		ผลลัพธ์		
A	B	A&&B	A||B	!A
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

ตัวอย่าง กำหนดให้ i เป็นตัวแปรชนิดจำนวนเต็ม ซึ่งมีค่าเท่ากับ 7 และ f เป็นตัวแปรจำนวนจริง มีค่าเท่ากับ 5.5 ส่วน c คือ ตัวแปรชนิดตัวอักษร มีค่าเท่ากับ ‘w’ ผลลัพธ์ดังนี้

นิพจน์	ผลที่ได้ค่า
(i>=6) && (c== ‘w’)	1
(i>=6) || (c== 119)	1
(f<11) && (i>100)	0
(c!= ‘p’) || ((i+f)<= 10)	1

4. ตัวดำเนินการระดับบิต (Bitwise Operators)

ตัวดำเนินการประเภทนี้ ทำให้การเขียนโปรแกรมด้วยภาษาซีทำงานคล้ายกับภาษาแอสเซมบลี ได้โดยการกระทำแบบบิต มักจะใช้ในการทดสอบบิต เลื่อนบิต เซ็ตบิต โดยนำไบต์หรือเวิร์ดข้อมูลมากระทำต่อกัน แต่ข้อมูลจะถูกดำเนินการแบบบิตต่อบิต ตัวดำเนินการแบบนี้จะใช้กับตัวแปรประเภท char หรือ int เท่านั้น ไม่สามารถใช้กับตัวแปรประเภทอื่นได้ สมมติว่า ตัวแปร 'A' เก็บ 60 และตัวแปร 'B' เก็บ 13 ตัวดำเนินการแบบบิตแสดงได้ดังตารางต่อไปนี้

ตาราง แสดงโอเปอเรเตอร์ระดับบิต

ตัว ดำเนินการ	ความหมาย	ตัวอย่าง
&	ดำเนินการ AND	(A & B) = 12, i.e., 0000 1100
|	ดำเนินการ OR	(A | B) = 61, i.e., 0011 1101
^	ดำเนินการ Exclusive OR (XOR)	(A ^ B) = 49, i.e., 0011 0001
~	กลับค่าบิต(1’s complement)	(~A) = -61, i.e,. 1100 0011 in 2's complement form.
<<	เลื่อนทุกบิตไปทางซ้าย	A << 2 = 240 i.e., 1111 0000
>>	เลื่อนทุกบิตไปทางขวา	A >> 2 = 15 i.e., 0000 1111

ซึ่งผลที่ได้จากตารางดังกล่าวจะเป็นไปตามตารางค่าความจริงดังนี้

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

ตัวอย่าง สมมุติว่า A = 60 และ B = 13 ในรูปแบบไบนารี จะได้ผลดังนี้

A = 0011 1100

B = 0000 1101

ผลลัพธ์

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A = 1100 0011

5. ตัวดำเนินการกำหนดค่า (Assignment Operators)

ในภาษาซีมีหลายวิธีในการกำหนดค่าให้กับตัวแปร ในภาษา C มีหลายวิธีด้วยกันในการกำหนดค่าให้กับตัวแปร ซึ่งปกติตัวดำเนินการกำหนดค่ามักใช้เครื่องหมาย = และการกำหนดค่า

นิพจน์ก็จะใช้เครื่องหมาย = เช่นกัน สำหรับรูปแบบของตัวดำเนินการกำหนดค่า สามารถเขียนได้ตามรูปแบบดังนี้

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
=	กำหนดค่า	C = A + B จะกำหนดค่า A + B = C
+=	การบวกแบบลดรูป	C += A มีค่าเท่ากับ C = C + A
-=	การลบแบบลดรูป	C -= A มีค่าเท่ากับ C = C - A
*=	การคูณแบบลดรูป	C *= A มีค่าเท่ากับ C = C * A
/=	การหารแบบลดรูป	C /= A มีค่าเท่ากับ C = C / A
%=	การหารเอาเศษแบบลดรูป	C %= A มีค่าเท่ากับ C = C % A
<<=	โอเปอเรเตอร์กำหนดค่าเลื่อนเลื่อนบิตไปทางซ้าย	C <<= 2 เหมือนกับ C = C << 2
>>=	โอเปอเรเตอร์กำหนดค่าเลื่อนเลื่อนบิตไปทางขวา	C >>= 2 เหมือนกับ C = C >> 2
&=	โอเปอเรเตอร์กำหนดค่าระดับบิต AND	C &= 2 เหมือนกับ C = C & 2
^=	Bitwise exclusive OR และโอเปอเรเตอร์กำหนดค่า	C ^= 2 เหมือนกับ C = C ^ 2

ตัวอย่าง ต่อไปมาดูตัวอย่างการใช้ตัวดำเนินการกับตัวแปรและผลที่ได้จากการการะทำ

คำสั่ง	**ลักษณะการทำงาน**	**ผลที่ได้**
m += 4;	m = m + 4;	เพิ่มค่า 4 ให้ตัวแปร m
a--;	a=a-1;	ลดค่าตัวแปร a ลงไป 1
m *=2	m=m*2;	คูณค่า m ด้วย 2 แล้วเก็บใน m
a / = 2;	a = a / 2;	หาร a ด้วย 2 แล้วเก็บใน a

6. ตัวดำเนินการอื่นๆ (Misc Operators ↦ sizeof & ternary)

นอกจากนี้ผู้ประกอบการที่กล่าวถึงข้างต้นยังมีโอเปอเรเตอร์อื่น ๆ ที่สำคัญรวมทั้ง sizeof และ? : ที่สนับสนุนโดยภาษา C

โอเปอเรเตอร์	ความหมาย	ตัวอย่าง
sizeof()	ส่งกลับค่าขนาดของตัวแปร	sizeof(a), a คือจำนวนเต็มจะส่งคืนค่า 4
&	ส่งกลับค่าที่อยู่ของตัวแปร	&a; ส่งคืนค่าที่อยู่ของตัวแปร
*	ชี้ตำแหน่งของตัวแปร	*a;
? :	นิพจน์เงื่อนไข	ถ้าเงื่อนไขเป็นจริง แล้วค่า X ฉะนั้นค่า Y

7. ลำดับความสำคัญของตัวดำเนินการ

ในการเขียนโปรแกรมเพื่อใช้งานจำเป็นต้องเกี่ยวข้องกับนิพจน์ทางคณิตศาสตร์มากกว่าหนึ่งตัวในนิพจน์เดียวกัน จึงจำเป็นที่ผู้เรียนจำต้องเรียนรู้ลำดับความสำคัญก่อนหลังของตัวดำเนินการแต่ลำตัว เพื่อให้ได้ผลลัพธ์การคำนวณที่ถูกต้อง

ตาราง ลำดับความสำคัญของตัวดำเนินการ จากความสำคัญสูงสุดลงมาหาต่ำสุด

ประเภท	ตัวดำเนินการ	กรณีลำดับความสำคัญเท่ากัน	ลำดับความสำคัญ
Postfix	() [] -> . ++ - -	ซ้ายไปขวา	
Unary	+ - ! ~ ++ - - (type)* & sizeof	ขวาไปซ้าย	
Multiplicative	* / %	ซ้ายไปขวา	
Additive	+ -	ซ้ายไปขวา	
Shift	<< >>	ซ้ายไปขวา	
Relational	< <= > >=	ซ้ายไปขวา	
Equality	== !=	ซ้ายไปขวา	
Bitwise AND	&	ซ้ายไปขวา	
Bitwise XOR	^	ซ้ายไปขวา	
Bitwise OR	|	ซ้ายไปขวา	
Logical AND	&&	ซ้ายไปขวา	
Logical OR	||	ซ้ายไปขวา	
Conditional	?:	ขวาไปซ้าย	
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	ขวาไปซ้าย	
Comma	,	ซ้ายไปขวา	

■ สรุปท้ายบท

ตัวแปรและค่าคงที่ เป็นส่วนของข้อมูลพื้นฐานที่สามารถถูกกำหนด และยักย้ายเทค่าได้ในภาษา C ตัวแปรต่างๆ ที่นำมาใช้งาน จำเป็นต้องได้รับการประกาศขึ้นมาก่อนเสมอ ตัวแปรที่กำหนดขึ้นในโปรแกรม จะต้องระบุชนิดข้อมูลว่าเป็นข้อมูลชนิดใด

กฎการตั้งชื่อตัวแปร ประกอบด้วย

1. สามารถนำตัวอักขระ ไม่ว่าจะเป็นตัวอักษรพิมพ์ใหญ่ A – Z ตัวอักษรพิมพ์เล็ก a – z และตัวเลข 0 -9 รวมถึงเครื่องหมาย _ (Underscore) มาใช้เพื่อการตั้งชื่อตัวแปรได้
2. อักษรตัวพิมพ์ใหญ่ และอักษรตัวพิมพ์เล็กมีความแตกต่างกัน
3. ชื่อตัวแปรจะต้องไม่ตรงกับคำสงวน
4. ชื่อตัวแปรกำหนดได้ยาวไม่จำกัด แต่จะพิจารณาความแตกต่างของตัวอักษร 31 ตัวแรก
5. ชื่อตัวแปรไม่อนุญาตให้มีช่องว่าง หากต้องการแยกคำ สามารถใช้เครื่องหมาย _ ช่วยได้

6. ควรตั้งชื่อตัวแปรที่สามารถสื่อความหมาย เพื่อบอกให้รู้ว่าตัวแปรนั้นๆ นำไปใช้เพื่อการใด

ภาษา C จะมีชนิดข้อมูลพื้นฐาน 4 ชนิด ดังนี้

1. char เป็นชนิดข้อมูลแบบตัวอักษร
2. int เป็นชนิดข้อมูลแบบเลขจำนวนเต็ม
3. float เป็นชนิดข้อมูลแบบเลขจำนวนจริง
4. double เป็นชนิดข้อมูลแบบเลขจำนวนจริง 2 เท่า

นอกจากชนิดข้อมูลพื้นฐานทั้ง 4 แล้ว ยังสามารถปรับปรุงด้วยการเพิ่มเครื่องหมายนำหน้าชนิดข้อมูลได้อีก ซึ่งประกอบด้วย signed, unsigned, long และ short

การประกาศชนิดข้อมูลให้กับตัวแปรต่างๆ จะต้องกำหนดให้ตรงกับค่าของข้อมูลที่จัดเก็บและควรเลือกขนาดให้เหมาะสมด้วย เพื่อมิให้สิ้นเปลืองเนื้อที่หน่วยความจำโดยใช่เหตุ

ปกติค่าข้อมูลที่จัดเก็บลงในตัวแปร จะสามารถเปลี่ยนแปลงค่าได้ตลอดเวลาในโปรแกรม แต่หากเป็นค่าคงที่ จะหมายถึงค่าที่ถูกกำหนดขึ้นในโปรแกรมแล้ว จะไม่สามารถเปลี่ยนแปลงค่าในภายหลังได้อีก

ในภาษา C สามารถกำหนดค่าคงที่ได้ในรูปแบบของ

1. ค่าคงที่ที่ระบุโดยตรงลงไป
2. ค่าคงที่ที่นิยามด้วย #define
3. ค่าคงที่ที่เก็บในตัวแปร

ในภาษา C สามารถประกาศตัวแปรในรูปแบบต่างๆ ได้ดังนี้

1. ประกาศแบบทีละตัว หรือประกาศเป็นกลุ่มตัวแปรที่มีชนิดข้อมูลเหมือนกันภายในบรรทัดเดียวกันด้วยการใช้เครื่องหมาย , คั่นระหว่างชื่อตัวแปร
2. ประกาศตัวแปร พร้อมกำหนดค่าเริ่มต้นให้กับตัวแปร
3. ประกาศตัวแปรชนิดค่าคงที่

ตัวดำเนินการหรือตัวดำเนินการที่ใช้งานในภาษา C ประกอบด้วย

1. ตัวดำเนินการทางคณิตศาสตร์
2. ตัวดำเนินการเปรียบเทียบ
3. ตัวดำเนินการทางตรรกะ
4. ตัวดำเนินการระดับบิต
5. ตัวดำเนินการกำหนดค่า
6. ตัวดำเนินการอื่นๆ (Misc Operators ↦ sizeof & ternary)

ตัวดำเนินการแต่ละตัว จะมีลำดับความสำคัญที่แตกต่างกันออกไป โดยตัวดำเนินการที่มีความสำคัญสูงจะถูกประมวลผลก่อน ในขณะที่ตัวดำเนินการที่มีความสำคัญต่ำลงไป จะถูกประมวลผลทีหลัง

กรณีที่นิพจน์มีตัวดำเนินการที่มีลำดับความสำคัญเท่ากัน ก็จะต้องพิจารณาลำดับการประมวลผลของตัวดำเนินการนั้นๆ ว่าจะคำนวณจากซ้ายไปขวา หรือจากขวาไปซ้าย

การสร้างนิพจน์หรือสูตรทางคณิตศาสตร์ จำเป็นต้องคำนึงถึงความสำคัญของตัวดำเนินการแต่ละตัวด้วย แต่อย่างไรก็ตาม เราสามารถใช้เครื่องหมาย ( ) ช่วยได้ กรณีที่สูตรคำนวณหรือนิพจน์มี

ความซับซ้อน ทั้งนี้ตัวดำเนินการวงเล็บจะมีความสำคัญลำดับสูงสุดที่จะถูกประมวลผลก่อน และหากมีเครื่องหมายวงเล็บซ้อนอยู่ภายในอีก นิพจน์ที่อยู่ภายในวงเล็บภายในสุดจะถูกประมวลผลก่อน

■ การทดลอง

1. จงเขียนโปรแกรมคำนวณ $x^2 + 8x + 4$ โดยให้รับค่า x ทางแป้นพิมพ์
2. จงเขียนโปรแกรมรับค่าปี พ.ศ. ทางอินพุต แล้วแสดงผลเป็นปี ค.ศ.
3. จงเขียนโปรแกรมหาพื้นที่สามเหลี่ยม โดยอินพุตรับค่าฐานและความสูง

■ แบบฝึกหัดท้ายบทที่ 4

ตอนที่ 1 จงทำเครื่องหมาย√ หน้าข้อที่ถูกต้องและทำเครื่องหมาย × หน้าข้อที่ไม่ถูกต้อง

.....1. ถ้าหากต้องการประกาศตัวแปรสำหรับเก็บอายุของบุคคลทั่วไปควรประกาศเป็น int

.....2. การประกาศตัวแปรหลายตัวแล้วไม่ใช้จะทำให้เปลืองหน่วยความจำ

.....3. ถ้าหากเก็บค่า 40000 ลงในตัวแปรแบบ int ข้อมูลที่ได้จะเป็นค่าลบ

.....4.การประกาศเก็บตัวแปรสำหรับเก็บสตริง ควรประกาศให้มีขนาดใหญ่กว่าข้อความเสมอ

.....5.ถ้าประกาศตัวแปรเป็นแบบ char จะใช้หน่วยความจำ 2 ไบต์

.....6. คำว่า xyz123 สามารถใช้เป็นชื่อตัวแปรได้

.....7. นิพจน์ a = a+1 จะทำงานเหมือนกับ ++a

.....8. การหารแบบ mod (%) ผลลัพธ์จะเป็นข้อมูลแบบจำนวนเต็มเท่านั้น

.....9. ตัวดำเนินการทางตรรกะสามารถกระทำกับเลขทศนิยมได้

.....10. นิพจน์ y = a++; และ y = ++a; การทำงานจะเหมือนกัน

ตอนที่ 2 จงตอบคำถามต่อไปนี้

1. ถ้าเขียนประโยคภาษาซีให้คอมพิวเตอร์คำนวณค่า $Y = ax^3 + 7$ จะเขียนได้ดังข้อใน
   ก. Y = a * x * x * x + 7;
   ข. Y = a * x * x * (x + 7);
   ค. Y = (a * x)* x *(x + 7);
   ง. Y = (a * x) * x * x + 7;
   จ. Y = a * (x * x * x) + 7;
   ฉ. Y = a * x * (x * x + 7);
2. จงหาค่าผลลัพธ์จากการทำงานประโยคต่อไปนี้
   ก. X = 7 + 3 * 6 / 2 – 1 ;
   ข. X = 2 % 2 + 2 * 2 – 2 / 2
   ค. X =(3 * 9 * (3 + (9 * 3 / (3))));
3. จงอธิบายการทำงานต่อไปนี้
   ก. X = x + 1;
   ข. X += 1;
   ค. ++x;

ง. X++;

4. ถ้าหาก x เท่ากับ 2 และ y = 3 จงบอกผลลัพธ์จากการทำคำสั่งต่อไปนี้

ก. printf(“%d”,x);

ข. printf(“%d”,x + x);

ค. printf(“x =”);

ง. printf(“x = %d”,x);

จ. printf(“%d = %d”,x + y,y + x);

ฉ. z = x + y;

5. จงหาผลลัพธ์จากการทำนิพจน์ต่อไปนี้

ก. X = (int)8.30 * (int) 4.3;

ข. C&&(a>=b); เมื่อ a มีค่าเป็น 23 , b มีค่าเป็น -80 และ c มีค่าเป็น 0

บทที่ 5
การรับและแสดงผลข้อมูล

บทนี้จะกล่าวถึงการรับข้อมูลผ่านคีย์บอร์ดเข้ามาในโปรแกรม (input) เพื่อนำมาประมวลผล รวมทั้งวิธีการแสดงผลข้อความหรือผลลัพธ์ของโปรแกรมออกทางจอภาพ(output) ซึ่งจะขอกล่าวถึงรายละเอียดการทำงานของเอาท์พุตก่อน

■ การรับและแสดงผลข้อมูล (เฮดเดอร์ไฟล์ stdio.h)

ฟังก์ชันที่ใช้เพื่อการรับและแสดงผลข้อมูลที่ประกาศในเฮดเดอร์ stdio.h ประกอบด้วย 6 ฟังก์ชัน คือ getchar(), putchar(), scanf(), printf() และ puts()

1. ฟังก์ชัน getchar() และputchar()

1) ฟังก์ชัน getchar()

ฟังก์ชัน getchar() เป็นหนึ่งในฟังก์ชันที่บรรจุอยู่ในไลบรารีมาตรฐาน I/O โดยจะรีเทิร์นค่าอักขระหนึ่งตัวที่ถูกอินพุตเข้ามา ทั้งนี้ตัวอักขระที่ป้อนเข้ามาจะแสดงทางจอภาพ และจะต้องยืนยันการป้อนข้อมูลด้วยการเคาะปุ่ม Enter ทุกครั้ง อย่างไรก็ตาม กรณีที่มีการป้อนอักขระหลายๆ ตัว จะมีเพียงอักขระตัวแรกเท่านั้นที่ถูกนำไปใช้งานหรือจัดเก็บไว้ในตัวแปร และเนื่องจากฟังก์ชัน getchar() ไม่ต้องการค่าอาร์กิวเมนต์ใดๆ ดังนั้นจึงสามารถใส่วงเล็บว่างเปล่าได้ เช่น getchar()

รูปแบบ

```
Character_variable  =  getchar() ;
```

ตัวอย่างเช่น

```
getchar() ;
```

และในกรณีที่ต้องการนำค่าที่ป้อน จัดเก็บไว้ในตัวแปร ก็จะเขียนในรูปแบบดังนี้

```
char  ch1;
ch1  =  getchar();
```

2) ฟังก์ชัน putchar()

ฟังก์ชัน putchar() เป็นฟังก์ชันที่ใช้แสดงตัวอักขระหนึ่งตัวทางจอภาพ ซึ่งอาจนำมาใช้แสดงค่าที่ป้อนมาจากฟังก์ชัน getchar () หรืออาจกำหนดให้แสดงค่าอักขระโดยตรง

รูปแบบ

```
putchar  ( character_variable ) ;
```

ตัวอย่างเช่น

```
char ch1  = 'A' ;
putchar(ch1) ;
```

```
putchar('B');
```

```
putchar(66);  // display character B
```

ตัวอย่าง การใช้ฟังก์ชัน getchar() และputchar()

```
#include <stdio.h>
int main( ) {

   int c;

   printf( "Enter a value :");
   c = getchar( );

   printf( "\nYou entered: ");
   putchar( c );

   return 0;
}
```

ผลลัพธ์ของโปรแกรม

```
$./a.out
Enter a value : this is test
You entered: t
```

■ 2. ฟังก์ชัน gets() และ puts()

นอกจากจะให้ฟังก์ชั่น scanf() และ printf() ในการรับและแสดงผลข้อมูลแล้ว ยังมีฟังก์ชั่นที่ทำงานกับสตริงโดยเฉพาะด้วย คือ gets() ที่ใช้ในการรับข้อมูลสติง และ puts() ที่ใช้ในการแสดงผลข้อมูลสตริง

1) ฟังก์ชั่น gets()

มาจากคำว่า get string เป็นฟังก์ชั่นสำหรับอ่านข้อมูลจากคีย์บอร์ดมาเก็บไว้ในหน่วยความจำ ณ ตำแหน่งที่ตัวแปรชนิดอาร์เรย์ซึ่งอยู่ ซึ่งการอ่านข้อมูลของฟังก์ชั่น gets() นี้ หากข้อความที่ป้อนเข้ามา ประกอบด้วยช่องว่าง (space) ก็ไม่เป็นปัญหา ฟังก์ชั่น gets() ยังคงอ่านข้อความต่อไปได้ สิ่งที่จะทำให้ ฟังก์ชั่น gets() หยุดอ่านข้อความมีเพียงกรณีเดียว คือ เมื่อกด [Enter] ซึ่งเมื่อป้อนข้อความและกด [Enter] แสดงการสิ้นสุดการป้อนข้อความแล้ว ฟังก์ชั่น gets()จะใส่ '\0' (null string) ปิดท้ายให้กับข้อความ โดยอัตโนมัติด้วย

2) ฟังก์ชั่น puts()

มาจากคำว่า put string เป็นฟังก์ชั่นสำหรับพิมพ์สตริงออกทางจอภาพ หลักการใช้งานของฟังก์ชั่นนี้ง่ายๆ คือ ให้ส่งค่าที่อยู่ (address) ของสตริงเข้ามาที่ฟังก์ชั่นนี้เท่านั้น

ภาษา C ได้จัดเตรียมฟังก์ชันเพื่อการรับและการแสดงผลข้อมูลมาให้หลายรูปแบบด้วยกัน โดยเฉพาะการนำไปใช้เพื่อการถ่ายโอนข้อมูล ไม่ว่าจะเป็นการถ่ายโอนข้อมูลกันภายใน หรือส่งออกไปยังภายนอก และฟังก์ชันgets() ก็เป็นอีกฟังก์ชันหนึ่งที่นำมาใช้สำหรับการรับข้อมูลประเภทสตริง ส่วนฟังก์ชัน puts() ก็นำมาใช้สำหรับแสดงผลลัพธ์ข้อมูลประเภทสตริง

ข้อมูลประเภทสตริง คือกลุ่มข้อความ ซึ่งท้ายข้อความจะมีการผนวกค่า Null หรือรหัส \0 ปะต่อท้ายเพื่อใช้บ่งบอกถึงจุดสิ้นสุดของข้อความนั้นๆ ทั้งนี้การจัดเก็บข้อมูลสตริงในภาษา C จะจัดเก็บในรูปแบบของอาร์เรย์ สำหรับฟังก์ชัน gets() และ puts() ถือเป็นอีกทางเลือกหนึ่งของการนำไปใช้เพื่อการรับค่าและแสดงผล แทนที่จะใช้ฟังก์ชัน scanf() หรือ printf() เท่านั้น ให้ลองพิจารณาตัวอย่างโปรแกรมดังต่อไปนี้

```
#include <stdio.h>
int main( ) {

   char str[100];

   printf( "Enter a value :");
   gets( str );

   printf( "\nYou entered: ");
   puts( str );

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
$./a.out
```

Enter a value : this is test **You entered:** this is test

ตัวอย่าง เปรียบเทียบการทำงานระหว่างฟังก์ชั่น gets() และ scanf()

#include <stdio.h>
 	
main() {
char name[20];
printf("What's your name?\n");
gets(name);
printf("Hi %s, nice to meet you"
,name);
}
	#include <stdio.h>
	
	main() {
	char name[20];
	printf("What's your name?\n");
	scanf("%s", name);
	printf ("Hi %s, nice to meet you"
	,name);
	}

ผลลัพธ์ของโปรแกรม

What's your name? Dr.Lawan Dulyachart Hi Dr.Lawan Dulyachart, nice to meet you	What's your name? Dr.Lawan Dulyachart Hi Dr.Lawan, nice to meet you

อธิบายโปรแกรม

อย่างที่ได้กล่าวไว้แล้วข้างต้นว่า ฟังก์ชั่น gets() สามารถรับค่าได้ แม้กระทั่งคำที่เป็นช่องว่าง (space) โดยจะรับค่าไปจนกระทั่งมีการกด [Enter]จึงจะหยุดการรับคำ ดังนั้น ผลลัพธ์ของโปรแกรมทาง ด้านซ้ายมือที่ใช้ฟังก์ชั่นgets()ในการรับคำ จึงแสดงชื่อ Dr.Lawan ออกมาได้ต่างจากโปรแกรมทางด้านขวามือที่ใช้ฟังก์ชั่น scanf()ในการรับคำ ซึ่งแสดงชื่อได้เพียง Dr.Lawan เท่านั้น ทั้งที่ป้อนเข้ามาเป็น Dr.Lawan Dulyachart สาเหตุที่เป็นเช่นนี้เพราะฟังก์ชั่น scanf()นั้นเมื่ออ่านพบช่องว่างจะตัดข้อความออก คือ สำหรับกรณีนี้จะแยก Dr.Lawan และ Dulyachart ออกจากกัน (ต้องระบุ %s ในส่วนของ string format อีกอันหนึ่งและน่าตัวแปรชนิดสตริงมารับค่าอีกตัวแปรหนึ่งจึงจะได้ข้อความ Dulyachart เข้ามาในโปรแกรม) นอกจากนั้น scanf()ยังไม่มีการเติม '\0' เพื่อปิดสตริงให้

3. ฟังก์ชัน printf() และ scanf()

1) ฟังก์ชัน printf

ฟังก์ชั่นที่ใช้ในการแสดงผลข้อมูลออกทางหน้าจอ คือ **printf** (print formatted) มีหน้าที่หลัก คือ แปลงข้อมูลในลักษณะเลขฐานสอง (Binary) ที่คอมพิวเตอร์ประมวลผลได้ ให้อยู่ในรูปแบบที่มนุษย์เข้าใจ ก่อนแสดงผลข้อมูลออกทางจอภาพ

รูปแบบ

```
printf (“string_format” ,data_list);
```

string_format คือ สตริงที่ต้องการแสดงผล ซึ่งอาจเป็นข้อความธรรมดา เช่น Hello, C Language หรือเป็นสัญลักษณ์แทนชนิดข้อมูลต่างๆ ซึ่งจะถูกแทนด้วยค่าคงที่ ,ตัวแปร หรือนิพจน์ใดๆที่กำหนดมาเป็นพารามิเตอร์ (ในส่วนของ data_list)ตั้งแต่ตัวที่ 2 เป็นต้นไป เช่น %d ใช้แทนชนิดข้อมูลเลขจำนวนเต็ม %C ใช้แทนชนิดอักษร เป็นต้น

data_lits คือ ข้อมูลที่จะแสดงผล ซึ่งอาจเป็นค่าคงที่ ตัวแปร หรือนิพจน์ใดๆ

ตารางที่ 5.1 รหัสรูปแบบข้อมูลชนิดต่างๆที่ใช้ในฟังก์ชัน printf()

รหัสรูปแบบ	ชนิดข้อมูล
%c	ตัวอักขระหนึ่งตัว
%d	เลขจำนวนเต็ม
%ld	เลขจำนวนเต็มแบบยาว
%e	เลขจำนวนจริงแบบเอ็กซ์โปเนนต์
%f	เลขจำนวนจริง
%i	เช่นเดียวกับ %d
%o	เลขฐานแปด
%s	ข้อความสตริง
%u	เลขจำนวนเต็ม ไม่มีเครื่องหมาย
%x	เลขฐานสิบหก

นอกจากนี้ภายใน “format control string” ยังสามารถใส่รหัสควบคุม (Escape Sequence) เข้าไปได้อีก ซึ่งรหัสควบคุมเหล่านี้จัดเป็นส่วนหนึ่งของคำสั่งควบคุมการแสดงผล ด้วยการใช้เครื่องหมาย \ (backslash)และตามด้วยรหัสควบคุม ซึ่งรหัสดังกล่าวแสดงไว้ดังรูปที่ 5.2 ต่อไปนี้

ตารางที่ 5.2 แสดงรหัสควบคุม(Escape Character)

รหัส	ความหมาย
\0	ค่าว่าง (null character)
\a	ส่งเสียง 1 ครั้ง (bell)
\b	ถอยหลัง 1 หนึ่งช่องตัวอักษร(backspace)
\f	ขึ้นหน้าใหม่ (form feed)
\n	ขึ้นบรรทัดใหม่ (new line)
\r	เลื่อนเคอร์เซอร์ไปทางซ้ายสุด(carriage return)
\t	แท็บแนวนอน(horizontal tab)
\v	แท็บแนวตั้ง(vertical tab)

\’	พิมพ์เครื่องหมาย '
\"	พิมพ์เครื่องหมาย "
\\	พิมพ์เครื่องหมาย \

2) ฟังก์ชัน scanf()

ฟังก์ชันscanf() คือ ฟังก์ชั่นที่รับข้อมูลจากคีย์บอร์ด โดยข้อมูลที่รับเข้ามาเป็นตัวเลข อักขระหนึ่งตัว หรือข้อความสตริงก็ได้

รูปแบบ

```
scanf(“string_format” , address_list);
```

string_format ต่างจาก string format ของฟังก์ชั่น print() ตรงที่ string format ของฟังก์ชั่น scanf() จะอยู่ในรูปแบบของตัวแทนชนิดข้อมูลต่างๆ เช่น %d , %c , %s , %f,....เท่านั้น

address_list เป็นตัวระบุที่อยู่ (Address) ในหน่วยความจำที่จะใช้ในการเก็บข้อมูลที่รับเข้ามานั้นโดย address list จะต่างกับ data list ของฟังก์ชั่น printf() ตรงที่ data list เป็นระบุถึงข้อมูลโดยตรง ทำให้อ้างถึงตัวแปรที่เก็บข้อมูลนั้นๆได้โดยตรง ส่วน address list เป็นการอ้างถึงที่อยู่ ดังนั้นจะเรียกโดยอ้างอิงถึงแปรโดยตรงไม่ได้ แต่ต้องอ้างถึงโดยใช้ **ampersand (&)** นำหน้าชื่อตัวแปรที่ต้องการรับข้อมูลเข้ามาเก็บไว้ ซึ่งเป็นระบุที่อยู่ของตัวแปร

ตารางที่ 3.3 รหัสรูปแบบข้อมูลชนิดต่างๆที่ใช้ในฟังก์ชัน scanf()

รหัสรูปแบบ	ชนิดข้อมูล
%c	ตัวอักขระหนึ่งตัว
%d	เลขจำนวนเต็ม
%ld	เลขจำนวนเต็มแบบยาว
%e	เลขจำนวนจริงแบบเอ็กซ์โปเนนต์
%f	เลขจำนวนจริง
%i	เช่นเดียวกับ %d
%o	เลขฐานแปด
%s	ข้อความสตริง
%u	เลขจำนวนเต็ม ไม่มีเครื่องหมาย
%x	เลขฐานสิบหก

ตัวอย่าง โปรแกรมแสดงความแตกต่างระหว่าง address_list ของฟังก์ชั่น scanf() กับ data list ของฟังก์ชั่น printf()

```
#include <stdio.h>
int main( ) {
```

```
    char str[100];
    int i;
    printf( "Enter a value :");
    scanf("%s %d", str, &i);
    printf( "\nYou entered: %s %d ", str, i);
    return 0;
}
```

ผลลัพธ์โปรแกรม

```
$./a.out
Enter a value : seven 7
You entered: seven 7
```

การใช้ฟังก์ชั่น scanf() รับข้อมูลชนิดสตริงเข้าทางคีย์บอร์ดนั้น เราจะไม่ใส่ &นำหน้าตัวแปรที่ใช้รับค่าเนื่องจากภาษา C กำหนดให้ตัวแปรชนิดสตริง (ซึ่งก็คืออาร์เรย์ของข้อมูล char) เป็นการอ้างถึงที่อยู่ของตัวแปรนั้นๆ อยู่แล้ว

■ การรับแป้นคีย์ ล้างจอภาพ และกำหนดตำแหน่งแสดงผลทางจอภาพ (เฮดเดอร์ไฟล์ conio.h)

จากเนื้อหาที่ผ่านมา ได้เรียนรู้ถึงฟังก์ชันการรับข้อมูล ไม่ว่าจะเป็น getchar(), scanf() และ gets() มาแล้ว โดยการใช้งานฟังก์ชันดังกล่าว จำเป็นต้องผนวกเฮดเดอร์ไฟล์ stdio.h เข้าไปที่ต้นโปรแกรมด้วย และสำหรับเนื้อหาถัดจากนี้ไป จะเกี่ยวข้องกับเฮดเดอร์ไฟล์ conio.h ที่ประกอบด้วยฟังก์ชันต่างๆ ที่นำมาใช้เพื่อการรับแป้นคีย์ ฟังก์ชันล้างจอภาพ และฟังก์ชันกำหนดตำแหน่งข้อมูลทางจอภาพ เป็นต้น

1. ฟังก์ชัน getch() และ getche()

ทั้งฟังก์ชัน getch() และ getchar() ถูกประกาศไว้ในเฮดเดอร์ไฟล์ conio.h ดังนั้นเมื่อใช้งานฟังก์ชันทั้งสอง จึงต้องผนวกเฮดเดอร์ไฟล์ conio.h ที่ต้นโปรแกรม สำหรับการรับข้อมูลด้วยฟังก์ชัน getch() และ getchar()นั้น จะมีความคล้ายคลึงกัน กล่าวคือ จะเป็นฟังก์ชันที่รอรับการป้อนข้อมูลด้วยคีย์ใดก็ได้เพียงหนึ่งตัว โดยไม่ต้องยืนยันด้วยการเคาะแป้น Enter ทั้งนี้ฟังก์ชัน getch() จะไม่แสดงข้อมูลที่ป้อนเข้าไป ในขณะที่ฟังก์ชันgetch() จะแสดงข้อมูลที่ป้อนทางจอภาพ

getch() อ่าน 1 ตัวอักษรจากคีย์บอร์ด แต่ไม่แสดงตัวอักษรที่รับเข้ามาออกทางหน้าจอ

getche() อ่าน 1 ตัวอักษรจากคีย์บอร์ด และแสดงตัวอักษรที่รับเข้ามาออกทางหน้าจอด้วย

ทั้ง ฟังก์ชั่น getch() และ getche() ถูกประกาศไว้ในเฮดดอร์ไฟล์ conio.h ที่นำมาเพื่อใช้ในการรับแป้นคีย์ ฟังก์ชันล้างจอภาพ และฟังก์ชันกำหนดตำแหน่งข้อมูลจอภาพ เป็นต้น

ตัวอย่าง เปรียบเทียบการทำงานของฟังก์ชั่น getchar(), getch() แล: getche()

```
#include<stdio.h>

main() {
char c;
   do {
   c = getchar();
   } while(c != 'E');
}
```

```
#include<stdio.h>
#include<conio.h>
main() {
char c;
   do {
   c = getch();
   } while(c != 'E');
}
```

```
#include<stdio.h>
#include<conio.h>
main() {
char c;
   do {
   c = getche();
   } while(c != 'E');
}
```

ผลลัพธ์ของโปรแกรม

```
a
b
c
e
d
e
D
E
```

```

```

```
abcdeDE
```

อธิบายโปรแกรม

โปรแกรมที่1 : ใช้ฟังก์ชั่นgetchar() ในการรับตัวอักษรจากคีย์บอร์ด ซึ่งเมื่อป้อนตัวอักษรเข้ามาแล้วจะต้องกด บ[Enter] ทุกครั้ง กล่าวคือฟังก์ชั่น getchar() จะยัง ไม่รับตัวอักษรไปประมวลผลจนกว่าผู้ใช้จะกด Enterและเมื่อกด Enter แล้วเคอร์เซอร์ก็จะย้ายไปยังบรรทัดถัดไป

โปรแกรมที่2 : ใช้ฟังก์ชั่นgetch() ในการรับตัวอักษรจากคีย์บอร์ด จะเห็นว่าไม่ปรากฏผลลัพธ์ใดๆ บนจอภาพ ที่เป็นเช่นนี้เนื่องจากฟังก์ชั่น getch() จะไม่แสดงผลตัวอักษร ที่ป้อนเข้าไปออกทางหน้าจอ และเมื่อป้อนตัวอักษรแล้วตัวอักษรนั้นจะถูกนำ ไปประมวลผลทันที ผู้ใช้ไม่ต้องทำการกด Enter ใดๆทั้งสิ้น (ฟังก์ชั่น getch() มีการประกาศไว้ในเฮดเดอร์ไฟล์ conio.h ดังนั้น จึงต้องรวมเฮดเดอร์ไฟล์นี้ เข้าไว้ในโปรแกรมด้วย)

โปรแกรมที่ 3 : ใช้ฟังก์ชั่นgetche() ในการรับตัวอักษรจากคีย์บอร์ด จะเห็นว่าฟังก์ชั่นgetche() แสดงผลตัวอักษรทีป้อนเข้าไปออกทางหน้าจอเหมือนโปรแกรม ที่ 1 ด้วย แต่ฟังก์ชั่นgetche() ต่างจากฟังฟังก์ชั่น getchar() เพียงเล็กน้อย คือ สำหรับฟังก์ชั่น getche() นั้น ผู้ใช้ไม่ต้องทำการกด Enter โดยเมื่อป้อน ตัวอักษรแล้ว ตัวอักษรจะถูกนำไปประมวลผลทันที (ฟังก์ชั่น getche() มีการประกาศไว้ในเฮดเดอร์ไฟล์conio.h ดังนั้น จึงต้องรวมเฮดเดอร์ไฟล์นี้ เข้าไว้ในโปรแกรมด้วย)

ฟังก์ชัน	การกด Enter	แสดงผลทางจอภาพ
getchar()	✓	✓
getch()	✗	✗
getche()	✗	✓

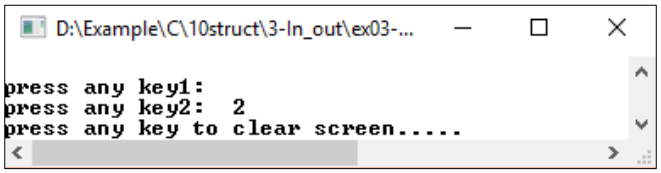
2. ฟังก์ชัน clrscr()

ฟังก์ชัน clrscr() เป็นฟังก์ชันที่ใช้สำหรับล้างจอภาพ

ตัวอย่าง โปรแกรมตัวอย่างการใช้งานฟังก์ชัน getch(), getche() และclrscr()

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

main()
{
   char ch;
   printf("\npress any key1:  ");    getch();
   printf("\npress any key2:  ");    getche();
   printf("\npress any key to clear screen.....");
   ch =   getch();
   system("cls");// ใช้ system("cls");แทน clrscr() ใน Dev-C++
   printf("\nAscii value is ch is %d\n", ch);

}
```

ผลลัพธ์โปรแกรม



ถ้านักศึกษาเขียนโปรแกรม **ภาษา C** ด้วย **Dev-C++** คงจะเคยเจอปัญหาเวลาที่เราต้องการ **ล้างหน้าจอ (Clear Screen)** ด้วยคำสั่ง **clrscr()** ปรากฏว่า Error เกิดอะไรขึ้น !! ให้นำคำสั่ง **system("cls")**; นี้ ไปใช้แทนคำสั่ง clrscr()

3. ฟังก์ชัน gotoxy()

ฟังก์ชัน gotoxy() เป็นฟังก์ชันที่ใช้สำหรับกำหนดตำแหน่งคอลัมน์และแถวบนจอภาพ โดยอาจใช้เป็นตำแหน่งแสดงข้อความหรือข้อมูลต่างๆ รวมถึงตำแหน่งรับข้อมูล เป็นต้น โดยจอภาพแบบเท็กซ์โหมดจะมีอยู่ 25 แถว แต่ละแถวจะมี 80 คอลัมน์

รูปแบบ

```
gotoxy (column, row) ;
```

ตัวอย่าง

```
void gotoxy(int x, int y)
{
  static HANDLE h = NULL;
  if(!h)
    h = GetStdHandle(STD_OUTPUT_HANDLE);
  COORD c = { x, y };
  SetConsoleCursorPosition(h,c);
}
```

gotoxy() เป็นฟังก์ชันมาตรฐาน C กำหนดไว้ใน <conio.h> แต่มันจะไม่ทำงานในคอมไพเลอร์ ANSI C เช่น Dev-C ++ ทำไม? เพราะ gotoxy () เป็นฟังก์ชันที่เฉพาะเจาะจง Turbo-C ++ ซึ่งหมายความว่ามันไม่ได้เป็นส่วนหนึ่งของมาตรฐาน แต่ถ้าคุณยืนยันในการใช้ฟังก์ชันคอนโซลคุณสามารถกำหนดฟังก์ชันของคุณเองโดยใช้ฟังก์ชันของสมาชิกที่มีอยู่ใน <windows.h>

getche(), clsscr() เป็นฟังก์ชันมาตรฐาน C กำหนดไว้ใน <conio.h> แต่มันจะไม่ทำงานในคอมไพเลอร์ ANSI C เช่น Dev-C ++ เพราะเป็นฟังก์ชันที่เฉพาะเจาะจง ของ Borland's C++

■ สรุปท้ายบท

ฟังก์ชั่นที่ใช้เพื่อรับข้อมูลและแสดงผลที่ประกาศไว้ในฮดเดอร์ไฟล์ stdio.h ประกอบด้วย 6 ฟังก์ชั่น คือ getchar(),putchar(),scanf(),printf(),gets() และ puts()

ฟังก์ชั่น getchar() เป็นฟังก์ชั่นที่นำมาใช้สำหรับรับอักขระทีละตัวอักษรจากคีย์บอร์ด และต้องยืนยันด้วยการกด Enter ทุกครั้ง

ฟังก์ชั่น putchar() เป็นฟังก์ชั่นที่นำมาใช้สำหรับแสดงตัวอักขระทางจอภาพ

ฟังก์ชั่น scanf() เป็นฟังก์ชั่นที่รับข้อมูลจากคีย์บอร์ด ซึ่งข้อมูลดังกล่าวอาจเป็นได้ทั้งตัวเลข อักขระหนึ่งตัว และข้อความสตริง

การใช้ฟังก์ชั่น scanf() รับข้อมูลชนิดสตริงเข้าทางคีย์บอร์ดนั้นเราจะใส่ & เข้าไปหน้าชื่อตัวแปร**ยกเว้น**ตัวแปรชนิดสตริง (ซึ่งก็คืออาร์เรย์ของข้อมูล char) เป็นการอ้างถึงที่อยู่ของตัวแปรนั้นๆ อยู่แล้ว

<table>
<tr><th>ฟังก์ชัน</th><th colspan="2">หน้าที่การทำงาน</th><th>ตัวอย่างการใช้งาน</th></tr>
<tr><td>getch()</td><td colspan="2">รับข้อมูลทีละตัวอักษรจากคีย์บอร์ด</td><td>ch = getch();</td></tr>
<tr><td>getchar()</td><td colspan="2">รับข้อมูลทีละตัวอักษรจากคีย์บอร์ด</td><td>ch = getchar();</td></tr>
<tr><td>gets()</td><td colspan="2">รับข้อมูลเป็นข้อความจากคีย์บอร์ด</td><td>gets(str);</td></tr>
<tr><td rowspan="12">scanf</td><td colspan="2">การแสดงผลทุกชนิด ซึ่งมีรหัสรูปแบบดังนี้</td><td>scanf(“%d”, intCount);</td></tr>
<tr><td>รหัสรูปแบบ</td><td>ชนิดข้อมูล</td><td rowspan="11">scanf (“%s”, &str);</td></tr>
<tr><td>%c</td><td>ตัวอักขระหนึ่งตัว</td></tr>
<tr><td>%d</td><td>เลขจำนวนเต็ม</td></tr>
<tr><td>%ld</td><td>เลขจำนวนเต็มแบบยาว</td></tr>
<tr><td>%e</td><td>เลขจำนวนจริงแบบเอ็กซ์โปเนนต์</td></tr>
<tr><td>%f</td><td>เลขจำนวนจริง</td></tr>
<tr><td>%i</td><td>เช่นเดียวกับ %d</td></tr>
<tr><td>%o</td><td>เลขฐานแปด</td></tr>
<tr><td>%s</td><td>ข้อความสตริง</td></tr>
<tr><td>%u</td><td>เลขจำนวนเต็ม ไม่มีเครื่องหมาย</td></tr>
<tr><td>%x</td><td>เลขฐานสิบหก</td></tr>
</table>

ฟังก์ชั่น printf() เป็นฟังก์ชั่นที่นำมาใช้สำหรับสั่งพิมพ์ข้อมูลออกทางจอภาพ ซึ่งข้อมูลดังกล่าวอาจเป็นได้ทั้งข้อความ ค่าคงที่ และค่าตัวแปร

<table>
<tr><th>ฟังก์ชัน</th><th colspan="2">หน้าที่การทำงาน</th><th>ตัวอย่างการใช้งาน</th></tr>
<tr><td>putchar()</td><td colspan="2">การแสดงผลที่ละตัวอักษร</td><td>putchar(ch);</td></tr>
<tr><td>puts()</td><td colspan="2">การแสดงผลข้อความ</td><td>puts(str);</td></tr>
<tr><td rowspan="12">printf</td><td colspan="2">การแสดงผลทุกชนิด ซึ่งมีรหัสรูปแบบดังนี้</td><td>printf”(C Programing)”;</td></tr>
<tr><td>รหัสรูปแบบ</td><td>ชนิดข้อมูล</td><td rowspan="11">printf(“%s”,str);
printf(“Value =%d”,intValue);
printf(“%s %d”,str,intValue);
printf(“Grade %c”,chGrade);</td></tr>
<tr><td>%c</td><td>ตัวอักขระหนึ่งตัว</td></tr>
<tr><td>%d</td><td>เลขจำนวนเต็ม</td></tr>
<tr><td>%ld</td><td>เลขจำนวนเต็มแบบยาว</td></tr>
<tr><td>%e</td><td>เลขจำนวนจริงแบบเอ็กซ์โปเนนต์</td></tr>
<tr><td>%f</td><td>เลขจำนวนจริง</td></tr>
<tr><td>%i</td><td>เช่นเดียวกับ %d</td></tr>
<tr><td>%o</td><td>เลขฐานแปด</td></tr>
<tr><td>%s</td><td>ข้อความสตริง</td></tr>
<tr><td>%u</td><td>เลขจำนวนเต็ม ไม่มีเครื่องหมาย</td></tr>
<tr><td>%x</td><td>เลขฐานสิบหก</td></tr>
</table>

ฟังก์ชั่น gets() เป็นฟังก์ชั่นที่นำมาใช้สำหรับรับข้อมูลประเภทสตริง

ฟังก์ชั่น puts()เป็นฟังก์ชั่นที่นำมาใช้สำหรับแสดงผลข้อมูลประเภทสตริง ท้ายข้อความจะผนวกค่า null หรือ รหัส \0 ปะต่อท้ายด้วย จัดเก็บเข้าไปในรูปแบบ อาร์เรย์

■ การทดลอง

1. จงเขียนโปรแกรมรับข้อมูลชื่อ นามสกุล และเกรดเฉลี่ยนักศึกษาจากคีย์บอร์ด

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

main()
{
	char chId[14], chName[20], chSurname[20];
	float fGrade;

	printf("Enter Code ID : ");
	scanf("%s", chId);
	printf("\nEnter My Name : ");
	scanf("%s", chName);
	printf("\nEnter My Surname : ");
	scanf("%s", chSurname);
	printf("\nEnter GPA : ");
	scanf("%f", &fGrade);
	printf("\n\n");
	printf("%s\t%s  %s\t%.2f", chId, chName, chSurname, fGrade);
	getch();
}
```

2. จงเขียนโปรแกรมรับข้อมูลบัตรประจำตัวสอบ เลขที่นั่งสอบ และรหัสวิชาสอบจากคีย์บอร์ดและแสดงผลออกทางจอภาพดังนี้

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

main()
{
	char chSubjectId[4], chNumberId[8];
	int intSeatId;
	printf("\nEnter Subject ID : ");
```

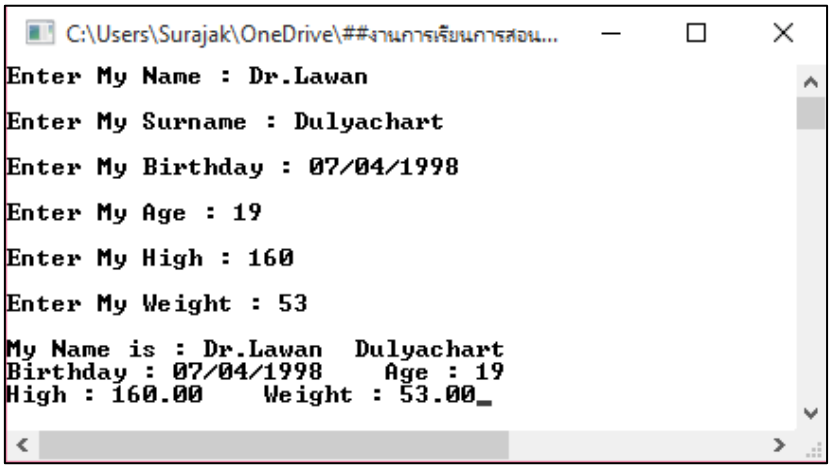
```
    scanf("%s", &chSubjectId);
    printf("\nEnter Identification Number : ");
    scanf("%s", &chNumberId);
    printf("\nEnter Seat Number : ");
    scanf("%d", &intSeatId);
    printf("\n\n");
    printf("%s\t%s\t%d", chSubjectId, chNumberId, intSeatId);
    getch();
}
```

3. จงเขียนโปรแกรมรับข้อมูลราคา และจำนวนเงินที่เสียภาษีคิดเป็นร้อยละ 7 ของราคาสินค้า

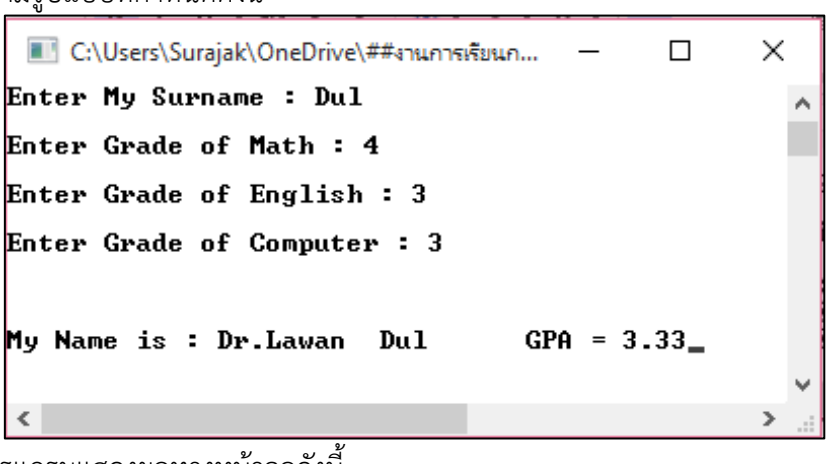
```
#include <stdio.h>
//#include <conio.h>

main()
{
    float fMoney, fTax;
    printf("Enter the income is yearly : ");
    scanf("%f", &fMoney);
    fTax = fMoney * 0.07;
    printf("\nThe income is yearly : %.2f", fTax);
    printf("\n123456 Dr.Lawan Dulyachart ");
 }
```

■ แบบฝึกหัดท้ายบทที่ 5

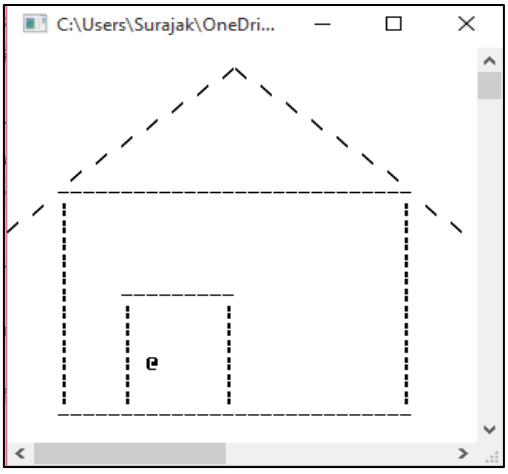
1. จงเขียนโปรแกรมรับข้อมูลส่วนบุคคล ซึ่งมีรูปแบบกรอกข้อมูลและแสดงผลข้อมูลในรูปแบบที่กำหนดดังนี้



2. จงเขียนโปรแกรมรับข้อมูลคะแนนสอบของนักศึกษา มี 3 วิชาพร้อมแสดงผลเกรดเฉลี่ยนักศึกษาตามรูปแบบที่กำหนดดังนี้



3. จงเขียนโปรแกรมแสดงผลทางหน้าจอดังนี้



บทที่ 6
คำสั่งควบคุม(Control-Flow Statement)

การเขียนโปรแกรมคอมพิวเตอร์มักจะต้องมีการสั่งให้โปรแกรมเลือกทำอย่างใดอย่างหนึ่ง ซึ่งจะต้องนำคำสั่งการเลือกทำมาใช้ ในการเขียนโปรแกรมภาษาซีก็มีคำสั่งสำหรับเลือกทำคือคำสั่ง if ซึ่งการทำคำสั่งนั้นจะต้องมีการตรวจสอบเงื่อนไขก่อนว่าจะทำสิ่งใดต่อไป ถ้าหากมีการเลือกทำหลายทางก็สามารถนำคำสั่ง if นี้มาซ้อนกันได้ หรือเลือกใช้คำสั่ง switch...case ก็ได้ ดังนั้นผู้เขียนโปรแกรมจะต้องเข้าใจการใช้คำสั่งเลือกทำนี้ จึงทำให้โปรแกรมทำตามเงื่อนไขได้ถูกต้อง

ในการเขียนโปรแกรมคอมพิวเตอร์บางครั้งจะต้องมีการให้โปรแกรมมีทางเลือกเพื่อที่จะทำงานอย่างใดอย่างหนึ่ง โดยถ้าเงื่อนไขเป็นจริงจะทำ ถ้าเงื่อนไขเป็นเท็จจะไม่ทำ ในการเขียนโปรแกรมทุกภาษาจะมีคำสั่งให้โปรแกรมตัดสินใจก่อนทำเงื่อนไข สำหรับภาษาซีมีคำสั่งที่ใช้ในการเลือกทำอยู่หลายคำสั่ง ในบทนี้จะกล่าวถึงคำสั่งเลือกทำง่ายๆ ที่ใช้กันอยู่ทั่วไป

รูปแบบการใช้งาน	การทำงานของคำสั่ง	ตัวอย่างการใช้งาน

If (กำหนดเงื่อนไขที่ 1)
{
ชุดคำสั่งเมื่อเงื่อนไขเป็นจริง;
}
	● เป็นคำสั่งตัดสินใจเลือกทำงานในชุดคำสั่งเมื่อเงื่อนไขที่กำหนดเป็นจริง	If (A > B)
		{
		printf(“A”);
		}

If (กำหนดเงื่อนไขที่ 1)
{
ชุดคำสั่งเมื่อเงื่อนไขเป็นจริง;
}
else
{
ชุดคำสั่งเมื่อเงื่อนไขเป็นเท็จ;
}
	●เป็นคำสั่งตัดสินใจเลือกทำงานในชุดคำสั่งหนึ่งจาก 2 ชุดคำสั่งซึ่งพิจารณาจากเงื่อนไขที่กำหนด
	●เมื่อเงื่อนไขที่กำหนดเป็นจริงให้ทำงานชุดคำสั่งหนึ่งและถ้าเงื่อนไขเป็นเท็จให้ทำงานอีกชุดคำสั่งหนึ่ง
		If (A > B)
		{
		printf(“A”);
		}
		else
		{
		printf(“B”);
		}

If (กำหนดเงื่อนไขที่ 1)
{
ชุดคำสั่งเมื่อเงื่อนไขเป็นจริง;
}
else if (กำหนดเงื่อนไขที่ 2)
{
ชุดคำสั่งที่ 2 ;
}
.
.
	●เป็นคำสั่งตัดสินใจเลือกทำงานในชุดคำสั่งหนึ่งใดคำสั่งหนึ่งจากหลายๆ ซึ่งพิจารณาจากเงื่อนไขที่กำหนด
	●เมื่อเงื่อนไขที่ 1 เป็นจริงให้ทำงานชุดคำสั่งหนึ่งและถ้าเงื่อนไขเป็นเท็จให้พิจารณาเงื่อนไขที่ 2
	●เมื่อเงื่อนไขที่ 2 เป็นจริงให้ทำงานชุดคำสั่งหนึ่งและถ้าเงื่อนไขเป็นเท็จให้พิจารณาเงื่อนไขต่อไป
		If (A > B)
		{
		printf(“A”);
		}
		else if (B>C)
		{
		printf(“B”);
		}
		else if (C>D)
		{

รูปแบบการใช้งาน	การทำงานของคำสั่ง	ตัวอย่างการใช้งาน

else
{
ชุดคำสั่งสุดท้าย ;
}
	●ทำเช่นนี้ต่อไป หากพบว่าทุกเงื่อนไขเป็นเท็จให้ทำงานชุดคำสั่งสุดท้าย	printf(“C”);
		}
		else
		{
		printf(“D”);
		}

switch(ตัวแปรหรือนิพจน์)
{
case ค่าที่ 1;
ชุดคำสั่งที่ 1
break;
case ค่าที่ 2;
ชุดคำสั่งที่ 2
break;
case ค่าที่ 3;
ชุดคำสั่งที่ 3
break;
default
ชุดคำสั่งสุดท้าย
}
	●เป็นคำสั่งตัดสินใจเลือกทำงานในชุดคำสั่งหนึ่งใดคำสั่งหนึ่งจากหลายๆ ชุดคำสั่ง ซึ่งพิจารณาจากตัวแปรหรือนิพจน์ที่กำหนด
	● หากตัวแปรหรือนิพจน์ที่กำหนดตรงกับค่าที่ 1 ให้ทำงานชุดคำสั่งที่ 1 และถ้าไม่ตรงให้พิจารณาค่าที่ 2
	●ทำเช่นนี้ต่อไป หากพบว่าตัวแปรหรือนิพจน์ที่กำหนดไม่ตรงกับค่าใดๆ ให้ทำงานชุดคำสั่งสุดท้าย
		switch (A)
		{
		case 1 ;
		printf(“A”);
		break;
		case 2 ;
		printf(“B”);
		break;
		case 3 ;
		printf(“C”);
		break;
		default ;
		printf(“D”);
		}

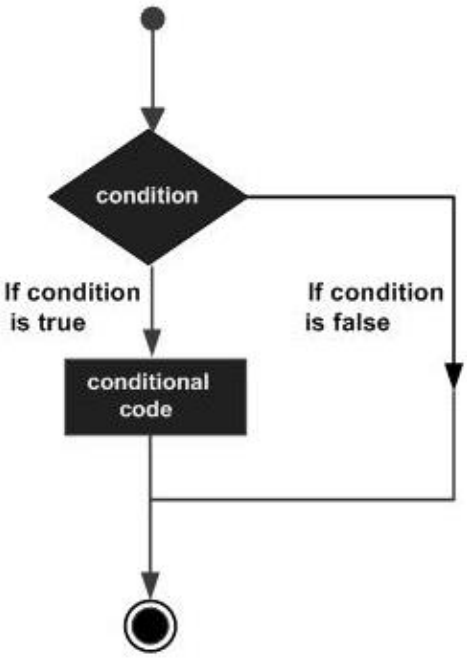
▪ คำสั่งเลือกทำแบบทางเดียว (if)

การเลือกทำแบบทางเดียวเพื่อตรวจสอบว่าชุดคำสั่งที่ตามมาจะทำหรือไม่ ในภาษาซีจะใช้คำสั่ง if ในการทำงานของคำสั่งคอมพิวเตอร์จะตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่ง หรือสเตตเมนต์ที่ตามหลัง หรือเป็นสเตตเมนต์รวมที่อยู่ในเครื่องหมาย { } แต่ถ้าเงื่อนไขเป็นเท็จคอมพิวเตอร์จะทำคำสั่ง หรือสเตตเมนต์ต่อไป รูปแบบของคำสั่งเป็นดังต่อไปนี้

รูปแบบ

```
if(boolean_expression) {
   /* statement(s) will execute if the boolean expression is true */
}
```

โดยการตรวจสอบเงื่อนไข จะเป็นการกระทำแบบบูลีน ผลลัพธ์ที่ได้จะเป็นจริงหรือเป็นเท็จเท่านั้น ถ้าหากมีการใช้ตัวดำเนินการจะใช้ตัวดำเนินการบูลีน สำหรับการทำงานของคำสั่ง if สามารถเขียนเป็นผังงานได้ดังนี้



ตัวอย่าง ถ้าหากคอมพิวเตอร์ทำคำสั่งต่อไปนี้

```
If (age >=18) print("of age\n");
```

ถ้าหากค่าในตัวแปร age เป็นค่าต่างๆ จะทำให้คอมพิวเตอร์แสดงผลดังต่อไปนี้

age = 25	age = 14	age = 18
of age good luck	good luck	of age good luck

สำหรับการตรวจสอบเงื่อนไขจะใช้ตัวดำเนินการเปรียบเทียบ ได้แก่

>	มากกว่า	>=	มากกว่าหรือเท่ากับ
<	น้อยกว่า	<=	น้อยกว่าหรือเท่ากับ
==	เท่ากัน	!=	ไม่เท่ากัน

ในการตรวจสอบเงื่อนไขนั้นตัวแปลที่นำมาเปรียบเทียบจะต้องเป็นข้อมูลประเภทเดียวกัน ตัวอย่างเช่น ถ้าให้ ch เป็น char ให้ num และ mark เป็น int และจะต้องตรวจสอบเงื่อนไขอาจเขียนได้ดังนี้

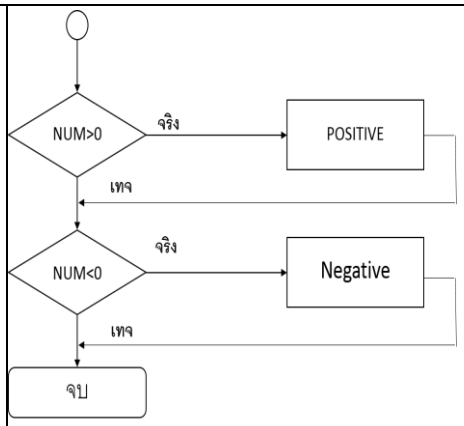
if (ch = 'A') ถ้า ch	เก็บรหัสแอสกีของ A จะเป็นจริง
if (num == 8)	ถ้า num มีค่าเท่ากับ 8 จะเป็นจริง
if (mark == num)	ถ้า mark มีค่าเท่ากับ num จะเป็นจริง

นอกจากนี้การตรวจสอบเงื่อนไขสามารถใช้ตัวดำเนินการทางตรรกะมาร่วมด้วยได้ อย่างเช่น ถ้าหากตัวแปร mark เก็บคะแนน และต้องการตรวจสอบว่าถ้าคะแนนมากกว่า 80 และน้อยกว่า หรือเท่ากับ 100 ให้ได้เกรด A จะเขียนคำสั่ง if ได้เป็น

```
If((mark > 80) && (mark <= 100));
   printf(“A”);
```

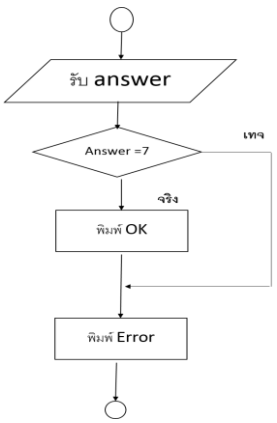
ตัวอย่างที่ 6.1 โปรแกรมทายตัวเลข โดยในโปรแกรม จะกำหนดตัวเลขไว้ในตัวแปรเป็น 123 และคอมพิวเตอร์จะให้ป้อนตัวเลขเข้าไป ถ้าค่าที่ป้อนเข้าไป มีค่าเท่ากันคอมพิวเตอร์จะแสดงคำว่า *** Right *** จากโปรแกรมจะสังเกตเห็นว่านิพจน์หลัง if จะใช้เครื่องหมาย == ซึ่งเป็นการเปรียบเทียบว่าเท่ากันหรือไม่

```
#include <stdio.h>
main()
{
int magic = 123;          /* กำหนดค่า 123
ให้กับตัวแปร */
          int guess;
          print(“Enter your guess : “);
          scanf(“%d”,&guess)
          if (guess == magic)
          printf(“***Right***);
}
```



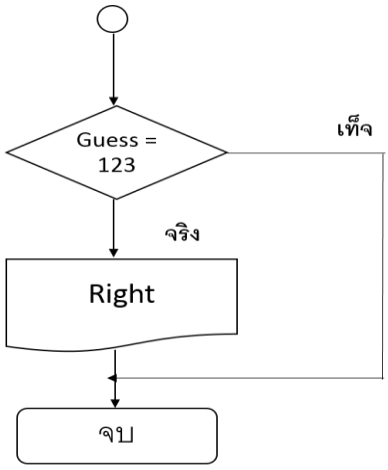
ตัวอย่างที่ 6.2 โปรแกรมต่อไป เมื่อรันโปรแกรมคอมพิวเตอร์จะถามว่า 3 + 4 มีค่าเท่า กับเท่าใด และให้ใส่คำตอบเข้าไป ถ้าตอบถูกคอมพิวเตอร์จะบอกว่า OK แต่ถ้าตอบผิดคอมพิวเตอร์จะบอกว่า Error โดยในโปรแกรมจะใช้คำสั่ง if ตรวจสอบว่าค่าเท่ากับ 3 + 4 หรือไม่

```
#include <stdio.h>
#include <conio.h>
main()
{
          int answer;
          printf("What is 3 + 4 = ");
          scanf("%d", &answer);
          if(answer ==3+4)
            printf("OK");
          printf("Error");
          getch();
}
```



ตัวอย่างที่ 6.3 ตัวอย่างนี้ จะให้คอมพิวเตอร์รับตัวเลขจำนวนเต็มเข้าไป แล้วให้แจ้งว่าตัว : เลขนั้นเป็นเลขบวกหรือเลขลบ โดยใช้คำสั่ง if ในการเลือกทำการตรวจสอบว่าเป็นเลขบวก หรือลบจะทำโดยนำตัวเลขที่รับเข้าไปเปรียบเทียบกับ 0 ว่ามี ค่ามากกว่าหรือน้อยกว่า

```
#include <stdio.h>
Main()
{
    int num;
    printf("Number : ");
    scanf("%d", &num);
    if(num > 0)printff'The Number is
Positive\n");
    if(num < 0)printf("The Number is
Negative\n");
    getch();
}
```



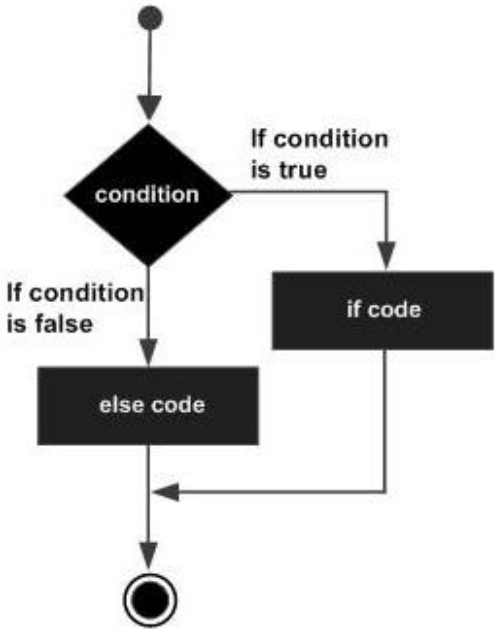
Tips การใช้คำสั่ง if ซึ่งใช้ในการตัดสินใจค่าที่ได้จะเป็นจริงหรือเท็จ จากตัวอย่างโปรแกรมที่ 6.1 จะเห็นว่าตัวดำเนินการใช้ในนิพจน์ของ if จะใช้ตัวดำเนินการเปรียบเทียบ == ซึ่งเป็นเครื่องหมายของการกำหนดค่า

■ คำสั่งเลือกทำอย่างใดอย่างหนึ่ง (if...else)

จากคำสั่ง if ที่ผ่านมาจะใช้ในการเขียนโปรแกรมที่ใช้ทดสอบว่าจะเลือกทำหรือไม่ ถ้าเงื่อนไขเป็นจริงจะทำคำสั่ง หรือสเตตเมนต์หลัง if ถ้าพิจารณาโปรแกรมที่ 6.2 จะพบว่าถ้าหากเราป้อนค่าถูกต้อง (ค่า 7) คอมพิวเตอร์จะพิมพ์ทั้ง 0K และ Error ซึ่งการทำงานในลักษณะนี้ควรจะให้โปรแกรมทำงานแบบเลือกทำอย่างใดอย่างหนึ่ง ในกรณีที่คอมพิวเตอร์ต้องเลือกทำอย่างใดอย่างหนึ่งโดยตรวจสอบเงื่อนไขที่กำหนดจะใช้คำสั่ง if...else โดยถ้าเงื่อนไขเป็นจริงจะทำคำสั่งหลัง if แต่ ถ้าเงื่อนไขเป็นเท็จจะทำคำสั่งหลัง else โดยนิพจน์ที่ตามหลัง if จะเป็นข้อมูลทางตรรกะ รูปแบบคำสั่งเป็นดังนี้

รูปแบบ

```
if(boolean_expression) {
   /* statement(s) will execute if the boolean expression is true */
}
else {
   /* statement(s) will execute if the boolean expression is false */
}
```



ตัวอย่าง 6.4 ตัวอย่างนี้จะรับค่าเลขจำนวนเต็มจากแป้นพิมพ์จากนั้นจะนำไปเปรียบเทียบ กับ 0 ว่าค่าที่รับเข้ามานั้นเป็นบวกหรือลบ จากนั้นจะแสดงประเภทของ ตัวเลขออกมา

```
#include <stdio.h>
#include <conio.h>
main()
{
        Int num;
        Printf (“enter a number:  ");
        scanf("%d', &num);
        if(num < 0)
        */ ตรวจสอบว่าค่าที่รับน้อยกว่า 0 หรือไม่ */
            Printf(“number is negative");
        /* ถ้าน้อยกว่า 0 บอกว่าเป็นค่าลบ */
        else
            printf(“number is positive”);
        /* ถ้าค่ามากกว่า 0 บอกว่าเป็นค่าบวก */
```

ตัวอย่าง 6.5 โปรแกรมต่อไปนี้เป็นโปรแกรมคำนวณราคาต้นทุนสินค้า ถ้าหากผลิต มากกว่า 10 ชิ้น จะราคาชิ้นละ 6.5 บาท แต่ถ้าไม่เกิน 10 ชิ้น ราคาชิ้นละ 7 บาท

```
#include <stdio.h>
 int Number;
float cost;
main()
{
        printf("Enter number:   ");
        scanf("%d", &Number);
        if (Number >=10)
              cost= Number * 6.5
        else cost= Number * 7;
        printf("Cost= %2.2f\n", cost);
}
```

ผลการรัน

```
Enter number:5
Cost = 35.00
```

การใช้คำสั่ง if-else สามารถเขียนได้สองรูปแบบ แต่โปรแกรมเมอร์นิยมใช้แบบที่สอง มากกว่า เพราะจะดูสวยงามและถ้าโปรแกรมมีขนาดใหญ่รูปแบบที่สองจะทำให้ดูโปรแกรมได้ง่ายขึ้น

```
If (condition) {statement 1 }
        else  {statement 2 }
```

สเตตเมนต์หลังเงื่อนไขของ if และตามหลัง else อาจรวมสเตตเมนต์ย่อยๆ เข้าไว้ด้วยกัน ได้ที่เรียกว่า compound statement หรือสเตตเมนต์รวม โดยสเตตเมนต์หรือคำสั่งต่างๆ จะอยู่ภายใน { กับ } การเขียนโปรแกรมในลักษณะนี้มักจะใช้คำสั่ง if - else ในรูปแบบที่สอง ตัวอย่าง เช่น ถ้านำโปรแกรมที่ 6.5 มาเขียนใหม่ในรูปแบบสเตตเมนต์รวม อาจเขียนได้เป็น

```
If (condition)
        {statement 1 }
        else
        {statement 2 }
```

```
if (number >= 10)
{
    printf(“Discount “);
    cost = number * 6.5;
}
else
```

```
{
    printf("No Discount" );
    cost = number * 7;
}
```

ตัวอย่างที่ 6.6 คอมพิวเตอร์จะรับตัวเลขจำนวนเต็มเข้าไปสองตัว จากนั้นจะนำมา หารกัน แต่ก่อนหารจะตรวจสอบก่อนว่าค่าที่นำมาหารเป็น 0 หรือไม่ ถ้าเป็น จะแจ้งว่าหารไม่ได้

```
#include <stdio.h>
#include <conio.h>
 main()
{
    int num1, num2:
    printf("enter first number: ");
    scanf("%d", &num1);
    printf("enter second number:" );
    scanf("%d", &num2):
    if(num2 == 0)      /* ตรวจสอบว่าค่า num2 ที่นำมาหารเป็น 0 หรือไม่ */
            printf("cannot divide by zero");
    else
          printf("answer is: %d", num1/num2); /* ถ้าไม่เป็น 0 แสดงผลหาร */
getch();
}
```

ตัวอย่างที่ 6.7 โปรแกรมต่อไปเป็นการรับค่าตัวเลขจำนวนเต็มเข้ามา ไม่ว่าจะเป็นเลขบวกหรือ ลบ จากนั้นจะแสดงข้อมูลเป็นค่าสัมบูรณ์ของเลขหรือเลขบวกทางหน้าจอ โดย ข้อมูลที่รับมานั้นจะถูกตรวจสอบว่าเป็นเลขลบหรือไม่ ถ้าเป็นให้คูณด้วย -1

```
    include <stdio.h>
    int x;
    main()
    {
          printf("INPUT NUMBER : ");
          scanf("%d", &x);
          If (x < 0)
                x = x * -1;
          printf("INPUT %d", x);
    }
```

ตัวอย่างที่ 6.8 โปรแกรมตรวจสอบว่าเป็นเลขคู่หรือคี่ ถ้าหากตัวแปร x เป็นเลขจำนวนเต็ม การตรวจสอบว่าเป็นเลขคู่หรือเลขคี่ ทำได้โดยเอา 2 ไปหาร แล้วดูว่าหารได้ ลงตัวหรือไม่ ถ้าหากหารได้ลงตัว เศษที่ได้จะเป็น 0 หมายความว่าเป็นเลขคู่ เราสามารถเขียนส่วนของโปรแกรมได้ดังนี้

```
if (x % 2 == 0)
    printf("Even number");
else
    printf("odd number");
```

NOTE: จะสังเกตเห็นได้ว่าการตรวจสอบเงื่อนไขหลัง if สามารถเขียนเป็นนิพจน์ได้ดังตัวอย่างที่ 6.8 และหลังจากที่ตรวจสอบเงื่อนไขแล้ว จะต้องมีเครื่องหมาย ; ถ้าหากมีเครื่องหมาย ; ตัวคอมไฟล์เลอร์จะมองว่าเป็นสเตตเมนต์ว่าง

ตัวอย่างที่ 6.9 ตัวอย่างนี้คอมพิวเตอร์จะให้ใส่ค่าผลบวกของเลข ถ้าตอบไม่ถูกต้องคอมพิวเตอร์จะแจ้งข้อมูลที่ถูกต้องออกมา โดยสเตตเมนต์หลัง else จะเป็น สเตตเมนต์รวม

```
#include<stdio.h>
#include<conio.h>
main()
{
    int ans;
        printf("What is 10 + 14 = ?");
        scanf("%d", &ans);
    if(ans == 10+14)
      printf("Right !");
    else
        {
        printf("Sorry, you are wrong. ");
        printf("The answer is 24");
    }
    getch();
}
```

ตัวอย่าง 6.10 ตัวอย่างนี้เป็นโปรแกรมเปลี่ยนหน่วยวัด โดยจะแสดงเมนูให้เลือกว่าต้องการ เปลี่ยนจากหน่วยฟุตเป็นเมตรหรือจากเมตรเป็นฟุต โดยใช้คำสั่ง if-else กับสเตตเมนต์รวม เนื่องจากหนึ่งเมตรมี 100 เซนติเมตร แต่หนึ่งฟุตมี 12 นิ้ว ซึ่งใกล้เคียงกับ 30 เซนติเมตร ถ้าหากรับข้อมูลเป็นฟุตแล้วต้องการทำเป็นเมตรจะทำโดยการเอา 3.28 ไปหาร แต่ถ้าหากเปลี่ยน จากเมตรเป็นฟุตทำโดยการเอา 3.28 ไปคูณ โปรแกรมเขียนได้ดังนี้

```
#include <stdio.h>
#include <conio.h>
main()
{
        float num;
        int choice;
        printf("1:feet to meters, 2:meters to feet ");
        printf("enter choice: ");
        scanf("%d", &choice);
        if (choice == 1)
        {
                printf("enter number of feet; ");
                scanf("%f", &num);
                printf("meters; %f", num/3.28);
        }
        else
        {
        printf("enter number of meters: ");
        scanf("%f", &num);
        printf("feet: %f", num*3.28);
        }
        getch();
}
```

ผลการรัน

```
1: feet to meters, 2: meters to feet
enter choice 1
```

เมื่อรันโปรแกรมจะแสดงเป็นเมนู
ถ้าเลือก 1 เป็นการเปลี่ยนจากฟุตเป็นเมตรจะทำให้เงื่อนไขของ if เป็นจริง
โปรแกรมก็จะทำตามสเตตเมนต์รวมหลัง if
แต่ถ้าเลือก 2 โปรแกรมจะไปทำสเตตเมนต์รวมหลัง else

■ คำสั่งแบบหลายทางเลือก (nested if statement)

การใช้คำสั่งเลือกทำนี้สามารถนำหลายๆ คำสั่ง if – else มาซ้อนกันได้ เรียกว่า nested if statement ดังรูปแบบคำสั่งต่อไปนี้

ตัวอย่างที่ 6.11 โปรแกรมต่อไปจะเป็นโปรแกรมที่ใช้คำสั่งเลือกทำซ้อนๆ กัน โดย ให้ใส่ค่าคะแนนระหว่าง 0 ถึง 100 เข้าไปในตัวแปร X จากนั้นจะบอกว่าผลการสอบ เป็นอย่างไร

```
#include <stdio.h>
#include <conio.h>
main()
{
  int x;
        printf("Enter Score(0..100) : ");
        scanf("%d", &x);
        if (x>=90)
printf("Excellent" );
        else if (x>=80)
             printf("Good");
        else if(x>=70)
             printf("FAIR");
        else
         printf("FAIL");
  getch();
}
```

■ การเลือกทำแบบ switch statement

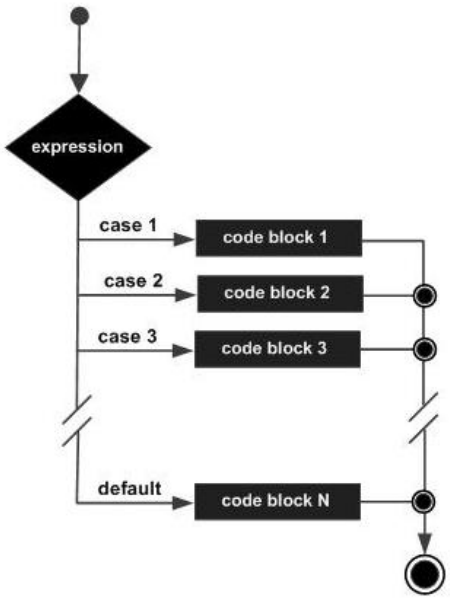
การเขียนโปรแกรมที่ต้องมีการเลือกทำหลาย ทางเลือก เราสามารถนำประโยคคำสั่ง If...else มาซ้อนกันได้ แต่ถ้าเงื่อนไขที่ต้องตัดสินใจขึ้นกับตัวแปรตัวเดียว เราสามารถใช้คำสั่ง switch..case แทนได้ ตัวอย่างเช่น ถ้าเขียนโปรแกรมเป็นลักษณะ เมนูดังต่อไปนี้

```
เมนู
1. คำนวณบวกเลข
2. คำนวณลบเลข
3. คำนวณคูณเลข
4. คำนวณหารเลข
   โปรดเลือกการคำนวณ
```

และให้ผู้ใช้โปรแกรมเลือกวิธีการคำนวณเข้าไปโดยป้อนค่าอินพุตเข้าไป เราสามารถเขียนโปรแกรม โดยนำค่าอินพุตที่รับเข้าไปเก็บไว้ในตัวแปรตัวหนึ่ง และใช้คำลัง switch เลือกว่ามีค่าเท่ากับคำใด (1, 2, 3, 4) จากนั้นให้ไปทำงานตามที่เลือก ประโยคคำสั่ง switch..case มีรูปแบบดังนี้

รูปแบบ

```
switch(expression) {

   case constant-expression  :
      statement(s);
      break; /* optional */

   case constant-expression  :
      statement(s);
      break; /* optional */

   /* you can have any number of case statements */
   default : /* Optional */
   statement(s);
}
```



คำสั่ง switch นี้จะนำค่าใน variable มาตรวจสอบว่าเท่ากับค่าคงที่ค่าใดหลัง case จากนั้นโปรแกรมจะไปทำสเตตเมนต์หลังค่าคงที่ตัวนั้น และออกจาก switch เมื่อถึงคำสั่ง break แต่ถ้าไม่เท่าค่ากับคำคงที่ค่าใดเลย โปรแกรมจะไปทำสเตตเมนต์หลัง default สำหรับค่าที่ใช้ตรวจ สอบจะเป็นตัว

แปร นิพจน์ หรือฟังก์ชันก็ได้ สำหรับในแต่ละ case สามารถมีคำสั่งได้มากกว่าหนึ่ง คำสั่งหรืออาจไม่มีก็ได้ โดยถ้าไม่มีคำสั่ง โปรแกรมจะไปทำงานใน case ถัดไป **และค่าคงที่หลัง case จะต้องเป็น int หรือ char เท่านั้น**

การใช้คำสั่ง switch มีสิ่งสำคัญที่ควรรู้ 4 ประการดือ
1. คำสั่ง switch ต่างจากคำสั่ง if ตรงที่ switch สามารถทดสอบเงื่อนไขได้หลายอย่าง แต่คำสั่ง if จะตรวจสอบเฉพาะความสัมพันธ์ หรือลอจิกเท่านั้น
2. ค่าคงที่ของคำสั่ง switch สามารถมีค่าซ้ำกันได้
3. ถ้าค่าคงที่เป็นตัวอักษร คำสั่ง switch จะมองเป็นเลขจำนวนเต็ม
4. ค่า default จะมีหรือไม่มีก็ได้

ตัวอย่าง 6.12 ถ้าหากตัวแปร num เป็นเลขจำนวนเต็ม การกำหนดค่าหลัง case ในการกำหนดเงื่อนไข

แบบ switch จะเป็นดังนี้

```
switch(num)
{
     case 4     : ชุดคำสั่ง;       break;  /* ถูกต้องเพราะเป็นตัวเลข */
     case 2.5   : ชุดคำสั่ง;       break:  /* ไม่ถูกต้องเพราะเป็นทศนิยม */
     case m     : ชุดคำสั่ง;       break;  /* ไม่ถูกต้องเพราะเป็นตัวแปร */
     case '2'   : ชุดคำสั่ง;       break;  /* ไม่ถูกต้องเพราะเป็นตัวอักขระ */
     default    : ชุดคำสั่ง
}
```

ตัวอย่างที่ 6.13 โปรแกรมต่อไปเป็นตัวอย่างการใช้คำสั่ง switch โดยโปรแกรมจะให้ใส่ตัว เลข 1 ถึง 4 จากนั้นโปรแกรมจะพิมพ์ชื่อของเลขที่ใส่เข้าไป แต่ถ้าหากไม่ ใส่เลขในช่วงดังกล่าว โปรแกรมจะพิมพ์คำว่า "unrecognized number"

```
#include <stdio.h>
#include <conio.h>
main()
{
     int i;
     printf("Enter a number between 1 and 4:    ");
     scanf("%d", &i);
     switch(i)
          {
```

```
        case 1:printf("one");
                break;
        case 2:printf("two");
                break;
        case 3:printf("three");
                break;
        case 4:printf("four");
                break;
        default:
                  printf("unrecognized number");
           }
           getch();
}
```

ตัวอย่างที่ 6.14 โปรแกรมต่อไปเป็นตัวอย่างการใช้คำสั่ง switch โดยหลังค่าคงที่ของ case จะไม่มีคำสั่ง ซึ่งจะทำให้โปรแกรมไปทำงานคำสั่งหลัง case ถัดไปเรื่อยๆ ในโปรแกรมนี้เครื่องจะให้เราพิมพ์ตัวอักษรภาษาอังกฤษที่เป็นสระ โดย คอมพิวเตอร์จะแจ้งว่า vowel ถ้าใส่ตัวอักษรอื่นที่ไม่ใช่สระ คอมพิวเตอร์ จะแจ้งว่า consonant

```
#include <stdio.h>
#include <conio.h>
main()
{
          char ch;
          printf("Enter the letter: ");
          ch = getche();
          switch(ch)
{
          case 'a' :
          case 'e' :
          case 'i'  :
          case 'o' :
          case 'u' :
               printf(" is a vowel\ n");
break;
           default    : printf(" is a consonant");
     }
     getch();
```

```
}
```

จากตัวอย่างนี้จะเห็นว่า ถ้าตัวแปรหลัง switch เป็นตัวอักขระ ค่าคงที่หลัง case ต้อง เป็นตัวอักขระด้วย โดยจะเขียนอยู่ในเครื่องหมาย ‘ ’

■ การเลือกทำแบบ nested switch statements

การเลือกใช้คำสั่ง switch ซ้อนคำสั่ง switch โดยมีรูปแบบการทำงานดังนี้

รูปแบบ

```
switch(ch1) {

   case 'A':
      printf("This A is part of outer switch" );

      switch(ch2) {
         case 'A':
            printf("This A is part of inner switch" );
            break;
         case 'B': /* case code */
      }
      break;
   case 'B': /* case code */
}
```

ตัวอย่าง 6.15

```
#include <stdio.h>
int main () {
   /* local variable definition */
   int a = 100;
   int b = 200;
   switch(a) {
      case 100:
         printf("This is part of outer switch\n", a );
         switch(b) {
            case 200:
               printf("This is part of inner switch\n", a );
         }
   }
   printf("Exact value of a is : %d\n", a );
```

```
   printf("Exact value of b is : %d\n", b );
   return 0;
}
```

■ สรุปท้ายบท

ในการเขียนโปรแกรมคอมพิวเตอร์ มักจะต้องมีการสั่งให้โปรแกรมเสือกทำอย่างใดอย่างหนึ่ง การทางานประเภทนี้จะนำคำลังเลอกทำมาใช้ คำลังประเภทนี้ประกอบด้วยคำลังเสือกทำแหบทาง เดียว (if) คำสั่งเลือกทำอย่างใดอย่างหนึ่ง (if - else) เมื่อโปรแกรมจะทำคำสั่งเลือกทำจะต้องมีการตรวจสอบเงื่อนไขก่อน การตรวจสอบเงื่อนไขนั้นผลลัพธ์,ที่ได้จะเป็นจริงหรือเท็จเท่านั้น1 แต่ถ้าหากโปรแกรมมีทางเสือกที่จะทำหลายๆ ทาง การเขียนโปรแกรมจะทำได้สองวิธี คือการนำคำสั่ง if มาซ้อนๆ กัน และการนำคำลัง switch..case มาใช้

■ การทดลอง

1. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int intScore = 63;
        if (intScore > 50)
        {
                printf("My Score = %d", intScore);
        }
        getch();
}
```

2. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int intScore = 50;
```

```
        if (intScore > 50)
        {
                printf("My Score Pass");
        }
        else
        {
                printf("My Score Not Pass");
        }
        getch();
}
```

3. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main() {
        int A = 17, B = 3, C = 7;
        if (A > 23)
        {
                A = A % B;
                B = C;
                C = A - B;
        }
        else
        {
                C = A - B;
                A = C * 2;
                C = A - C;
        }
        printf("A = %d, B = %d, C = %d", A, B, C);

        getch();
}
```

4. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

```
main()
{
	int X = 11, Y = 8, Z = 7;
	if (Z <= Y)
	{
		Z = X % Z;
		Z = X + Y;
		Y = Z + X;
	}
	else
	{
		Z = Y;
		Z = X * Y;
		X = Z - X;
	}
	printf("X = %d, Y = %d, Z = %d", Y, Z, X);

	getch();
}
```

5. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
	char strProvince[4] = "BKK";
	if (strProvince[1] == 'K')
	{
		printf("Province : Bangkok");
	}
	else
	{
		printf("Province : %s", strProvince);
	}

	getch();
}
```

6. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
	int A = 3, B = 5, C = 8;
	if (A == B)
	{
		C = A % 3;
		B = B - A;
	}
	else if (A == C)
	{
		B = A;
		A = C + B * 2;
	}
	else
	{
		C = C - B;
		B = C;
		C = A;
	}
	printf("A = %d, B = %d, C = %d", A, B, C);
	getch();
}
```

7. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
	int M = 11, N = 79;
	if (M >= 10)
	{
		N = N - 3;
	}
	else if (M >= 20)
```

```
        {
                N = N % 3;
        }
        else
        {
                N = N * 3;
        }
        printf("N = %d", M);

        getch();
}
```

8. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A = 2, B = 3, C = 5;
        if (A >= B)
        {
                A = A + B;

                if (A >= B)
                {
                        C = A--;
                }
                else
                {
                        C = ++B;
                }
        }
        else
        {
                if (A < B)
                {
                        C += A;
                }
```

```
                else
                {
                        C -= ++A;
                }
        }
        printf("C = %d", C);

        getch();
}
```

9. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        char strTHA[10] = "THAILAND";
        if ((strTHA [1] == 'T') || (strTHA [2] == 'H'))
        {
                printf("THAILAND");
        }
        else if (strTHA [4] == 'L')
        {
                printf("ENGLAND");
        }
        else
        {
                printf("My World");
        }
        getch();
}
```

10. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
```

```
int A = 72, B = 13, C = 45;
if (A >= 5)
{
        C = B++;

        if (C >= 10)
        {
                C += B * 2;
        }
        else
        {
                C = C * 3;
        }
}
if (C >= A)
{
        A = ++C;

        if (C >= B)
        {
                A = A - C;
        }
        else
        {
                C = B % C;
        }
}
else
{
        if (A == B)
        {
                C = A;
        }
        else if (A > B)
        {
                C = B;
        }
```

```
                else
                {
                        C = A % C;
                }
        }
        printf("A = %d, B = %d, C = %d", A, B, C);

        getch();
}
```

11. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        char chGrade = 'B';
        switch (chGrade)
        {
      case 'A' :
           printf("Excellent");
           break;
      case 'B' :
           printf("Good");
           break;
      case 'C' :
           printf("OK");
           break;
      case 'D' :
           printf("Improve");
           break;
      default :
           printf("Durchfallen");
        }
        getch();
}
```

12. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
        int intGroup = 17;
        switch (intGroup % 4)
        {
      case 0 :
            printf("Science Group");
            break;
      case 1 :
            printf("Thai Group");
            break;
      case 2 :
            printf("English Group");
            break;
      default :
            printf("Math Group");
        }
        getch();
}
```

13. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
        int intGroup = 2;
        int intScore = 73;
        switch (intGroup)
        {
                case 0 :
                        printf("Science Group : ");
                        if (intScore > 50)
                        {
```

```
                        printf("Pass");
                }
                else
                {
                        printf("Not Pass");
                }
                break;
        case 1 :
                printf("Thai Group : ");
                if (intScore > 60)
                {
                        printf("Pass");
                }
                else
                {
                        printf("Not Pass");
                }
                break;
        case 2 :
                printf("English Group : ");
                if (intScore > 80)
                {
                        printf("Pass");
                }
                else
                {
                        printf("Not Pass");
                }
                break;
        default :
                printf("Not Group");
    }

    getch();
}
```

14. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
	int A = 3, B = 23, C = 6, intAns = 0;

	if ((A >= 2) && (B >= 20))
	{
		intAns = A % 5;
		switch (intAns)
		{
		case 0 :
		case 1 :
			case 2 :
				printf("Answer = %d", intAns);
				break;
			case 3 :
			case 4 :
				intAns *= 2;
				printf("Answer = %d", intAns);
				break;
			default :
				printf("Error");
		}
	}
	else
	{
		intAns *= 3;
		printf("Answer = %d", intAns);
	}

	getch();
}
```

15. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
	int intMath = 73, intSci = 87;
	char chGrade;

	if ((intMath >= 60) && (intSci >= 60))
	{
		if ((intMath >= 70) && (intSci >= 80))
		{
			chGrade = 'G';
		}
		else
		{
			chGrade = 'P';
		}
	}
	else
	{
		chGrade = 'F';
	}

	intMath *= 0.40;
	intSci *= 0.60;

	switch (chGrade)
	{
		case 'G' :
		case 'P' :
			printf("Science Sroce : %d", intSci);
			printf("\nMath Sroce : %d", intMath);
			printf("\nGrade : %c", chGrade);
			break;
		default :
```

```
                printf("\nGrade : %c", chGrade);
        }

        getch();
}
```

16. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    char chCharTest;
    printf("Enter Character : ");
    scanf("%c", &chCharTest);
    switch (chCharTest)
    {
        case '0' :
        case '1' :
        case '2' :
        case '3' :
        case '4' :
        case '5' :
        case '6' :
        case '7' :
        case '8' :
        case '9' :
            printf("\n\nCharacter %c is integer", chCharTest);
                break;
        default :
            printf("\n\nCharacter %c isn't integer", chCharTest);
    }
    getch();
}
```

17. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

```
main()
{
    int intScore;

    printf("Enter Score : ");
    scanf("%d", &intScore);

    if (intScore > 79)
    {
        printf("\nMy Grade : A");
    }
    else if (intScore > 69)
    {
        printf("\nMy Grade : B");
    }
    else if (intScore > 59)
    {
        printf("\nMy Grade : C");
    }
    else if (intScore > 49)
    {
        printf("\nMy Grade : D");
    }
    else
    {
        printf("\nMy Grade : F");
    }
    getch();
}
```

■ แบบฝึกหัดท้ายบทที่ 6

ตอนที่ 1 จงทำเครื่องหมาย ✓ หน้าข้อที่ถูกต้อง และทำเครื่องหมาย ✗ หน้าข้อที่ไม่ถูกต้อง

..... 1. ประโยคหลัง if ต้องให้ผลลัพธ์เป็นจริงหรือเท็จเท่านั้น

..... 2. คำสั่ง if-else ไม่ใช่คำสั่งควบคุมของโปรแกรม

..... 3. การตรวจสอบเงื่อนไขของคำสั่ง if สามารถใช้ตัวดำเนินการแบบตรรกะได้

..... 4. ประโยคหลัง if สามารถเขียนนิพจน์ที่มีการคำนวณได้

..... 5. คำสั่ง switch จะต้องมี default ด้วย

..... 6. ทุกๆ case ของคำสั่ง switch จำเป็นต้องมีค่าคงที่เสมอ

..... 7. ถ้าหากใส่เครื่องหมาย ; หลังการตรวจสอบเงื่อนไขของ if โปรแกรมจะรันไม่ได้

..... 8. การเลือกทำแบบสองทางจะใช้คำสั่ง if ที่ไม่มี else

..... 9. ทุกๆ case ของคำสั่ง switch ถ้าไม่มี break โปรแกรมจะรันไม่ได้

..... 10. ระหว่าง if กับ else ถ้ามีหลายคำสั่งจะต้องอยู่ในเครื่องหมายปีกกา

ตอนที่ 2 จงตอบคำถามต่อไปนี้

1. ถ้าหาก x = 4, y = 0 และ z = 2 จงหาค่าของ x, y และ z หลังทำคำสั่งต่อไปนี้

```
if (x != 0)
        y = 3;
else
        z = 2;
```

2. ถ้าหาก X = -2, y = 5, z = 0 และ t = -4 จงหาผลลัพธ์จากการทำนิพจน์ต่อไปนี้

ก. x + y < z + t

ข. x – 2 * y + y < z * 2 / 3

3. จงหาค่า x และ y หลังจากทำคำสั่งต่อไปนี้

```
int x = 3, y = 10;
if ((3/x) < j)
    x = x + 2;
    y = y + 1;
```

ตอนที่ 3 ฝึกเขียนโปรแกรม

1. จงเขียนโปรแกรมให้คอมพิวเตอร์ถามอายุ
   - ถ้าอายุมากกว่า 40 ปีให้บอกว่า "คุณแก่จัง"
   - แต่ถ้ายังไม่ถึง 40 ปีให้บอกว่า "คุณเด็กจัง"
2. จงเขียนโปรแกรมคำนวณจำนวนเงินจากการขายสินค้า โดยให้อินพุตเป็นจำนวนชิ้นของสินค้า โดยกำหนดให้ราคาขายเป็น
   - ถ้าซื้อน้อยกว่า 10 ชิ้น ให้ราคาชิ้นละ 20 บาท
   - ถ้าซื้อตั้งแต่ 10 ชิ้นขึ้นไป ให้ราคาชิ้นละ 8 บาท
3. จากข้อ 2 จงเขียนโปรแกรมให้คอมพิวเตอร์บวกภาษีมูลค่าเพิ่มอีก 7%
4. จงเขียนโปรแกรมรับค่าคะแนน เพื่อแสดงผลเกรด ด้วย if-else โดยที่

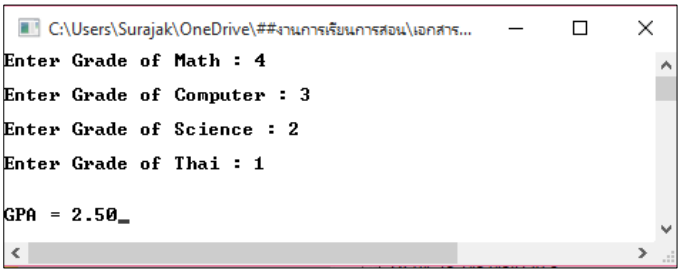
ช่วงคะแนน	เกรด
80-100	A
75-79	B+

70-74	B
65-69	C+
60-64	C
55-59	D+
50-54	D
0-49	F

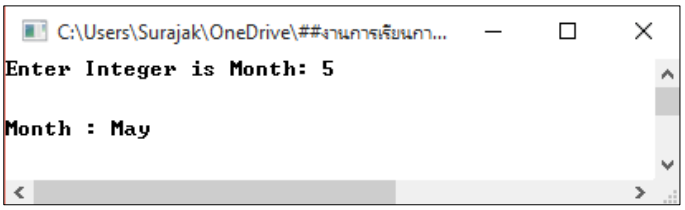
5. จงนำโปรแกรมข้อที่ 1 มาปรับปรุง และเขียนด้วย switch-case โดยที่

เกรด	ความหมาย
A	Excellent
B+	Very Good
B	Good
C+	Fairly Good
C	Fair
D+	Poor
D	Very Poor
F	Fail

6. จงเขียนโปรแกรมรับระดับคะแนน(เกรด)ใน 4 รายวิชา คือ



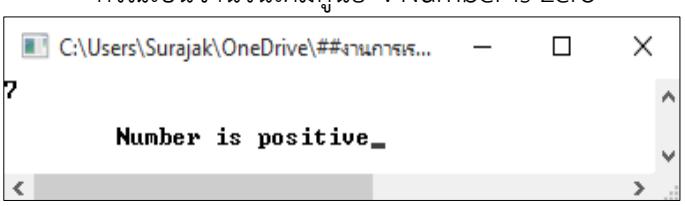
7. จงเขียนโปรแกรมรับเลขเดือน 1-12 แล้วแสดงผลชื่อเดือนที่ได้รับมาบจอภาพ เช่น เลข 5 ให้แสดงเป็น May เป็นต้น



8. จงเขียนโปรแกรมตรวจสอบค่าที่รับจากคีย์บอร์ดว่าเป็นเลขจำนวนเต็ม เต็มลบ เต็มศูนย์ โดยกำหนดรูปแบบการแสดงผลดังนี้

   - กรณีเป็นจำนวนเต็มบวก : Number is positive

- กรณีเป็นจำนวนเต็มลบ : Number is negative
- กรณีเป็นจำนวนเต็มศูนย์ : Number is zero



บทที่ 7
คำสั่งทำซ้ำ (Loops)

คำสั่งทำซ้ำ (Loops) เป็นคำสั่งให้เกิดการทำงานที่ชุดคำสั่งเดียวกันหลายๆ ครั้ง ภาษา C มีคำสั่งทำซ้ำ 3 คำสั่ง คือ for, while และ do-while

■ คำสั่ง for

for เป็นคำสั่งที่สั่งให้โปแกรมมีการทำงานซ้ำ ๆ วนลูปจนกว่าเงื่อนไขที่กำหนดไว้เป็นเท็จ จึงออกจากคำสั่ง for ไปทำคำสั่งถัดไป ควรใช้คำสั่ง for ในกรณีที่ทราบจำนวนรอบของการทำงาน

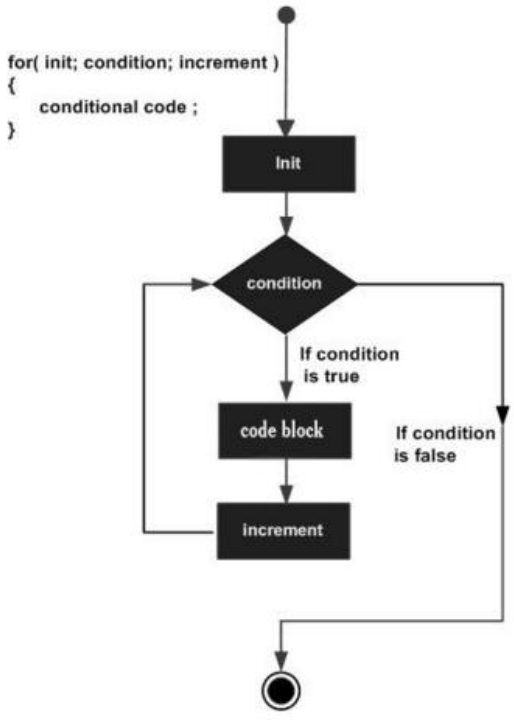
รูปแบบ

```
for ( init; condition; increment ) {
   statement(s);
}
```

หรือ

```
for (ค่าเริ่มต้น; ทดสอบ; เปลี่ยนค่า) {
      คำสั่ง;
}
```

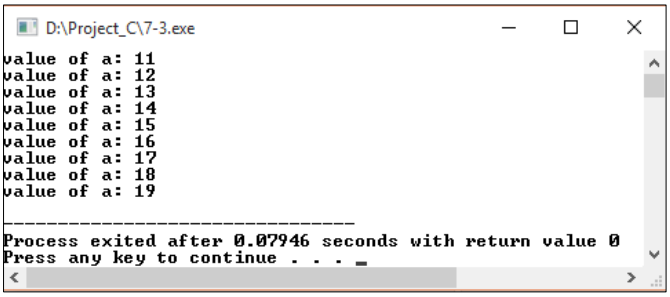
เขียนเป็นแผนผังได้ดังนี้



ตัวอย่าง

```
#include <stdio.h>
main () {
   int a;
   /* for loop execution */
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }
}
```

ผลลัพธ์โปรแกรม

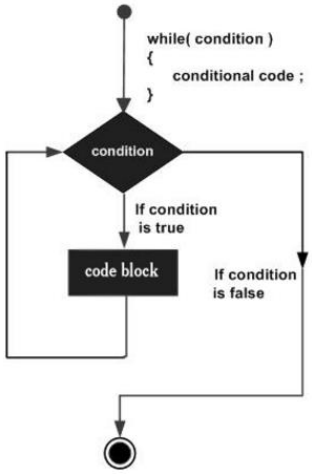


■ คำสั่ง while

รูปแบบ

```
while(condition) {
   statement(s);
}
```

เขียนเป็นแผนผังได้ดังนี้



ลำดับการทำงาน เริ่มต้นด้วยการตรวจสอบนิพจน์ condition ถ้านิพจน์ส่งค่าเป็น true โปรแกรมจะข้ามไปทำงานใน code block เสร็จแล้วจะไปตรวจสอบนิพจน์อีก และถ้านิพจน์ส่งค่าเป็น false โปรแกรมก็จะออกจาก body

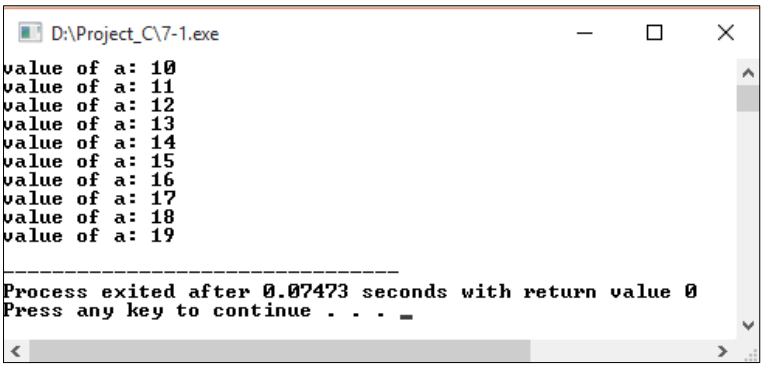
ลักษณะการทำงานของลูป while

1. ชุดคำสั่งภายในลูปจะทำงานก็ต่อเมื่อเงื่อนไขเป็นจริง
2. การออกจากลูปก็ต่อเมื่อเงื่อนไขเป็นเท็จ
3. เงื่อนไขหรือนิพจน์ที่นำมาตรวจสอบ สามารถใช้ตัวดำเนินการเปรียบเทียบ หรือตัวดำเนินการตรรกะได้

ตัวอย่าง

```
#include <stdio.h>
main () {
   /* local variable definition */
   int a = 10;
   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }
}
```

ผลลัพธ์โปรแกรม

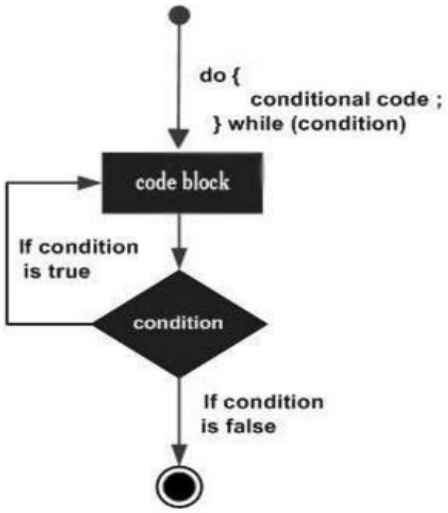


■ คำสั่ง do – while

รูปแบบ

```
do {
   statement(s);
} while( condition );
```

เขียนเป็นแผนผังได้ดังนี้



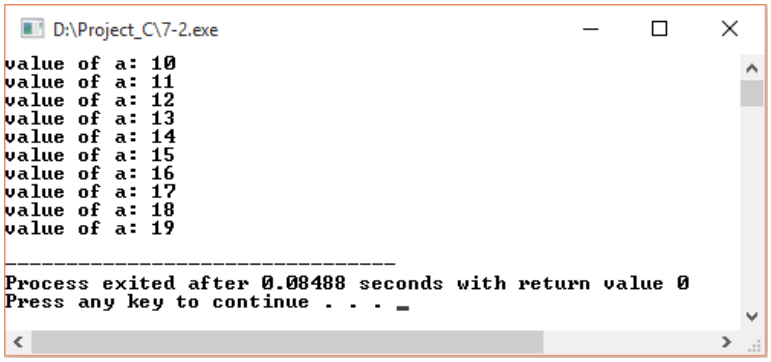
ลักษณะการทำงานของลูป do-while

1. ชุดคำสั่งภายในลูป do-while อย่างน้อยจะถูกทำงาน 1 ครั้งเสมอ ถึงแม้ว่าผลการทำงานตรวจสอบเงื่อนไขจะเป็นเท็จตั้งแต่แรกก็ตาม
2. การออกจากลูป do-while ก็ต่อเมื่อเงื่อนไขเป็นเท็จ

ตัวอย่าง

```
#include <stdio.h>
main () {
   /* local variable definition */
   int a = 10;
   /* do loop execution */
   do {
      printf("value of a: %d\n", a);
      a = a + 1;
   }while( a < 20 );
}
```

ผลลัพธ์โปรแกรม



■ สรุปท้ายบท

คำสั่งทำซ้ำ (Loops) เป็นคำสั่งให้เกิดการทำงานที่ชุดคำสั่งเดียวกันหลายๆ ครั้ง ภาษา C มีคำสั่งทำซ้ำ 3 คำสั่ง คือ for, while และ do-while

for เป็นคำสั่งที่สั่งให้โปแกรมมีการทำงานซ้ำ ๆ วนลูปจนกว่าเงื่อนไขที่กำหนดไว้เป็นเท็จ จึงออกจากคำสั่ง for ไปทำคำสั่งถัดไป ควรใช้คำสั่ง for ในกรณีที่ทราบจำนวนรอบของการทำงาน

while คือ ชุดคำสั่งภายในลูปจะทำงานก็ต่อเมื่อเงื่อนไขเป็นจริง การออกจากลูปก็ต่อเมื่อเงื่อนไขเป็นเท็จ เงื่อนไขหรือนิพจน์ที่นำมาตรวจสอบ สามารถใช้ตัวดำเนินการเปรียบเทียบ หรือตัวดำเนินการตรรกะได้

do-while **คือ** ชุดคำสั่งภายในลูป do-while อย่างน้อยจะถูกทำงาน 1 ครั้งเสมอ ถึงแม้ว่าผลการทำงานตรวจสอบเงื่อนไขจะเป็นเท็จตั้งแต่แรกก็ตาม การออกจากลูป do-while ก็ต่อเมื่อเงื่อนไขเป็นเท็จ

■ การทดลอง

1. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
      int intNum, intSum;
      for (intNum = 1; intNum <= 5; intNum++)
      {
          printf("\t%d", intNum);
      }
      getch();
}
```

2. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

```
main()
{
      int intNum, intSum;
      intSum = 0;
      for (intNum = 0; intNum < 10; intNum++)
      {
          intSum = intSum + intNum;
      }

      printf("Sum = %d", intSum);
      getch();
}
```

3. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
      int intNum, intAns;
      intAns = 0;
      for (intNum = 1; intNum <= 12; intNum++)
      {
          intAns = intNum * 2;
          printf("2 * %d = %d", intNum, intAns);
          printf("\n");
      }
      getch();
}
```

4. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
      int intNum, intAns;
      intNum = 1;
      intAns = 0;
```

```
    while(intAns < 100)
    {
        intAns =  intNum * intNum;
        printf("%d ^ %d = %d", intNum, intNum, intAns);
        printf("\n");
        intNum++;
    }
    getch();
}
```

5. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A = 3, B = 7, C = 2;
    A = A * C;
    B = A % C;
    while(B < 7)
    {
        A =  A + B;
        B = B + C;
        C = A - B++;
    }
    printf("A = %d\n", A);
    printf("B = %d\n", B);
    printf("C = %d", C);
    getch();
}
```

6. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A = 4, B = 1, C = 2;
    C = B - C;
```

```
    A = A % C;
    while(A >= 11)
    {
        A =  B % 3;
        C = C - A;
        C = A - B++;
    }
    printf("A = %d\n", A);
    printf("B = %d\n", B);
    printf("C = %d", C);
    getch();
}
```

7. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A = 3, B = 2, C = 9;
    C = A - B;
    while(A <= 11)
    {
        if (A == 7)
        {
            A = A + 3;
        }
        B = A * C;
        C = B - C;
        A++;
                printf("A = %d, B = %d, C = %d", A, B, C);
                printf("\n");
    }
    getch();
}
```

8. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

```
#include <conio.h>
main()
{
    int A = 1, B = 2, C = 7;
    A += B * 5;
    while(B <= 3)
    {
        if (C > 4)
        {
            C = C - 2;
            B+= 1;
        }
        else
        {
            B = A * C;
            A+= 2;
        }
        printf("A = %d, B = %d, C = %d", A, B, C);
        printf("\n");
    }
    getch();
}
```

9. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A = 1, B = 2, C = 7;
    A += B * 5;
    while(B <= 3)
    {
        for (A = 0; A < B; A++)
        {
            C = C + B - 1;
                        A = A + B;
        }
```

```
                B = B + 2;
        printf("A = %d, B = %d, C = %d", A, B, C);
        printf("\n");
    }
    getch();
}
```

10. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B, C, D;
    C = 0;
    for (A = 0; A < 7; A++)
    {
        for (B = 1; B < A; B++)
        {
            D = A + B - C;
            C++;
        }
    }
        printf("A = %d, B = %d, C = %d, D = %d", A, B, C, D);
    printf("\n");
    getch();
}
```

11. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B, C, D;
    C = 0;
        D = 0;
    for (A = 0; A < 10; A++)
```

```
    {
        for (B = 1; B < 10; B++)
        {
            D += A;
            C++;
        }
        printf("A = %d, B = %d, C = %d, D = %d", A, B, C, D);
        printf("\n");
    }
    getch();
}
```

12. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 1; A < 10; A++)
        {
                for (B = 1; B <= A; B++)
                {
                        printf("%d", A);
                }
                printf("\n");
        }
   getch();
}
```

13. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 9; A > 0; A--)
        {
```

```
                for (B = 1; B <= A; B++)
                {
                        printf("%d", A);
                }
                printf("\n");
        }
    getch();
}
```

14. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
        int A, B;
        for (A = 9; A > 0; A--)
        {
                for (B = 1; B <= A; B++)
                {
                        printf("%d", B);
                }
                printf("\n");
        }
        getch();
}
```

15. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 9; A > 0; A--)
        {
                for (B = 9; B > A; B--)
                {
```

```
                printf(" ");
        }

        for (B = 1; B <= A; B++)
        {
                printf("%d", B);
        }

        for (B = B-2; B > 0; B--)
        {
                printf("%d", B);
        }
        printf("\n");
    }
    getch();
}
```

16. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B;
    for (A = 9; A > 0; A--)
    {
        for (B = 9; B > A; B--)
        {
                printf(" ");
        }

        for (B = 1; B <= A; B++)
        {
                printf("%d", B);
        }
        printf("\n");
    }
    getch(); }
```

17. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 1; A < 10; A++)
        {
                for (B = 1; B <= A; B++)
                {
                        if (A == B)
                        {
                                printf("%d", A);
                        }
                        else
                        {
                                printf(" ");
                        }
                }
                        printf("\n");
        }
    getch();
}
```

18. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
	int A, B;
	for (A = 1; A < 10; A++)
	{
		for (B = 9; B > A; B--)
		{
			printf(" ");
		}

		for (B = 1; B <= A; B++)
		{
			printf("%d", B);
		}

		for (B = B-2; B > 0; B--)
		{
			printf("%d", B);
		}
		printf("\n");
	}
	getch();
}
```

19. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
	int A, B;

	for (A = 1; A < 10; A++)
	{
		for (B = 1; B < A; B++)
```

```
                {
                        printf("%d", B);
                }
                        printf("\n");
        }

        for (A = A - 1; A > 0; A--)
        {
                for (B = 1; B <= A; B++)
                {
                        printf("%d", B);
                }
                printf("\n");
        }

        getch();
}
```

20. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 1; A < 10; A++)
        {
                for (B = 9; B > A; B--)
                {
                        printf(" ");
                }

                for (B = 1; B <= A; B++)
                {
                        if (B == 1)
                        {
                                printf("%d", A);
                        }
```

```
                        else
                        {
                                printf(" ");
                        }
                }
                for (B = B-2; B > 0; B--)
                {
                        if (B == 1)
                        {
                                printf("%d", A);
                        }
                        else
                        {
                                printf(" ");
                        }
                }
                printf("\n");
        }
        getch();
}
```

21. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

```
#include <stdio.h>
#include <conio.h>
main()
{
        int A, B;
        for (A = 1; A < 10; A++)
        {
                for (B = 9; B > A; B--)
                {
                        printf(" ");
                }

                for (B = 1; B <= A; B++)
                {
                        printf("%d", B);
```

```
                }
                printf("\n");
        }

        for (A = A - 2; A > 0; A--)
        {
                for (B = 9; B > A; B--)
                {
                        printf(" ");
                }

                for (B = 1; B <= A; B++)
                {
                        printf("%d", B);
                }
                printf("\n");
        }
        getch();
}
```

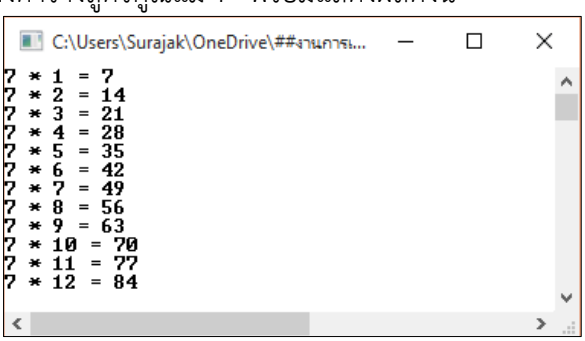
22. จงแสดงผลลัพธ์ และอธิบายการทำงานของโปรแกรม

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

main()
{
        int intNum, intAns;
        intAns = 0;

        for (intNum = 1; intNum <= 12; intNum++)
        {
                intAns = intNum * 17;
                printf("17 * %d = %d", intNum, intAns);
                printf("\n");
        }
        getch();
}
```

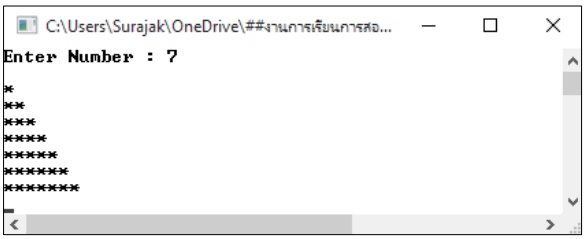
■ แบบฝึกหัดท้ายบทที่ 7

➢ จงเขียนโปรแกรมดังต่อไปนี้

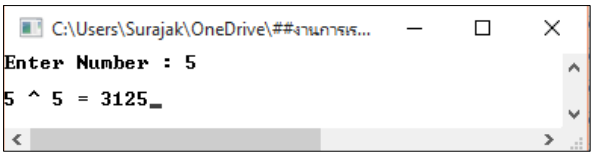
1. จงเขียนโปรแกรมแสดงตารางสูตรคูณแม่ 7 พร้อมแสดงผลดังนี้



2. จงเขียนโปรแกรมรับค่าเป็นจำนวนเต็ม พร้อมแสดงผลดังนี้



3. จงเขียนโปรแกรมรับค่าเป็นจำนวนเต็ม พร้อมแสดงผลเลขยกกำลังดังนี้



4. จงเปลี่ยนโค้ดโปรแกรมต่อไปนี้ จากคำสั่ง for เป็นคำสั่ง while

```
#include <stdio.h>
#include <conio.h>
main()
{
      int intNum, intSum;
      for (intNum = 1; intNum <= 5; intNum++)
      {
          printf("\t%d", intNum);
      }
      getch();
}
```

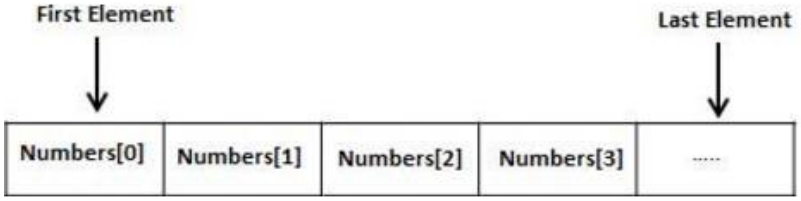
5. จากโค้ดข้อ 4 จงเปลี่ยนโค้ดคำสั่งจาก while เป็น do-while

บทที่ 8
อาร์เรย์(Array)

ตัวแปรประเภทหนึ่งที่ใช้ชื่อตัวแปรชื่อเดียวแต่สามารถเก็บข้อมูลเป็นกลุ่มได้เรียกว่าตัวแปรแบบอาร์เรย์ ถ้าหากประกาศตัวแปรประเภทนี้ขึ้นมาผู้ใช้สามารถเก็บข้อมูลหลายๆ ค่าติดๆ กันได้และสามารถเรียกข้อมูลแต่ละค่าขึ้นมาได้ ตัวแปรแบบอาร์เรย์นี้มีทั้งแบบหนึ่งมิติและหลายมิติ สำหรับตัวแปรแบบสตริงก็คือตัวแปรที่นำมาตัวอักขระมาต่อกันเป็นอาร์เรย์ประเภทหนึ่ง

ประเภทของข้อมูลที่กล่าวมาในบทต้นๆ เป็นข้อมูลแบบข้อมูลเดี่ยว โดยตัวแปรหนึ่งตัวสามารถเก็บข้อมูลได้หนึ่งตัว ถ้าหากต้องการเก็บข้อมูลหลายตัวจะต้องประกาศตัวแปรหลายตัว ซึ่งจะไม่สะดวกในการเก็บข้อมูลจำนวนมาก ในภาษาซีถ้าหากต้องการเก็บข้อมูลเป็นกลุ่มจะต้องใช้ตัวแปรประเภทอาร์เรย์ (Array) โดยอาร์เรย์จะทำหน้าที่จองเนื้อที่ในหน่วยความจำตามขนาดที่ระบุ และแบ่งหน่วยความจำนั้นออกเป็นช่องๆ ถ้าหากต้องการใช้หน่วยความจำตัวใดก็สามารถอ้างหน่วยความจำนั้นแล้วดึงมาใช้ได้

อาร์เรย์จะประกอบด้วยข้อมูลหลายๆ ตัวรวมกันเป็นกลุ่ม ข้อมูลแต่ละตัวในกลุ่มนั้นจะเรียกว่าอีลีเมนต์ **(Element)** หรือ **(Cell)** ในการอ้างถึงข้อมูลแต่ละเซลล์จะใช้อินเด็กซ์ **(Index)** เป็นตัวชี้



ตัวอย่างเช่น ถ้าเรามีข้อมูลอยู่กลุ่มหนึ่งซึ่งเป็นคะแนนของนักศึกษา 8 คน สามารถเก็บได้ดังนี้

หมายเลข	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]
คะแนน	18	20	35	84	21	45	65	71	39

จากข้อมูลข้างต้น สามารถบอกได้ว่าข้อมูลอยู่ในอาร์เรย์ชื่อ X ในแต่ละเซลล์จะเก็บเลขจำนวนเต็ม ส่วนตัวเลขที่อยู่ในเครื่องหมาย square brackets ([ ]) เรียกว่าอินเด็กซ์ ซึ่งจะต้องเป็นข้อมูลประเภทจำนวนเต็มเท่านั้น ถ้าหากต้องการติดต่อกับเซลล์ใดก็ให้ใช้อินเด็กซ์เป็นตัวชี้ ถ้าหากเราอ้างตัวแปร X[3] หมายความว่าเป็นการติดต่อกับอาร์เรย์ X เซลล์ที่ 3

ตัวอย่าง X[2] อ้างเซลล์ที่ 2 มีค่าเท่ากับ 35
X[2] + X[3] นำเซลล์ที่ 2 บวกกับเซลล์ที่ 3 จะได้ 35 + 84 เท่ากับ 119
X[1+3] อ้างเซลล์ที่ 4 มีค่าเท่ากับ 21
X[5] + 1 นำเซลล์ที่ 5 มาบวกด้วย 1 จะได้เท่ากับ 46

ถ้าหากในโปรแกรมมีการประกาศตัวแปรอาร์เรย์เอาไว้ เราสามารถนำตัวแปรนี้มาใช้ได้เหมือนกับตัวแปรทั่วไป เช่น

X[5] = 45; ใส่ค่า 45 ลงในตัวแปรอาร์เรย์ X เซลล์ที่ 5

printf("%d\n", X[6]); พิมพ์ค่าในตัวแปรอาร์เรย์ X เซลล์ที่ 6

■ ตัวแปรอาร์เรย์ 1 มิติ

ลักษณะของตัวแปรอาร์เรย์แบบ 1 มิติจะเก็บข้อมูลต่อเนื่องกันไปเป็นแถว การประกาศตัวแปรจะใช้ชื่อเดียวตามเครื่องหมาย [ ] คร่อมค่าตัวเลขที่บอกจำนวนของข้อมูลที่ต้องการ โดยมีรูปแบบดังนี้

รูปแบบ

```
type arrayName [ arraySize ];
```

หรือ

```
ประเภทข้อมูล   ชื่อตัวแปร [จำนวนสมาชิก];
```

โดย type จะเป็นประเภทของข้อมูลในภาษาซีที่จะเก็บ, arrayName เป็นชื่อตัวแปรอาร์เรย์ และ arraySize เป็นจำนวนเซลล์ข้อมูลในอาร์เรย์ ตัวอย่าง เช่น ถ้าประกาศตัวแปรอาร์เรย์ชื่อว่า balanceมีชนิดข้อมูลเป็น double โดยมีข้อมูลจำนวน 10 ตัว แต่ละตัวเก็บค่าเลขทศนิยมสามารถเขียนได้ดังนี้

ตัวอย่างการประกาศตัวแปรอาร์เรย์

```
double balance[10];
```

1. การกำหนดค่าข้อมูลให้ตัวแปรอาร์เรย์

การกำหนดค่าให้ตัวแปรอาร์เรย์นั้น มีรูปแบบเหมือนกับการกำหนดค่าให้ตัวแปรทั่วไป ต่างกันเพียงแต่การกำหนดค่าให้ตัวแปรอาร์เรย์นั้น เราต้องกำหนดตำแหน่งอินเด็กซ์เพื่อระบุตำแหน่งของตัวแปรที่จะกำหนดค่า ยกตัวอย่างเช่น

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

หรือ

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

2. การอ้างถึงข้อมูลในตัวแปรอาร์เรย์

ข้อมูลที่มีการเข้าถึงโดยการจัดทำดัชนีชื่ออาร์เรย์ จะกระทำโดยการวางดัชนีของข้อมูลภายในวงเล็บหลังชื่อของอาร์เรย์ ยกตัวอย่างเช่น

```
double salary = balance[9];
```

ข้อความข้างต้นจะใช้สมาชิก 10 ตัว จากอาร์เรย์ และกำหนดค่าให้กับตัวแปร salary ตัวอย่างต่อไปนี้แสดงวิธีการใช้ทั้งสามแนวความคิดดังกล่าวข้างต้น ได้แก่ การประกาศ การกำหนดค่า และการเข้าถึงอาร์เรย์

ตัวอย่างที่ 8.1

```
#include <stdio.h>

int main () {

   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ ) {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

■ ตัวแปรอาร์เรย์ 2 มิติ

อาร์เรย์แบบ 2 มิติ เปรียบได้กับการนำตัวแปรมาเรียงต่อกันหลายๆ ตัวในลักษณะของตารางข้อมูล จะเป็นการเก็บข้อมูลในแนวแถวและคอลัมน์ ซึ่งการนำตัวแปรอาร์เรย์ 2 มิติมาใช้มีรูปแบบดังนี้

รูปแบบ

```
type arrayName [ x ][ y ];
```

ตัวอย่าง

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

จะได้ตู้เก็บหน่วยความจำที่ใช้สำหรับเก็บค่า n จำนวน 12 ตัว ดังนี้

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

1. การกำหนดค่าให้กับตัวแปรอาร์เรย์

การกำหนดค่าให้กับตัวแปร อาร์เรย์นั้น มีรูปแบบเหมือนการกำหนดค่าให้ตัวแปรทั่วไป ต่างกันเพียงเราต้องกำหนดตำแหน่ง index เพื่อระบุตำแหน่งของตัวแปรที่กำหนดค่า ตัวอย่างเช่น

```
int a[3][4] = {
   {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
   {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
   {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

หรือ

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

2. การอ้างถึงข้อมูลในตัวแปรอาร์เรย์

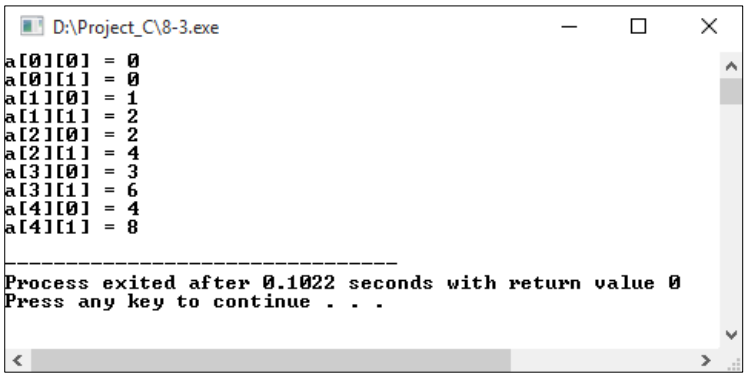
เมื่อมีการกำหนดค่าให้ตัวแปรแล้ว ต่อไปต้องรู้วิธีการอ่านข้อมูลจากตัวแปรโดยการระบุตำแหน่งข้อมูลที่ต้องการลงไป หรือเรียกว่า index มีตัวอย่างดังนี้

```
int val = a[2][3];
```

ตัวอย่าง การใช้ nested loops กับ อาร์เรย์ 2 มิติ

```c
#include <stdio.h>
int main () {
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;
    /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {
      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }
   return 0;
}
```

ผลลัพธ์โปรแกรม



3. การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

โปรแกรมภาษาซี ไม่อนุญาตส่งค่าผ่านอาร์เรย์เป็นอาร์กิวเมนต์ของฟังก์ชัน แต่คุณสามารถส่งค่าพ้อยเตอร์ไปยังอาร์เรย์โดยระบุชื่ออาร์เรย์โดยไม่ต้องใช้ดัชนี หากนักศึกษาต้องการที่จะส่งค่าอาร์เรย์มิติเดียวจากฟังก์ชั่นที่ จะต้องมีการประกาศฟังก์ชั่น ที่เป็นพ้อยเตอร์ดังตัวอย่าง

ตัวอย่าง

```c
int * myFunction() {
   .
   .
   .
}
```

```c
#include <stdio.h>
/* function to generate and return random numbers */
```

```
int * getRandom( ) {
   static int  r[10];
   int i;
   /* set the seed */
   srand( (unsigned)time( NULL ) );

   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf( "r[%d] = %d\n", i, r[i]);
   }
   return r;
}
/* main function to call above defined function */
int main () {

   /* a pointer to an int */
   int *p;
   int i;
   p = getRandom();
   for ( i = 0; i < 10; i++ ) {
      printf( "*(p + %d) : %d\n", i, *(p + i));
   }
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
r[0] = 313959809
r[1] = 1759055877
r[2] = 1113101911
r[3] = 2133832223
r[4] = 2073354073
r[5] = 167288147
r[6] = 1827471542
r[7] = 834791014
r[8] = 1901409888
r[9] = 1990469526
*(p + 0) : 313959809
```

```
*(p + 1) : 1759055877
*(p + 2) : 1113101911
*(p + 3) : 2133832223
*(p + 4) : 2073354073
*(p + 5) : 167288147
*(p + 6) : 1827471542
*(p + 7) : 834791014
*(p + 8) : 1901409888
*(p + 9) : 1990469526
```

■ สรุปท้ายบท

ตัวแปรแบบอาร์เรย์ เป็นตัวแปรที่รวมข้อมูลประเภทเดียวกันเอาไว้เป็นกลุ่มภายใต้ชื่อเดียวกัน ข้อมูลแต่ละตัวในกลุ่มนั้นเรียกว่า อีลีเมนต์ หรือ เซลล์ การระบุสมาชิกแต่ละตัวในอาร์เรย์จะใช้อินเด็กซ์เป็นตัวชี้ ตัวแปรแบบอาร์เรย์นี้อาจมองว่ามีข้อมูลต่อกันเป็นแถว ถ้าเป็นแถวเดี่ยวเรียกว่าอาร์เรย์แบบหนึ่งมิติ นอกจากนี้ในภาษาซียังมีอาร์เรย์แบบหลายมิติให้ใช้อีกด้วย

■ การทดลอง

1. จงเขียนโปรแกรมตรวจสอบข้อความที่รับจากคีย์บอร์ด (ข้อความไม่เกิน 30 ตัวอักษร) ว่ามีตัวอักษรทั้งหมดกี่ตัว และเป็นอักษรตัวเล็กกี่ตัว

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

main()
{
        int intCount = 0, intCheck = 0, nStr = 0;
        char str[50], strAns[50];
        printf("Enter String : ");
        scanf("%s", &str);

        while (str[nStr] != '\0')
        {
                if ((str[nStr] > 96) && (str[nStr] < 123))
                {
                        strAns[intCount] = str[nStr];
                        intCount++;
                }
                nStr++;
        }
```

```
        printf("\Small character = ");

        for (intCheck = 0; intCheck < intCount; intCheck++)
        {
                putchar(strAns[intCheck]);
        }

        printf("\nThe number of All character = %d", nStr);
        printf("\nThe number of Small  character = %d", intCount);
        getch();
}
```

2. จงเขียนโปรแกรมตรวจสอบข้อความที่รับจากคีย์บอร์ด (ข้อความไม่เกิน 30 ตัวอักษร) ว่ามีตัวอักษรทั้งหมดกี่ตัว และเป็นสระกี่ตัว

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

main()
{
        int intCount = 0, intNum = 0;
        int intCount_A = 0, intCount_E = 0, intCount_I = 0, intCount_O = 0,
intCount_U = 0;
        char str[50];
        printf("Enter String : ");
        scanf("%s", &str);

        intNum = sizeof(str);

        for(intCount = 0; intCount < intNum; intCount++)
        {
                switch(str[intCount])
                {
                        case 'A' :
                        case 'a' :
                                intCount_A ++;
                                break;
                        case 'E' :
```

```
                        case 'e' :
                                intCount_E ++;
                                break;
                        case 'I' :
                        case 'i' :
                                intCount_I ++;
                                break;
                        case 'O' :
                        case 'o' :
                                intCount_O ++;
                                break;
                        case 'U' :
                        case 'u' :
                                intCount_U ++;
                                break;
                }
        }
        printf("\nThe number of A character = %d", intCount_A);
        printf("\nThe number of E character = %d", intCount_E);
        printf("\nThe number of I character = %d", intCount_I);
        printf("\nThe number of O character = %d", intCount_O);
        printf("\nThe number of U character = %d", intCount_U);
        getch();
}
```

3. จงเขียนโปรแกรมตรวจสอบจำนวนเต็มที่รับจากคีย์บอร์ด 10 จำนวนว่าจำนวนคี่กี่ตัว และมีจำนวนคู่กี่ตัว

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

main()
{
        int intNum[10],intOdd[10],intEven[10];
        int intCount,intN;
        int intCountOdd = 0, intCountEven = 0;

        printf("Enter total number : ");
```

```
	scanf("%d",&intN);

	for (intCount=0; intCount < intN; intCount++)
	{
		printf("Enter number %d : ", intCount + 1);
		scanf("%d",&intNum[intCount]);
	}

	for (intCount = 0; intCount < intN; intCount++)
	{
		if ((intNum[intCount] % 2) == 0)
		{
			intEven[intCountEven] = intNum[intCount];
			intCountEven++;
		}
		if ((intNum[intCount] % 2) != 0)
		{
			intOdd[intCountOdd] = intNum[intCount];
			intCountOdd++;
		}
	}

	printf("\n\nEven number : ");

	for (intCount = 0; intCount< intCountEven; intCount++)
	{
		printf("\t%d",intEven[intCount]);
	}

	printf("\n\nOdd number : ");

	for (intCount = 0; intCount < intCountOdd; intCount++)
	{
		printf("\t%d",intOdd[intCount]);
	}
	getch();
}
```

4. จงเขียนโปรแกรมรับข้อความที่รับจากคีย์บอร์ดไม่เกิน 30 ตัวอักษร และตรวจสอบอักขระว่ามีข้อความกี่ตัว และที่ตำแหน่งใดของข้อความบ้าง

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

main()
{
	char str[30];
	int intPosition[30];
	char chSearch;
	int intCount = 0, intCountChar = 0;

	printf("Enter string : ");
	gets(str);
	printf("Enter a character : ");
	scanf("%c", &chSearch);

	while (str[intCount] != '\0')
	{
		if (str[intCount] == chSearch)
		{
			intPosition[intCountChar] = intCount + 1;
			intCountChar++;
		}
		intCount++;
	}

	printf("\n\nFind character %c = %d  in the intPositionition : ", chSearch,
intCountChar);

	for (intCount = 0; intCount < intCountChar; intCount++)
	{
		printf("\t%d ",intPosition[intCount]);
	}
	getch();
}
```

5. จงเขียนโปรแกรมรับข้อความที่รับจากคีย์บอร์ดไม่เกิน 30 ตัวอักษร และตรวจสอบอักขระที่รับมาว่ามีข้อความกี่ตัว พร้อมแสดงผลข้อความสลับจากหลังมาหน้า

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

main()
{
        char str[30];
        int i,j,count = 0;

        printf("Enter string : ");
        gets(str);

        while (str[count] != '\0')
        {
                count++;
        }

        count--;

        while (count >= 0)
        {
                printf("%c", str[count]);
                count--;
        }

        getch();
}
```

■ แบบฝึกหัดท้ายบทที่ 8

ตอนที่ 1 จงตอบคำถาม

1. จงพิจารณาความถูกต้องของการประกาศอาร์เรย์ หากข้อใดไม่ถูกต้องให้อธิบายพร้อมแก้ไขให้ถูกต้อง

   1.1 long double ldbTax[4];

   1.2 char ch[2][2];

   1.3 char chName[ ];

1.4 unsigned int uintMark[10] = {324, 726, 85};

1.5 float fWeight = {63.1, 45.9, 76.7};

2. จงพิจารณาความถูกต้องของการกำหนดค่าให้กับตัวแปรอาร์เรย์ หากข้อใดไม่ถูกต้องให้อธิบายพร้อมแก้ไขให้ถูกต้อง

 2.1 int intCount[ ] = {1, 2, 3, 4, 5};

 2.2 int intCount[5] = {1; 2; 3; 4; 5};

 2.3 char chStr[10] = {S; 2; 3; P; R};

 2.4 float fArea[ ] = {‘A’, ‘E’, ‘I’, ‘O’, ‘U’};

 2.5 int intValue[ ][ ] = {6, 2, 8, 4, 5, 4};

 2.6 float fWeight[ ][ ] = {3.1, 3.7, 1.9};

3. จงบอกจำนวนข้อมูลที่ตัวแปรอาร์เรย์แต่ละตัวสามารถเก็บค่าได้ พร้อมอธิบายความหมาย

ตัวอย่างเช่น A[2][3]; จะเห็นได้ว่าตัวแปร A เป็นตัวแปรอาร์เรย์ 2 มิติ ชนิดจำนวนเต็ม มี 2 แถว 3 คอลัมน์ สามารถเก็บค่าได้ 6 ค่าเป็นต้น

3.1 int intAge[2][10];

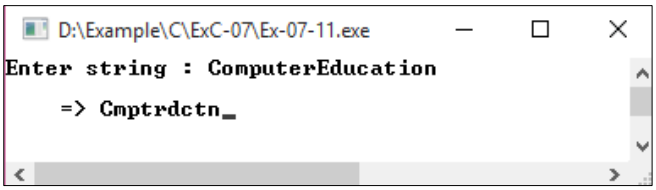
3.2 char chGrade[3][6];

3.3 double db[2][4];

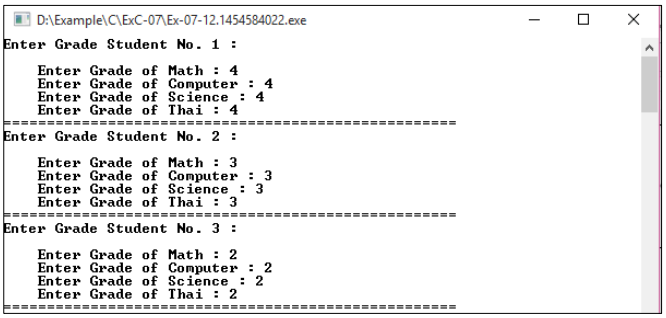
3.4 float fGPA[2][4];

ตอนที่ 2 จงเขียนโปรแกรม

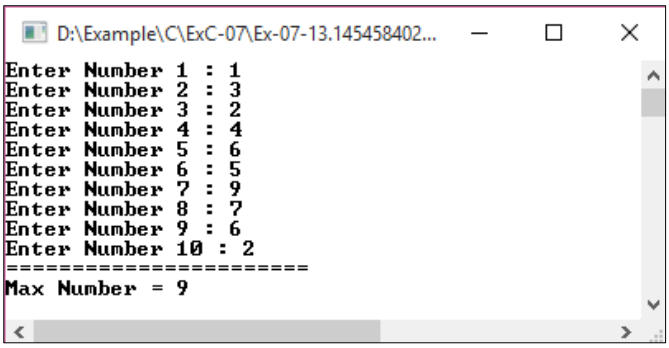
1. จงเขียนโปรแกรมรับข้อความจากคีย์บอร์ดไม่เกิน 30 ตัวอักษร พร้อมแสดงผลข้อความที่ตัดสระจากข้อความ ดังนี้



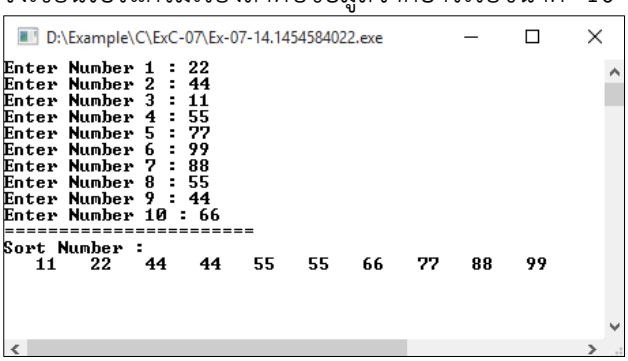
2. จงเขียนโปรแกรมรับค่าเกรดแต่ละรายวิชาของนักเรียน 4 วิชา เป็นจำนวน 3 คน พร้อมแสดงเกรดเฉลี่ยโดยใช้ตัวแปรอาร์เรย์ ดังนี้



3. จงเขียนโปรแกรมหาค่าสูงสุดของอาร์เรย์ขนาด 10 จำนวน ดังนี้



4. จงเขียนโปรแกรมเรียงลำดับข้อมูลจากอาร์เรย์ขนาด 10 จำนวน ดังนี้



บทที่ 9
พอยน์เตอร์ (Pointer)

ตัวแปรพอยน์เตอร์(Pointers) คือ ตัวแปรชนิดพิเศษในภาษาซี มีหน้าที่เก็บตำแหน่งที่อยู่(Address) ของตัวแปรอื่นๆ ที่อยู่ในหน่วยความจำ ซึ่งต่างจากตัวแปรทั่วไปเพราะจะเก็บตำแหน่งที่อยู่จากนั้นก็จะให้ที่อยู่นี้อ้างอิงด้วยการชี้ไปยังที่อยู่ตัวแปรนั้นแทน มีรูปแบบการใช้งานดังนี้

รูปแบบ

```
type *var-name;
```

โดยที่

type คือ ชนิดของตัวแปรพอยน์เตอร์
var-name คือ ชื่อของตัวแปรพอยน์เตอร์
* คือ เครื่องหมายที่กำหนดให้ตัวแปรที่ประกาศเป็นพอยน์เตอร์

ตัวอย่างการประกาศตัวแปรพอยน์เตอร์

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

■ การใช้งานพอยน์เตอร์

การใช้งานพอยน์เตอร์จะมีการใช้งาน 5 ประเด็นดังนี้

1. ตัวดำเนินการพอยน์เตอร์ (Pointer arithmetic)
2. พอยน์เตอร์ของอาร์เรย์ (Array of pointers)
3. พอยน์เตอร์ไปยังพอยน์เตอร์ (Pointer to pointer)
4. การส่งพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Passing pointers to functions in C)
5. การคืนค่าพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Return pointer from functions in C)

มีการดำเนินงานที่สำคัญของพอยน์เตอร์ ดังนี้ (ก) การกำหนดตัวแปรพอยน์เตอร์ (ข) กำหนดที่อยู่ของตัวแปรที่จะชี้ไป และ (ค) การเข้าถึงค่าที่อยู่ที่มีอยู่ในตัวแปรพอยน์เตอร์

- การใช้งานเครื่องหมาย & เพื่ออ้างอิงตำแหน่งที่อยู่ของตัวแปร
- การใช้งานเครื่องหมาย * เพื่ออ้างอิงถึงข้อมูลที่ตำแหน่งที่อยู่ของตัวแปร สามารถศึกษาได้จากตัวอย่างดังต่อไปนี้

ตัวอย่าง 9.1

```
#include <stdio.h>

int main () {

   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Address of var variable: bffd8b3c
Address stored in ip variable: bffd8b3c
Value of *ip variable: 20
```

1. การดำเนินการกับพอยน์เตอร์ (Pointer arithmetic)

การดำเนินการกับพอยน์เตอร์มี 4 ตัวดำเนินการ คือ ++,--,+,-

1) พอยน์เตอร์การเพิ่ม (Incrementing a Pointer)

การอ้างถึงการใช้พอยน์เตอร์ ในโปรแกรมของเราแทนอาร์เรย์เพราะตัวแปรพอยน์เตอร์สามารถเพิ่มขึ้น ซึ่งแตกต่างจากชื่ออาร์เรย์ที่ไม่สามารถเพิ่มขึ้นเพราะมันเป็น พอยเตอร์ค่าคงที่ โปรแกรมสามารถเพิ่มตัวแปรพอยเตอร์ในการเข้าถึงแต่ละข้อมูลของอาเรย์

ตัวอย่าง 9.2

```
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = var;

   for ( i = 0; i < MAX; i++) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* move to the next location */
      ptr++;
   }

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Address of var[0] = bf882b30
Value of var[0]   = 10
Address of var[1] = bf882b34
Value of var[1]   = 100
Address of var[2] = bf882b38
Value of var[2]   = 200
```

2) พอยน์เตอร์การลด (Decrementing a Pointer)

พอยน์เตอร์จะเป็นการลดค่าของมันจากจำนวนไบต์ของชนิดข้อมูลที่แสดง

ตัวอย่าง 9.3

```
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = &var[MAX-1];

   for ( i = MAX; i > 0; i--) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* move to the previous location */
      ptr--;
   }

   return 0;
}
```

3) การเปรียบเทียบพอยน์เตอร์ (Pointer Comparisons)

พอยน์เตอร์สามารถนำมาเปรียบเทียบเชิงสัมพันธ์ระหว่างตัวดำเนินการ เช่น == <, and > ถ้า p1 และ p2 ชี้ไปยังตัวแปรที่เกี่ยวข้องกับอื่น ๆ เช่น องค์ประกอบของอาร์เรย์เดียวกันแล้ว p1 และ p2 สามารถเทียบความหมายได้

โปรแกรมต่อไปนี้จะปรับเปลี่ยน ดังตัวอย่างก่อนหน้านี้ โดยการเพิ่มตัวแปรพอยน์เตอร์ address จะชี้ให้เป็นทั้งน้อยกว่าหรือเท่ากับไปยังที่อยู่ของข้อมูลตัวสุดท้ายของอาร์เรย์ซึ่งเป็น & var [MAX - 1]

ตัวอย่าง 9.4

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have address of the first element in pointer */
   ptr = var;
   i = 0;

   while ( ptr <= &var[MAX - 1] ) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* point to the previous location */
      ptr++;
      i++;
   }

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Address of var[0] = bfdbcb20
Value of var[0] = 10
Address of var[1] = bfdbcb24
Value of var[1] = 100
Address of var[2] = bfdbcb28
Value of var[2] = 200
```

2. พอยน์เตอร์ของอาร์เรย์ (Array of pointers)

สามารถกำหนดอาร์เรย์ที่เก็บจำนวนของจำนวนในพอยน์เตอร์

ตัวอย่าง 9.5

```
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int i;

   for (i = 0; i < MAX; i++) {
      printf("Value of var[%d] = %d\n", i, var[i] );
   }

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Value of var[0] = 10
Value of var[1] = 100
Value of var[2] = 200
```

อาจจะมีสถานการณ์เมื่อเราต้องการที่จะเก็บอาร์เรย์ซึ่งสามารถเก็บพอยน์เตอร์ไปยัง int หรือchar หรือชนิดข้อมูลอื่น ๆ การประกาศของอาเรย์ของพอยน์เตอร์ไปยังจำนวนเต็ม ดังนี้

```
int *ptr[MAX];
```

นอกจากนี้คุณยังสามารถใช้อาร์เรย์ของพอยน์เตอร์ไปยังตัวอักษร(charactor) ในการจัดเก็บรายการของสตริงดังต่อไปนี้

ตัวอย่าง 9.5

```
#include <stdio.h>
const int MAX = 4;
int main () {

   char *names[] = {
```

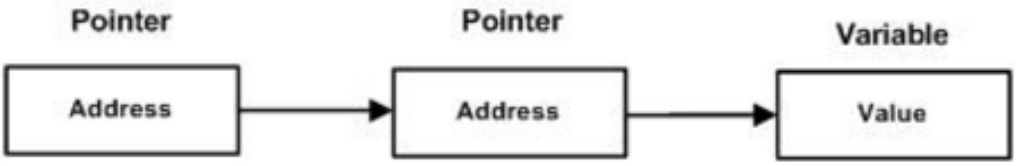
```
      "Zara Ali",
      "Hina Ali",
      "Nuha Ali",
      "Sara Ali",
   };

   int i = 0;

   for ( i = 0; i < MAX; i++) {
      printf("Value of names[%d] = %s\n", i, names[i] );
   }

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Value of names[0] = Zara Ali
Value of names[1] = Hina Ali
Value of names[2] = Nuha Ali
Value of names[3] = Sara Ali
```

3. พอยน์เตอร์ไปยังพอยน์เตอร์ (Pointer to pointer)

ภาษา C อนุญาตให้มีพอยน์เตอร์บนพอยน์เตอร์ และ อื่นๆ



ตัวอย่าง การประกาศตัวแปรพอยน์เตอร์ไปยังพอยน์เตอร์แบบจำนวนเต็ม

```
int **var;
```

ตัวอย่าง 9.7

```
#include <stdio.h>

int main () {
```

```
   int  var;
   int  *ptr;
   int  **pptr;

   var = 3000;

   /* take the address of var */
   ptr = &var;

   /* take the address of ptr using address of operator & */
   pptr = &ptr;

   /* take the value using pptr */
   printf("Value of var = %d\n", var );
   printf("Value available at *ptr = %d\n", *ptr );
   printf("Value available at **pptr = %d\n", **pptr);

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

4. การส่งพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Passing pointers to functions in C)

การส่งอาร์กิวเมนต์โดยการอ้างอิง หรือโดยการเปิด address ที่ใช้งานอยู่ผ่านอาร์กิวเมนต์ส่งไปเปลี่ยนแปลงในฟังก์ชั่น ในการเรียกฟังก์ชั่นโดยตัวเรียกฟังก์ชั่น

การเขียนโปรแกรมภาษา C ช่วยให้ผ่านพอยน์เตอร์ไปยังฟังก์ชัน ต้องการทำเช่นนั้นก็ประกาศพารามิเตอร์ของฟังก์ชันเป็นชนิดพอยน์เตอร์ ดังตัวอย่างที่ส่งพอยน์เตอร์ไปยัง unsigned long pointer ไปยัง ฟังก์ชันและเปลี่ยนค่าฟังก์ชั่นภายในที่สะท้อนให้เห็นถึงการเรียกคืนฟังก์ชั่น

ตัวอย่าง 9.8

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

```
void getSeconds(unsigned long *par);
int main () {
   unsigned long sec;
   getSeconds( &sec );

   /* print the actual value */
   printf("Number of seconds: %ld\n", sec );
   return 0;
}
void getSeconds(unsigned long *par) {
   /* get the current number of seconds */
   *par = time( NULL );
   return;
}
```

ผลลัพธ์โปรแกรม

```
Number of seconds :1294450468
```

ตัวอย่าง 9.8 ฟังก์ชั่นที่สามารถรับพอยน์เตอร์และรับอาร์เรย์

```
#include <stdio.h>

/* function declaration */
double getAverage(int *arr, int size);

int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argument */
   avg = getAverage( balance, 5 ) ;

   /* output the returned value  */
   printf("Average value is: %f\n", avg );
   return 0;
}
```

```
double getAverage(int *arr, int size) {

   int  i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = (double)sum / size;
   return avg;
}
```

ผลลัพธ์โปรแกรม

```
Average value is: 214.40000
```

5. การคืนค่าพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Return pointer from functions in C)

ภาษา C อนุญาตให้ฟังก์ชันไปเรียกคืนพอยน์เตอร์ ไปยัง ตัวแปรเฉพาะ (local variable) ตัวแปรคงที่(static variable) และ หน่วยความจำจัดสรรแบบไดนามิก

เราได้เห็นในบทที่ผ่านมาว่าการเขียนโปรแกรมภาษาซีอนุญาตให้คืนค่าอาร์เรย์จากฟังก์ชั่น ในทำนองเดียวกันภาษาซียังอนุญาตให้คืนค่าพอยน์เตอร์จากฟังก์ชั่น ต้องการทำเช่นนั้นคุณจะต้องประกาศฟังก์ชั่นที่กลับมาเป็นพอยน์เตอร์ในตัวอย่างต่อไปได้

```
int * myFunction() {
   .
   .
   .
}
```

ประเด็นที่สองที่ต้องจำไว้คือว่ามันไม่ได้เป็นความคิดที่ดีที่จะกลับไปที่อยู่ของตัวแปรท้องถิ่น (local variable)นอกฟังก์ชั่น เพื่อให้คุณจะต้องกำหนดตัวแปรท้องถิ่น(local variable)เป็นตัวแปรคงที่(static variable)

ตอนนี้พิจารณาฟังก์ชันต่อไปนี้ซึ่งจะสร้างตัวเลขสุ่ม 10 และ คืนค่าโดยใช้ชื่ออาร์เรย์ซึ่งหมายถึงพอยน์เตอร์ นั่นคือที่อยู่ของข้อมูลของอาร์เรย์แรก

ตัวอย่าง 9.10

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

```c
#include <time.h>

/* function to generate and retrun random numbers. */
int * getRandom( ) {

   static int  r[10];
   int i;

   /* set the seed */
   srand( (unsigned)time( NULL ) );

   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf("%d\n", r[i] );
   }

   return r;
}

/* main function to call above defined function */
int main () {

   /* a pointer to an int */
   int *p;
   int i;
   p = getRandom();
   for ( i = 0; i < 10; i++ ) {
      printf("*(p + [%d]) : %d\n", i, *(p + i) );
   }
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
1523198053
1187214107
1108300978
430494959
```

```
1421301276
930971084
123250484
106932140
1604461820
149169022
*(p + [0]) : 1523198053
*(p + [1]) : 1187214107
*(p + [2]) : 1108300978
*(p + [3]) : 430494959
*(p + [4]) : 1421301276
*(p + [5]) : 930971084
*(p + [6]) : 123250484
*(p + [7]) : 106932140
*(p + [8]) : 1604461820
*(p + [9]) : 149169022
```

■ สรุปท้ายบท

ตัวแปรพอยน์เตอร์(Pointers) คือ ตัวแปรชนิดพิเศษในภาษาซี มีหน้าที่เก็บตำแหน่งที่อยู่(Address) ของตัวแปรอื่นๆ ที่อยู่ในหน่วยความจำ ซึ่งต่างจากตัวแปรทั่วไปเพราะจะเก็บตำแหน่งที่อยู่จากนั้นก็จะให้ที่อยู่นี้อ้างอิงด้วยการชี้ไปยังที่อยู่ตัวแปรนั้นแทน

การใช้งานพอยน์เตอร์จะมีการใช้งาน 5 ประเด็นดังนี้

1. ตัวดำเนินการพอยน์เตอร์ (Pointer arithmetic)
2. พอยน์เตอร์ของอาร์เรย์ (Array of pointers)
3. พอยน์เตอร์ไปยังพอยน์เตอร์ (Pointer to pointer)
4. การส่งพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Passing pointers to functions in C)
5. การคืนค่าพอยน์เตอร์ไปยังฟังก์ชั่นใน C (Return pointer from functions in C)

■ การทดลอง

1. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
```

```
        int A[5] = {1,3,5,7,9};
        int *ptA;
        ptA = &A[0];

        printf("%d\n", A[1]);
        printf("%d\n", *ptA);
        printf("%d\n", *(ptA + 1));
        printf("%d\n", *(A + 1));
        printf("%d\n", ptA[1]);
        getch();
}
```

2. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
        int B[5] = {5,10,15,20,25};
        int *ptB;

        ptB = &B[0];

        printf("%d\n",  B[1]);
        printf("%d\n", *ptB);
        printf("%d\n", *(ptB + 2));
        printf("%d\n", *(B + 2));
        printf("%d\n", ptB[1]);
        getch();
}
```

3. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
```

```
        int A[5] = {10,9,3,12,15};
        int *ptA;
        ptA = &A[2];

        A[1] = A[3] + 7;
        A[3] = *(ptA + 2) + A[2];

        printf("%d\n", A[1]);
        printf("%d\n", *ptA);
        printf("%d\n", *(ptA + 1));
        printf("%d\n", *(A + 1));
        printf("%d\n", ptA[1]);
        getch();
}
```

4. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
        int intCount, A[5] = {4,21,62,22,12};
        int *ptA;

        ptA = &A[0];

        if (A[3] > 50)
        {
                A[0] = A[1] + 1;
                A[1] = A[1] + 2;
        }

        printf("%d\n", A[4]);
        printf("%d\n", *ptA);
        printf("%d\n", *(ptA + 3));
        printf("%d\n", *(A + 1));
        printf("%d\n", ptA[0]);
```

```
        getch();
}
```

5. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
        int intCount, A[5] = {4,6,3,6,8};
        int *ptA;
        ptA = &A[1];
        if (*(ptA + 1) > 5)
        {
                A[0] = *(ptA + 2)+ 1;
                A[1] = A[1] + 2;
        }
        else
        {
                A[2] = A[2] + 3;
                A[3] = *(ptA + 1)+ 4;
        }
        printf("%d\n", A[2]);
        printf("%d\n", *ptA);
        printf("%d\n", *(ptA + 2));
        printf("%d\n", *(A + 1));
        printf("%d\n", ptA[3]);
        getch();
}
```

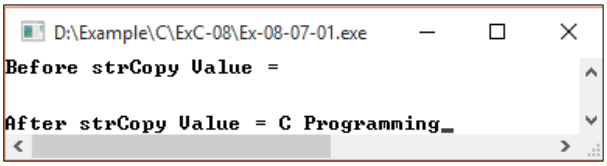
6. จงแสดงผลลัพธ์และอธิบายการทำงานของโปรแกรมต่อไปนี้

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

main()
{
        int intCount, intNum[5] = {31,35,62,17,82};
```

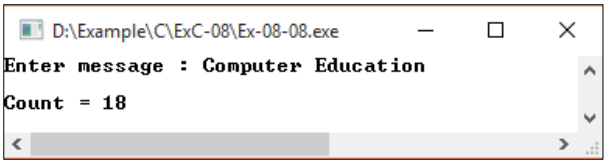
```
        int *ptC;
        ptC = &intNum[0];
        if (*(ptC + 1) > 50)
        {
                intNum[0] = *(ptC + 2) + *(ptC + 1);
                intNum[1] = intNum[1] + 2;
        }
        else
        {
                intNum[1] = intNum[2] + *(ptC + 3);
                intNum[2] = *(ptC + 1)+ 4;
        }
        printf("%d\n", intNum[1]);
        printf("%d\n", *ptC);
        printf("%d\n", *(ptC + 4));
        printf("%d\n", *(intNum + 1));
        printf("%d\n", ptC[3]);
        getch();
}
```

■ แบบฝึกหัดท้ายบทที่ 9

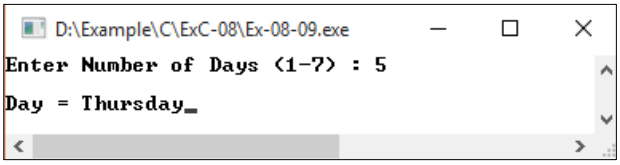
1. จงเขียนโปรแกรมคัดลอกข้อความจากตัวแปร X ไปยังตัวแปร Y โดยใช้พอยน์เตอร์ ดังตัวอย่างต่อไปนี้



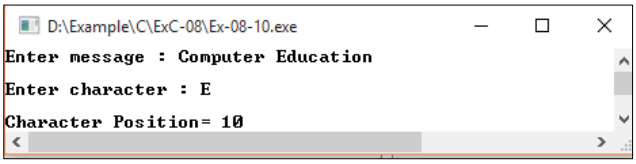
2. จงเขียนโปรแกรมรับข้อความคีย์บอร์ด แล้วแสดงว่าข้อความมีตัวอักษรทั้งหมดกี่ตัว โดยใช้พอยน์เตอร์ ดังตัวอย่างต่อไปนี้



3. จงเขียนโปรแกรมแสดงชื่อ วันที่ที่กำหนดค่าไว้ที่ตัวแปรอาร์เรย์ของพอยน์เตอร์ ดังตัวอย่างต่อไปนี้



4. จงเขียนโปรแกรมแสดงตำแหน่งข้อมูลในตัวแปรอาร์เรย์ที่ได้รับจากคีย์บอร์ด ไม่เกิน 10 ตัวอักษร ดังตัวอย่างต่อไปนี้



บทที่ 10
การจัดการข้อมูลชนิดสตริง

■ ข้อมูลชนิดสตริง

ตัวแปรสตริง คือ ชุดของตัวอักษรที่นำมาต่อเรียงกัน การกำหนดตัวแปรอาร์เรย์ให้เก็บข้อมูลชนิดอักขระจะทำให้เกิดสตริง โดยอักขระตัวสุดท้ายที่เก็บจะต้องเป็น null หรือ '\0' ซึ่งเป็นรหัสสิ้นสุดสตริง

สำหรับรหัส\0 ที่ปิดท้ายข้อความ ไม่จำเป็นต้องกำหนดลงไปก็ได้เนื่องจากภาษา C จะทำการแทรกเพิ่มเติมเข้าไปอัตโนมัติ

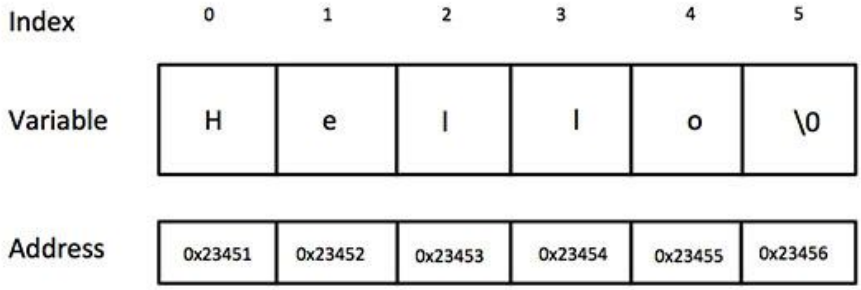
ตัวอย่างเช่น ถ้าจะประกาศ และกำหนดค่าเริ่มต้นตัวแปรสตริงที่ประกอบด้วยคำว่า "Hello" จะมีอักขระ null ปิดท้ายของอาร์เรย์ ขนาดของอักขระในอาร์เรย์ ที่มีสตริงเป็นมากกว่าหนึ่งจำนวนของตัวอักษรในคำว่า "Hello"

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

จะเหมือนกันกับการประกาศแบบนี้

```
char greeting[] = "Hello";
```

ต่อไปนี้เป็นตัวแทนหน่วยความจำของสตริงที่กำหนดไว้ข้างต้นใน C / C ++



ตัวอย่าง

```
#include <stdio.h>

int main () {

   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   printf("Greeting message: %s\n", greeting );
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Greeting message: Hello
```

■ ฟังก์ชั่นเกี่ยวกับสตริง

ในการเขียนโปรแกรมด้วยภาษาซีมีฟังก์ชั่นเกี่ยวกับสตริงที่น่าสนใจอยู่หลายฟังก์ชั่น โดยฟังก์ชั่นต่างๆ จะเก็บอยู่ในเฮดเดอร์ไฟล์ <string.h> ฟังก์ชั่นดำเนินการกับสตริงที่น่าสนใจและถูกนำมาใช้งานบ่อยมีดังนี้

ฟังก์ชัน	ความหมาย
strlen(s1)	ส่งคืนค่าความยาวสตริง s1
strcpy(s1,s2)	คัดลอกค่าสตริง s2 ไปเก็บไว้ที่สตริง s1
strcmp(s1,s2)	เปรียบเทียบค่าสตริง s1 กับ s2
strlwr(s1)	ใช้เปลี่ยนสตริง s1 ให้เป็นตัวพิมพ์เล็ก
strupr(s1)	ใช้เปลี่ยนสตริง s1 ให้เป็นตัวพิมพ์ใหญ่
strcat(s1,s2)	เชื่อมต่อ s2 ไปให้ s1
strrev(s1)	ใช้กลับตัวอักษรในสตริง s1 จากซ้ายไปขวา

ตัวอย่าง

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

int main () {

   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ;

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );
```

```
   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
strcpy( str3, str1) :  Hello
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```

■ การแปลงสตริงเป็นค่าตัวเลข

เนื้อหาต่อไปนี้ จะกล่าวถึงฟังก์ชันที่นำมาใช้แปลงค่าสตริงให้เป็นตัวเลข โดยฟังก์ชันแปลงสตริงเหล่านี้สามารถนำมาใช้ประโยชน์ได้ดี ในกรณีที่มีการรับค่าตัวเลขด้วยฟังก์ชัน gets() ซึ่งฟังก์ชัน gets() จะจัดเก็บข้อมูลในลักษณะข้อความ ดังนั้นหากต้องการแปลงข้อความที่เป็นตัวเลขเหล่านี้กลับมาเป็นตัวเลขที่สามารถนำไปคำนวณได้ ก็สามารถทำได้ด้วยฟังก์ชันแปลงสตริง โดยฟังก์ชันดังกล่าวจะถูกประกาศไว้ในเฮดเดอร์ไฟล์ stdlib.h

ฟังก์ชัน	ความหมาย
Atof	แปลงสตริงที่เก็บค่าตัวเลข มาเป็นตัวเลขชนิด double
Atoi	แปลงสตริงที่เก็บค่าตัวเลข มาเป็นตัวเลขชนิด int
Atol	แปลงสตริงที่เก็บค่าตัวเลข มาเป็นตัวเลขชนิด long int

■ สรุปท้ายบท

สำหรับข้อมูลแบบสตริง ประกอบด้วยตัวอักขระหลายๆ ตัวมาต่อเรียงกัน โดยอาจมองว่าสตริงคืออาร์เรย์ของตัวอักขระก็ได้ แต่จะต้องมีรหัสสิ้นสุดสตริงที่เรียกว่า null เป็นตัวปิดท้ายสตริง

ในการเขียนโปรแกรมด้วย Dev-C++ มีฟังก์ชั่นเกี่ยวกับสตริงมากมายที่เก็บอยู่ในไฟล์ string.h

strlen(s1)	ส่งคืนค่าความยาวสตริง s1
strcpy(s1,s2)	คัดลอกค่าสตริง s2 ไปเก็บไว้ที่สตริง s1
strcmp(s1,s2)	เปรียบเทียบค่าสตริง s1 กับ s2
strlwr(s1)	ใช้เปลี่ยนสตริง s1 ให้เป็นตัวพิมพ์เล็ก
strupr(s1)	ใช้เปลี่ยนสตริง s1 ให้เป็นตัวพิมพ์ใหญ่
strcat(s1,s2)	เชื่อมต่อ s1 ไปให้ s2
strrev(s1)	ใช้กลับตัวอักษรในสตริง s1 จากซ้ายไปขวา

ฟังก์ชันที่นำมาใช้เพื่อแปลงสตริงที่เป็นตัวเลข ให้กลับมาเป็นค่าตัวเลขที่สามารถนำไปคำนวณได้ จะถูกประกาศใช้งานในเฮดเดอร์ไฟล์ stdlib.h

ฟังก์ชันatof() ทำหน้าที่แปลงสตริงที่เก็บคั่วเลข มาเป็นตัวเลขชนิด double

ฟังก์ชันatoi() ทำหน้าที่แปลงสตริงที่เก็บค่าตัวเลข มาเป็นตัวเลขชนิด int

ฟังก์ชันatol() ทำหน้าที่แปลงสตริงที่เก็บค่าตัวเลข มาเป็นตัวเลขชนิด long int

■ การทดลอง

1. จงเขียนโปรแกรมเพื่อนับจำนวนตัวอักขระที่อยู่ในสตริง (ฟังก์ชัน strlen)

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

main () {

   char str1[] = "Programing with C ";
   int len;

   len = strlen(str1);
   printf("\nString = %s\nLenght = %d\n\n", str1, len);
}
```

2. จงเขียนโปรแกรมเพื่อคัดลอกตัวแปรสตริงหนึ่งไปยังตัวแปรของอีกสตริงหนึ่ง

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

main () {

   char source[] = "Programing with C ";
   char target[20];

   strcpy(target,source);
   printf("Source string = %s\n", source);
   printf("Target string = %s\n\n", target);
}
```

3. จงเขียนโปรแกรมเพื่อแปลงสตริงหรือข้อความให้เป็นตัวพิมพ์เล็ก

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

main () {

   char txt[] = "STRING ";

   printf("%s\n", txt);
   printf("%s\n\n", strlwr(txt));
}
```

4. จงเขียนโปรแกรมเพื่อแปลงสตริงหรือข้อความให้เป็นตัวพิมพ์ใหญ่

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

main () {

   char txt[] = "STRING ";

   printf("%s\n", txt);
   printf("%s\n", strupr(txt));
   printf("%s\n\n", strupr("FUNCTION strupr()"));
}
```

5. จงเขียนโปรแกรมเพื่อนำสตริง 2 สตริงมาเชื่อมเข้าด้วยกัน โดยนำค่าไปเก็บไว้ที่สตริงตัวแรก

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

main () {
   char str1[50] = "Computer Education ";
   char str2[50] = "Kalasin University ";
   strcat(str2, str1);
   printf("str1 string = %s\n", str1);
   printf("str2 string = %s\n\n", str2);
}
```

■ แบบฝึกหัดท้ายบทที่ 10

1. จงเขียนโปรแกรมหาว่าสตริงในอาร์เรย์ msg[100] มี “The” กี่ตัว
2. จงเขียนโปรแกรมให้รับค่าสตริง(ชื่อนักศึกษา) แล้วให้โปรแกรมแสดงอักขระกลับด้านจากข้างหลังมาข้างหน้า
3. จงเขียนโปรแกรมรับค่าสตริงเข้ามาหนึ่งบรรทัด(80 ตัว) แสดงผลด้วยว่าสตริงที่รับเข้ามานั้นเป็นอะไร แล้วจงหาว่าสตริงนั้นมีสระในภาษาอังกฤษ (E) อยู่ทั้งหมดกี่ตัว
4. จงเขียนโปรแกรมรับค่าสตริงเข้ามาหนึ่งบรรทัด(80 ตัว) แสดงผลด้วยว่าสตริงที่รับเข้ามานั้นเป็นอะไร แล้วจงหาว่าสตริงนั้นมีสระในภาษาอังกฤษ (a,e,i,o,u) ทั้งตัวเล็กและตัวใหญ่อยู่ทั้งหมดกี่ตัว

บทที่ 11
ข้อมูลชนิดโครงสร้าง

ปกติการประกาศตัวแปรชนิดอาร์เรย์ ข้อมูลที่บรรจุอยู่ภายในอาร์เรย์จำเป็นต้องมีชนิดข้อมูลเหมือนกันทั้งหมด ซึ่งจัดเป็นข้อจำกัดประการหนึ่งของตัวแปรชนิดอาร์เรย์ อย่างไรก็ตาม ในความเป็นจริงแล้ว ข้อลมูลที่จัดเก็บรวมกันในรูปแบบของเรคอร์ด ย่อมมีชนิดข้อมูลที่แตกต่างกันได้ ตัวอย่างเช่น ข้อมูลพนักงาน ซึ่งประกอบด้วย หมายเลขพนักงาน ชื่อ นามสกุล เพศ วันเกิด แผนก และรายได้ต่อเดือน ฯลฯ จะพบว่าภายในเรคอร์ดจะประกอบด้วยชนิดข้อมูลหลายประเภทรวมเข้าด้วยกัน ที่อยู่ภายใต้แบบโครงสร้างเดียวกันนี้ และเมื่อมีหลายๆเรคอร์ดรวมเข้าด้วยกัน ก็จะกลายเป็นแฟ้มข้อมูลนั่นเอง

ดังนั้นข้อมูลชนิดโครงสร้าง จึงหมายถึงแบบข้อมูลที่เป็นแหล่งรวมของกลุ่มข้อมูลที่มีความสัมพันธ์กันทั้งนี้ข้อมูลต่างๆที่บรรจุอยู่ภายใน สามารถมีชนิดข้อมูลเหมือนกันและแตกต่างกันก็ได้ และโดยทั่วไปข้อมูลแบบโครงสร้างนั้นจะเป็นตัวกำหนดถึงเรคอร์ดต่างๆ ที่เก็บไว้ในแฟ้มข้อมูลนั่นเอง

อย่างไรก็ตาม สำหรับข้อดีของตัวแปรอาร์เรย์ก็คือ ความสามารถในการเข้าถึงสมาชิกภายในเรคอร์ดได้โดยตรง ดังนั้นก็เข้าถึงข้อมูลชนิดโครงสร้าง จึงจำเป็นต้องใช้ฟิลด์ใดฟิลด์หนึ่งที่ถูกระบุไว้เป็นคีย์เพื่อนำมาใช้ในการชี้ระบุถึงตำแหน่งเรคอร์ดของข้อมูลชุดนั้น ซึ่งคีย์ดังกล่าวจะต้องมีความเป็นหนึ่งเดียวที่ไม่ซ้ำกับใคร และจากข้อมูลพนักงาน เมื่อนำมาพิจารณาแล้ว คีย์ที่เหมาะสมกับการนำมาใช้เพื่อชี้ระบุถึงพนักงานคนนนั้นๆก็ควรจะเป็นหมายเลขพนักงาน โดยหากได้ชี้ระบุหมายเลขพนักงานรายใดลงไป ข้อมูลภายในเรคอร์ดนั้น ก็คือข้อมูลส่วนตัวทั้งหมดของพนักงานรายนั้นๆ นั่นเอง

อาร์เรย์อนุญาตให้มีการกำหนดประเภทของตัวแปรที่สามารถเก็บรายการข้อมูลหลายชนิดเดียวกันโครงสร้างในทำนองเดียวกันนั้นใช้ชนิดข้อมูลอื่นที่กำหนดไว้ใช้ได้ใน C ที่ช่วยให้การรวมรายการข้อมูลที่แตกต่างกัน โครงสร้างที่ใช้แทนการบันทึก สมมติว่าคุณต้องการที่จะติดตามหนังสือในห้องสมุด คุณอาจต้องการที่จะติดตามแอตทริบิวต์ต่อไปนี้เกี่ยวกับหนังสือแต่ละเล่ม

- Title
- Author
- Subject
- Book ID

■ นิยามตัวแบบโครงสร้าง

เพื่อกำหนดโครงสร้างนักศึกษาจะต้องใช้คำสั่งที่ struct คำสั่ง struct กำหนดชนิดข้อมูลใหม่ที่มีสมาชิกมากกว่าหนึ่ง รูปแบบของคำสั่ง struct เป็นดังนี้

รูปแบบ

```
struct [structure tag] {

   member definition;
   member definition;
```

```
   ...
   member definition;
} [one or more structure variables];
```

โดยที่ struct เป็นคำสำคัญที่ระบุ ต่อไปนี้ไปนิยามตัวแบบโครงสร้าง

structure tag คือ ชื่อของโครงสร้าง

member definition คือ สมาชิกที่ประกาศใช้งานภายในโครงสร้าง โดยสมาชิกที่มีนี้สามารถเป็นตัวแปรพอยน์เตอร์ อาร์เรย์ หรือ โครงสร้างอื่นๆ ก็ได้ ซึ่งใช้เครื่องหมาย { } คลุมตัวแปรที่ประกาศไว้ทั้งหมด และลงท้ายด้วย;

■ การประกาศตัวแปรโครงสร้าง (Defining Structure Variables)

ภายหลังจากการนิยามตัวแบบโครงสร้างขึ้นมาแล้ว ก็เปรียบเสมือนกับการมีตัวแบบหรือเท็มเพลต (Template) ของตัวโครงสร้างแล้ว ดังนั้นเมื่อต้องการนำมาใช้งาน ก็ต้องประกาศตัวแปรให้กับตัวแบบโครงสร้างอีกทอดหนึ่ง และตัวแปรสร้างนี้เอง ที่จะนำไปใช้เพื่ออ้างอิงสมาชิกภายในโครงสร้าง ดังตัวอย่าง

ตัวอย่างที่ 10.1 นิยามตัวแบบโครงสร้างชื่อ Books และประกาศตัวแปรชื่อ book

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

■ การเข้าถึงสมาชิกของข้อมูลชนิดโครงสร้าง

สำหรับหัวข้อต่อไปนี้ จะกล่าวถึงวิธีการเข้าถึงสมาชิกของข้อมูลชนิดโครงสร้าง ซึ่งเป็นไปตามรูปแบบดังต่อไปนี้

รูปแบบ

```
Variable.member
```

ในการเข้าถึงสมาชิกของโครงสร้าง เราจะใช้โอเปอเรเตอร์จุด (.) เพื่อเข้าถึงสมาชิก จะเป็นโค้ดระหว่างชื่อตัวแปร(**variable name**) และสมาชิกโครงสร้าง (**structure member**)ที่เราต้องการที่จะเข้าถึง โดยต้องใช้คีย์เวิร์ดว่า struct เพื่อประกาศตัวแปรชนิดโครงสร้าง ตัวอย่างต่อไปนี้แสดงให้เห็นถึงวิธีการใช้โครงสร้างในโปรแกรม

ตัวอย่างที่ 10.2

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

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Lawan Dul");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, " Computer Network ");
   strcpy( Book2.author, "Surajak Piri");
   strcpy( Book2.subject, " Computer Network Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printf( "Book 1 title : %s\n", Book1.title);
   printf( "Book 1 author : %s\n", Book1.author);
   printf( "Book 1 subject : %s\n", Book1.subject);
   printf( "Book 1 book_id : %d\n", Book1.book_id);

   /* print Book2 info */
   printf( "Book 2 title : %s\n", Book2.title);
```

```
   printf( "Book 2 author : %s\n", Book2.author);
   printf( "Book 2 subject : %s\n", Book2.subject);
   printf( "Book 2 book_id : %d\n", Book2.book_id);

   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Book 1 title : C Programming
Book 1 author : Lawan Dul
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Computer Network
Book 2 author : Surajak Piri
Book 2 subject : Computer Network Tutorial
Book 2 book_id : 6495700
```

■ อาร์เรย์ของโครงสร้าง

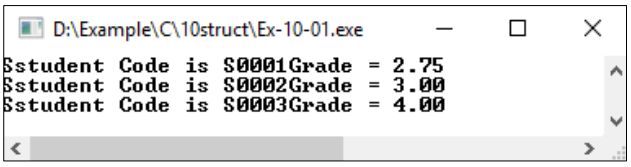
โครงสร้างของตัวแปรแบบ struct เป็นโครงสร้างที่เหมาะสำหรับการเก็บรายการข้อมูล ซึ่งในความเป็นจริงแล้วนั้นข้อมูลต่างๆ อาจมีมากกว่า 1 รายการ เช่น รายชื่อนักนักศึกษา รายชื่อสินค้า เป็นต้น

ดังนั้นเราสามารถประกาศตัวแปรแบบ struct ขึ้นมาใช้งานแบบอาร์เรย์ เพื่อให้เหมาะกับการเขียนโปรแกรม ซึ่งมีรูปแบบเหมือนการประกาศอาร์เรย์ทั่วไป ตัวอย่างเช่น

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

typedef struct
{
        char strCode[5]         ;
        float fGPA;
} student;

main()
{

          student std[3];
```

```
        strcpy(std[0].strCode, "S0001");
        std[0].fGPA = 2.75,

        strcpy(std[1].strCode, "S0002");
        std[1].fGPA = 3.00,

        strcpy(std[2].strCode, "S0003");
        std[2].fGPA = 4.00,


    printf("Sstudent Code is %s", std[0].strCode );
    printf("Grade = %.2f\n", std[0].fGPA );

    printf("Sstudent Code is %s", std[1].strCode );
    printf("Grade = %.2f\n", std[1].fGPA );

    printf("Sstudent Code is %s", std[2].strCode );
    printf("Grade = %.2f\n", std[2].fGPA );

    getch();
};
```

ผลลัพธ์โปรแกรม



■ พอยน์เตอร์ของโครงสร้าง

พอยเตอร์กับโครงสร้างข้อมูลชนิด struct จะมีลักษณะการใช้งานเหมือนกับการใช้งานพอยเตอร์กับตัวแปรทั่วไป

```
struct Books *struct_pointer;
```

โดยสามารถจัดเก็บที่อยู่ของตัวแปรโครงสร้าง ประกาศตัวแปรพอยเตอร์ และระบุที่อยู่ของตัวแปรโครงสร้าง โดยวางโปเปอเรเตอร์ '&; ก่อนชื่อของโครงสร้างดังต่อไปนี้

```
struct_pointer = &Book1;
```

เข้าถึงสมาชิกของโครงสร้างโดยใช้พอยน์เตอร์ ด้วยโอเปอเรเตอร์ -> ดังนี้

```
struct_pointer->title;
```

สามารถเขียนพอยน์เตอร์ชนิดโครงสร้างได้ดังตัวอย่าง

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

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

/* function declaration */
void printBook( struct Books *book );
int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Lawan Dul");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Computer Network");
   strcpy( Book2.author, "Surajak Piri");
   strcpy( Book2.subject, "Computer Network Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info by passing address of Book1 */
   printBook( &Book1 );

   /* print Book2 info by passing address of Book2 */
   printBook( &Book2 );
```

```
   return 0;
}

void printBook( struct Books *book ) {

   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
```

ผลลัพธ์โปรแกรม

```
Book title : C Programming
Book author : Lawan Dul
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Computer Network
Book author : Surajak Piri
Book subject : Computer Network Tutorial
Book book_id : 6495700
```

■ คำสำคัญ typedef

สำหรับหัวข้อต่อไปนี้จะขอกล่าวถึงคีย์เวิร์ดหรือคำสำคัญตัวหนึ่งที่น่าสนใจ คือ typedaf ที่สามารถนำมาประยุกต์ใช้กับข้อมูลชนิดทั่วไป รวมถึงข้อมูลชนิดโครงสร้าง โดย typrdef จะนำมาใช้เพื่อนิยามข้อมูลขึ้นมาใหม่ (define a synonym) และเรายังสามารถนำชื่อนิยามนี้ไปใช้งานแทนซึ่งมีความหมายเดียวกัน ทำให้การอ้างอิงใช้งานมีความสะดวกยิ่งขึ้น โดยเฉพาะการนำคีย์เวิร์ด typedef มานิยามกับชุดคำสั่งที่มีประโยคยาวๆ แทนการเขียนหลายๆ ครั้ง ซึ่งจะช่วยลดขั้นตอนลงได้มาก

สำหรับรูปแบบการประกาศใช้คีย์เวิร์ด typedef กับข้อมูลทั่วไป จะมีรูปแบบการเขียน ดังนี้

รูปแบบ

```
typedef type new_type;
```

โดยที่ type คือ ชนิดข้อมูล
new_type คือ ชนิดข้อมูลใหม่ที่นิยามขึ้นเอง

ตัวอย่าง การใช้คีย์เวิร์ด typedef เพื่อนิยามข้อมูล

```
typedef unsigned char BYTE;
```

หลังจากประกาศ type, ชนิดของ BYTE แล้วสามารถเรียกใช้ type เป็น unsigned char ดังตัวอย่าง

```

BYTE  b1, b2;
```

ตัวอย่าง การใช้คีย์เวิร์ด typedef เพื่อนิยามข้อมูล

```

typedef  int  age;
age  male, female;
```

หมายความว่าประกาศให้ age ซึ่งชนิดข้อมูลที่นิยามขึ้น มีชนิดข้อมูลเช่นเดียวกันกับ int ประกาศตัวแปร male และ female มีชนิดข้อมูลเป็น int ซึ่งมีความหมายเดียวกันกับ

```

int  male, female;
```

สำหรับรูปแบบการประกาศใช้คีย์เวิร์ด typedef กับชนิดข้อมูลโครงสร้าง จะมีรูปแบบการเขียนดังนี้

รูปแบบ

```
typedef struct {
        member_1;
        member_2;
        member_3;
              :
        member_n;
} new_type;
```

โดยที่ type struct เป็นคำสำคัญที่ระบุว่า ต่อไปนี้จะเป็นการนิยามโครงสร้างขึ้นมาใหม่
member คือ สมาชิกในโครงสร้าง
new_type คือ ชื่อโครงสร้างที่กำหนดขึ้นเอง (user-defined structure)

ตัวอย่าง การใช้คีย์เวิร์ด typedef กับตัวแปรชนิดโครงสร้าง

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

```

```c
typedef struct Books {
   char title[50];
   char author[50];
   char subject[100];
   int book_id;
} Book;

int main( ) {

   Book book;

   strcpy( book.title, "C Programming");
   strcpy( book.author, "Lawan Dul ");
   strcpy( book.subject, "C Programming Tutorial");
   book.book_id = 6495407;

   printf( "Book title : %s\n", book.title);
   printf( "Book author : %s\n", book.author);
   printf( "Book subject : %s\n", book.subject);
   printf( "Book book_id : %d\n", book.book_id);
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Book  title : C Programming
Book  author : Lawan Dul
Book  subject : C Programming Tutorial
Book  book_id : 6495407
```

■ การใช้ typedef กับ #define

#define คือ C ไดเร็กทีฟ ซึ่งใช้ในการกำหนดชื่อแทนสำหรับชนิดข้อมูลต่างๆคล้ายกับ typedef แต่มีความแตกต่างกันดังต่อไปนี้

- typedef จะถูกจำกัดให้ใช้ชื่อสัญลักษณ์ชนิดเดียวเท่านั้น คือ # define สามารถนำมาใช้ในการประกาศนามแฝงสำหรับค่าเช่นกัน สามารถกำหนด 1 เป็น ONE ฯลฯ
- •typedef จะดำเนินการโดยคอมไพเลอร์ในขณะที่มีการประมวลผล #define statements โดยก่อนการประมวลผล

ตัวอย่างต่อไปนี้แสดงให้เห็นถึงวิธีการใช้ #define ในโปรแกรม

```c
#include <stdio.h>

#define TRUE  1
#define FALSE 0

int main( ) {
   printf( "Value of TRUE : %d\n", TRUE);
   printf( "Value of FALSE : %d\n", FALSE);

   return 0;
```

ผลลัพธ์โปรแกรม

```
Value of TRUE : 1
Value of FALSE : 0
```

■ ยูเนียน (Union)

ยูเนียนจัดเป็นชนิดข้อมูลที่คล้ายกับข้อมูลโครงสร้าง การดำเนินการต่างๆ กับข้อมูลรวมถึงการนิยามใดๆก็จะใช้วิธีเดียวกันกับข้อมูลชนิดโครงสร้าง เพียงแต่ใช้คีย์เวิร์ด union แทน struct เท่านั้น อย่างไรก็ตาม จะมีสิ่งสำคัญสิ่งหนึ่งที่ union แตกต่างไปจาก struct ก็คือ สมาชิกของยูเนียนใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูลร่วมกัน กล่าวคือ จะแชร์หน่วยความจำเพื่อใช้งานร่วมกันนั่นเอง ซึ่งแตกต่างจากข้อมูลชนิดโครงสร้างที่แต่ละสมาชิกภายในโครงสร้างจะมีเนื้อที่หน่วยความจำของตนเอในการจัดเก็บข้อมูล และด้วยวิธีการแชร์หน่วยความจำในยูเนียนนี้เอง จึงทำให้ยูเนียนสามารถนำไปใช้จัดการกับโครงสร้างที่มีสมาชิกจำนวนมากได้ ถึงแม้ว่าเครื่องจะมีหน่วยความจำขนาดเล็กก็ตาม

สำหรับขนาดหน่วยความจำในยูเนียนนั้น จะพิจารณาจากชนิดข้อมูลที่มีขนาดใหญ่ที่สุดภายในโครงสร้างนั้นเป็นหลัก เช่น ภายในโครงสร้างมีชนิดข้อมูลที่ประกอบด้วย int และ char ดังนั้นขนากหน่วยความจำของยูเนียนที่จะนำมาใช้งานก็คือ ขนาดเท่ากับขนาดของข้อมูลชนิดนี้ int นั่นเอง ถึงแม้ว่ายูเนียนจะมีวิธีการจัดสรรหน่วยความจำที่ประหยัดและมีประสิทธิภาพก็ตาม แต่ด้วยการแชร์หน่วยความจำร่วมกันในการจัดเก็บข้อมูลขอสมาชิกภายในยูเนียนนี้เอง จึงทำให้การจัดเก็บข้อมูลภายในหน่วยความจำ จะกระทำได้ทีละตัวในช่วงเวลาหนึ่งเท่านั้น กล่าวคือในช่วงเวลาหนึ่งจะมีเพียงสมาชิกเพียงตัวเดียวเท่านั้น ที่สามารถจัดเก็บข้อมูลที่อยู่ภายในหน่วยความจำได้ และข้อมูลเดิมที่เก็บในหน่วยความจำก็จะถูกทับด้วยข้อมูลล่าสุด จึงทำให้ไม่สามารถอ้างอิงข้อมูลเดิมได้ ดังนั้นการดำเนินงานกับข้อมูลในยูเนียนจึงจำเป็นต้องนำข้อมูลที่บันทึก ณ ขณะนั้นไปใช้งานทันที ก่อนที่จะถูกทับด้วยข้อมูลชุดใหม่ และปกติยูเนียนมักมีจำนวนสมาชิกมากกว่าหนึ่งตัว รวมถึงมีชนิดข้อมูลหลายประเภทที่อยู่ภายในโครงสร้าง ดังนั้นการใช้โครงสร้างแบบยูเนียน จึงดำเนินการด้วยความระมัดระวังเป็นพิเศษ

1. การประกาศข้อมูลแบบยูเนียน

การประกาศยูเนียน ต้องใช้ union statement ในลักษณะเดียวกับที่คุณทำในขณะที่การประกาศโครงสร้าง union statement ประกาศชนิดข้อมูลใหม่ที่มีสมาชิกมากกว่าหนึ่งตัวในโปรแกรม รูปแบบของคำสั่งยูเนี่ยนมีดังนี้

รูปแบบ

```
union [union tag] {
   member definition;
   member definition;
   ...
   member definition;
} [one or more union variables];
```

ตัวอย่าง

```
union Data {
   int i;
   float f;
   char str[20];
} data;
```

ตัวอย่าง โปรแกรม

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

union Data {
   int i;
   float f;
   char str[20];
};

int main( ) {
   union Data data;
   printf( "Memory size occupied by data : %d\n", sizeof(data));
   return 0;
}
```

ผลลัพธ์โปรแกรม

```
Memory size occupied by data : 20
```

■ สรุปท้ายบท

ข้อมูลชนิดโครงสร้าง หมายถึง แบบข้อมูล ที่เป็นแหล่งรวมของกลุ่มข้อมูลที่มีความสัมพันธ์กัน ทั้งนี้ข้อมูลต่างๆ ที่บรรจุอยู่ภายใน สามารถมีชนิดข้อมูลเหมือนกันหรือต่างกันก็ได้

ข้อมูลแบบโครงสร้างนั้นจะเป็นตัวกำหนดถึงเรคอร์ดต่างๆ ที่เก็บในแฟ้มข้อมูล

ข้อดีของตัวแปรอาร์เรย์ คือ การเข้าถึงสมาชิกภายในอาร์เรย์ได้โดยตรง ดังนั้นการเข้าถึงข้อมูลชนิดโครงสร้าง จึงจำเป็นต้องใช้ฟิลด์ใดฟิลด์หนึ่งที่ถูกระบุไว้คีย์ เพื่อนำมาใช้ในการระบุถึงตำแหน่งเรคอร์ดของข้อมูลชุดนั้น ซึ่งคีย์ดังกล่าวจะต้องมีความเป็นหนึ่งเดียวที่ไม่ซ้ำกับใคร

ภายหลังจากประกาศตัวแบบโครงสร้างขึ้นมาแล้ว เมื่อต้องการนำมาใช้งาน ก็จะต้องประกาศตัวแบบโครงสร้างให้กับตัวแปรอีกทอดหนึ่ง และตัวแปรโครงสร้างนี้เอง ที่จะนำไปใช้เพื่ออ้างอิงสมาชิกภายในโครงสร้าง

การอ้างอิงสมาชิกภายในโครสร้าง จะใช้ชื่อตัวแปรโครงสร้าง แล้วคั่นด้วยจุด จากนั้นก็ตามด้วยชื่อสมาชิกที่ต้องการอ้างถึง

คำสำคัญ typedef นำมาใช้เพื่อนิยามข้อมูลขึ้นมาใหม่ ทั้งนี้เรายังสามารถนำชื่อที่นิยามนี้ไปใช้งานแทนซึ่งจะแทนความหมายเดียวกัน ทำให้การอ้างอิงเพื่อใช้งานมีความสะดวกยิ่งขึ้น โดยเฉพาะการนำมานิยามกับชุดคำสั่งที่มีประโยคนิยามๆ แทนการเขียนหลายๆครั้ง

ในการส่งผ่านโครงสร้างไปยังฟังก์ชัน สามารถดำเนินการได้โดยวิธีการส่งผ่านสมาชิกในโครงสร้างหรือผ่านทั้งตัวโครงสร้างก็ได้ ซึ่งขึ้นอยู่กับรูปแบบของการถ่ายโอนข้อมูลเป็นสำคัญ ว่าการส่งผ่านแบบสมาชิกบางตัวหรือทั้งตัวโครงสร้าง

ยูเนียน จัดเป็นชนิดข้อมูลที่คล้ายกับข้อมูลโครงสร้าง การดำเนินการต่างๆ กับข้อมูลรวมถึงการนิยามใดๆ ก็จะใช้วิธีเดียวกันกับข้อมูลชนิดโครงสร้าง เพียงแต่ใช้คีย์เวิร์ด union แทน struct เท่านั้น

ความแตกต่างระหว่าง union และ struct ก็คือ สมาชิกของยูเนียนจะใช้พื้นที่หน่วยความในการจัดเก็บข้อมูลร่วมกัน ในขณะที่ข้อมูลชนิดโครงสร้าง สมาชิกแต่ละตัวภายในโครงสร้างจะมีเนื้อที่หน่วยความจำเป็นของตนเองในการจัดเก็บข้อมูล

ด้วยวิธีการแชร์หน่วยความจำในยูเนียน จึงทำให้ยูเนียนสามารถนำไปจัดการกับโครงสร้างที่มีสมาชิกจำนวนมากได้ ถึงแม้ว่าเครื่องจะมีหน่วยความจำขนาดเล็กก็ตาม

ขนาดหน่วยความจำในยูเนียน จะพิจารณาจากชนิดข้อมูลชองสมาชิก ทีมีขนาดใหญ่ที่สุด

ถึงแม้ว่า ยูเนียนจะมีวิธีการจัดสรรหน่วยความจำที่ประหยัดและมีประสิทธิภาพก็ตาม แต่ด้วยการแชร์การหน่วยความจำร่วมกันในการจัดเก็บข้อมูลของสมาชิกภายในยูเนียนนี้เอง จึงทำให้การจัดเก็บค่าข้อมูลภายในหน่วยความจำ จะกระทำได้ทีละตัวในช่วงเวลาหนึ่งเท่านั้น

การดำเนินงานกับข้อมูลในยูเนียน จำเป็นต้องนำข้อมูลที่บันทึก ณ ขณะนั้นไปใช้งานทันทีก่อนที่จะถูกทับด้วยข้อมูลชุดใหม่

■ การทดลอง

1. จงเขียนโปรแกรม และอธิบายการทำงาน

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

typedef struct
{
        int intCode;
    char strName[30];
        char strSurname[30];
    int intYear;
    char strMajor[30];
}Student;

main()
{
    Student pdStudent;

    printf("Enter Code ID : ");
    scanf("%d", &pdStudent.intCode);
    printf("Enter Name : ");
    scanf("%s", &pdStudent.strName);
    printf("Enter Surname : ");
    scanf("%s", &pdStudent.strSurname);
    printf("Enter Year : ");
    scanf("%d", &pdStudent.intYear);
    printf("Enter Major : ");
    scanf("%s", &pdStudent.strMajor);

    printf("\n\nStudent => %d  ", pdStudent.intCode);
    printf("%s %s\n", pdStudent.strName, pdStudent.strSurname);
    printf("\tYear %d", pdStudent.intYear);
    printf("\tMajor %s\n", pdStudent.strMajor);
    getch();
}
```

2. จงเขียนโปรแกรม และอธิบายการทำงาน

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

```
typedef struct
{
    int intCode;
    char strName[30];
    char strSurname[30];
    char strPosition[30];
    char strDivision[30];
}Person;

main()
{
    Person pdPerson;

    printf("Enter Code ID : ");
    scanf("%d", &pdPerson.intCode);
    printf("Enter Name : ");
    scanf("%s", &pdPerson.strName);
    printf("Enter Surname : ");
    scanf("%s", &pdPerson.strSurname);
    printf("Enter Position : ");
    scanf("%s", &pdPerson.strPosition);
    printf("Enter Division : ");
    scanf("%s", &pdPerson.strDivision);

    printf("\n\nStudent => %d  ", pdPerson.intCode);
    printf("%s %s\n", pdPerson.strName, pdPerson.strSurname);
    printf("\tPosition %s", pdPerson.strPosition);
    printf("\tDivision %s\n", pdPerson.strDivision);
    getch();
}
```

3. จงเขียนโปรแกรม และอธิบายการทำงาน

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

char CalGrade (int intScore);
```

```
typedef struct
{
        char strCode[5];
    int intScoreHomework;
        int intScoreDivide;
        int intScoreMidterm;
        int intScoreFinal;
        int intSumScore;
        char chGrade;
}Student;

main()
{
        int intCount;
        Student stStudent[10];
        for(intCount = 0;intCount < 10;intCount++)
        {
                printf("\nEnter Data Student %d", intCount + 1);
                printf("\n   Enter Code ID : ");
                scanf("%s", &stStudent[intCount].strCode);
                printf("\n   Enter Score Homework : ");
                scanf("%d", &stStudent[intCount].intScoreHomework);
                printf("\n   Enter Score Divide : ");
                scanf("%d", &stStudent[intCount].intScoreDivide);
                printf("\n   Enter Score Midterm : ");
                scanf("%d", &stStudent[intCount].intScoreMidterm);
                printf("\n   Enter Score Final : ");
                scanf("%d", &stStudent[intCount].intScoreFinal);
                stStudent[intCount].intSumScore =
stStudent[intCount].intScoreHomework +

        stStudent[intCount].intScoreDivide +

        stStudent[intCount].intScoreMidterm +

        stStudent[intCount].intScoreFinal;
```

```
                stStudent[intCount].chGrade =
CalGrade(stStudent[intCount].intSumScore);
        }

        //แสดงผล
        printf("\n\n\t\tID\tHomework Divide\tMidterm\tFinal\tSum\tGrade");
        for(intCount = 0;intCount < 10;intCount++)
        {
                printf("\nData Student %d", intCount + 1);
                printf("\t%s", stStudent[intCount].strCode);
                printf("\t  %d", stStudent[intCount].intScoreHomework);
                printf("\t  %d", stStudent[intCount].intScoreDivide);
                printf("\t  %d", stStudent[intCount].intScoreMidterm);
                printf("\t  %d", stStudent[intCount].intScoreFinal);
                printf("\t %d", stStudent[intCount].intSumScore);
                printf("\t  %c", stStudent[intCount].chGrade);
        }
        getch();
}

char CalGrade (int intScore)
{
        char chGrade;
        if (intScore > 79)
        {
                chGrade = 'A';
        }
        else if (intScore > 69)
        {
                chGrade = 'B';
        }
        else if (intScore > 59)
        {
                chGrade = 'C';
        }
        else if (intScore > 49)
        {
```

```
                chGrade = 'D';
        }
        else
        {
                chGrade = 'E';
        }
        return chGrade;
}
```

4. จงเขียนโปรแกรม และอธิบายการทำงาน

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

typedef struct
{
        char strCode[5];
        float fMath;
        float fCom;
        float fSci;
        float fThai;
        float fGPA;
}Student;

main()
{
        Student GStudent;
        printf("Enter Code ID Student : ");
        scanf("%s", &GStudent.strCode);
        printf("Enter Grade of Math : ");
        scanf("%f", &GStudent.fMath);
        printf("\nEnter Grade of Computer : ");
        scanf("%f", &GStudent.fCom);
        printf("\nEnter Grade of Science : ");
        scanf("%f", &GStudent.fSci);
        printf("\nEnter Grade of Thai : ");
        scanf("%f", &GStudent.fThai);
```

```
        GStudent.fGPA = (GStudent.fMath + GStudent.fCom + GStudent.fSci +
GStudent.fThai)/4;
    printf("=====================================================");
        printf("\n\n\nCode ID : %s", GStudent.strCode);
        printf("\n\nMath : %.0f", GStudent.fMath);
        printf("\tMath : %.0f", GStudent.fCom);
        printf("\tComputer : %.0f", GStudent.fSci);
        printf("\tThai : %.0f", GStudent.fThai);
        printf("\n\nGPA = %.2f", GStudent.fGPA);
        getch();
}
```

■ แบบฝึกหัดท้ายบทที่ 11

1. จงเขียนโปรแกรมรับข้อมูลรายการสั่งซื้อ ซึ่งประกอบด้วยรหัสใบสั่งซื้อ ราคาสินค้า จำนวนสินค้า และราคารวมของแต่ละรายการ จำนวน 5 รายการ พร้อมแสดงผลทางจอภาพ โดยรับข้อมูลผ่านตัวแปรอาร์เรย์ของ struct
2. จงเขียนโปรแกรมรับข้อมูลนักศึกษา โดยที่ข้อมูลประกอบด้วย รหัสนักศึกษา ชื่อ นามสกุล ชั้นปี วิชาเอกตามจำนวนที่ต้องการ โดยรับข้อมูลผ่านตัวแปร struct พร้อมแสดงผลทางจอภาพ
3. จงเขียนโปรแกรมรับข้อมูลพนักงาน โดยที่ข้อมูลประกอบด้วยรหัสพนักงาน ชื่อ นามสกุล ตำแหน่งงาน และสังกัดแผนกตามจำนวนที่ต้องการ โดยรับข้อมูลผ่านตัวแปร struct พร้อมแสดงผลทางจอภาพ
4. จงเขียนโปรแกรมรับข้อมูลบุคคล โดยที่ข้อมูลประกอบด้วยเลขที่บัตรประชาชน ชื่อ นามสกุล วันเกิด และวันที่ออกบัตร โดยรับข้อมูลผ่านตัวแปร struct พร้อมแสดงผลทางจอภาพ

ภาคผนวก

- มคอ. 3 วิชาขั้นตอนวิธีและการเขียนโปรแกรม

รายละเอียดของรายวิชา

ชื่อสถาบันอุดมศึกษา	มหาวิทยาลัยกาฬสินธุ์
วิทยาเขต/คณะ/ภาควิชา	คณะศึกษาศาสตร์และนวัตกรรมการศึกษา สาขาวิชาคอมพิวเตอร์

หมวดที่ 1 ข้อมูลโดยทั่วไป

1. รหัสและชื่อรายวิชา

ED-032-109 ขั้นตอนวิธีและการเขียนโปรแกรม (Algorithms and Programming)

2. จำนวนหน่วยกิต

3 หน่วยกิต (2-2-5)

3. หลักสูตรและประเภทของรายวิชา

หลักสูตรครุศาสตรบัณฑิต สาขาวิชาคอมพิวเตอร์ (หลักสูตรปรับปรุง พ.ศ.2562)

ประเภทวิชาเอกบังคับ

4. อาจารย์ผู้รับผิดชอบรายวิชาและอาจารย์ผู้สอน

อาจารย์ผู้รับผิดชอบรายวิชา : ดร. ลาวัณย์ ดุลยชาติ

อาจารย์ผู้สอน 1) อาจารย์นคินทร พัฒนชัย **สัดส่วน** ร้อยละ 50

2) ดร. ลาวัณย์ ดุลยชาติ **สัดส่วน** ร้อยละ 50

5. ภาคการศึกษา/ชั้นปีที่เรียน

ภาคการศึกษาที่ 2 ปีการศึกษา 2562

หลักสูตรครุศาสตรบัณฑิต สาขาวิชาคอมพิวเตอร์ ชั้นปีที่เรียน ปี 1 หมู่เรียนที่ 1

6. รายวิชาที่ต้องเรียนมาก่อน (Pre-requisites) (ถ้ามี)

ไม่มี

7. รายวิชาที่ต้องเรียนพร้อมกัน (Co-requisites) (ถ้ามี)

ไม่มี

8. สถานที่เรียน

อาคารศูนย์ภาษาและคอมพิวเตอร์

9. วันที่จัดทำหรือปรับปรุงรายละเอียดของรายวิชาครั้งล่าสุด

วันที่ 22 เดือน ตุลาคม พ.ศ.2562

หมวดที่ 2 จุดมุ่งหมายและวัตถุประสงค์

1. จุดมุ่งหมายของรายวิชา

1.1 เพื่อให้ผู้เรียนมีสามารถคิดเชิงคำนวณการแก้ปัญหาโดยใช้แนวคิดเชิงคำนวณ ออกแบบขั้นตอนวิธีวิเคราะห์ปัญหาโดยวิธีการโปรแกรมได้

1.2 เพื่อให้ผู้เรียนมีทักษะการเขียนโปรแกรมเพื่อพัฒนาโปรแกรมโปรแกรมคอมพิวเตอร์ขนาดเล็กได้

2. วัตถุประสงค์ในการพัฒนา/ปรับปรุงรายวิชา

พัฒนาองค์ความรู้ทางด้านโปรแกรมคอมพิวเตอร์ที่ทันสมัยให้เหมาะสมกับปัจจุบันและการนำไปใช้งาน

หมวดที่ 3 ลักษณะและการดำเนินการ

1. คำอธิบายรายวิชา

หลักการคิดเชิงคำนวณการแก้ปัญหาโดยใช้แนวคิดเชิงคำนวณ ฟังชั่น ขั้นตอนวิธี การเขียนผังงาน การแก้ปัญหา การคิดเชิงตรรกะ ชนิดข้อมูล ตัวแปร โครงสร้างควบคุม การนำเข้าข้อมูล การแสดงผล หลักการเขียนโปรแกรม วิเคราะห์ปัญหาโดยวิธีการโปรแกรม ฝึกปฏิบัติการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ สร้างโปรแกรมแก้ปัญหาขนาดเล็กได้

Calculation concepts, problem solving using computational functions. Introduction to programming, problem solving, algorithms, language structures, data types, operator, variables, input / output data, flow control, loops, string, array, pointer, computer

2. จำนวนชั่วโมงที่ใช้ต่อภาคการศึกษา

บรรยาย	สอนเสริม	การฝึกปฏิบัติ/งานภาคสนาม/การฝึกงาน	การศึกษาด้วยตนเอง
16 สัปดาห์ x 2 ชม. = 32 ชั่วโมง/ภาคการศึกษา	ตามความต้องการของนักศึกษาเฉพาะราย	16 สัปดาห์ x 2 ชม. = 32 ชั่วโมง/ภาคการศึกษา	16 ชม x 5 ชม. = 80 ชั่วโมง/ภาคการศึกษา

3. จำนวนชั่วโมงต่อสัปดาห์ที่อาจารย์ให้คำปรึกษาและแนะนำทางวิชาการแก่นักศึกษาเป็นรายบุคคล

3.1 อาจารย์จัดเวลาให้คำปรึกษาเป็นรายบุคคล หรือ รายกลุ่มตามความต้องการ ประมาณ 1 ชั่วโมง ต่อ สัปดาห์ (เฉพาะรายที่ต้องการ)

3.2 ให้คำปรึกษาเป็นรายบุคคลผ่านทางเครือข่ายทางสังคมออนไลน์: https://www.facebook.com

หมวดที่ 4 การพัฒนาการเรียนรู้ของนักศึกษา

มาตรฐาน / ระดับผลการเรียนรู้		1. คุณธรรม จริยธรรม				2. ความรู้					3. ทักษะทางปัญญา			4. ทักษะความสัมพันธ์ระหว่างบุคคล ฯลฯ				5. ทักษะการวิเคราะห์เชิงตัวเลข การสื่อสาร และการใช้เทคโนโลยี			6. วิธีวิทยาการจัดการเรียนรู้				
ชื่อวิชา		1	2	3	4	1	2	3	4	5	1	2	3	1	2	3	4	1	2	3	1	2	3	4	5
วิชาเอกบังคับ																									
ED-032-109	ขั้นตอนวิธีและการเขียนโปรแกรม		●	○			●				●	○		○		●				●	●				

1. การเรียนรู้ด้านคุณธรรม จริยธรรม

1.1. คุณธรรม จริยธรรมที่ต้องพัฒนา

*2) มีจิตอาสา จิตสาธารณะ อดทนอดกลั้น มีความเสียสละ รับผิดชอบและซื่อสัตย์ต่องานที่ได้รับมอบหมายทั้งด้านวิชาการและวิชาชีพ และสามารถพัฒนาตนเองอย่างต่อเนื่อง ประพฤติตนเป็นแบบอย่างที่ดีแก่ศิษย์ ครอบครัว สังคมและประเทศชาติ และเสริมสร้างการพัฒนาที่ยั่งยืน

3) มีค่านิยมและคุณลักษณะเป็นประชาธิปไตย คือ การเคารพสิทธิ และให้เกียรติคนอื่นมีความสามัคคีและทำงาน

1.2. วิธีการสอน

1) สอนด้วยวิธีการบรรยาย แทรกคุณธรรมและจริยธรรม โดยมีการยกตัวอย่างจากประสบการณ์ของผู้สอนและข่าวจากสื่อต่างๆ

2) อภิปรายร่วมกันเพื่อแลกเปลี่ยนความคิดเห็นก่อให้เกิดจิตสำนึกที่ดี

3) มีการจัดให้ทำงานกลุ่มเพื่อให้นักศึกษามีทักษะในการทำงานร่วมกันและมีไมตรีจิตที่ดีต่อกัน

4) ให้มีความรับผิดชอบ ตรงต่อเวลา มีความซื่อสัตย์ต่อตนเองผู้อื่น

1.3. วิธีการประเมินผล

1) ประเมินจากการร่วมกันอภิปรายร่วมกันในชั้นเรียน

2) ประเมินจากการเข้าเรียนและพฤติกรรมในชั้นเรียน

3) ประเมินจากความตรงต่อเวลาในการส่งงานตรงต่อเวลาและประสิทธิผลของงานที่ได้รับมอบหมาย

4) ประเมินจากการมีส่วนร่วมในงานกลุ่มและมนุษย์สัมพันธ์

2. การเรียนรู้ด้านความรู้

2.1.ความรู้ที่ต้องได้รับ

*2) มีความรอบรู้ในหลักการ แนวคิด ทฤษฎี เนื้อหาวิชาที่สอน สามารถวิเคราะห์ความรู้และเนื้อหาวิชาที่สอนอย่างลึกซึ้ง สามารถติดตามความก้าวหน้าด้านวิทยาการและนำไปประยุกต์ใช้ในการพัฒนา

ผู้เรียน โดยมีผลลัพธ์การเรียนรู้และเนื้อหาสาระด้านมาตรฐานผลการเรียนรู้ด้านความรู้ของสาขา โดยสามารถบูรณาความรู้ที่ครอบคลุมสาระต่อไปนี้

(1) วิทยาศาสตร์และคณิตศาสตร์พื้นฐาน

(2) ความรู้เฉพาะสาขาเทคโนโลยี

(2.1) พื้นฐานเทคโนโลยีสารสนเทศและเทคโนโลยีดิจิทัล

(2.2) หลักการคิดเชิงคำนวณการแก้ปัญหาโดยใช้แนวคิดเชิงคำนวณ ฟังชั่น

(2.3) การออกแบบ เทคโนโลยีการวางแผน และการดำเนินการ และการใช้เทคโนโลยีสารสนเทศอย่างปลอดภัย

(2.4) การใช้อินเทอร์เน็ต ซอฟแวร์และการพัฒนาแอบพลิเคชั่น การเขียนโปรแกรม

(2.5) เครื่องมือทางฮาร์ดแวร์ การใช้งานโปรแกรมไมโครคอนโทรเลอร์ และสมองกล

(2.6) การสืบค้น การรวบรวม การวิเคราะห์ การประมวลผล การเลือกใช้แหล่งข้อมูล ประเมินและนำเสนอข้อมูลสารสนเทศ

(2.7) ออกแบบ พัฒนาระบบสารสนเทศ การออกแบบและพัฒนานวัตกรรม การพัฒนาโครงงาน

(2.8) การนำแนวคิดเชิงคำนวณไปพัฒนาโครงงานที่เกี่ยวกับชีวิตประจำวัน ธุรกิจและบริการ

(2.9) การเพิ่มมูลค่าให้บริการหรือผลิตภัณฑ์

(2.10) วิทยาการสมัยใหม่เกี่ยวกับเทคโนโลยี การรู้เท่าทัน

(2.11) กฎหมายคอมพิวเตอร์และการใช้ลิขสิทธิ์ของผู้อื่นโดยชอบธรรม

2.2.วิธีการสอน

1) การเรียนการสอนด้วยวิธีการบรรยายและสาธิตการเขียนโปรแกรม

2) จัดให้มีการเรียนรู้จากภาคปฏิบัติในการใช้โปรแกรมสำหรับเขียนโปรแกรม

3) นักศึกษาค้นคว้าเพิ่มเติมละจัดให้มีการทำรายงานพร้อมทั้งนำเสนอหน้าชั้นเรียน

2.3.วิธีการประเมินผล

1) การสอบกลางภาคเรียนและปลายภาคเรียน

2) ประเมินจากรายงานการศึกษาค้นคว้าของนักศึกษาและการนำเสนอการรายงานหน้าชั้นเรียน

3) ประเมินจากพฤติกรรมการทำงานต่างๆ ทั้งงานเดี่ยวและงานกลุ่ม

4) การทดลองเขียนโปรแกรมคอมพิวเตอร์

5) แบบฝึกหัดท้ายบท

3. การเรียนรู้ด้านทักษะทางปัญญา

3.1.ทักษะทางปัญญาที่ต้องพัฒนา

*1) คิด ค้นหา วิเคราะห์ข้อเท็จจริง และประเมินข้อมูล สื่อ สารสนเทศจากแหล่งข้อมูลที่หลากหลายอย่างรู้เท่าทัน เป็นพลเมืองตื่นรู้ มีสำนึกสากล สามารถเผชิญและก้าวทันกับการเปลี่ยนแปลงในโลกยุคดิจิทัล เทคโนโลยีข้ามแพลทฟอร์ม (Platform) และโลกอนาคต นำไปประยุกต์ใช้ในการปฏิบัติงานและวินิจฉัยแก้ปัญหาและพัฒนางานได้อย่างสร้างสรรค์ โดยคำนึงถึงความรู้ หลักการทางทฤษฎี ประสบการณ์ภาคปฏิบัติ ค่านิยม แนวคิด นโยบายและยุทธศาสตร์ชาติ บรรทัดฐานทางสังคมและผลกระทบที่อาจเกิดขึ้น

2) สามารถคิดริเริ่มและพัฒนางานอย่างสร้างสรรค์

3.2.วิธีการสอน

1) จัดกระบวนการเรียนการสอนที่ฝึกทักษะการคิดร่วมกันในชั้นเรียน

2) จัดกิจกรรมให้นักศึกษามีโอกาสใช้ความรู้ที่ได้ไปปฏิบัติงานจริง

3.3.วิธีการประเมินผล

1) ประเมินจากการอภิปรายร่วมกันในชั้นเรียน

2) การนำเสนอแนวความคิดและนำเสนอผลงาน

3) การใช้ข้อสอบหรือแบบฝึกหัดที่ให้นักศึกษาคิดแก้ปัญหา

4. การเรียนรู้ด้านทักษะความสัมพันธ์ระหว่างตัวบุคคลและความรับผิดชอบ

4.1.ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบที่ต้องพัฒนา

1) เข้าใจและใส่ใจอารมณ์ความรู้สึกของผู้อื่น มีความคิดเชิงบวก มีวุฒิภาวะทางอารมณ์และทางสังคม

*3) มีความรับผิดชอบต่อหน้าที่ ต่อตนเอง ต่อผู้เรียน ต่อผู้ร่วมงาน และต่อส่วนรวมสามารถช่วยเหลือและแก้ปัญหาตนเอง กลุ่มและระหว่างกลุ่มได้อย่างสร้างสรรค์

4.2.วิธีการสอน

1) มอบหมายงานรายกลุ่ม และรายบุคคล เกี่ยวกับการเขียนโปรแกรมคอมพิวเตอร์และการนำไปใช้งาน

2) สอดแทรกเรื่องความรับผิดชอบ การมีมนุษยสัมพันธ์ จากแนวความคิดจากประสบการณ์ของผู้สอนหรือเรื่องราวจากแหล่งข้อมูลอื่นๆ เพื่อให้เกิดจิตสำนึก

4.3.วิธีการประเมินผล

1) ประเมินจากการอภิปรายร่วมกันในชั้นเรียน

2) ประเมินจากความสม่ำเสมอการเข้าเรียนและพฤติกรรมในชั้นเรียนของนักศึกษา

3) ประเมินความรับผิดชอบในหน้าที่ที่ได้รับมอบหมาย เช่น การส่งงานเขียนโปรแกรมในเวลาที่กำหนด

4) ประเมินโดยเพื่อนร่วมชั้น

5. การเรียนรู้ด้านทักษะการวิเคราะห์เชิงตัวเลข การสื่อสาร และการใช้เทคโนโลยีสารสนเทศ

5.1. ทักษะการวิเคราะห์เชิงตัวเลข การสื่อสาร และการใช้เทคโนโลยีสารสนเทศที่ต้องพัฒนา

*3) ใช้เทคโนโลยีสารสนเทศในการสืบค้นข้อมูลหรือความรู้จากแหล่งการเรียนรู้ต่างๆได้อย่างมีประสิทธิภาพ สามารถใช้โปรแกรมสาเร็จรูปที่จำเป็นสำหรับการเรียนรู้ การจัดการเรียนรู้ การทำงานการประชุม การจัดการและสืบค้นข้อมูลและสารสนเทศ รับและส่งข้อมูลและสารสนเทศโดยใช้ดุลยพินิจที่ดีในการตรวจสอบความน่าเชื่อถือของข้อมูลและสารสนเทศ อีกทั้งตระหนักถึงการละเมิดลิขสิทธิ์และการลอกเลียนผลงาน

5.2.วิธีการสอน

1) จัดให้มีการนำเสนองานกลุ่มหรือรายบุคคล พร้อมทั้งเสนอแนะการใช้ภาษาที่ถูกต้องและชัดเจน

2) ในการค้นคว้าผู้เรียนต้องรู้จักเลือกใช้เทคโนโลยีสารสนเทศและการสื่อสารหลากหลายและเหมาะสม

5.3.วิธีการประเมินผล

1) ประเมินจากทักษะการพูดในการนำเสนอผลงาน

2) ประเมินจากทักษะการนำเสนอโดยใช้เทคโนโลยีสารสนเทศ

3) ประเมินจากการนำเสนอโดยใช้ข้อมูลทางสถิติคณิตศาสตร์เข้ามานำเสนออย่างเป็นระบบ

6. การเรียนรู้ด้านวิธีวิทยาการจัดการเรียนรู้

6.1. ทักษะวิธีวิทยาการจัดการเรียนรู้

*1) สามารถเลือกใช้ปรัชญาตามความเชื่อในการสร้างหลักสูตรรายวิชา การออกแบบเนื้อหาสาระ กิจกรรมการเรียนการสอน สื่อและเทคโนโลยีการสื่อสาร การวัดและประเมินผู้เรียน การบริหารจัดการชั้นเรียน การจัดการเรียนโดยใช้แหล่งการเรียนรู้ในโรงเรียนและนอกโรงเรียน แหล่งการเรียนรู้แบบเปิดได้อย่างเหมาะสมกับสภาพบริบทที่ต่างกันของผู้เรียนและพื้นที่

6.2.วิธีการสอน

ใช้การสอนในหลากหลายรูปแบบ โดยเน้นทักษะการใช้คอมพิวเตอร์ให้ทันต่อการเปลี่ยนแปลงทางเทคโนโลยี ทั้งนี้ให้เป็นไปตามลักษณะของรายวิชาตลอดจนเนื้อหาสาระของรายวิชานั้น

1) บรรยายและสอดแทรกในเนื้อหาวิชาที่เรียน
2) จัดให้มีการนำเสนองานกลุ่มหรือรายบุคคล พร้อมทั้งเสนอแนะการใช้ภาษาที่ถูกต้องและชัดเจน
3) ในการค้นคว้าผู้เรียนต้องรู้จักเลือกใช้เทคโนโลยีสารสนเทศและการสื่อสารหลากหลายและเหมาะสม

6.3.วิธีการประเมินผล

1) ประเมินพฤติกรรมการเรียนรู้ในชั้นเรียน ผ่านกระบวนการการทำกิจกรรมกลุ่ม
2) ประเมินจากทักษะการพูดในการนำเสนอผลงาน และ การนำเสนอโดยใช้เทคโนโลยีสารสนเทศ
3) ประเมินจากผลงานวิชาการที่เกิดจากการศึกษาค้นคว้าด้วยตนเอง

หมวดที่ 5 แผนการสอนและการประเมินผล

5.1 แผนการสอน

สัปดาห์ที่	รายละเอียด	จน.ชม.	กิจกรรมการเรียนการสอน	สื่อการสอน	ผลการเรียนรู้						การประเมินและวัดผล	ผู้สอน
					1	2	3	4	5	6		
1	บทนำสู่ขั้นตอนวิธีและการเขียนโปรแกรม (Algorithms and Programming)	4	- อธิบายเนื้อหารายวิชาที่นักศึกษาต้องเรียนตลอดทั้งเทอม - ชี้แจงแนวการสอน และConcept mapping - ชี้แจงวิธีการให้คะแนนในรายวิชา -อธิบายความหมายของขั้นตอนวิธีและการเขียนโปรแกรม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การมีส่วนร่วมในชั้นเรียน 4. การสังเกตในชั้นเรียน	อาจารย์นคินทร พัฒนชัย
2	บทที่ 1 ขั้นตอนวิธี (Algorithms) และ การเขียนผังงาน (Flowchart)	4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน	อาจารย์นคินทร พัฒนชัย
3		4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด	อาจารย์นคินทร พัฒนชัย

สัปดาห์ที่	รายละเอียด	จน. ชม.	กิจกรรมการเรียนการสอน	สื่อการสอน	ผลการเรียนรู้						การประเมินและวัดผล	ผู้สอน
					1	2	3	4	5	6		
			-การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	- google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม							4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน	
4	ชนิดข้อมูล (Data types) บทที่ 2 แถวคอย (Queue) บทที่ 3 สแตกส์ (Stack)	4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	2 3	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน	อาจารย์นคินทร พัฒนชัย
5	บทที่ 4 แถวลำดับ (Array) บทที่ 5 รายการโยง (Linked List)	4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	2 3	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน	อาจารย์นคินทร พัฒนชัย
6	บทที่ 6 การเรียกซ้ำ (Recursion)	4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	2 3	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน	อาจารย์นคินทร พัฒนชัย
7	บทที่ 7 ต้นไม้และ ขั้นตอนวิธีของต้นไม้ (Tree and Tree algorithms)	4	-บรรยาย -การสาธิตโปรแกรม -ปฏิบัติการ	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท	2 3	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน	อาจารย์นคินทร พัฒนชัย

สัปดาห์ที่	รายละเอียด	จน.ชม.	กิจกรรมการเรียนการสอน	สื่อการสอน	ผลการเรียนรู้						การประเมินและวัดผล	ผู้สอน
					1	2	3	4	5	6		
			-การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	- google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม							5. การสังเกตในชั้นเรียน	
8-9	บทที่ 8 กราฟ (Graph) - การนำเสนองานหน้าชั้นเรียน และสอบกลางภาค	4	-บรรยาย -การนำเสนอและเสวนาแลกเปลี่ยนเรียนรู้ -สไลด์ประกอบการนำเสนอ -สอบกลางภาค	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4.การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้นเรียน 6. การทดสอบกลางภาคและการกระทำทุจริตในการสอบ	อาจารย์นคินทร พัฒนชัย
10	- ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรม - ขั้นตอนการทำงานของอัลกอริทึม	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ
11-12	- โครงสร้างภาษา - ประเภทของข้อมูล และตัวดำเนินการ	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชา ขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ

สัปดาห์ที่	รายละเอียด	จน.ชม.	กิจกรรมการเรียนการสอน	สื่อการสอน	ผลการเรียนรู้						การประเมินและวัดผล	ผู้สอน
					1	2	3	4	5	6		
13	- การรับและแสดงผลข้อมูล - คำสั่งควบคุม	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ
14	- คำสั่งทำซ้ำ(Loops)	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น 6.การทดสอบกลางภาค	ดร.ลาวัณย์ ดุลยชาติ
15	- อาร์เรย์ (Array)	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ
16	- พอยเตอร์ (Pointers) - การจัดการข้อมูลชนิดสตริง	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ

สัปดาห์ที่	รายละเอียด	จน.ชม.	กิจกรรมการเรียนการสอน	สื่อการสอน	ผลการเรียนรู้						การประเมินและวัดผล	ผู้สอน
					1	2	3	4	5	6		
17	- ข้อมูลชนิดโครงสร้าง - การนำเสนองานหน้าชั้นเรียน	4	-บรรยาย -การสาธิตโปรแกรม -การมอบหมายให้ศึกษาค้นคว้าเพิ่มเติม	-สไลด์ประกอบการบรรยาย -เอกสารประกอบการสอน -แบบฝึกหัดท้ายบท - google classroom รายวิชาขั้นตอนวิธีและการเขียนโปรแกรม	23	2	1 2	1 3	3	1	1. การตรงเวลาของนักศึกษาในการเข้าชั้นเรียน 2. การตอบข้อซักถาม 3. การทำแบบฝึกหัด 4. การมีส่วนร่วมในชั้นเรียน 5. การสังเกตในชั้น	ดร.ลาวัณย์ ดุลยชาติ
18	สอบปลายภาค	4	-	-		2	1 2				**ภาคทฤษฎี** -ข้อสอบอัตนัย วัดความรู้ความเข้าใจเกี่ยวกับการเขียนโปรแกรมคอมพิวเตอร์ **ภาคปฏิบัติ** -การเขียนโปรแกรมคอมพิวเตอร์	ดร.ลาวัณย์ ดุลยชาติ

2. การประเมินผลการเรียนรู้

2.1 การวัดผล

วิธีการประเมินผล ผู้ผ่านรายวิชานี้ต้องมีเวลาการเข้าชั้นเรียนไม่น้อยกว่าร้อยละ 80 ของเวลาเรียน เกณฑ์การตัดสินผลการเรียนคือผู้ผ่านรายวิชาต้องได้คะแนนร้อยละ 50 ขึ้นไป ให้ระดับผลการเรียนแบบอิงเกณฑ์

2.1.1 คะแนนระหว่างเรียน ร้อยละ 70
- คะแนน สอบกลางภาค ร้อยละ 30
- คะแนนการเข้าเรียน การมีส่วนร่วมอภิปรายเสนอความคิดเห็นในชั้นเรียน
  และการสอบย่อย ร้อยละ 20
- ใบงาน ร้อยละ 20

2.1.2 คะแนน สอบปลายภาค ร้อยละ 30

รวมคะแนนทั้งหมด ร้อยละ 100

2.2 แผนการประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีสอนที่ระบุในรายละเอียดวิชา	วิธีประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
คุณธรรม จริยธรรม	1. ฝึกความวินัย ตรงต่อเวลาและรับผิดชอบ ต่องานที่ได้รับมอบหมาย 2. มอบหมายงานกลุ่ม เพื่อฝึกภาวะความเป็นผู้นำและผู้ตาม สามารถทำงานเป็นทีมร่วมกับผู้อื่น 3. อภิปรายกลุ่ม สะท้อนความคิดเห็น เพื่อฝึกให้เคารพสิทธิและรับฟังความคิดเห็นของผู้อื่น	1) วัดและประเมินจากการตรงเวลาของนักศึกษาในการเข้าชั้นเรียน พฤติกรรมการเรียน การส่งงานตามกำหนดระยะเวลาที่มอบหมาย 2) วัดและประเมินจากการเข้าร่วมกิจกรรมเสริมหลักสูตร	1-18	10
ความรู้	1. ใช้สื่อการสอน PowerPoint, เอกสารประกอบการสอน 2. การสาธิตโปรแกรมตัวอย่าง 3. ทำงานกลุ่ม งานรายบุคคล 4. ทำ Lab การทดลองท้ายชั่วโมง และส่งทาง Google Classroom ตามเวลาที่กำหนด 5.ต้องส่งการบ้านแบบฝึกหัดทุกสัปดาห์	1) ทดสอบกลางภาคเรียน 2) ทดสอบปลายภาคเรียน 3) การทดลองเขียนโปรแกรม 4) แบบฝึกหัด/การบ้าน	1-18	25 25 10 10

ผลการเรียนรู้	วิธีสอนที่ระบุในรายละเอียดวิชา	วิธีประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
ทักษะทางปัญญา	1. อภิปรายกลุ่ม 2. วิเคราะห์โปรแกรม 3. สะท้อนความคิดเห็น	1) วัดและประเมินจากการอภิปราย ทักษะการสร้างผลงานโปรแกรมคอมพิวเตอร์	1-18	10
ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	1. มอบหมายรายงานกลุ่มกรณีศึกษา ร่วมกันเก็บค้นคว้าข้อมูล 2. ให้งานค้นคว้าความรู้การเขียนโปรแกรมภาษา นำเสนอวิจารณ์ประเด็นสำคัญ	1) วัดและประเมินจากผลงานนำเสนอผลงานกลุ่ม และการเป็นผู้นำในการอภิปรายซักถาม	8,16	10
ทักษะการวิเคราะห์เชิงตัวเลข การสื่อสาร และการใช้เทคโนโลยีสารสนเทศ	1. มอบหมายงานให้ศึกษาและค้นคว้าจากเว็บไซต์ และนำเสนอโดยใช้เทคโนโลยีที่เหมาะสม	1) วัดและประเมินผลจากการประยุกต์ใช้โปรแกรมคอมพิวเตอร์	8,16	5
ทักษะการจัดการเรียนรู้	1.ทดสอบปฏิบัติ 2.การสังเกตในชั้นเรียน	1) วัดและประเมินผลจากการทดสอบเชิงปฏิบัติ 2) วัดและประเมินผลจากการสังเกตในชั้นเรียน	8,16	15

2.3 การประเมินผล

การประเมินผลใช้แบบอิงเกณฑ์ พิจารณาจากคะแนนรวมโดยใช้เกณฑ์ตัดสิน ดังนี้

ลำดับที่	คะแนน	ผลการเรียน	ค่าผลคะแนน	ระดับคะแนน(เกรดที่ได้)
1	80 –100	A	ดีเยี่ยม	4.0
2	75 – 79	B+	ดีมาก	3.5
3	70 – 74	B	ดี	3.0
4	65-69	C+	ดีพอใช้	2.5
5	60-64	C	พอใช้	2.0
6	55-59	D+	อ่อน	1.5
7	50-54	D	อ่อนมาก	1.0
8	0-49	F	ตก	0

หมวดที่ 6 ทรัพยากรประกอบการเรียนการสอน

1. เอกสารและตำราหลัก

1.1 นคินทร พัฒนชัย.(2562). เอกสารประกอบการสอน เรื่อง ขั้นตอนวิธีและโครงสร้างข้อมูล.

1.2 ลาวัณย์ ดุลยชาติ(2562) .เอกสารประกอบการสอนการเขียนโปรแกรมคอมพิวเตอร์.

2. เอกสารและข้อมูลสำคัญ

ไม่มี

3. เอกสารและข้อมูลแนะนำ

3.1 สมชาย ประสิทธิ์ตระกูล.(2550). โครงสร้างข้อมูลฉบับวาจาจาวา. กรุงเทพมหานคร: สำนักพิมพ์แห่ง จุฬาลงกรณ์มหาวิทยาลัย.

3.2 สมพิศ โกศัลวัฒน์.(2532). โครงสร้างข้อมูลและอัลกอริทึม. CS 243 (341) กรุงเทพฯ: คณะวิทยาศาสตร์
มหาวิทยาลัยรามคำแหง.

3.3 D.S. Malik. (2013). C++ Programming: Program Design Including Data Structures 6th ED. South-Western College.

3.4 D. S. Malik. Data Structures Using C++ 2nd ED. United State of America.

3.5 https://classroom.google.com/c/NDcxODk1OTc2Mjha
(ED-032-109 ขั้นตอนวิธีและการเขียนโปรแกรม Algorithms and Programming)

3.6 อรพิน ประวัติบริสุทธิ์ (2547). คู่มือเรียนภาษาซี. บริษัท โปรวิชั่น จำกัด กรุงเทพฯ พิมพ์ครั้งที่ 1

3.7 Josuttis, Nicolai M. The C++ standard library : a tutorial and reference. – 2nd ed.

3.8 Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. **C++ primer**. – 5th ed.

3.9 https://classroom.google.com/c/NDcxODU3NDE2MjJa (Part 2 การเขียนโปรแกรม)

หมวดที่ 7 การประเมินและปรับปรุงการดำเนินการของรายวิชา

1. กลยุทธ์การประเมินประสิทธิผลของรายวิชาโดยนักศึกษา

ให้นักศึกษาประเมินประสิทธิผลในรายวิชานี้ ได้แก่ วิธีการสอน การจัดกิจกรรมในระหว่างเรียน ทั้งภาคทฤษฎีและภาคปฏิบัติ อุปกรณ์การเรียนการสอนต่างๆ ที่มีผลต่อประสิทธิผลในการเรียนการสอนและผลการเรียนรู้ของนักศึกษา โดยมีการเก็บข้อมูลดังนี้

- แบบประเมินผู้สอน และแบบประเมินรายวิชา
- ข้อแนะนำผ่านระบบ ESS ของมหาวิทยาลัยฯ

2. กลยุทธ์การประเมินการสอน

ใช้กลยุทธ์ในการเก็บข้อมูลเพื่อประเมินการสอนดังนี้

- ประเมินจากผลการประเมินผู้สอนและผลการเรียนของนักศึกษา
- การทวนสอบผลประเมินผลการเรียนรู้

3. การปรับปรุงการสอน

หลังจากได้รับผลการประเมินการสอนในข้อ 2 จะมีการปรับปรุงการสอน และสรรหาข้อมูลเพิ่มเติมในการปรับปรุงการสอน
4. การทวนสอบมาตรฐานผลสัมฤทธิ์ของนักศึกษาในรายวิชา ในระหว่างกระบวนการสอนรายวิชา มีการทวนสอบผลสัมฤทธิ์ในรายหัวข้อ ตามที่คาดหวังจากการเรียนรู้ในรายวิชา ได้จากการสอบถามนักศึกษา หรือการสุ่มตรวจผลงานของนักศึกษา รวมถึงพิจารณาจากผลการทดสอบย่อย และหลังการออกผลการเรียนรายวิชา มีการทวนสอบผลสัมฤทธิ์โดยรวมในวิชาได้ดังนี้ - การทวนสอบการให้คะแนนจากการสุ่มตรวจผลงานของนักศึกษาโดยอาจารย์อื่น หรือผู้ทรงคุณวุฒิ ที่ไม่ใช่อาจารย์ประจำหลักสูตร - มีการตั้งคณะกรรมการในสาขาวิชา ตรวจสอบผลการประเมินการเรียนรู้ของนักศึกษา โดยตรวจสอบข้อสอบ รายงาน วิธีการให้คะแนนสอบ และการให้คะแนนพฤติกรรม
5. การดำเนินการทบทวนและการวางแผนปรับปรุงประสิทธิผลของรายวิชา - ไม่มี

ชื่ออาจารย์ผู้รับผิดชอบรายวิชา

ลงชื่อ _____ วันที่รายงาน 1 ตุลาคม 2563

(อาจารย์ ดร. ลาวัณย์ ดุลยชาติ)

ชื่อผู้ตรวจสอบ (ประธานหลักสูตร)

ลงชื่อ _____ วันที่รับรายงาน _____

(อาจารย์นคินทร พัฒนชัย)

บรรณานุกรม

กิตินันท์ พลสวัสดิ์.(2554). รวมโจทย์และแบบฝึกหัดภาษา C + JAVA. นนทบุรี : ไอซีซีฯ

ธีรวัฒน์ ประกอบผล.(2554) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์ กรุงเทพฯ : ไรวีว่า.

สมพันธุ์ ชาญศิลป์. (2544). เอกสารคำสอนวิชา Computer Programming(ภาษาซี),สาขาวิชาวิศวกรรมคอมพิวเตอร์ : มหาวิทยาลัยเทคโนโลยีสุรนารี.

โอภาส เอี่ยมสิริวงศ์. (2552). การเขียนโปรแกรมด้วยภาษา C (Programing with C). กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

Alex Allain. C Programming and C++ Programming. Retrieved January 1, 2015, form http://www.cprogramming.com/tutorial/c-tutorial.html.

Brian W. Kernighan and Dennis M. Ritchie.(1988).The C Programming Language. 2nd PRENTICE HALL, Englewood Cliffs, Newjersy.

Kjell Blackman. (2012). Structured programming with C++. Retrieved December 1, 2015. http://bookboon.com/en/c-cpp-csharp-ebooks.

Programmingsimplified. C programming. Retrieved February 19, 2015, form http://www.programmingsimplified.com/c.

Mohtashim M. Learn C Programing. Retrieved January 1, 2015, form http://www.tutorialspoint.com/cprogramming/c_program_structure.htm.

WIKIBOOKS. C Programming. Retrieved February 19, 2015, form https://en.wikibooks.org/wiki/C_Programming.

ประวัติผู้เขียน

ชื่อ- สกุล **ดร. ลาวัณย์ ดุลยชาติ**

ประวัติการศึกษา

ปริญญาเอก	ปรัชญาดุษฎีบัณฑิต (ปร.ด.) สาขาวิชาคอมพิวเตอร์ศึกษา มหาวิทยาลัยราชภัฏมหาสารคาม
ปริญญาโท	ศิลปศาสตรมหาบัณฑิต (ศศ.ม.) สาขาวิชาสารสนเทศศาสตร์ มหาวิทยาลัยสุโขทัยธรรมาธิราช
ปริญญาตรี	วิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ

ประวัติการทำงาน

ปัจจุบัน	อาจารย์ประจำสาขาวิชาคอมพิวเตอร์ คณะศึกษาศาสตร์และนวัตกรรมการศึกษา มหาวิทยาลัยกาฬสินธุ์

งานวิจัย

พ.ศ. 2563	การส่งเสริมทักษะศตวรรษที่ 21 (4Cs) ด้านการสร้างสรรค์และนวัตกรรม ด้วยการจัดการเรียนรู้แบบสร้างสรรค์เป็นฐาน ในรายวิชาโครงการพิเศษด้านคอมพิวเตอร์ศึกษา สาขาวิชาคอมพิวเตอร์ มหาวิทยาลัยกาฬสินธุ์
พ.ศ. 2558	แอพพลิเคชั่นเพื่อการเรียนรู้เครื่องดนตรีโปงลางบนอุปกรณ์แท็บเล็ต
พ.ศ. 2554	วิจัยในชั้นเรียน : การจัดการเรียนการสอน เรื่อง การวิเคราะห์และออกแบบอัลกอริทึม ด้วยเทคนิคกระบวนการกลุ่ม
พ.ศ. 2552	ระบบสารสนเทศแหล่งท่องเที่ยวทางวัฒนธรรม กรณีศึกษา : วัดพุทธนิมิต(ภูค่าว)
พ.ศ. 2551	การพัฒนาศักยภาพบุคลากรชุมชนเพื่อสร้างภูมิคุ้มกันด้านเศรษฐกิจชุมชนอย่างยั่งยืนของจังหวัดกาฬสินธุ์: กรณีศึกษากลุ่มสตรีทอผ้าบ้านกุดหว้า อำเภอกุฉินารายณ์ จังหวัดกาฬสินธุ์
พ.ศ. 2549	การจัดการที่พักทางวัฒนธรรม กรณีศึกษา : ชุมชนผู้ไทยบ้านโพน
พ.ศ. 2547	การพัฒนาระบบสารสนเทศผ้าไหมแพรวาบ้านโพน