



KMITL 4G

9:00 A.M.

100%

## 01006012 การเขียนโปรแกรมคอมพิวเตอร์ (Computer Programming)

<http://www.ce.kmitl.ac.th>

**Version 0.41**

Status: unstable, developing

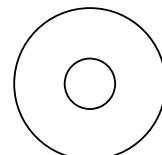
□ □ □

C  
main()

Lab

uBook

www.  
ce.kmitl.  
ac.th



## คำนำ รุ่น 0.41

หนังสือฉบับนี้เป็นรุ่น 0.41 สิ่งที่แก้ไขเพิ่มเติมจากรุ่นก่อนหน้าคือ แก้ไขปรับปรุงคำผิด การจัดเรียงเอกสาร และ จำนวนในการอธิบาย โดยยังคงสาระเดิมจากรุ่นก่อนหน้า

เอกสารนี้ทำขึ้นเพื่อใช้ประกอบการทดลองเท่านั้นไม่สามารถใช้แทนหนังสือได้ นักศึกษาควรมีหนังสือสำหรับศึกษา เพิ่มเติมหรือใช้ทบทวน

ผู้ประสานงานวิชา Computer Programming ปีการศึกษา 2556

## คำนำ รุ่น 0.4

หนังสือฉบับนี้เป็นรุ่น 0.4 สิ่งที่แก้ไขเพิ่มเติมจากรุ่นก่อนหน้าคือ ได้ปรับปรุงการทดลองในส่วนของครึ่งแรก โดยหวังว่าจะช่วยให้นักศึกษาทำความเข้าใจกับบทเรียนได้ดีขึ้น

เอกสารนี้ทำขึ้นเพื่อใช้ประกอบการทดลองเท่านั้นไม่สามารถใช้แทนหนังสือได้ นักศึกษาควรมีหนังสือสำหรับศึกษาเพิ่มเติมหรือใช้ทบทวน

ผู้ประสานงานวิชา Computer Programming ปีการศึกษา 2556

## คำนำ รุ่น 0.3

หนังสือฉบับนี้เป็นรุ่น 0.3 ลิํงที่แก้ไขเพิ่มเติมจากรุ่นก่อนหน้าคือ ได้ปรับปรุงการทดลองในส่วนของครึ่งแรก โดยหวังว่าจะช่วยให้นักศึกษาทำความเข้าใจกับบทเรียนได้ดีขึ้น

เอกสารนี้ทำขึ้นเพื่อใช้ประกอบการทดลองเท่านั้นไม่สามารถใช้แทนหนังสือได้ นักศึกษาควรรีฟังหนังสือสำหรับศึกษาเพิ่มเติมหรือใช้ทบทวน

ผู้ประสานงานวิชา Computer Programming ปีการศึกษา 2555

## คำนำ รุ่น 0.1

หนังสือฉบับนี้เป็นรุ่น 0.1 ซึ่งเนื้อหาส่วนใหญ่ในหนังสือ มาจากสไลด์ ซึ่งคณาจารย์ที่สอนวิชา Computer and Programming และ Principle of Programming ได้แก่

- อ. ประสาร ตั้งติศาనันท์
- อ. คณะรุํ ตั้งติศาnanท์
- อ. วิบูลย์ พร้อมพานิชย์
- อ. ปันธิต พัฒนา
- อ. วัฒพงศ์ เกษมศิริ
- อ. จิระศักดิ์ สิทธิกร
- อ. ธนาภูชัย ตรีภาค
- อ. เกียรตินรังค์ ทองประเสริฐ
- ดร. ปกรณ์ วัฒนาจตุรพร
- รศ. ดร. เกียรติกุล เจียรนัยธนະกิจ
- ดร. สุรินทร์ กิตติธรกุล
- ดร. สุภกิจ นุตยกะสกุล

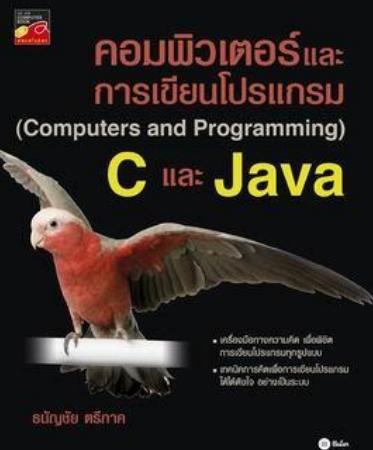
ท่านเหล่านี้ได้เขียนหรือแก้ไขปรับปรุงเนื้อหาเอาไว้ ไม่ได้รวมเนื้อหาเหล่านี้เป็นหนังสือ โดยจุดประสงค์เพื่อเก็บเนื้อหาอย่างเป็นระบบ และนศ.สามารถค้นหาสิ่งที่สงสัยได้อย่างรวดเร็ว เนื้อหาในช่วยแรกฯย่อมมีจุดผิดพลาด Bug ต่างๆ ก็ได้เช่น ถ้าทั้งความสมบูรณ์ของเนื้อหาในทางวิชาการอาจจะไม่ลงลึกในบางเรื่อง แนะนำว่าต้องปรับปรุงต่อไป เมื่อถึงรุ่น 1.0 แล้วเนื้อหาสมบูรณ์ ณ เวลานั้น คงพร้อมเป็นตำราที่เหมาะสมต่อไป  
ผู้ปรับปรุงข้อเสนอแนะในการปรับปรุงเนื้อหา

ผู้ประสานงานวิชา Computer and Programming ปีการศึกษา 2554

## แนะนำหนังสือภาษาซี (โดย ดร. ธนัญชัย ตรีภาค)

ในการเรียนวิชา คอมพิวเตอร์และการเขียนโปรแกรม (Computers and Programming) นอกจากจะใช้เอกสารที่อาจารย์ใช้ในการเรียนการสอนในห้อง และเอกสารการทดลองแล้ว นักศึกษาควรมีหนังสืออื่นๆ เพื่อเป็นเอกสารอ้างอิงคำสั่งต่างๆ ศึกษาโจทย์ปัญหา และแนวคิดต่างๆ โดยนักศึกษาสามารถใช้หนังสือในรายการต่างๆ ต่อไปนี้ หรือหนังสืออื่นๆ ในห้องตลาดได้

	<p><b>ชื่อหนังสือ :</b> รวมโจทย์ข้อสอบภาษา C  <b>ผู้เขียน :</b> ประภาพร ช่างไม้  <b>สำนักพิมพ์ :</b> Dev Book</p> <p>หนังสือเล่มนี้นำเสนอบอกโจทย์ปัญหาต่างๆ ของภาษา C ในรูปแบบคำตาม และคำตอบ ซึ่งนักศึกษาสามารถนำมาใช้ทบทวน และประเมินตัวเองในการสอบกลางภาคและการสอบปลายภาคได้</p>
	<p><b>ชื่อหนังสือ :</b> กรณีศึกษาการเขียนโปรแกรมด้วยภาษา C  <b>ผู้เขียน :</b> ธนัญชัย ตรีภาค  <b>สำนักพิมพ์ :</b> SE-ED</p> <p>หนังสือเล่มนี้นำเสนองานกรณีศึกษาในการเขียนโปรแกรมภาษา C เพื่อให้ผู้เริ่มต้นสามารถเขียนโปรแกรมเป็นเร็วขึ้น โดยนำเสนองานกรณีศึกษาจากโจทย์ปัญหาที่ง่ายไปยังโจทย์ปัญหาที่ยาก จนถึงการนำเสนอการนำเทคนิค หลักการทางคณิตศาสตร์ ตรรกศาสตร์ หรือทฤษฎีการคำนวณ อื่นๆ มาประยุกต์ใช้ในโปรแกรม เพื่อให้โปรแกรมสามารถทำงานได้อย่าง</p>

	ถูกต้อง รวดเร็ว และเขียนโปรแกรมไม่ยุ่งยากซับซ้อน
	<b>ชื่อหนังสือ :</b> คอมพิวเตอร์และการเขียนโปรแกรม C และ Java <b>ผู้เขียน :</b> ชนัญชัย ตรีภาค <b>สำนักพิมพ์ :</b> SE-ED <p>เป็นหนังสือที่มุ่งเน้นกระบวนการคิดในการเขียนโปรแกรม มีการกำหนดขั้นตอนการคิดโดยละเอียด จากโจทย์ปัญหา โค้ดเทียม ไปจนถึงลำดับการเขียนโค้ดโปรแกรม หมายเหตุที่สำคัญ เช่น การเขียนโปรแกรมภาษา C และ Java ให้เข้าใจง่าย รวมถึงการแก้ไขข้อบกพร่องในโค้ด ตลอดจนการใช้งานของเครื่องคอมพิวเตอร์ ที่จำเป็นต่อการเขียนโปรแกรม</p>
	<b>ชื่อหนังสือ :</b> คู่มือเรียนภาษา C <b>ผู้เขียน :</b> อรพิน ประวัติบริสุทธิ์ <b>สำนักพิมพ์ :</b> Provision <p>เป็นหนังสือที่นำเสนอบุคลิกภาพของภาษา C อย่างลึกซึ้งและน่าสนใจ ผ่านการอธิบายความหมายของคำศัพท์ ไวยากรณ์ และโครงสร้างข้อมูล พร้อมตัวอย่างเชิงลึก ที่ช่วยให้ผู้อ่านเข้าใจการทำงานของภาษา C ได้มากยิ่งขึ้น ไม่ว่าจะเป็นการจัดการหน้าจอ หรือการเชื่อมต่อเครือข่าย ก็ได้รับการอธิบายอย่างละเอียด</p>
	<b>ชื่อหนังสือ :</b> คู่มือการเขียนโปรแกรมภาษา C <b>ผู้เขียน :</b> รศ. ริริวัฒน์ ประกอบผล <b>สำนักพิมพ์ :</b> Clean Code <p>หนังสืออธิบายหลักการและแนวคิดของภาษา C การเขียนโปรแกรมแบบโครงสร้าง ผังงาน และไซโตโค้ด การบันทึกลงไฟล์ และอ่านไฟล์ การเขียนโปรแกรมกราฟิก และภาพเคลื่อนไหว โดยมีตัวอย่างโจทย์ โค้ดโปรแกรม และคำอธิบายโค้ดชัดเจน</p>

## สารบัญ

คำนำ รุ่น 0.41 .....	2
คำนำ รุ่น 0.4 .....	3
คำนำ รุ่น 0.3 .....	4
คำนำ รุ่น 0.1 .....	5
สารบัญ .....	8
สารบัญโปรแกรม .....	15
บทที่ 0 แนะนำวิชา .....	18
วัตถุประสงค์ของวิชา .....	18
การวัดผล .....	18
ระเบียบการใช้ห้องปฏิบัติการ .....	18
บทที่ 1 พื้นฐานเกี่ยวกับคอมพิวเตอร์ .....	20
1. ความเป็นมาของคอมพิวเตอร์ .....	20
1.1 คอมพิวเตอร์คืออะไร .....	20
1.2 ขั้นตอนการทำงานของคอมพิวเตอร์ .....	20
1.3 องค์ประกอบหลักของคอมพิวเตอร์ .....	20
1.4 ภาษาคอมพิวเตอร์ .....	22
1.5 ระบบจำนวนและตัวเลข .....	23
1.6 ขนาดตัวเลขในระบบคอมพิวเตอร์ .....	24
1.7 ตัวอักษรที่ใช้ในคอมพิวเตอร์ .....	26
1.8 รู้จักภาษาซี .....	27
การทดลองที่ 1 การพัฒนาโปรแกรมภาษาซีเบื้องต้น .....	29
ทฤษฎี .....	29
ขั้นตอนการพัฒนาโปรแกรม .....	29
การทดลองที่ 1.1 .....	30
การทดลองที่ 1.2 .....	35
การทดลองที่ 1.3 .....	36
การทดลองที่ 1.4 .....	37
บทที่ 2 การเขียนโปรแกรม คำสั่งแสดงผล .....	38
2.1 โครงสร้างภาษาซี .....	38
2.2 ฟีลิปเปรเซอร์ไดเรกทีฟ (Preprocessor directive) .....	38
ฟังก์ชันหลัก .....	39
2.3 การแสดงผลด้วยคำสั่ง printf .....	39
2.4 แสดงชนิดข้อมูลที่กำหนดรูปแบบการแสดงผล .....	42
2.5 คำอธิบายโปรแกรม .....	44
2.5 คำถามท้ายบท .....	45
การทดลองที่ 2 การเขียนโปรแกรมเบื้องต้นและคำสั่งแสดงผล .....	46

## สารบัญ (ต่อ)

ทฤษฎี.....	46
การทดลองที่ 2.1 การใช้งาน <code>\t</code> และ จำนวนตัวอักษรในหนึ่งบรรทัด.....	50
การทดลองที่ 2.2 การใช้งาน <code>%d %d %f</code> เพื่อแสดงผลข้อมูลปกติ .....	51
การทดลองที่ 2.3 การแสดงผลจำนวนเต็ม ด้วย <code>%d %x %c %f</code> .....	52
การทดลองที่ 2.4 การแสดงผลจำนวนทศนิยมแบบต่าง ๆ .....	53
การทดลองที่ 2.5 การแสดงผลข้อความแบบต่าง ๆ .....	54
การทดลองที่ 2.6.....	54
การทดลองที่ 2.7.....	55
การทดลองที่ 2.8.....	55
การทดลองที่ 2.9.....	56
บทที่ 3 ชนิดของข้อมูลในภาษาซี คำสั่งรับข้อมูล และการคำนวณ.....	57
3.0 ตัวแปรในภาษาซี .....	57
3.1 ชนิดของข้อมูลในภาษาซี .....	57
3.1.1 ข้อมูลชนิดว่างเปล่า (void).....	57
3.1.2 ข้อมูลชนิดจำนวนเต็ม (integer) .....	57
3.1.3 ข้อมูลชนิดจำนวนทศนิยม (float) .....	57
3.1.4 ข้อมูลชนิดอักขระ หรือตัวอักษร (character).....	58
3.1.5 ตารางสรุปชนิดข้อมูล .....	58
3.2 รูปแบบการประกาศตัวแปร.....	58
3.2.1 ประกาศตัวแปรแบบไม่กำหนดค่าเริ่มต้น .....	58
3.2.2 ประกาศตัวแปรแบบกำหนดค่าเริ่มต้น .....	59
3.2.3 หลักการตั้งชื่อตัวแปร .....	59
3.3 ตัวแปรชนิดข้อความ .....	60
3.3.1 รูปแบบการประกาศตัวแปรชนิดข้อความ .....	60
3.3.2 รูปแบบการประกาศตัวแปรชนิดข้อความพร้อมกำหนดค่า .....	61
3.3.3 รูปแบบการอ้างอิงในตัวแปรข้อความ .....	61
3.4 การใช้งานตัวแปรร่วมกับคำสั่ง <code>printf</code> .....	62
3.5 การรับข้อมูลจาก Keyboard ด้วย <code>scanf</code> .....	62
3.5.1 รูปแบบคำสั่ง <code>scanf</code> .....	62
3.5.2 การรับข้อความที่มีการเว้น (spacebar) .....	63
3.6 เครื่องหมายคำนวนทางคณิตศาสตร์ .....	64
3.6.1 การเพิ่มลดค่าตัวแปร.....	65
3.6.2 เครื่องหมายแบบลดรูป.....	65
3.6.3 ตัวดำเนินการ และตัวถูกดำเนินการ .....	66
3.6.4 นิพจน์.....	66
3.6.4 การเปลี่ยนชนิดของข้อมูลZ .....	67

## สารบัญ (ต่อ)

3.7 คำสั่งอื่นที่ใช้ แสดงผล และรับข้อมูล .....	67
3.8 คำถ้าท้ายบท .....	69
การทดลองที่ 3 การใช้งานคำสั่ง Input Output และการคำนวณต่างๆ .....	70
ทฤษฎี.....	70
การทดลองที่ 3.1 การแสดงผลข้อความแบบต่าง ๆ .....	72
การทดลองที่ 3.2 การรับข้อมูลด้วยตัวแปรจำนวนทศนิยม .....	73
การทดลองที่ 3.3 การรับข้อมูลด้วยตัวแปรข้อความ .....	74
การทดลองที่ 3.4 การคำนวณ.....	75
การทดลองที่ 3.5 การรับข้อมูลจากคีย์บอร์ด .....	76
การทดลองที่ 3.6 ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง .....	77
บทที่ 4 ผังงาน และการเขียนโปรแกรมกำหนดเงื่อนไข .....	79
4.1 การพัฒนาโปรแกรมคอมพิวเตอร์ .....	79
อัลกอริทึม (Algorithm).....	79
ขั้นตอนการคิด .....	80
4.1 ผังงาน (Flowchart).....	80
4.2 การเขียนโปรแกรมแบบกำหนดเงื่อนไข .....	82
4.2.1 การเปรียบเทียบ .....	83
4.2.2 เปรียบเทียบทางตรรกศาสตร์ .....	83
4.3 คำสั่งเงื่อนไข if .....	84
4.4 คำสั่งเงื่อนไข if-else .....	86
4.4 คำสั่งเงื่อนไข if-else ลักษณะผูกกัน .....	87
4.4 คำสั่ง switch case .....	89
4.5 คำถ้าท้ายบท .....	90
การทดลองที่ 4 การเขียนโปรแกรมแบบกำหนดเงื่อนไข .....	91
ทฤษฎี.....	91
การทดลองที่ 4.1 จากตัวอย่างการใช้งานคำสั่ง ให้นักศึกษาประยุกต์ใช้งานดังต่อไปนี้ .....	94
การทดลองที่ 4.2 เครื่องคิดเลข การบวกและการลบ.....	95
การทดลองที่ 4.3 ความแตกต่างระหว่างคำสั่ง if และ if . . . else .. .	96
การทดลองที่ 4.4 การเปรียบเทียบตัวเลข 3 จำนวน .....	97
การทดลองที่ 4.5 การเปรียบเทียบตัวเลขจำนวนทศนิยม .....	98
การทดลองที่ 4.6 : ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง .....	99
บทที่ 5 การเขียนโปรแกรมแบบวนซ้ำ.....	101
5.1 บทนำ .....	101
5.2 คำสั่ง while .....	102
5.3 คำสั่ง do-while .....	104
5.4 คำสั่ง for .....	105

## สารบัญ (ต่อ)

5.4 คำสั่งที่ยังไม่ได้บรรยาย .....	108
การทดลองที่ 5 การเขียนโปรแกรมแบบวนซ้ำ .....	109
ทฤษฎี.....	109
การทดลองที่ 5.1 while loop .....	113
การทดลองที่ 5.2 การหาผลรวมของเลขตั้งแต่ 1 ถึงค่าที่รับเข้ามา while loop .....	114
การทดลองที่ 5.3 do . . . while loop.....	115
การทดลองที่ 5.4 for loop .....	116
การทดลองที่ 5.5 Infinite Loop (ลูปที่รันไม่รู้จบ) .....	117
การทดลองที่ 5.6 ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง .....	118
บทที่ 6 การเขียนโปรแกรมแบบวนซ้ำ และกำหนดเงื่อนไข .....	120
6.1 ตัวอย่างการประยุกต์ใช้คำสั่งวนซ้ำ กับเงื่อนไข เพื่อแก้ปัญหาต่างๆ .....	120
6.2 คำสั่งที่ยังไม่ได้บรรยาย .....	126
การทดลองที่ 6 การเขียนโปรแกรมการวนรอบ และกำหนดเงื่อนไข .....	128
การทดลองที่ 6.1.....	128
การทดลองที่ 6.2.....	128
การทดลองที่ 6.3.....	128
การทดลองที่ 6.4.....	128
การทดลองที่ 6.5.....	128
การทดลองที่ 6.6.....	129
บทที่ 7 ตัวแปรและลำดับ Array .....	130
วัตถุประสงค์การศึกษา .....	130
7.1 ตัวแปรและลำดับ Array .....	130
7.2 อาร์เรย์ 1 มิติ.....	131
7.2.1 การกำหนดค่าเริ่มต้นให้อาร์เรย์.....	131
7.2.2 การอ้างอิงข้อมูลอาร์เรย์.....	132
7.3 ตัวแปรและลำดับและเก็บข้อมูล .....	134
7.4 อาร์เรย์ 2 มิติหรือมากกว่า.....	134
7.4.1 การกำหนดค่าเริ่มต้นให้อาร์เรย์ 2 มิติ .....	135
7.5 การสับค่าในอาร์เรย์ .....	138
7.6 คำสั่งที่ยังไม่ได้บรรยาย .....	138
การทดลองที่ 7 ตัวแปรและลำดับ .....	140
ทฤษฎี.....	140
การทดลองที่ 7.1.....	142
การทดลองที่ 7.2.....	142
การทดลองที่ 7.3.....	143
บทที่ 8 ตัวแปรโครงสร้าง Structure .....	144

## สารบัญ (ต่อ)

8.0 ตัวแปรโครงสร้าง .....	144
8.1 การประกาศตัวแปรโครงสร้าง .....	145
8.2 การอ้างถึงตัวแปรในโครงสร้าง .....	145
8.3 การกำหนดข้อมูลให้ตัวแปรโครงสร้าง .....	145
8.4 การกำหนดค่าเริ่มต้นให้โครงสร้าง .....	147
8.5 ประกาศตัวแปรโครงสร้างแบบอาร์เรย์ .....	147
8.6 โครงสร้างชั้อนโครงสร้าง (Nest structure) .....	149
8.7 คำถາມท้ายบท .....	150
การทดลองที่ 8 โครงสร้างข้อมูล .....	152
ทฤษฎี .....	152
การทดลองที่ 8.1 .....	154
การทดลองที่ 8.2 .....	154
บทที่ 9 ตัวแปรชี้ตำแหน่ง (Pointer) .....	156
9.1 โครงสร้างของหน่วยความจำและตัวชี้ .....	156
9.1.1 การประกาศตัวแปร pointer .....	156
9.1.2 ตัวดำเนินการ (Reference Operator) “&” .....	157
9.1.3 ตัวดำเนินการเชิงอ้อม (Indirect Operator) “*” .....	157
9.2 การแสดงตัวชี้ด้วยฟังก์ชัน printf .....	157
9.3 Pointer ชั่วคราว Pointer .....	160
9.4 ตัวชี้และแผลงคำบัญ .....	160
9.4.1 การย้าย Pointer ที่ชี้ตัวแปรอาร์เรย์ ด้วย += และ -= .....	161
9.4.2 การย้าย Pointer ที่ชี้ตัวแปรอาร์เรย์ ด้วย + และ - .....	161
9.4.3 การคำนวณจำนวน element ด้วย pointer .....	161
9.4 ความสัมพันธ์ระหว่าง Pointer กับ Array .....	162
9.4.1 กรณีที่ 1 ใช้ array ปกติ .....	162
9.4.2 กรณีที่ 2 ใช้ array เขียนแบบ pointer .....	163
9.4.3 กรณีที่ 3 ใช้ pointer เขียนแบบ array .....	163
9.4.4 กรณีที่ 4 ใช้ pointer ปกติ .....	163
9.5 ตัวชี้กับตัวแปรโครงสร้าง .....	164
9.6 คำถາມท้ายบท .....	166
การทดลองที่ 9 ตัวชี้ (Pointer) .....	167
ทฤษฎี .....	167
การทดลองที่ 9.1 .....	169
การทดลองที่ 9.2 .....	170
การทดลองที่ 9.3 .....	170
การทดลองที่ 9.4 .....	171

## สารบัญ (ต่อ)

คำถ้ามพิเศษ.....	172
บทที่ 10 พังก์ชัน (Function) .....	173
10.1 พังก์ชันคืออะไร .....	173
10.2 ประเภทของพังก์ชัน .....	174
10.2.1 พังก์ชันไลบรารีมาตรฐาน .....	174
10.2.2 พังก์ชันที่ผู้ใช้งานสร้างขึ้นเอง (User-defined function) .....	176
10.3 รูปแบบโครงสร้างของพังก์ชัน.....	177
10.4 การรับค่าและส่งค่าจากพังก์ชัน .....	178
10.5 ตัวแปรและขอบเขตการใช้งานพังก์ชัน .....	180
10.6 การส่งค่าตัวแปร.....	181
10.6.1 Pass by value .....	181
10.6.2 Pass by reference .....	181
10.7 คำถ้ามท้ายบท.....	183
การทดลองที่ 10 พังก์ชัน.....	186
ทฤษฎี.....	186
การทดลองที่ 10.1 .....	187
การทดลองที่ 10.2 .....	187
การทดลองที่ 10.3 .....	187
การทดลองที่ 10.4 .....	187
การทดลองที่ 10.5 .....	188
การทดลองที่ 10.6 .....	188
การทดลองที่ 10.7 .....	188
การทดลองที่ 10.8 .....	189
การทดลองที่ 10.9 .....	189
บทที่ 11 ไฟล์ข้อมูล.....	190
11.1 ไฟล์ข้อมูล .....	190
11.2 เปิดไฟล์ .....	191
11.3 ปิดไฟล์ .....	192
11.4 อ่านไฟล์รูปแบบ Text .....	193
11.5 เขียนไฟล์รูปแบบ text file .....	195
11.6 เขียน array หรือ struct ลงไฟล์รูปแบบ binary .....	196
11.7 อ่าน array หรือ struct จากไฟล์รูปแบบ binary.....	197
11.8 ย้าย file pointer.....	197
11.9 คำสั่งอื่นเกี่ยวกับไฟล์ .....	198
11.10 คำถ้ามท้ายบท .....	199
การทดลองที่ 11 การเขียนโปรแกรมติดต่อกับ Text File.....	200

## สารบัญ (ต่อ)

ทฤษฎี.....	200
การทดลองที่ 11.1 .....	202
การทดลองที่ 11.2 .....	202
การทดลองที่ 11.3 .....	202
การทดลองที่ 11.4 .....	202
การทดลองที่ 11.5 .....	202
การทดลองที่ 11.6 .....	203
หนังสืออ้างอิง.....	204

## สารบัญโปรแกรม

โปรแกรม 2.1 การใช้งานคำสั่ง printf.....	40
โปรแกรม 2.2 การใช้งานคำสั่ง printf หลายๆคำสั่ง .....	40
โปรแกรม 2.3 การใช้งานคำสั่ง printf มีสัญลักษณ์พิเศษควบคุมการแสดงผล.....	41
โปรแกรม 2.4 printf มีสัญลักษณ์พิเศษควบคุมการแสดงผล .....	41
โปรแกรม 2.4 แสดงรายได้ต่อเดือน .....	44
โปรแกรม 2.5 แสดงรายได้ต่อเดือน .....	44
โปรแกรม 2.6 โปรแกรมแสดงเกรด.....	45
โปรแกรม 3.1 คำสั่ง scanf .....	63
โปรแกรม 3.2 คำสั่ง scanf .....	64
โปรแกรม 3.3 แสดงการคำนวณ .....	64
โปรแกรม 3.4 เพิ่ม/ลดค่าตัวแปร .....	65
โปรแกรม 3.5 เพิ่ม/ลดค่าตัวแปร .....	66
โปรแกรม 3.6 แสดงลำดับความสำคัญสัญลักษณ์ทางคณิตศาสตร์ .....	66
โปรแกรม 3.7 เรียกใช้ putchar, puts .....	67
โปรแกรม 3.8 เรียกใช้ getch, getch, puts .....	68
โปรแกรม 4.1 การสร้างโปรแกรมคำนวณหาพื้นที่วงกลม .....	81
โปรแกรม 4.2 โปรแกรมตรวจสอบคะแนน .....	85
โปรแกรม 4.3 หารเลข 2 จำนวน.....	87
โปรแกรม 4.4 แสดงเกรด A ถึง F .....	88
โปรแกรม 5.1 ไม่มีการวนซ้ำ .....	101
โปรแกรม 5.2 แสดงเลข 0..10 ด้วย while.....	102
โปรแกรม 5.3 หากรวมของเลขใช้ while .....	103
โปรแกรม 5.4 หากรวม 1 ถึง 100 ใช้ do-while .....	104
โปรแกรม 5.5 โปรแกรมหากรวม 1 ถึง 100 ใช้ for .....	106
โปรแกรม 5.6 แสดงผล a – z ใช้ for .....	106
โปรแกรม 5.7 แสดงผลรูปสี่เหลี่ยม ใช้ for.....	107
โปรแกรม 6.1 โปรแกรมแสดงสูตรคูณแม่ 2.....	120
โปรแกรม 6.2 โปรแกรมแสดงเลข 0 – 100 ใช้ while .....	121

## สารบัญโปรแกรม (ต่อ)

โปรแกรม 6.3 โปรแกรมแสดงเลขคู่ 0 – 100 ใช้ while .....	121
โปรแกรม 6.4 โปรแกรมตรวจสอบจำนวนสรุป ใช้ for .....	122
โปรแกรม 6.5 โปรแกรมแสดงผลสรุปสี่เหลี่ยมกล่อง .....	123
โปรแกรม 6.6 โปรแกรมตู้ ATM.....	124
โปรแกรม 6.7 โปรแกรมคำนวณค่าอาหาร .....	125
โปรแกรม 7.1 โปรแกรมรับอายุของคน 20 คน.....	132
โปรแกรม 7.2 โปรแกรมวิเคราะห์ส่วนสูงของคน ก คน .....	133
โปรแกรม 7.3 โปรแกรมรับข้อมูลและแสดงผลลัพธ์แบบเมทริกซ์ .....	136
โปรแกรม 7.4 หาผลรวมในเมทริกซ์.....	137
โปรแกรม 7.5 นับจำนวนข้อความ.....	137
โปรแกรม 7.6 หาค่ามากที่สุด.....	138
โปรแกรม 8.1 ประกาศตัวแปรโครงสร้าง .....	146
โปรแกรม 8.2 กำหนดค่าให้ตัวแปรโครงสร้าง .....	146
โปรแกรม 8.3 โปรแกรมเก็บข้อมูลหนังสือ .....	146
โปรแกรม 8.4 เก็บข้อมูลคน.จำนวน 10 คน .....	148
โปรแกรม 8.5 โปรแกรมเก็บข้อมูลสถานศึกษา .....	150
โปรแกรม 9.1 โปรแกรมแสดงข้อมูลผ่านทาง pointer .....	158
โปรแกรม 9.2 โปรแกรมการใช้ referencing และ dereferencing .....	159
โปรแกรม 9.3 โปรแกรมการใช้ referencing และ dereferencing แบบ2.....	159
โปรแกรม 9.4 โปรแกรมแสดงการใช้ pointer กับ array .....	162
โปรแกรม 9.5 โปรแกรมแสดงการใช้ pointer กับ array .....	162
โปรแกรม 9.6 โปรแกรมเก็บข้อมูลคน. 10 คน.....	165
โปรแกรม 10.1 โปรแกรมคำนวณ sin,cos,tan,sqrt,pow,log,log10,exp และ fabs .....	175
โปรแกรม 10.2 โปรแกรมใช้ strcpy, strcat, strcmp, strncat, strlen, tolower และ toupper .....	176
โปรแกรม 10.3 โปรแกรมแสดง kmitl ในกรอบสี่เหลี่ยม.....	177
โปรแกรม 10.4 โปรแกรมคำนวณ degree radian .....	180
โปรแกรม 10.5 โปรแกรมแสดง pass by value และ pass by reference .....	182
โปรแกรม 10.6 โปรแกรมใช้ pass by value.....	182

## สารบัญโปรแกรม (ต่อ)

โปรแกรม 10.7 โปรแกรมใช้ pass by reference .....	183
โปรแกรม 10.8 โปรแกรมรู้จัก global variable .....	183
โปรแกรม 11.1 โปรแกรมเปิด text file.....	192
โปรแกรม 11.2 โปรแกรมเปิดและปิด text file.....	193
โปรแกรม 11.3 โปรแกรมอ่านข้อมูลจาก text file (string หนึ่งตัว).....	193
โปรแกรม 11.4 โปรแกรมอ่านข้อมูลจาก text file (string หลายตัว) .....	194
โปรแกรม 11.5 โปรแกรมอ่านข้อมูลจาก text file (character).....	194
โปรแกรม 11.6 โปรแกรมอ่านข้อมูลจาก text file (string กับ int).....	194
โปรแกรม 11.7 โปรแกรมรับชื่ออายุ 3 คน เก็บลงไฟล์ .....	195
โปรแกรม 11.8 โปรแกรมเก็บข้อมูลจาก struct ลงไฟล์ .....	196
โปรแกรม 11.9 โปรแกรมอ่านข้อมูลจากไฟล์เก็บลง struct.....	197
โปรแกรม 11.10 โปรแกรมอ่านข้อมูลจากไฟล์เก็บลง struct ใช้ fseek .....	198

## บทที่ 0 แนะนำวิชา

วิชาคอมพิวเตอร์และการโปรแกรม (Computer & Programming) มีอาจารย์สอนทั้งหมดสิบสามท่านดังนี้

อ.ประสาร	ตั้งติศาնนท์
อ.ณัฐ	ตั้งติศาնนท์
อ.บัณฑิต	พัชยา
อ.วิญญาลัย	พร้อมพานิชย์
ดร.ธนัญชัย	ตรีภาค
อ.เกียรตินรังค์	ทองประเสริฐ
อ.วัฒนพงศ์	เกษมศิริ
ดร.พิกุลแก้ว	ตั้งติศาնนท์
ผศ.มยุรี	เลิศเวชกุล
รศ.ทรงชัย	วีระพัฒนาศ
อ.สุธรรมรัตน์	สัทธิธรรมสกุล
อ.สรยุทธ	กลมกล่อม

### วัตถุประสงค์ของวิชา

- เข้าใจองค์ประกอบ และการทำงานของระบบคอมพิวเตอร์
- รู้จักโครงสร้างภาษา และคำสั่งต่างๆ ของภาษาซี เพื่อให้สามารถเขียนเป็นโปรแกรมคอมพิวเตอร์ได้
- สามารถเขียนโปรแกรมสั่งให้คอมพิวเตอร์ทำงานตามที่กำหนดได้
- สามารถประมวลผลผลลัพธ์ของโปรแกรมตามคำสั่งที่กำหนดให้ได้
- สามารถหาข้อผิดพลาด และแก้ไขโปรแกรมที่กำหนดให้ถูกต้อง

### การวัดผล

การวัดผลแบบอิงเกณฑ์ โดย ปีการศึกษา 255 นี้จะเน้นและการวัดผลเป็นดังนี้

ภาคปฏิบัติ (Laboratory)	เข้าห้องปฏิบัติ	20% (เช็คชื่อเข้าห้อง + เช็คโปรแกรม)
ภาคทฤษฎี (Lecture)	สอบกลางภาค	40%
	สอบปลายภาค	40%

### ระเบียบการใช้ห้องปฏิบัติการ

- แต่งกายให้เรียบร้อยถูกระเบียบสถาบันฯ
- ไม่นำอาหารเครื่องดื่มเข้าห้องปฏิบัติการ
- เตรียมเอกสารประกอบการทดลองพร้อมทั้งทำความสะอาดเข้าใจก่อนเข้าห้อง
- ให้นักศึกษาทำการทดลองตามเอกสารด้วยตนเอง หากไม่เข้าใจให้สอบถามอาจารย์ และผู้ช่วยสอนประจำห้อง
- ในกรณีการลาป่วยต้องมีใบรับรองแพทย์ แบบพร้อมใบลา กรณีลากิจอนุญาตเฉพาะการทำกิจกรรมในฐานะตัวแทนสถาบันเท่านั้น การลาป่วย และลากิจ ต้องแจ้งอาจารย์ผู้ประสานงานวิชาฯ เท่านั้น ลาป่วยต้องส่งเอกสารภายใน 7 วัน (นับตั้งแต่วันลา) ส่วนลากิจต้องส่งเอกสารล่วงหน้า 7 วัน

6. การคิดคะแนนในการเข้าห้องปฏิบัติการ 1 ครั้งประกอบด้วย 1) คะแนนเข้าห้องปฏิบัติการ 0.5 คะแนน โดยนักศึกษาเข้าห้องปฏิบัติการไม่เกินเวลา 15 นาที ถ้าเข้าหลังจาก 15 นาที จะไม่ได้คะแนนนี้ 2) คะแนนการส่งโปรแกรมในห้องปฏิบัติการ 0.5 คะแนน นักศึกษาจะได้คะแนนนี้จากการส่งโปรแกรมให้เจ้าหน้าที่ดูแลห้องปฏิบัติการตรวจ ภายในเวลาขั้วโมงทำปฏิบัติครั้งต่อครั้ง โดยโจทย์จะมาจากเอกสารการทดลอง ที่เจ้าหน้าที่จะกำหนดให้นักศึกษาทำภายในแต่ละห้อง ครั้งละไม่ต่ำกว่า 1 ข้อ

# บทที่ 1 พื้นฐานเกี่ยวกับคอมพิวเตอร์

## วัตถุประสงค์การศึกษา

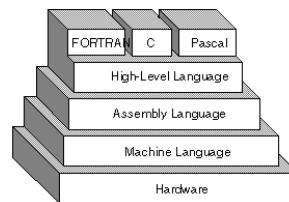
- เข้าใจองค์ประกอบ และการทำงานของระบบคอมพิวเตอร์
- เข้าใจขั้นตอนการเขียนโปรแกรมภาษาซีเบื้องต้น
- รู้จักโปรแกรม Visual C++ 2008 Express Edition

## 1. ความเป็นมาของคอมพิวเตอร์

### 1.1 คอมพิวเตอร์คืออะไร

เป็นอุปกรณ์อิเล็กทรอนิกส์อย่างหนึ่ง ที่มีมุขย์ประดิษฐ์ขึ้น เพื่อนำมาเสริมความสามารถของมนุษย์ในด้านการรับรู้ การจำ การคำนวณ การเปรียบเทียบตัดสินใจ เป็นต้น การที่จะส่งให้เครื่องคอมพิวเตอร์ทำงาน จะเป็นต้องป้อนคำสั่ง และ จะต้องเป็นคำสั่งที่เครื่องคอมพิวเตอร์เข้าใจ คำสั่งเหล่านี้มีหลายคำสั่งเรียกว่า ภาษาเครื่อง (Machine Language) เป็นภาษาระดับต่ำสุด (Low-level language) มีลักษณะอยู่ในรูปเลขฐานสอง ประกอบด้วยตัวเลข 0 กับ 1 จำกัดด้านล่าง จะเห็นว่า Machine language อยู่ใกล้ Hardware หากที่สุด

Machine language อยู่ในรูปเลขฐานสอง  
 10000011 00111101 11010111  
 00001010 10111000 11100010  
 11111111 11111111 11111111  
 01101010 01000000 11101000



### 1.2 ขั้นตอนการทำงานของคอมพิวเตอร์

คอมพิวเตอร์ประกอบด้วยส่วนหลักๆ สี่ส่วนคือ input device, output device, processor หรือ CPU และ memory



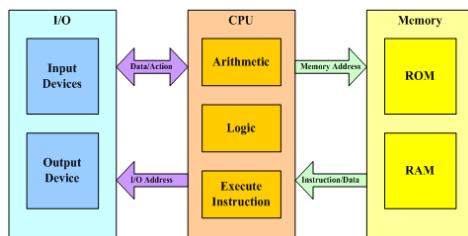
### 1.3 องค์ประกอบหลักของคอมพิวเตอร์

- ฮาร์ดแวร์ (Hardware) หมายถึง ส่วนประกอบทางอิเล็กทรอนิกส์ และแมคคานิคส์ (Mechanical) หรือส่วนที่เป็นกลไก ทั้งหมดที่ประกอบกันเป็นเครื่องคอมพิวเตอร์
- ซอฟต์แวร์ (Software) หมายถึง ส่วนที่เป็นชุดคำสั่ง หรือโปรแกรมที่ส่งให้อุปกรณ์ฮาร์ดแวร์ร่วมกันทำงานตามที่ต้องการ เช่น การคำนวณ การบันทึกผล การแสดงผล ฯลฯ

## ส่วนประกอบของฮาร์ดแวร์

- หน่วยประมวลผลกลาง (Central Processing Unit) เช่น 80486 Celeron Pentium Athlon Duron
- หน่วยความจำ (Memory) เช่น SDRAM 128 MB, ROM BIOS ฮาร์ดดิสก์ ชีดีรอม

- หน่วยนำเข้าข้อมูล (Input Data) เช่น คีย์บอร์ด เม้าท์ กล้อง
- หน่วยส่งออกข้อมูล (Output Data) เช่น จอภาพ เครื่องพิมพ์ เครื่องฉายภาพ ลำโพง



### ส่วนประกอบของซอฟต์แวร์

เป็นชุดคำสั่งที่ควบคุมการทำงานของคอมพิวเตอร์ โดยชุดคำสั่งเหล่านี้เรียกว่าโปรแกรมคอมพิวเตอร์

- ซอฟต์แวร์ระบบ (Operating system: OS หรือ System software)

ได้แก่

DOS (Disk Operating System) ปี 1981-1995 เป็นระบบปฏิบัติการในยุค 80 ใช้การสั่งงานผ่านคำสั่ง command line เช่น ต้องการตรวจสอบจำนวนไฟล์ในแฟลเดอร์ ใช้คำสั่ง dir

```
C:\Temp>dir
Volume in drive C has C:
Volume Serial Number is 74F5-B93C
Directory of C:\Temp
2009-08-25 11:59 <DIR> .
2009-08-25 11:59 <DIR> ..
2009-04-03 10:01 2,321,600 AddressUpdater12345.exe
2009-04-03 10:01 27,988 dd_despatch_dotnetfx30.txt
2009-04-03 10:01 32,572 dd_dotnetfx/install.txt
2009-06-09 13:46 35,145 GameProfile.log
2009-04-20 08:17 35,200 GameProfile.log.old
2009-04-20 08:17 402 MS1270B.LOG
2009-04-20 10:01 38,893 OfficePatches
2009-04-03 16:02 <DIR> OfficePatches
2009-08-25 10:52 16,384 PerfLib_Perfdata_c30.dat
2009-08-25 11:42 50,245,632 W27.log
2009-08-25 11:42 1,291 {AC76dABE-7A07-1033-7B44-4E1200000003}.ini
2009-04-20 10:01 52,723,795 bytes
4 File(s) 83,570,786,768 bytes free
```

Windows ปี 1985-ปัจจุบัน เป็นระบบ GUI (Graphic User Interface) ภาษาที่ใช้ในการสร้างคือ C/C++ และ Assembly language



OS2 (Operating System /2) ปี 1987-2001 เป็นระบบปฏิบัติการที่พัฒนาโดย Microsoft และ IBM ในช่วงแรก และต่อมา IBM เป็นผู้พัฒนาต่อ ภาษาที่ใช้ในการสร้างคือ C/C++



Unix ปี 1969-ปัจจุบัน ภาษาที่ใช้ในการสร้างคือ C	
---	--

## 2) ซอฟต์แวร์ประยุกต์ (Application software)

2.1) ซอฟต์แวร์สำเร็จรูป (Commercial applications) คือ Microsoft Offices, โปรแกรมสื่อสารข้อมูล

2.2) ซอฟต์แวร์ใช้งานเฉพาะ (Particular applications) คือโปรแกรมคำนวนเกรด, โปรแกรมยืมคืน

หนังสือ, โปรแกรมคำนวนดอกเบี้ย, โปรแกรมร้านเช่าหนังสือ โปรแกรมดู DVD, โปรแกรมฟังเพลง

ซอฟต์แวร์ หรือโปรแกรมต่างๆ ที่ทำงานโดยคอมพิวเตอร์จะใช้ชุดคำสั่งภาษาเครื่องในการทำงาน แต่ภาษาเครื่องเป็นภาษาที่มนุษย์ยากที่จะจำและเข้าใจการทำงาน จึงมีการคิดภาษาต่างๆ ที่ใกล้เคียงกับภาษาของมนุษย์ขึ้นมาเพื่อให้มนุษย์ใช้งานในการเขียนสร้างโปรแกรมอื่นๆ

## 1.4 ภาษาคอมพิวเตอร์

ภาษาคอมพิวเตอร์แบ่งออกเป็น

- 1) **ภาษาเครื่อง (Machine Language)** เป็นภาษาระดับต่ำสุด (Low-level language) อยู่ในรูปเลขฐานสอง (Binary number) ประกอบด้วยเลข 0 กับ 1 ภาษาเครื่องสามารถสั่งให้คอมพิวเตอร์ทำงานได้ทันที อย่างไรก็ตาม ข้อเสียของภาษาเครื่องคือลักษณะที่เป็นเลขฐานสอง ทำให้โปรแกรมเมอร์พัฒนาโปรแกรมได้ยาก
- 2) **ภาษา Assembly** เป็นกีงภาษาเครื่องเขียนเป็นคำสั่ง Neumonic สามารถแปลงเป็นภาษาเครื่องได้ง่าย โดยการเทียบตาราง หรือใช้ Assembler
- 3) **ภาษาขั้นสูง** เป็นภาษาที่ใกล้เคียงกับภาษาของมนุษย์ (ประโยชน์ข้อความส่วนใหญ่เป็นภาษาอังกฤษ) เช่น ภาษา C, PASCAL, FORTRAN ซึ่งมนุษย์สามารถเขียนได้ง่ายแต่ต้องมีกระบวนการแปลงเป็นภาษาเครื่องก่อนโดยผ่านโปรแกรม Compiler

Address	Label	Instruction (AT&T syntax)	Object code <sup>[29]</sup>
		.begin	
		.org 2048	
a_start		.equ 3000	
2048		ld length,%	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1,-4,%r1	10000010 10000000 01111111 11111100
2072		addcc %r1,%r2,%r4	10001000 10000000 01000000 00000010
2076		ld %r4,%r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3,%r5,%r3	10000110 10000000 11000000 00000101
2088	done:	jmpl %r15+4,%r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

ตัวอย่างด้านข้างมีอ ภาษา Assembly ของ AT&T และภาษาเครื่อง แปลง成รูป object code ภาษา assembly คำสั่ง addcc บวกเลข เมื่อแปลงเป็นภาษาเครื่องจะได้เลข 10000010 10000....1000010 เป็นต้น

## 1.5 ระบบจำนวนและตัวเลข

ระบบจำนวนและตัวเลข เป็นระบบตัวเลขที่มีนิยมทุกคุณเคย ที่เห็นได้ชัดเจนที่สุดคือการนับจำนวนสิ่งของต่างๆ ซึ่งพบว่าระบบจำนวนส่วนใหญ่ที่ใช้กันจะมีเลข 0 – 9 แต่ยังมีระบบจำนวนชนิดอื่นอีกแบบ เช่น ในเลขโรมัน ระบบคอมพิวเตอร์ หรือภาษาคอมพิวเตอร์จะใช้เลขเพียง 2 ตัว คือ 0 และ 1 ซึ่งไม่เหมือนกับที่มีนิยมใช้กันปกติ เราเรียกระบบนับจำนวนที่มีจำนวนตัวเลขที่ใช้งานต่างกันว่าระบบเลขฐาน เช่น ระบบตัวเลขฐานสิบ ระบบตัวเลขฐานสอง เป็นต้น

### ระบบเลขฐาน

ฐานสิบ ประกอบด้วย ตัวเลข 0-9

$$(525)_{10} = 5 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

ฐานสอง ประกอบด้วย ตัวเลข 0 กับ 1

$$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

ฐานสิบหก ประกอบด้วยตัวเลข 0-9 และ A-F (โดยแทน 10 = A, 11 = B, 12 = C, 13 = D, 14 = E และ 15 = F)

$$(3B2)_{16} = 3 \times 16^2 + 11 \times 16^1 + 2 \times 16^0$$

### ประโยชน์ที่ymbฐานสิบกับฐานสอง

ระบบเลขฐานสิบ

$$10^n \quad 10^{n-1} \quad \dots \quad 10^2 \quad 10^1 \quad 10^0$$

ระบบเลขฐานสอง

MSB					
$2^n$	$2^{n-1}$	$\dots$	$2^2$	$2^1$	$2^0$

MSB ย่อมาจาก Most Significant bit บิตที่มีนัยสำคัญสูง ส่วน LSB ย่อมาจาก Least significant bit บิตที่มีนัยสำคัญต่ำ

### การแปลงเลขฐานสองเป็นฐานสิบ

ตัวอย่าง จงแปลง 10001112 ไปเป็นเลขฐานสิบ

1	0	0	0	1	1	1
$\times$						
$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
64	32	16	8	4	2	1

คำนวณได้

$$\begin{aligned}
 & (1 \times 64) + (0 \times 32) + (0 \times 16) + (0 \times 8) + (1 \times 4) + (1 \times 2) + (1 \times 1) \\
 & = 64 + 0 + 0 + 0 + 4 + 2 + 1 \\
 & = 71
 \end{aligned}$$

การแปลงเลขฐานสิบเป็นฐานสอง

จะแปลง 45 ฐานสิบ ให้เป็นฐานสอง

$$\begin{array}{r}
 2) \underline{45} \\
 2) \underline{22} \quad \text{เศษ 1 LSB} \\
 2) \underline{11} \quad \text{เศษ 0} \\
 2) \underline{5} \quad \text{เศษ 1} \\
 2) \underline{2} \quad \text{เศษ 1} \\
 1 \quad \text{เศษ 0}
 \end{array}$$

↑  
MSB

$$45_{10} = 101101_2$$

การแปลงเลขฐานสิบเป็นฐานสิบหก

จะแปลง 946 ฐานสิบ ให้เป็นฐานสิบหก

$$\begin{array}{r}
 16) \underline{946} \\
 16) \underline{59} \quad \text{เศษ 2} \\
 16) \underline{3} \quad \text{เศษ } 11 = B \\
 0 \quad \text{เศษ 3}
 \end{array}$$

$$946_{10} = 03B2_{16}$$

การแปลงตัวเลขฐานสองเป็นฐานสิบหก

ฐานสิบหก	ฐานสอง	ฐานสิบหก	ฐานสอง
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

## ตัวอย่าง

แปลงเลขจาก  $12EC_{16}$  ฐาน 16 เป็น ฐานสองจะได้

$$12EC = 0001-0010-1110-1100$$

แปลงเลขจาก  $101110011100011110011_2$  เป็นเลขฐาน 16 จะได้

$$\underline{0001-0111-0011-1000-1111-0011} = 1738F3_{16} \quad (\text{จากตัวอย่างมีการเติม } 000 \text{ ที่ข้างเดียวเพื่อทำให้ตัวเลขครบ } 4 \text{ ตัว})$$

## 1.6 ขนาดตัวเลขในระบบคอมพิวเตอร์

บิต (bit) คือตัวเลขฐานสอง (binary) เพียงตัวเดียว

ไบต์ (Byte) คือ ตัวเลขขนาด 8 บิต

เวิร์ด (Word) คือ ตัวเลขขนาด 16 บิต หรือ 2 ไบต์

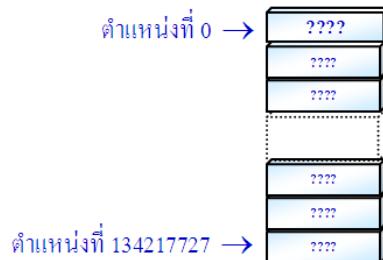
กิโลไบท์ (KB) คือ ตัวเลขขนาด  $2^{10} = 1024$  ไบท์

เมกะไบท์ (MB) คือ ตัวเลขขนาด  $2^{10} \times 2^{10}$  ไบท์

จิกะไบท์ (GB) คือ ตัวเลขขนาด  $2^{10} \times 2^{10} \times 2^{10}$  ไบท์

### ขนาดหน่วยความจำของเครื่องคอมพิวเตอร์

หน่วยความจำ 128 MB หมายความว่าสามารถเก็บข้อมูลได้  $128 \times 2^{10} \times 2^{10} = 134217728$  ตำแหน่ง



### ตัวเลขที่ใช้งานคอมพิวเตอร์

ตัวเลขจำนวนเต็มที่มีเฉพาะค่าบวก

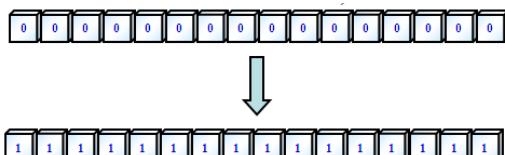
ขนาด 8 bit เรียกว่า 1 Byte มีค่าเริ่มจาก 0 ถึง 255



ตัวอย่าง มีข้อมูล 8 bit มีค่า 11001110 คำนวนแล้วมีค่าเท่าได้

$$11001110_2 = 128 + 64 + 8 + 4 + 2 = 206_{10}$$

ขนาด 16 bit เรียกว่า 1 Word (2 Byte) มีค่าเริ่มจาก 0 ถึง 65,536



ตัวอย่าง มีข้อมูล 16 bit มีค่า 1100-1110-1100-1110 คำนวนแล้วมีค่าเท่าได้

$$1100-1110-1100-1110_2 = 52942_{10}$$

### จำนวนเต็ม ที่มีทั้งค่าบวกและค่าลบ

ขนาด 8 bit มีค่าเริ่มจาก -128 ถึง 127 (2's Complement)



บิตแรกสุด เป็น sign bit (1 เป็นค่าลบ, 0 เป็นค่าบวก)

ขนาด 16 bit เรียกว่า integer มีค่าเริ่มจาก -32,768 ถึง 32,767

ขนาด 32 bit เรียกว่า long integer มีค่าเริ่มจาก -2,147,483,648 ถึง 2,147,483,647

ตัวอย่าง การแปลงเลขจาก -5 ฐาน 10 แบบ 8 Bit 2's Complement มีขั้นตอนดังนี้

1) เขียน เลข 5 ในรูปฐานสองปกติ ได้ 0000-0101

2) ทำการกลับบิตจาก 0 เป็น 1 และ 1 เป็น 0 เรียกว่า 1's Complement จะได้

0000-0101 กลับบิตเป็น 1111-1010

3) นำผลจาก 1's Complement แปลงเป็น 2's Complement โดยการบวก 1 จะได้

$$1111-1010 + 1 = 1111-1011 \text{ เป็นคำตอบ}$$

### จำนวนทศนิยม

ขนาด 4 Byte เรียกว่า float มีค่าเริ่มจาก  $3.4 \times 10^{-38}$  ถึง  $3.4 \times 10^{38}$

ขนาด 8 Byte เรียกว่า double มีค่าเริ่มจาก  $1.7 \times 10^{-308}$  ถึง  $1.7 \times 10^{308}$

ขนาด 10 Byte เรียกว่า long double มีค่าเริ่มจาก  $3.4 \times 10^{-4932}$  ถึง  $3.4 \times 10^{4932}$



### 1.7 ตัวอักษรที่ใช้ในคอมพิวเตอร์

ข้อมูลที่เป็นตัวอักษรและสัญลักษณ์ จะใช้รหัสแอลซี (American National Standard Code for Information Interchange หรือย่อว่า ASCII) ขนาด 8 บิต ซึ่งเป็นรหัสมาตรฐาน ถูกกำหนดโดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (American National Standard Institute หรือย่อว่า ANSI) เช่น

ตัวอักษร A มีค่าเท่ากับ  $65_{10}$  หรือ  $41_{16}$

ตัวอักษร B มีค่าเท่ากับ  $66_{10}$  หรือ  $42_{16}$

ตัวอักษร a มีค่าเท่ากับ  $97_{10}$  หรือ  $61_{16}$

ตัวอักษร 7 มีค่าเท่ากับ  $55_{10}$  หรือ  $37_{16}$

### ตาราง ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Γ	Ἴ	Ἵ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ	Ϛ
1	+				⊥	T	⊣	⊣								
2	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	-
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8																
9																
A																
B																
C																
D																
E																
F																

### ข้อสังเกตในการเก็บค่าลงหน่วยความจำ

7 ที่เป็นค่าตัวเลขจะเก็บลงในหน่วยความจำของคอมพิวเตอร์ดังนี้

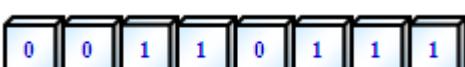
กรณีเก็บเป็นชนิด 1 byte



กรณีเก็บเป็นชนิด Integer (2 byte)



7 ที่เป็นอักขระจะเก็บลงในหน่วยความจำของคอมพิวเตอร์ดังนี้



### ลักษณะการใช้งานตัวอักษรในภาษาซี

ตัวอักษร 1 ตัว เรียกว่า อักขระ (Character) จะใช้สัญลักษณ์ Single quote (' ') แสดงขอบเขตของข้อมูล ดังนี้

เข่น	O	การใช้งานในภาษาซี	'O'
	z	การใช้งานในภาษาซี	'z'

ตัวอักษรเป็นชุด เรียกว่า ข้อความ (string) จะใช้สัญลักษณ์ Double quote (" ") แสดงขอบเขตของข้อมูล ดังนี้

เข่น	Com	การใช้งานในภาษาซี	"Com"
	kmitl	การใช้งานในภาษาซี	"kmitl"

## 1.8 รู้จักภาษาซี

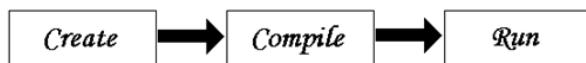
- ปี ค.ศ. 1966 มาร์ติน ริ查ร์ด (Martin Richards) พัฒนาภาษา BCPL (Basic Combined Programming Language) ที่ University of Cambridge ซึ่งเป็นต้นกำเนินภาษา C
- ปี ค.ศ. 1970 เคน ทอมพ์สัน (Ken Thompson) พัฒนาภาษา B จากภาษา BCPL เดิม
- ปี ค.ศ. 1972 เดนนิส ริทชี่ (Dennis Ritchie) พัฒนาภาษา C ขึ้นที่ AT&T Bell Labs เพื่อเพิ่มประสิทธิภาพ ภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก
- ปี ค.ศ. 1978 ไบรอัน เกอร์นิกเอน (Kernighan) และ เเดนนิส ริทชี่ (Dennis Ritchie) ได้เขียนหนังสือเล่มหนึ่งชื่อว่า "The C Programming Language" และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียน โปรแกรมมากขึ้น
- ปัจจุบัน ภาษา C ได้รับการออกเป็นมาตรฐานในปี ค.ศ. 1999 เรียกว่า C99 (ซี99) ตามมาตรฐาน ISO/IEC 9899:1999 ปัจจุบันมาตรฐานภาษา C สากระดูแลและควบคุมโดยกลุ่ม ISO/IEC JTC1/SC22/WG14

### จุดเด่นของภาษาซี

- เป็นภาษาโครงสร้าง
- เป็นภาษาระดับสูง
- ไม่เขียนอยู่กับระบบปฏิบัติการ
- ไม่เขียนอยู่กับชนิดของเครื่องคอมพิวเตอร์

วิธีการแปลภาษาที่นั้นสูงให้เป็นภาษาเครื่อง

- อินเตอร์พลีเตอร์ (Interpreter) แปลครั้งละคำสั่งแล้วส่งให้เครื่องทำงานจากนั้นจึงแปลงคำสั่งตัดไป เช่น java, perl, python, shell script, vb script
- คอมไพล์เลอร์ (Compiler) แปลคำสั่งทั้งหมดแล้วส่งให้เครื่องทำงาน เช่น C, C++, Pascal

ขั้นตอนการเขียนโปรแกรมภาษาซี

เริ่มต้นด้วย การสร้างไฟล์ภาษา C นามสกุล \*.cpp หรือ \*.c และจึงคอมไابل์เป็นภาษาเครื่อง ได้เป็นไฟล์ \*.exe ต่อจากนั้นจึงรันโปรแกรม

## การทดลองที่ 1 การพัฒนาโปรแกรมภาษาซีเบื้องต้น

### วัตถุประสงค์

1. สามารถเข้าใจขั้นตอนการพัฒนาโปรแกรมเบื้องต้น
2. สามารถใช้คอมไพล์เตอร์ Visual C++ 2008 Express Edition เบื้องต้น
3. สามารถเขียนโปรแกรมภาษา C อย่างง่าย
4. สามารถเข้าใจการทำงานของคำสั่ง printf เบื้องต้น
5. สามารถตรวจสอบหาที่ผิดและแก้ไขโปรแกรมเบื้องต้น

### การทดลอง

1. ทดลองใช้คอมไпал์เตอร์ Visual C++ 2008 Express Edition เบื้องต้น
2. ทดลองเขียนและรันโปรแกรมภาษา C อย่างง่าย
3. ทดลองหาที่ผิดและแก้ไขโปรแกรมภาษา C
4. ทดลองกำหนดรูปแบบการพิมพ์ด้วย \n และ \t ของคำสั่ง printf

### ทฤษฎี

การเขียนโปรแกรมคือการเขียนชุดคำสั่ง เพื่อให้คอมพิวเตอร์ปฏิบัติตามความต้องการของผู้ใช้งาน การเขียนชุดคำสั่งทำได้โดยการใช้ภาษาคอมพิวเตอร์ ซึ่งมีมากมายหลายภาษา อาทิ PHP, Visual Basic, C , JAVA ฯลฯ การที่จะเลือกใช้ภาษาใดภาษาหนึ่งนั้น จะต้องพิจารณาถึงความสอดคล้องกันระหว่างงานที่จะสร้างและภาษาคอมพิวเตอร์ กล่าวคือภาษาที่ใช้ต้องเหมาะสมกับงานที่ต้องการ เช่นหากต้องการเขียนโปรแกรมที่ทำงานบน WWW ภาษาที่เหมาะสมก็คือ PHP หรือ JAVA เป็นต้น

วิชาคอมพิวเตอร์และการเขียนโปรแกรมนี้ใช้ภาษาซีในการเขียนชุดคำสั่ง ซึ่งภาษาซีมีข้อดีหลายประการ อาทิเขียนเป็นภาษาที่ไม่เขียนกับhardware สามารถนำไปใช้ได้กับคอมพิวเตอร์ส่วนใหญ่ซึ่งมีระบบปฏิบัติการที่แตกต่างกัน เป็นภาษาที่มีโครงสร้าง สามารถทำความเข้าใจได้ไม่ยากนัก และเป็นที่นิยมสำหรับนักเขียนโปรแกรมทั่วไป ดังนั้นหากนักศึกษาสามารถเขียนโปรแกรมด้วยภาษาซีได้ นักศึกษาก็สามารถนำความรู้นี้ไปใช้ให้เป็นประโยชน์ได้

### ขั้นตอนการพัฒนาโปรแกรม

โปรแกรมคอมพิวเตอร์ถูกเขียนขึ้นมาด้วยวัตถุประสงค์ คือ นำไปใช้แก้ปัญหา ซึ่งในปัจจุบันโปรแกรมได้ถูกนำมาใช้แก้ปัญหาให้กับวงการต่างๆ มากมาย ขั้นตอนในการเขียนโปรแกรมก็จะคล้ายกับขั้นตอนในการแก้ไขปัญหานั้นเอง คือ เมื่อเราต้องการแก้ไขปัญหาก็จะมีขั้นตอนหลักๆ ดังนี้

- 1) กำหนดขอบเขตของปัญหา
- 2) วางแผนหาวิธีการแก้ไขปัญหา
- 3) ดำเนินการตามแผน
- 4) ทดสอบและประเมินผลว่าปัญหาได้ถูกแก้หรือไม่

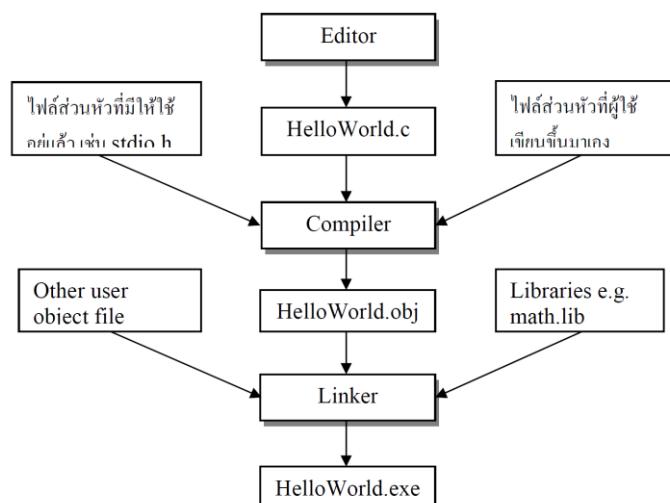
การเขียนโปรแกรม จะมีขั้นตอนหลักๆ ดังนี้

- 1) กำหนดวัตถุประสงค์ของโปรแกรม
- 2) หาวิธีการที่โปรแกรมจะใช้ในการแก้โจทย์ด้วยการเขียนเป็นแผนภาพ (Flowchart) หรือรหัสเทียม(Pseudo code)
- 3) เปลี่ยนรหัสเทียมหรือแผนภาพให้อยู่ในรูปแบบของคำสั่งคอมพิวเตอร์โดยใช้ภาษาคอมพิวเตอร์ เช่น C
- 4) ทดลองรันโปรแกรมและตรวจสอบผลลัพธ์

ขั้นตอนในการพัฒนาโปรแกรมเบื้องต้นที่โปรแกรมมีขนาดและจำนวนบรรทัดของคำสั่งไม่มากนัก จะมีขั้นตอนในการพัฒนาโปรแกรมดังนี้

- 1) ใช้โปรแกรม **Editor** ในการเขียนโปรแกรมต้นฉบับ
- 2) ใช้โปรแกรม **Compiler** ในการแปลภาษาต้นฉบับ
- 3) ใช้โปรแกรม **Linker** ในการรวมไฟล์หรือฟังก์ชันต่างๆ ที่จำเป็นเพื่อสร้างเป็นโปรแกรมที่สามารถทำงานได้หรือรันได้บนเครื่อง
- 4) สั่งให้โปรแกรมทำงาน ในกรณีที่ไม่มีข้อผิดพลาดเราจะได้โปรแกรมที่ต้องการ
- 5) ในกรณีมีข้อผิดพลาดก็จะต้องทำการแก้ไขโปรแกรม โดยอาจจะใช้โปรแกรม Debugger เข้าช่วย
- 6) หลังจากนั้นก็นำโปรแกรมไปใช้งานหรือเผยแพร่ต่อไป

ดังนั้นจะเห็นได้ว่าในการพัฒนาโปรแกรมจะต้องอาศัยโปรแกรมอีกหลายโปรแกรมเข้ามาช่วยในการทำงาน ซึ่งในปัจจุบันนี้ เพื่อให้ผู้พัฒนาโปรแกรมทำงานได้จ่ายขึ้น บริษัทผู้ผลิตซอฟต์แวร์ก็จะรวมเอาโปรแกรมต่างๆเหล่านี้ไว้ใน โปรแกรมหลัก โปรแกรมเดียว ซึ่งเรามักจะเรียกโปรแกรมหลักนี้ว่าเป็น โปรแกรมร่วมพัฒนา หรือ โปรแกรม IDE (Integrated Development Environment) สำหรับซอฟต์แวร์ Visual C++ 2008 Express Edition ก็ถือว่าเป็นโปรแกรมประเภท IDE ตัวหนึ่งซึ่งผลิตโดยบริษัท Microsoft และได้เผยแพร่ให้ใช้ฟรีเด็กวัยใส่เงินไปที่กำหนด

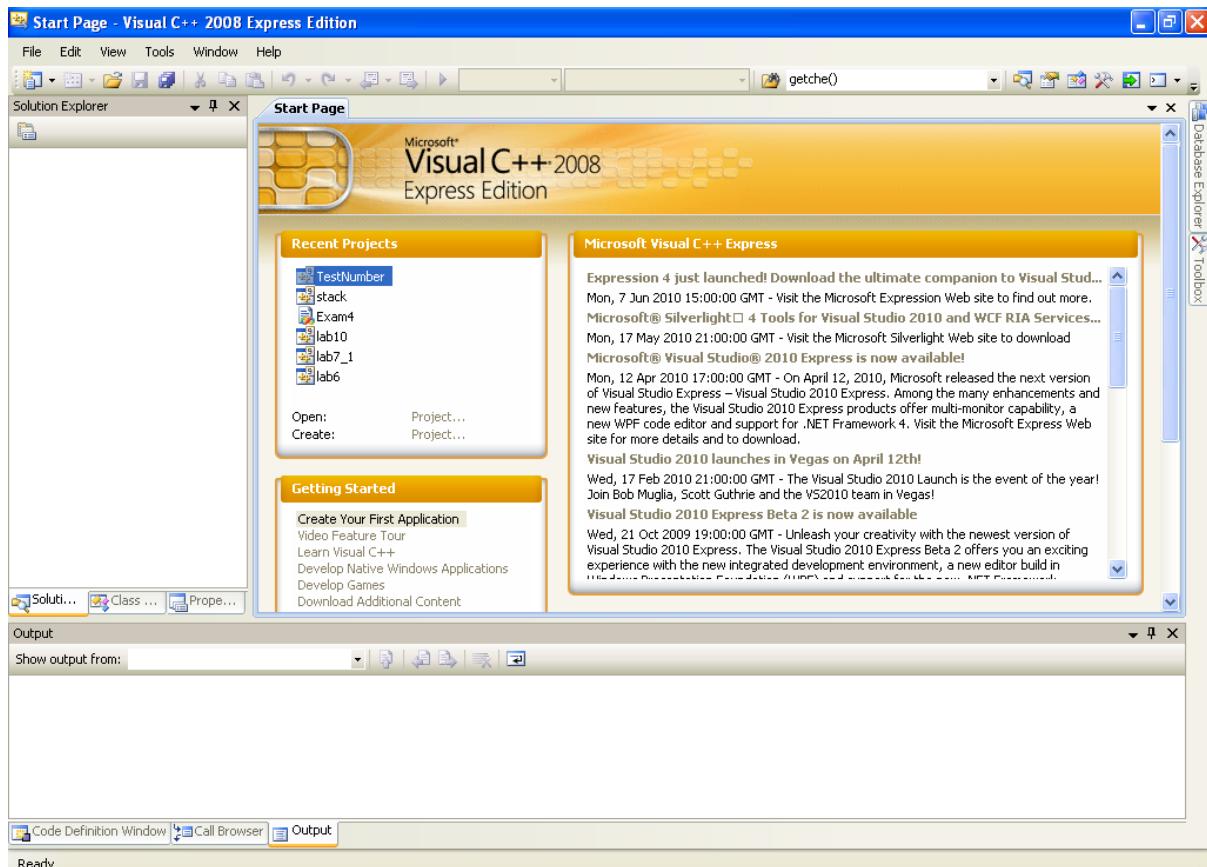


## การทดลองที่ 1.1

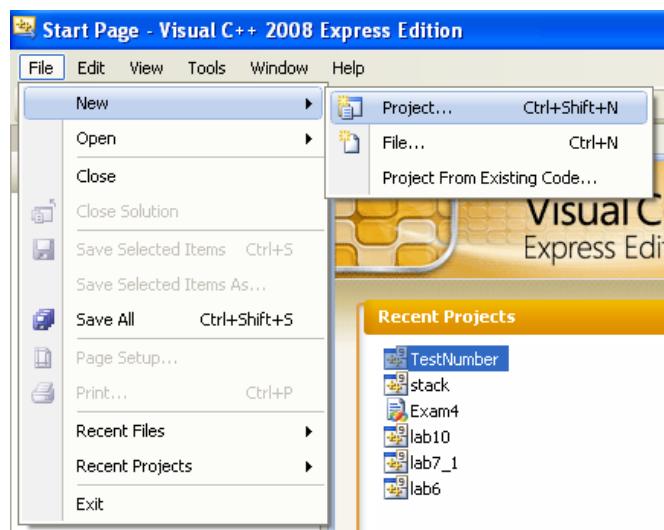
ทดลองใช้คอมพิวเตอร์ Visual C++ 2008 Express Edition เบื้องต้น

1. เปิดโปรแกรม Visual C++ 2008 โดยการ Double Click ที่ไอคอนของโปรแกรม Visual C++ หรือ กด

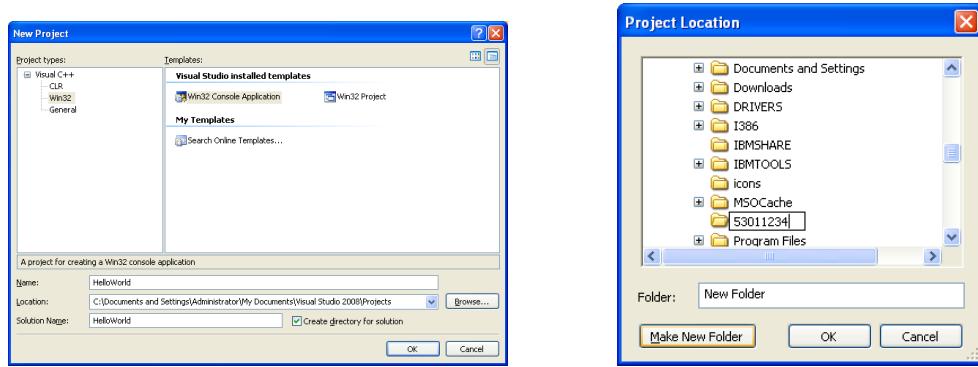
**Start -> All Program -> Visual C++ Express Edition -> Microsoft Visual C++ 2008 Express Edition**



2. สร้างโปรเจคเพื่อสำหรับเก็บโปรแกรมที่จะเขียนขึ้นในภายหลัง ในแต่ละโปรเจคอาจจะมีได้หลาย โปรแกรม ที่ เมนูบาร์ของ Visual C++ ให้กด File -> New -> Project

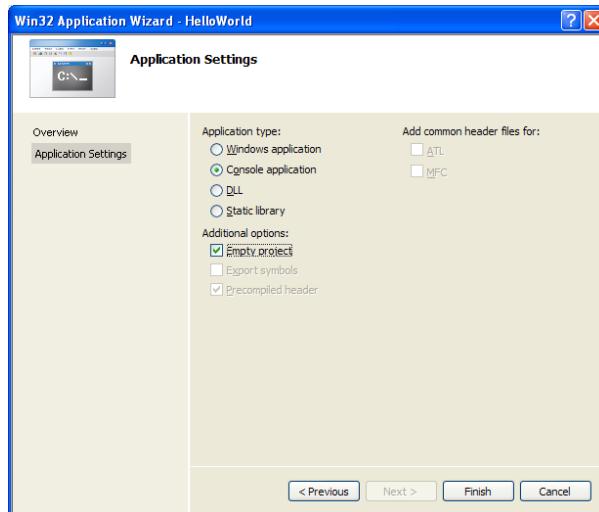


3. เลือกประเภทของโปรแกรมที่จะทำการสร้าง ในการทดลองนี้จะสร้างโปรแกรมที่จะรันบนหน้าต่างคอนโซล คือ ข้อมูลที่เป็น เอาท์พุตและอินพุตจะปรากฏอยู่ที่หน้าต่างนี้ (หน้าต่างที่เป็นสีดำ) ที่หน้าจอ New Project ให้เลือก Win32 -> Win32 Console Application ที่ Name ให้ตั้งชื่อโปรเจคว่า HelloWorld

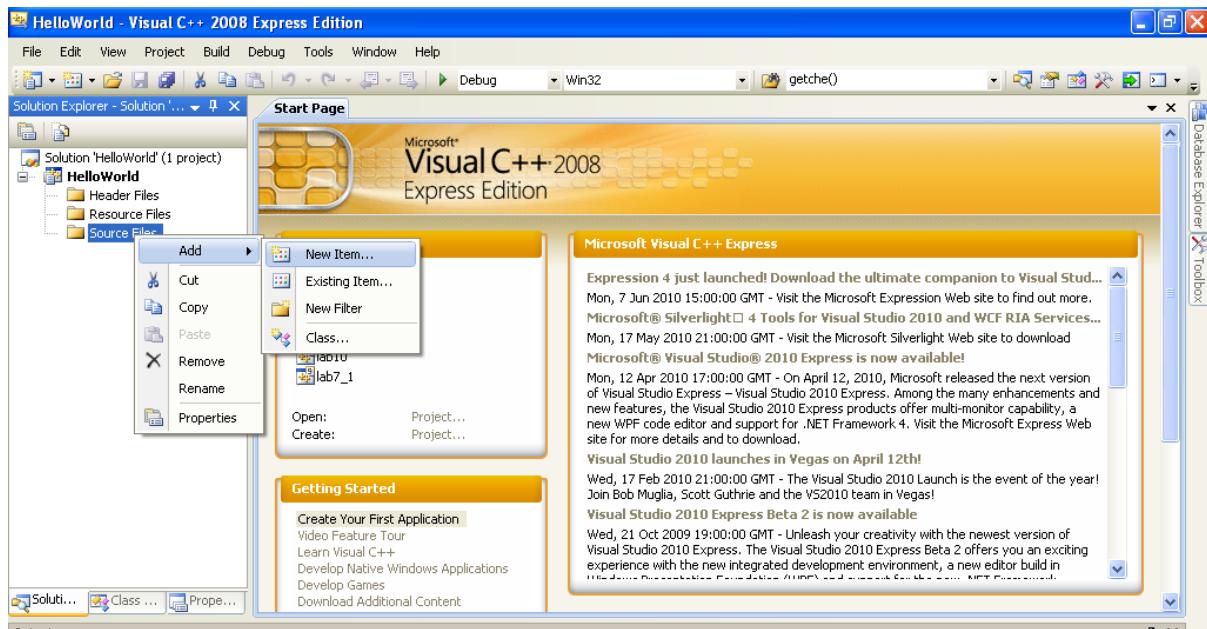


4. เลือกตำแหน่งที่จะเก็บโปรเจค ในการทดลองนี้ให้เก็บโปรเจคไว้ที่ไดร์ D: โดยที่ชื่อโฟลเดอร์ให้ใช้ รหัสนักศึกษา โดยการกด Browse -> เลือกไดร์ D: -> Make New Folder -> ตั้งชื่อโดยใช้รหัส นศ. ->OK -> Next

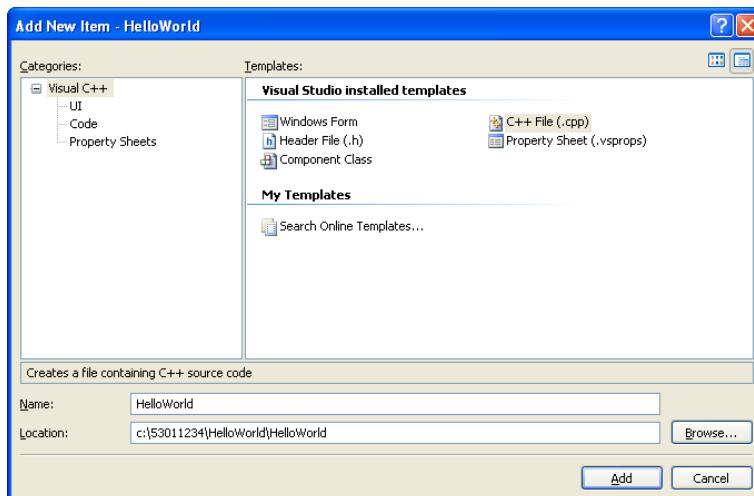
5. กำหนดรูปแบบของโปรแกรมที่จะทำการสร้าง ที่หน้าจอ Win32 Application Wizard ให้เลือก **Console Application** -> **Empty Project** -> **Finish**



6. ทำการสร้างโปรแกรมใหม่ ภายใต้โปรเจคที่ได้สร้างไว้แล้ว ที่ Solution Explorer ให้คลิกขวาที่ **Source File** -> **Add** -> **New Item**

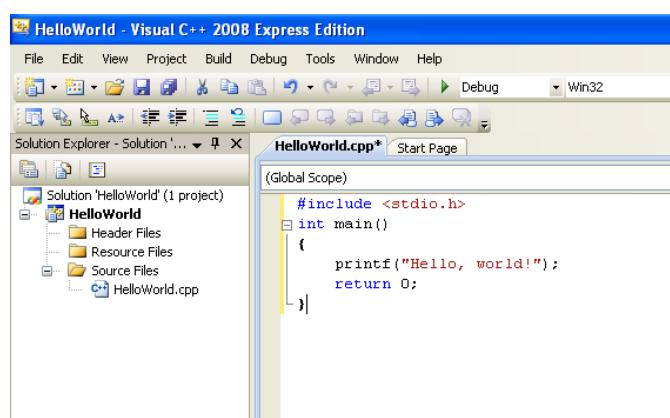


7. กำหนดรูปแบบของโปรแกรมที่จะทำการสร้าง ในการทดลองนี้ให้เลือก C++ File ที่ชื่อ Name ให้ตั้งชื่อโปรแกรม HelloWorld และกด Add

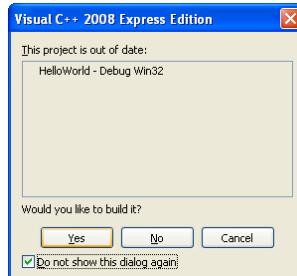


8. เขียนโปรแกรมตามตัวอย่างต่อไปนี้

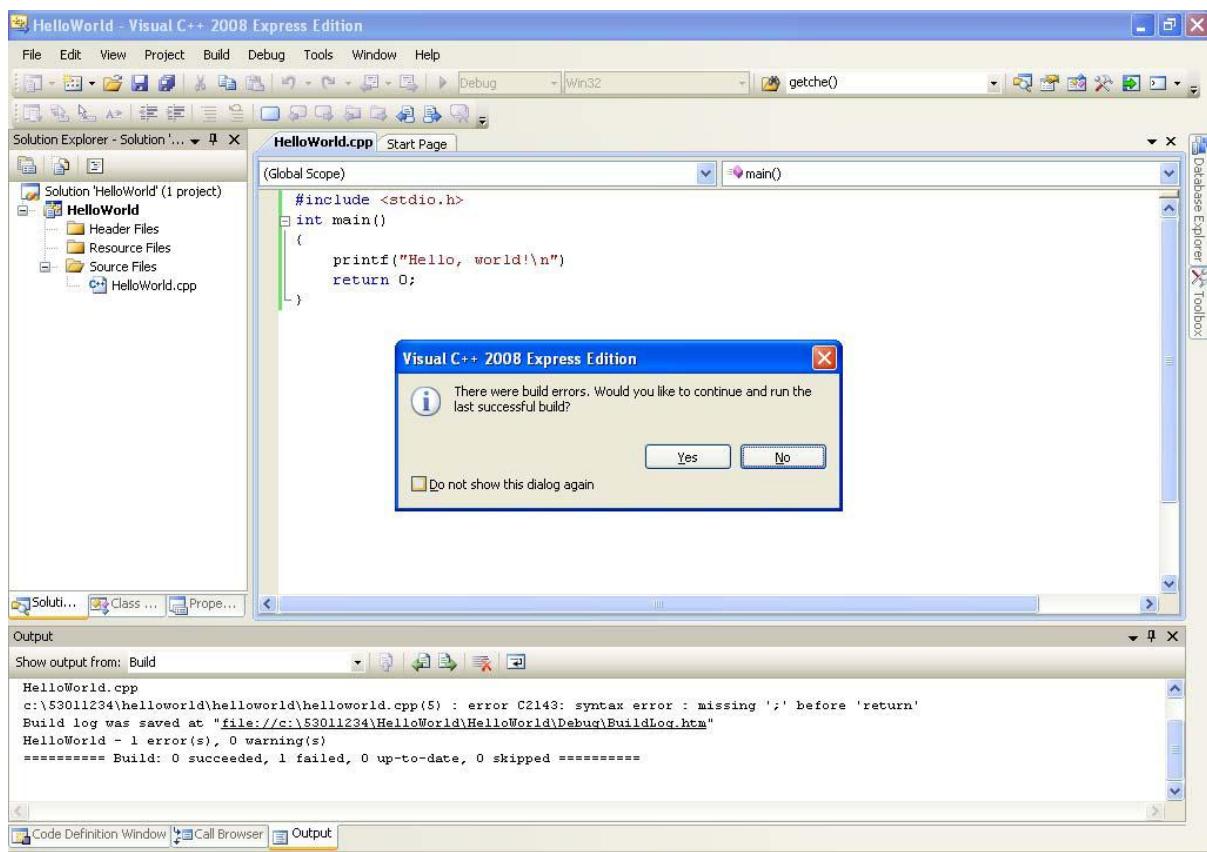
```
#include <stdio.h>
int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

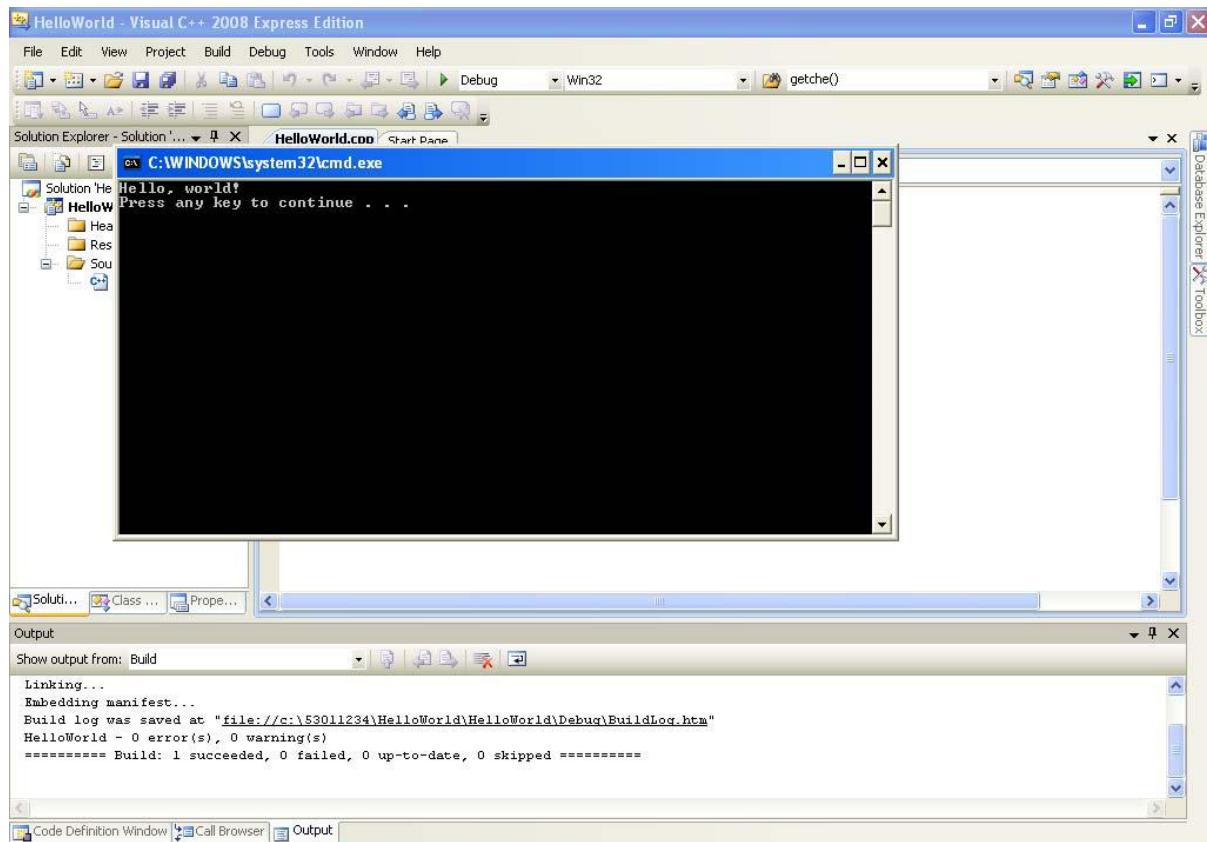


9. สั่งรันโปรแกรม ที่เมนูบาร์เลือก Debug -> Start Without Debugging หรือ Ctrl+F5 (ให้กดปุ่ม Ctrl และ F5 พร้อมๆ กัน) หลังจากนั้นให้ยืนยันการสร้างโปรแกรมโดยการกดปุ่ม Yes หากไม่ต้องการให้หน้าจอนี้ ปรากฏบ่อยๆ ให้เลือก Do not show this dialog again



10. ในกรณีที่โปรแกรมมีผิด จะมีข้อความบอกอยู่ด้านล่างว่าผิดที่ไหน กีต์ตำแหน่ง และที่หน้าจอจะขึ้นมาถามว่าจะให้นำเอาโปรแกรมเก่าที่เขียนถูกไว้แล้วมารันหรือไม่ ถ้าไม่ต้องการให้เลือก No หลังจากนั้นก็ทำการตรวจสอบว่าโปรแกรมเขียนผิดที่ไหน ให้ทำการแก้ไข แล้วรันโปรแกรมใหม่





## การทดลองที่ 1.2

ทดลองเขียนและรันโปรแกรมภาษา C อย่างง่าย

- ให้นักศึกษาสร้างโปรเจกใหม่ชื่อ Lab01 โดยให้เก็บไว้ที่ไดร์ฟ D: และสร้างโปรแกรมใหม่ชื่อ Lab01\_1.c แล้วป้อนโปรแกรมตามตัวอย่างด้านล่างนี้ใน Editor Screen

```
#include <stdio.h>
int main()
{
    printf("Hello");
    printf("Ladkrabang");
    return 0;
}
```

- ให้คอมไพล์โปรแกรมดังกล่าว โดยใช้ Ctrl+F5 แล้วตรวจสอบผลลัพธ์

- ให้ทำการแก้ไขโปรแกรมตามตัวอย่างด้านล่าง

```
#include <stdio.h>
int main()
{
    printf("Hello\n");
    printf("Ladkrabang");
    return 0;
}
```

ผลลัพธ์ของโปรแกรมต่างจากโปรแกรมในข้อที่ 1 อย่างไร

บันทึก \_\_\_\_\_

การที่จะแสดงผลให้อยู่ต่างบรรทัดกันต้องทำอย่างไร

---

4. ให้เปลี่ยนโปรแกรมในข้อ 3 โดยแทนที่ \t ด้วย \t และ Run โปรแกรมอีกครั้ง \t ทำให้ผลลัพธ์เปลี่ยนแปลงอย่างไร
- 

### การทดลองที่ 1.3

ทดลองหาที่ผิดและแก้ไขโปรแกรมเบื้องต้น

ให้นักศึกษาสร้างโปรแกรมใหม่ภายใต้โปรเจคเดิม ดังต่อไปนี้

1. ที่ Solution Explorer ที่ไฟล์เดิมชื่อ Lab01\_1 ให้คลิกขวาแล้วเลือก Remove
2. ที่ Solution Explorer -> Source File -> คลิกขวาเลือก New Item -> เลือก C++ File -> โดยใช้ชื่อ ไฟล์ว่า Lab01\_2.c และป้อนโปรแกรมตามด้านล่าง

```
#include <stdio.h>
int main()
{
    prntf("Bangkok");
    printf("Thailand");
    return 0;
}
```

3. ให้คอมไพล์โปรแกรมดังกล่าว โดยใช้เมนู Build->Compile โปรแกรมมีข้อผิดพลาดหรือไม่ \_\_\_\_\_  
ถ้าไม่มีข้อผิดพลาด เกิดข้อผิดพลาดในบรรทัดใด (เขียน Code ทั้งบรรทัด)
- 

ถ้ามีข้อผิดพลาด ผิดพลาดเพราะอะไร

---

ถ้ามีข้อผิดพลาด จะแก้ไขอย่างไร

---

ถ้ามีข้อผิดพลาด แก้ไขแล้วบรรทัดที่แก้ไขมี Code อย่างไรบ้าง (เขียน Code ทั้งบรรทัด)

---

4. สั่ง Run โปรแกรม

ผลลัพธ์ที่แสดงหลังโปรแกรมทำงานคือ

---

## การทดลองที่ 1.4

ทดลองกำหนดรูปแบบการพิมพ์ด้วย \n และ \t ของคำสั่ง printf

ให้นักศึกษาเขียนโปรแกรมแสดงข้อมูลของตัวนักศึกษาเอง ซึ่งประกอบด้วย รหัสนักศึกษา, เบอร์โทรศัพท์, ชื่อ-นามสกุล และ จังหวัดที่อยู่ โดยมีรูปแบบการแสดงผลดังต่อไปนี้ เก็บไว้ภายใต้โปรเจคเดิม คือ Lab01 ชื่อโปรแกรม Lab01\_4

ID : 50015541

phone : 0815555555

Name : Somsak Jaidee

province : Bangkok

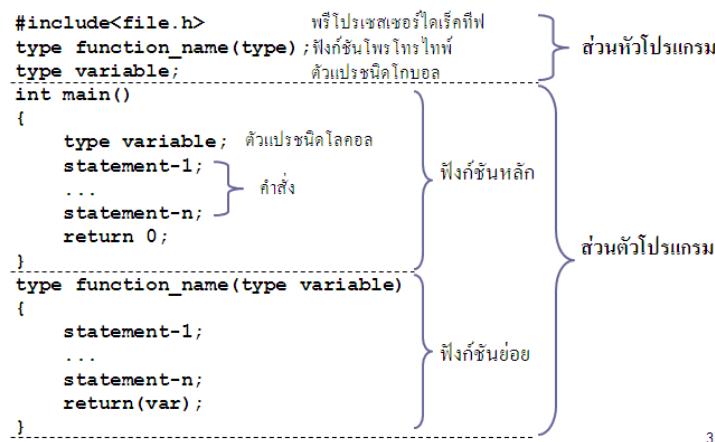
บันทึกโค๊ดของโปรแกรมที่นศ.เขียน

## บทที่ 2 การเขียนโปรแกรม คำสั่งแสดงผล

### วัตถุประสงค์การศึกษา

- 1) สามารถเขียนโปรแกรมภาษาซีอย่างง่าย ๆ ได้
- 2) สามารถแสดงข้อความอักขระทางจอภาพได้
- 3) สามารถแสดงข้อมูลชนิดต่างๆ ทางจอภาพได้
- 4) สามารถกำหนดรูปแบบการแสดงข้อมูลอักขระทางจอภาพได้

### 2.1 โครงสร้างภาษาซี



3

#### 1. ส่วนหัวโปรแกรม (Header Program)

- พาร์เชนเซอร์ไดเร็คทีฟ (Preprocessor Directive)

#include

#define

- พังก์ชันไฟร์โทร์ไทย (Function Prototype Declaration)
- ตัวแปรชนิดโภคอล (Global Variable Declaration)

#### 2. ส่วนตัวโปรแกรม (Body Program)

- พังก์ชันหลัก (Main Function)

ตัวแปรชนิดโลคอล (Local Variable Declaration)

คำสั่ง (Statement)

- พังก์ชันอื่น ๆ (Functions)

#### 3. คำอธิบายโปรแกรม (Comment Program)

### 2.2 พาร์เชนเซอร์ไดเร็คทีฟ (Preprocessor directive)

ส่วนประมวลผลก่อน เป็นส่วนที่สั่งให้คอมไพล์เรียบร้อยการทำงานที่กำหนดไว้ ก่อนที่จะทำงานในฟังก์ชันหลัก

#include <file-name.h>

ให้โปรแกรมดึงคำสั่งจาก file-name.h ใน Include Directory

```
#include "file-name.h"
```

ให้โปรแกรมดึงคำสั่งจาก file-name.h ใน Current Directory

file-name คือชื่อไฟล์นามสกุล h

ตัวอย่างคำสั่งพรีprocessor directive

#include<stdio.h> ทำการเรียกไฟล์ stdio.h เพื่อทำให้สามารถใช้คำสั่งที่ว่าไปได้

#include<conio.h> ทำการเรียกไฟล์ conio.h เพื่อทำให้สามารถใช้คำสั่งจัดการหน้าจอ และคำสั่งรับและแสดงผลเพิ่มเติม

#include<math.h> ทำการเรียกไฟล์ math.h เพื่อทำให้สามารถใช้คำสั่งเกี่ยวกับคณิตศาสตร์ได้

#include<string.h> ทำการเรียกไฟล์ string.h เพื่อทำให้สามารถใช้คำสั่งจัดการเกี่ยวกับข้อความได้

## ฟังก์ชันหลัก

ฟังก์ชันหลักเป็นฟังก์ชันที่โปรแกรมภาษาซีต้องมีอยู่เสมอ เพราะคอมไපเลอร์ของภาษาซีจะเริ่มต้นการทำงานจากฟังก์ชันหลัก ตัวอย่างการเขียนฟังก์ชันหลักที่ไม่มีการทำงานใดๆ

int main(void)	int main()
{	{
return (0);	return 0;

## กฎเกณฑ์การใช้คำสั่งในภาษาซี

- ใช้เครื่องหมาย semi colon ; เป็นจุดสิ้นสุดคำสั่ง
- ใช้อักษรตัวเล็กสำหรับเรียกใช้คำสั่ง (statement)
- ใช้เครื่องหมาย comma , สำหรับคั่นตัวแปร และพารามิเตอร์
- หากคำสั่งใดมีคำสั่งส่วนบุญภัยในหลาย ๆ คำสั่ง ให้ใช้เครื่องหมายปีกกา { } สำหรับกำหนดขอบเขต

## 2.3 การแสดงผลด้วยคำสั่ง printf

การเขียนโปรแกรมจำเป็นต้องมีการแสดงผล เพื่อให้ผู้ใช้งานทราบว่าโปรแกรมสามารถทำงานอะไร จำเป็นต้องป้อนค่าอะไรบ้าง และเมื่อโปรแกรมทำงานเสร็จ ผลลัพธ์ที่ได้เป็นอย่างไร

คำสั่งที่ใช้สำหรับการแสดงผลในภาษาซีมีหลายคำสั่ง แต่ที่สามารถใช้งานได้ครอบคลุมและนิยมใช้กัน คือ คำสั่ง

printf

รูปแบบคำสั่ง

```
printf (format-string, data-list);
```

- format-string คือรูปแบบของข้อความซึ่งจะประกอบด้วย ข้อความธรรมดา, ค่ารหัส ASCII และ ส่วนแสดงชนิดข้อมูล โดย format-string จะอยู่ในเครื่องหมาย Double quote "
- data-list คือชื่อข้อมูล หรือตัวแปรที่จะทำการแสดงผลตามส่วนแสดงชนิดข้อมูลใน format-string คำสั่ง printf ต้องเรียกใช้ Preprocessor Directive #include<stdio.h>

## โปรแกรม 2.1 การใช้งานคำสั่ง printf

```
#include <stdio.h> // เรียกว่า preprocessing directive ใช้ฟังก์ชันใน stdio.h
int main() // ประกาศฟังก์ชัน main
{
    //เริ่มต้นฟังก์ชัน
    printf ("Hello World!"); //ฟังก์ชัน printf
    return 0; //ส่งค่า 0 ออกจากฟังก์ชัน main
} //สิ้นสุดฟังก์ชัน
```

### รูปแบบของข้อความในคำสั่ง printf

1. ข้อความธรรมดา เป็นส่วนที่แสดงตัวอักษร ตัวเลขหรือโดยตรง
2. ค่ารหัส ASCII เป็นส่วนที่ควบคุมรูปแบบการแสดงผล เช่น การจัดย่อหน้า การขึ้นต้นบรรทัดใหม่ เป็นต้น
3. ส่วนแสดงชนิดข้อมูล เป็นการกำหนดรูปแบบของชนิดข้อมูลที่จะแสดงผลข้อมูล หรือตัวแปร

- ส่วนแสดงชนิดข้อมูลแบบปกติ
- ส่วนแสดงชนิดข้อมูลที่กำหนดรูปแบบการแสดงผล

การแสดงข้อความธรรมดาเป็นการใช้คำสั่ง printf ให้แสดงข้อความที่ต้องการอุทิ�าจากภาพ โดยจะอยู่ในส่วนของ format-string ในคำสั่ง (อยู่ในเครื่องหมาย Double quote " ")

printf (format-string, data-list);

เช่น คำสั่งในตัวอย่างโปรแกรมที่ผ่านมา

printf ("Hello World!");

จะทำการแสดงข้อความ Hello World! อุทิ�าจากภาพหลังจากสั่ง Run โปรแกรม

## โปรแกรม 2.2 การใช้งานคำสั่ง printf หลายคำสั่ง

#include<stdio.h> #include<conio.h> int main() {     printf ("Hello World!");     printf ("This is my first Program.");     printf ("I am a programmer.");     return 0; }	ผลลัพธ์ของโปรแกรมแสดงอุทิ�าจากภาพ
	Hello World!This is my first Program.I am a programmer

### ลักษณะพิเศษควบคุมการแสดงผล

อักษรพิเศษที่ใช้งานในภาษาซี ซึ่งใช้ควบคุมการแสดงผลในคำสั่ง printf ในส่วนของ format-string

ค่ารหัส ASCII	การใช้งาน
\t	แท็บช่องว่างทุก 8 ช่องตัวอักษร
\n	ขึ้นบรรทัดใหม่
\0	เป็นอักขระว่าง
\'	แสดง single quote อุทิ�าจากภาพ
\"	แสดง double quote อุทิ�าจากภาพ
\\\	แสดง backslash อุทิ�าจากภาพ

### โปรแกรม 2.3 การใช้งานคำสั่ง printf มีลักษณ์พิเศษควบคุมการแสดงผล

<pre>#include&lt;stdio.h&gt; int main() {     printf ("Hello World!\n");     printf ("This is my first Program.");     printf ("\n\tI'm a programmer.");     return 0; }</pre>	ผลลัพธ์ของโปรแกรมแสดงออกจอภาพ  Hello World! This is my first Program. I'm a programmer
--	--

#### ส่วนแสดงชนิดข้อมูล

การเขียนโปรแกรมที่มีการประยุกต์ขึ้น เช่นโปรแกรมคำนวณเลข จะต้องมีการแสดงผลลัพธ์ของค่าที่ทำการคำนวณ ซึ่งคำสั่ง printf สามารถแสดงผลค่าตัวแปรได้ แต่จะต้องเขียนโปรแกรมให้มีความสัมพันธ์กันในส่วนของ format-string และ data-list

- โดยที่ใน format-string จะมีส่วนแสดงชนิดข้อมูล
- และใน data-list จะมีข้อมูล หรือตัวแปรที่จะแสดงผล

ส่วนแสดงชนิดข้อมูล	การใช้งาน
%d	แสดงผลข้อมูลชนิดจำนวนเต็ม
%n	แสดงผลข้อมูลชนิดจำนวนเต็มบวก (ไม่คิดเครื่องหมาย)
%o	แสดงผลข้อมูลเป็นเลขฐานแปด
%x	แสดงผลข้อมูลเป็นเลขฐานสิบหก
%f	แสดงผลข้อมูลชนิดจำนวนหนึ่ง (6 ตำแหน่ง)
%e	แสดงผลข้อมูลเป็นจำนวนทางคณิตและอยู่ในรูปยกกำลัง
%c	แสดงผลข้อมูลชนิดอักขระ
%s	แสดงผลข้อมูลชนิดข้อความ
%p	แสดงผลข้อมูลชนิดคัวชี้ค่า变量

20

### โปรแกรม 2.4 printf มีลักษณ์พิเศษควบคุมการแสดงผล

<pre>#include&lt;stdio.h&gt; int main() {     printf ("My name is : %s\n", "Kmitl");     printf ("My point : %d\n", 10+40+49);     printf ("Grade : %c\n", 'A');     printf ("GPA : %f", 3.99);     return 0; }</pre>	ผลลัพธ์ของโปรแกรมแสดงออกจอภาพ  My name is : Kmitl My point : 99 Grade : A GPA : 3.990000
---	--

#### การกำหนดรายชื่อตัวแปรในคำสั่ง printf

- การแสดงผลตัวแปรโดยใช้คำสั่ง printf สามารถใช้งานโดยการใส่ข้อมูล หรือชื่อตัวแปรในส่วนของ data-list
- หากไม่ต้องการแสดงผลตัวแปร ไม่ต้องมีส่วนของ data-list

- หากต้องการแสดงผลข้อมูล หรือตัวแปรมากกว่าหนึ่งตัวในคำสั่งให้ใช้ เครื่องหมาย comma , สำหรับคันชี้อีกตัวแปร โดยจะทำการแสดงผลตามลำดับตัวแปร และส่วนการแสดงชนิดข้อมูล

```
printf ("%? %? ... %?", data-1, data-2, ..., data-n);
```

**data-1 data-2 ... data-n**

#include<stdio.h>  int main() { printf ("Age = %d, GPA = %f\n",17,3.75); printf ("Programming: %f\nMechanics: %f",4.0,3.5); return 0; }	ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ  Age = 17, GPA = 3.750000 Programming: 4.000000 Mechanics: 3.500000
--	--

#include<stdio.h>  int main() { printf ("Subject : %s(%d)\n","Programming",2552); printf ("Point : %d\nGrade : %c",99,'A'); return 0; }	ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ  Subject : Programming(2552) Point : 99 Grade : A
--	--

## 2.4 แสดงชนิดข้อมูลที่กำหนดรูปแบบการแสดงผล

```
printf ("%m?", data);  
printf ("% -m?", data);
```

เป็นการจงพื้นที่หน้าจอจำนวน m ตัวอักษร และแสดงผล data ชิดด้านขวาของพื้นที่ส่วนที่จงไว้ โดยชนิดข้อมูลตาม ?  
(หากความยาวเกินส่วนที่จงไว้ ก็จะเลื่อนออกไป)

เป็นการจงพื้นที่หน้าจอขนาด m ตัวอักษร และแสดงผล data ชิดด้านซ้ายของพื้นที่ส่วนที่จงไว้ โดยชนิดข้อมูลตาม ?

<pre>#include&lt;stdio.h&gt; int main() {     printf ("123456789012345678901234567890");     printf ("\n%20d*", 46);     printf ("\n%-20d*", 46);     printf ("\n%3d*", 46);     printf ("\n%3d*", 2550);     return 0; }</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ</p> <pre>123456789012345678901234567890                                 46*                                 * 46         46* 2550*</pre>
---	--

<pre>#include &lt;stdio.h&gt; int main() {     printf ("123456789012345678901234567890");     printf ("\n%20c*", 'c');     printf ("\n%-20c*", 'c');     printf ("\n%10s*", "Pro");     printf ("\n%10s*", "Programming");     return 0; }</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ</p> <pre>123456789012345678901234567890                                 c*                                 * c         Pro* Programming*</pre>
--	--

แสดงชนิดข้อมูลที่กำหนดคุณรูปแบบการแสดงผล

```
printf("%n?", data);
printf("%m.n?", data);
```

เป็นการกำหนดให้แสดงจำนวนทศนิยม n ตำแหน่ง สำหรับ %f หรือแสดงอักขระจำนวน n ตัว สำหรับ %s

เป็นการของพื้นที่หน้าของนัด m ตัวอักษรแล้วแสดงผล data จำนวนทศนิยม n ตำแหน่ง สำหรับ %f หรือแสดงอักขระจำนวน n ตัว สำหรับ %s

<pre>//มีคำสั่งอื่นๆก่อนหน้านี้ printf ("123456789012345678901234567890"); printf ("\n%20s*", "programming"); printf ("\n%-20s*", "programming"); printf ("\n%.3s*", "programming"); printf ("\n%20.3s*", "programming"); printf ("\n%-20.3s*", "programming"); //มีคำสั่งอื่นๆหลังจากนี้</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ</p> <pre>123456789012345678901234567890                                 programming*                                 * programming                                 * pro*                                 pro*                                 * pro</pre>
---	--

<pre>//มีคำสั่งอื่นๆก่อนหน้านี้ printf ("123456789012345678901234567890"); printf ("\n%20f*", 1234.56789); printf ("\n%-20f*", 1234.56789); printf ("\n%.3f*", 1234.56789); printf ("\n%20.3f*", 1234.56789); printf ("\n%-20.3f*", 1234.56789); //มีคำสั่งอื่นๆหลังจากนี้</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากภาพ</p> <pre>123456789012345678901234567890                                 1234.567890*                                 * 1234.567890                                 * 1234.568*                                 1234.568*                                 * 1234.568</pre>
--	---

## 2.5 คำอธิบายโปรแกรม

คำอธิบายโปรแกรมเป็นส่วนที่เพิ่มในโปรแกรมเพื่อช่วยให้ผู้ที่เขียน หรือผู้ที่อ่านโปรแกรมสามารถเข้าใจกับตัวโปรแกรมได้ง่ายขึ้น

โดยที่คำอธิบายโปรแกรมนี้จะไม่มีผลต่อการทำงานของโปรแกรม คือเวลาที่ทำการคอมไพล์ จะข้ามชื่อความที่อยู่ในส่วนอธิบายโปรแกรมไป  
รูปแบบคำสั่ง

```
/* comment style1
 * สำหรับข้อความหลายบรรทัด */

//comment style2 ข้อความหนึ่งบรรทัด
```

### โปรแกรม 2.4 แสดงรายได้ต่อเดือน

โปรแกรมแสดงผล ชื่อ-นามสกุล อายุ

#include <stdio.h> int main() {     return 0; }	ผลลัพธ์ของโปรแกรมแสดงออกจากการ Name      : Kmitl Surname   : Engineering Age       : 46
---	---

### โปรแกรม 2.5 แสดงรายได้ต่อเดือน

โปรแกรมแสดงรายได้ต่อเดือนแล้วแสดงว่าต่อสัปดาห์คิดเป็นเท่าใด (ทศนิยม 2 ตำแหน่ง) ต่อวันคิดเป็นเท่าใด (ทศนิยม 3 ตำแหน่ง)

#include <stdio.h> int main() {     return 0; }	ผลลัพธ์ของโปรแกรมแสดงออกจากการ Salary     : 6500 Money/Week : 1625.00 Money/Day  : 232.143
---	--

## โปรแกรม 2.6 โปรแกรมแสดงเกรด

โปรแกรมแสดงผล เกรดวิชา Programming, Drawing, Mechanics, Math1 พร้อมแสดงเกรดเฉลี่ย (ทศนิยม 2 ตำแหน่ง)

<pre>#include &lt;stdio.h&gt; int main() {     Programming : A     Drawing      : B+     Mechanics    : B+     Math         : B     GPS          : 3.50      return 0; }</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากการ</p> <p>Programming : A Drawing : B+ Mechanics : B+ Math : B GPS : 3.50</p>
--	--

## 2.5 คำความท้ายบท

1) แสดงผลของโปรแกรมต่อไปนี้

<pre>#include&lt;stdio.h&gt; int main() {     /* float radius, pi, area;     pi = 22/7; // pi = 3.14;     printf ("Enter Radius of Circular : ");     scanf ("%f",&amp;radius);     area = pi * radius * radius; */     printf ("Programming Language",2008);     return 0; }</pre>	<p>ผลลัพธ์ของโปรแกรมแสดงออกจากการ</p> <hr/> <hr/> <p>เหตุใดจึงเป็นเช่นนั้น _____</p> <hr/>
---	--

2) จงเขียนส่วนของโปรแกรมให้แสดงผลดังต่อไปนี้

12\06\2008  
King Mongkut's Institute of Technology Ladkrabang  
Bangkok, Thailand.

3) จงเขียนส่วนของโปรแกรมให้แสดงผลตามเงื่อนไขต่อไปนี้

บรรทัดที่ 1 แสดง รหัสประจำตัว นศ.

บรรทัดที่ 2 แสดง ชื่อ นามสกุล

บรรทัดที่ 3 แสดง วัน/เดือน/ปี เกิด

บรรทัดที่ 8 แสดงข้อความ <http://www.ce.kmitl.ac.th> ขิดชอบของบรรทัด

บรรทัดที่ 10 แสดงข้อความ bye bye ทรงกลางบรรทัด

## การทดลองที่ 2 การเขียนโปรแกรมเบื้องต้นและคำสั่งแสดงผล

### วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมได้ตรงตามโครงสร้างของภาษา
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมเพื่อแสดงข้อมูลในหน้าจอได้ตรงตามความต้องการ

### ทฤษฎี

ในการเขียนโปรแกรมทุก ๆ ภาษา โปรแกรมเมอร์จำเป็นต้องเขียนโปรแกรมให้ตรงตามโครงสร้างของภาษานั้นๆ ไม่ว่าแม้แต่ในภาษา C ดังนั้นนักศึกษาจึงควรศึกษาโครงสร้างภาษา C ในแต่ละส่วนโดยละเอียดก่อน สำหรับโครงสร้างภาษา C ประกอบด้วยองค์ประกอบต่างๆ ดังตัวอย่างต่อไปนี้

```
#include<file.h>           // พาร์เพรสเซอร์ไดเรคทีฟ
type function_name(type);   // การประกาศฟังก์ชันโพร์โตไทร์
type variable;             // การประกาศตัวแปร global
int main()                 // การประกาศฟังก์ชันหลัก
{
    type variable;         // วงศ์บีกกาเปิด สำหรับฟังก์ชันหลัก
    statement-1;
    ...
    statement-n;
    return 0;              // การส่งค่ากลับของฟังก์ชัน
}
                                         // วงศ์บีกกาปิด สำหรับฟังก์ชันหลัก
type function_name(type variable) // ฟังก์ชันย่อย
{
    statement-1;
    ...
    statement-n;
    return (var);
}
```

อธิบายส่วนประกอบต่างๆ ดังนี้

- 1) ในส่วนของพาร์เพรสเซอร์ไดเรคทีฟ เป็นส่วนที่โปรแกรมเมอร์จะต้องให้ข้อมูลกับคอมไพล์เลอร์ ว่าจะใช้ฟังก์ชันในกลุ่มของไฟล์ .h ไฟล์ไหนบ้าง ซึ่งโปรแกรมเมอร์ต้องทราบว่าฟังก์ชันที่ตนใช้ จะต้องใช้พาร์เพรสเซอร์ไดเรคทีฟตัวใด เช่น ถ้าในโปรแกรมมีคำสั่ง printf จะต้องมีการเรียกใช้งาน stdio.h ซึ่งโปรแกรมจะไม่สามารถ compile ได้ถ้าเราไม่ได้กำหนด #include <stdio.h> ดังนั้นการกำหนดพาร์เพรสเซอร์ไดเรคทีฟ ถือว่าเป็นส่วนที่จำเป็น ถ้ามีการเรียกใช้งานคำสั่งใดๆ ต้องมีการกำหนดพาร์เพรสเซอร์ไดเรคทีฟทุกครั้ง
- 2) การประกาศฟังก์ชันโพร์โตไทร์ จะมีการกำหนดก็ต่อเมื่อมีการเขียนโปรแกรมแบ่งเป็นฟังก์ชันย่อย และมีการประกาศฟังก์ชันย่อยหลังการประกาศฟังก์ชันหลักเท่านั้น ถ้าไม่มีฟังก์ชันย่อยก็ไม่จำเป็นต้องประกาศฟังก์ชันโพร์โตไทร์

- 3) การประกาศตัวแปร global จะมีการกำหนดกีต่อเมื่อมีการเขียนโปรแกรมที่ต้องเรียกใช้ตัวแปร global ถ้าไม่มีการเรียกใช้ตัวแปร global ก็ไม่จำเป็นต้องประกาศตัวแปรในบรรทัดดังกล่าว
- 4) การประกาศฟังก์ชันหลัก ถือเป็นส่วนสำคัญทุกโปรแกรมต้องมี
- 5) วงเล็บปีกกาเปิดสำหรับฟังก์ชันหลัก ถือเป็นส่วนสำคัญทุกโปรแกรมต้องมี
- 6) การ return ค่าของฟังก์ชัน ถือเป็นส่วนสำคัญ ทุกโปรแกรมต้องมี ถ้ามีการประกาศ type ของฟังก์ชันหลัก
- 7) วงเล็บปีกกาปิดสำหรับฟังก์ชันหลัก ถือเป็นส่วนสำคัญ ทุกโปรแกรมต้องมี
- 8) ฟังก์ชันย่อย เป็นส่วนที่ไม่จำเป็นต้องใช้สำหรับผู้เริ่มต้นเขียนโปรแกรม

นักศึกษาจะเห็นว่าโดยตัวโครงสร้างของภาษา C มีรายละเอียดมาก เนื่องจากผู้ออกแบบได้ออกแบบไว้สำหรับการใช้งานที่หลากหลาย สำหรับผู้เริ่มเขียนโปรแกรม จะเริ่มจากการเขียนโปรแกรมอย่างง่าย มีขนาดเล็ก และไม่จำเป็นต้องมีองค์ประกอบของโครงสร้างภาษา C ทุก ๆ องค์ประกอบ แต่จะต้องมีข้อมูลองค์ประกอบหลักเพียงไม่กี่ตัวเท่านั้น

การเขียนโปรแกรมภาษา C ที่สามารถ compile ได้ตามโครงสร้างภาษานั้น ใช้องค์ประกอบเพียงสี่องค์ประกอบเท่านั้น ดังต่อไปนี้

```
int main(void)
{
    return 0;
}
```

ถ้านักศึกษาลองนำโค้ดดังกล่าวมา compile จะพบว่าเราสามารถ compile โค้ดนี้ได้ เนื่องจากมีองค์ประกอบหลักครบถ้วนตามโครงสร้างภาษาแต่จะไม่สามารถรันให้เกิดผลลัพธ์ใด ๆ ได้ เนื่องจากโปรแกรมไม่มีการเรียกคำสั่งใดๆ เลย ถ้าต้องการให้โปรแกรมทำงานอย่างโดยอ้างหนึ่ง จะต้องเพิ่มคำสั่ง หรือฟังก์ชันภายในเครื่องหมายวงเล็บปีกกา เช่น ถ้านักศึกษาต้องการให้คอมพิวเตอร์แสดงคำว่า Engineer นักศึกษาต้องเพิ่มคำสั่งหนึ่งคำสั่งในเครื่องหมายวงเล็บปีกกา และเพิ่มพรีโพรเซสเซอร์ไดเรคทีฟของคำสั่นนี้ ด้วยจะได้ผลลัพธ์ดังนี้

```
#include <stdio.h>
int main()
{
    printf("Engineer");
    return 0;
}
```

สาเหตุที่ต้องเพิ่มพรีโพรเซสเซอร์ไดเรคทีฟด้วยเนื่องจาก คอมไฟล์เลอร์จะต้องทำการเปลี่ยนคำสั่ง printf ให้กลายเป็นไบนาเรียวโค้ด และต้องใช้รายละเอียดในไฟล์ชื่อ stdio.h ในการคอมไฟล์โปรแกรม ถ้าไม่มีพรีโพรเซสเซอร์ ไดเรคทีฟจะทำให้คอมไฟล์ไม่สามารถทราบได้ว่าต้องเปลี่ยนเป็นไบนาเรียวโค้ดอย่างไร ส่งผลให้โปรแกรมคอมไฟล์ไม่ผ่าน โดยโปรแกรมจะแจ้งว่าไม่รู้จักคำสั่ง printf

ฟังก์ชัน printf

รูปแบบคำสั่ง printf (format string, data list);

รายละเอียดการใช้งานคำสั่ง

printf เป็นคำสั่งที่ใช้เพื่อแสดงผลออกที่หน้าจอ Output โดยภายในคำสั่งจะประกอบด้วย 2 ส่วนได้แก่

1. format string : ส่วนที่แสดงผลซึ่งจะอยู่ในเครื่องหมาย " " (Double quote) โดยจะมี 4 ลักษณะได้แก่

- 1) ข้อความหรือตัวอักษรธรรมดា : ข้อความหรือตัวอักษรทั่วไปรวมทั้งช่องว่าง (space) ที่ต้องการให้แสดงผลที่ Output
- 2) รหัสควบคุม (Escape Sequence) : ใช้สำหรับควบคุมการแสดงผลที่หน้าจอหรือเครื่องพิมพ์ เช่น การจัดย่อหน้า การจัดบรรทัด ฯลฯ รหัสควบคุมนี้เป็นต้นด้วยเครื่องหมาย \ (backslash) และตามด้วยรหัสควบคุม ตามทัวร์อย่างต่อไปนี้
  - \n : กำหนดให้ขึ้นบรรทัดใหม่
  - \t : กำหนดให้เว้นไป 1 แท็บແນวนอน (8 อักษร)
  - \v : กำหนดให้เว้นไป 1 แท็บແນວตั้ง
  - \b : ถอยหลังหนึ่งตัวอักษร (back space)
  - \a : ส่งเสียงบีป 1 ครั้ง
  - \f : ขึ้นหน้าใหม่ (form feed)
  - \r : ย้ายเคอร์เซอร์กลับไปที่ต้นบรรทัด
  - \' : พิมพ์เครื่องหมาย ‘
  - \” : พิมพ์เครื่องหมาย “
  - \\\ : พิมพ์เครื่องหมาย \
  - \0 : ค่าว่าง (null)
- 3) รหัสรูปแบบข้อมูล : กำหนดชนิดข้อมูล ซึ่งอาจจะเป็นตัวแปร นิพจน์ หรือค่าคงที่ ซึ่งจะสัมพันธ์กับส่วน data list ที่อยู่ในคำสั่ง โดยที่รหัสรูปแบบข้อมูลจะขึ้นต้นด้วยเครื่องหมาย % และตามด้วยตัวอักษรที่บ่งบอกถึง ชนิดของข้อมูล ตัวอย่างเช่น
  - %d : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนเต็ม
  - %f : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนจริง
  - %c : ชนิดข้อมูลที่พิมพ์เป็นตัวอักษรหนึ่งตัว
  - %s : ชนิดข้อมูลที่พิมพ์เป็นข้อความสตริง
  - %ld : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนเต็ม แบบบайต
  - %g : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนจริง
  - %e : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนจริง แบบ exponent
  - %i : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนเต็ม เช่นเดียวกับ %d
  - %o : ชนิดข้อมูลที่พิมพ์เป็นเลขฐานแปด
  - %x : ชนิดข้อมูลที่พิมพ์เป็นเลขฐานสิบหก
  - %u : ชนิดข้อมูลที่พิมพ์เป็นเลขจำนวนเต็ม ไม่มีเครื่องหมาย
  - %p : ชนิดข้อมูลที่พิมพ์เป็นตัวชี้คำແเนง
- 4) รูปแบบการแสดงผลอื่นๆ
  - printf ("%m?",data); เป็นการจ่องพื้นที่หน้าจอจำนวน m ตัวอักษร เลี้ยวแสดงผล data ชิดด้านขวาของพื้นที่ส่วนที่จ่องไว้ โดยชนิดข้อมูลตาม ? (หากความยาวเกินส่วนที่จ่องไว้ก็จะเลื่อนออกไป) เช่น ถ้าต้องการแสดงผลคำว่า Engineer ชิดขวาเมื่อของจากภาพ (จากภาพมีความกว้าง 80 ตัวอักษร) สามารถใช้คำสั่ง printf("%80s","Engineer"); ได้

- `printf ("%m?",data);` เป็นการจ่องพื้นที่หน้าจอขนาด m ตัวอักษร และแสดงผล data ชิดด้านซ้ายของพื้นที่ส่วนที่จงไว้ โดยชนิดข้อมูลตาม ?
- `printf ("%n?",data);` เป็นการกำหนดให้แสดงจำนวนทศนิยม n ตำแหน่ง สำหรับ %f หรือแสดงอักขระจำนวน n ตัว สำหรับ %s
- `printf ("%m.n?",data);` เป็นการจ่องพื้นที่หน้าจอขนาด m ตัวอักษรแล้วแสดงผล data จำนวนทศนิยม n ตำแหน่ง สำหรับ %f หรือแสดงยักษรจำนวน n ตัว สำหรับ %s

2. data list : ส่วนที่เป็นตัวแปร นิพจน์ หรือค่าคงที่ต่างๆ ที่คำสั่ง printf จะนำไปใช้แสดงผลตามรูปแบบที่กำหนดในส่วนของ format string โดยถ้าหากมีข้อมูลที่ต้องการแสดงผลหลายตัวให้ใช้เครื่องหมาย , (comma) คั่นข้อมูลแต่ละตัว ชนิดของข้อมูลตัวที่ 1 จะต้องตรงกับรหัสข้อมูลที่อยู่ใน format string ตัวที่ 1



## การทดลองที่ 2.2 การใช้งาน %s %d %c %f เพื่อแสดงผลข้อมูลปกติ

" data-bbox="161 119 833 372"/>

บันทึกผล ถ้าผลลัพธ์ที่ได้มีมากกว่า 30 ตำแหน่ง ให้บันทึกเฉพาะ 30 ตำแหน่งแรก


2.2.1 การแสดงผลตัวอักษร ใช้สัญลักษณ์พิเศษอะไร \_\_\_\_\_

2.2.2 การแสดงผลจำนวนเต็ม ใช้สัญลักษณ์พิเศษอะไร \_\_\_\_\_

2.2.3 การแสดงผลจำนวนทศนิยม ใช้สัญลักษณ์พิเศษอะไร \_\_\_\_\_

2.2.4 การแสดงตัวผลข้อความ ใช้สัญลักษณ์พิเศษอะไร \_\_\_\_\_

2.2.5 การแสดงผลจำนวนทศนิยม แสดงทศนิยมกี่ตำแหน่ง \_\_\_\_\_







3. (จากข้อ 1) ถ้าต้องการให้คำว่า Engineer ปรากฏอยู่ตรงกลางของจอภาพจะต้อง เปลี่ยนแปลงคำสั่ง printf("Engineer"); เป็น printf("%44s","Engineer");

Q. การจองพื้นที่ 44 ตัวอักษร สัมพันธ์อย่างไรกับจำนวนตัวอักษรของคำว่า "Engineer" กับความกว้างของหน้าจอ

---



---



---

### การทดลองที่ 2.7

ให้นักศึกษาเขียนโปรแกรมโดยให้แสดงคำว่า Computers and Programming ปรากฏอยู่กลางจอภาพ จดคำสั่งที่ใช้

---



---



---

### การทดลองที่ 2.8

ให้นักศึกษาลองเขียนโปรแกรมต่อไปนี้ลงสังเกตผลลัพธ์ขณะ compile โปรแกรม หรือรันโปรแกรม แล้วตอบคำถาม

1. ให้นักศึกษาทดลองเขียนโปรแกรมในตัวอย่างการใช้งานข้อ 1 โดยตัดบรรทัด #include <stdio.h> ออกเพื่อทดสอบว่าจะเกิดอะไรขึ้นถ้ามีการเรียกใช้ฟังก์ชันโดยไม่มีการกำหนดพารามิเตอร์ใดๆ เช่น

Q: จะเกิดข้อผิดพลาดอะไร

---



---

2. จากโปรแกรมในตัวอย่างการใช้งานคำสั่ง ข้อ 1 ถ้าไม่มีบรรทัด return 0; จะเกิดข้อผิดพลาดอย่างไร

---



---

3. เมื่อหน้าจอ มีความกว้าง 80 ตัวอักษร จะเกิดอะไรขึ้นเมื่อใช้คำสั่ง printf ("%82s", "engineer");

---



---

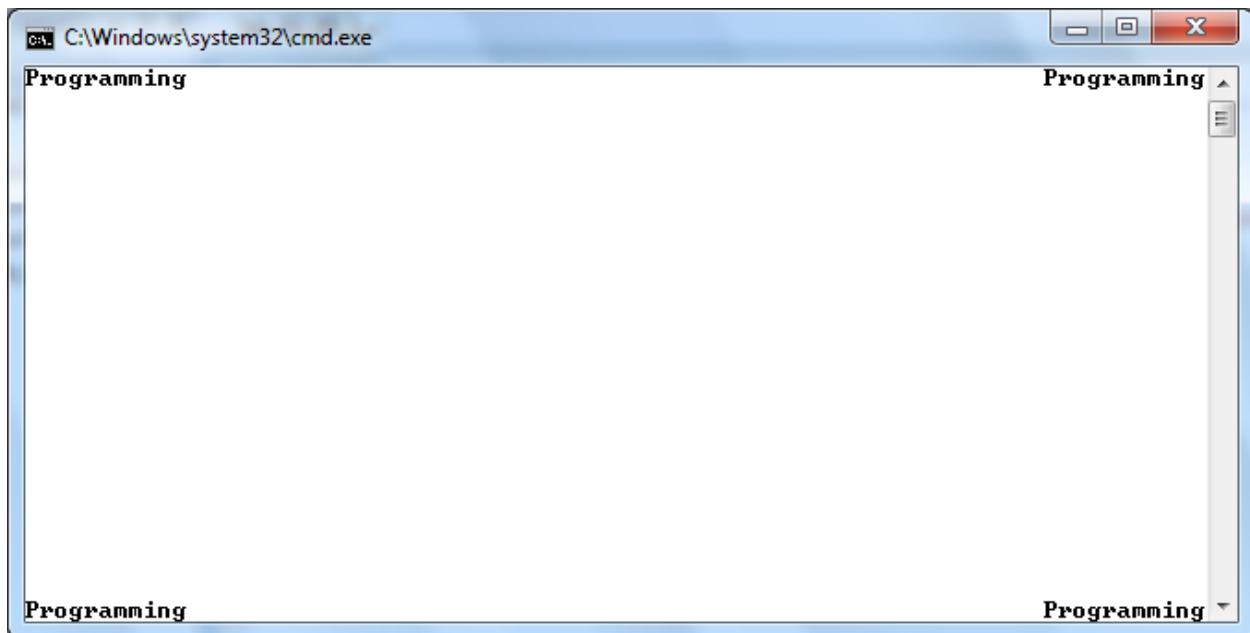
## การทดลองที่ 2.9

ให้นักศึกษาเขียนโปรแกรมต่อไปนี้

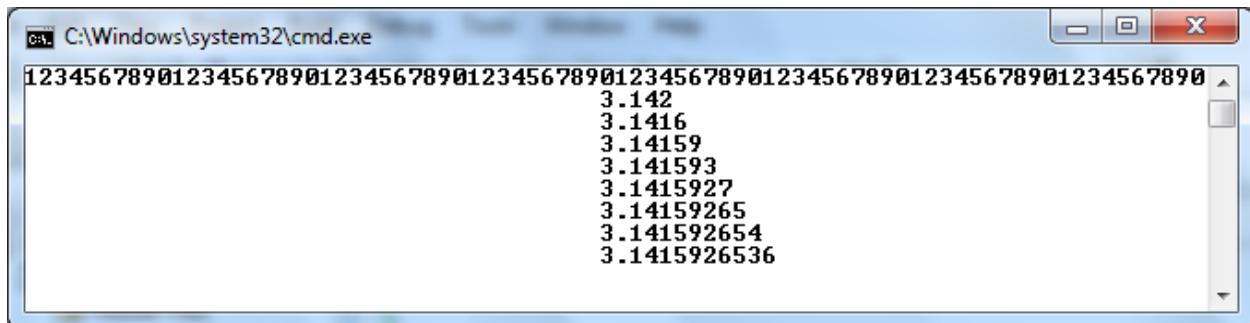
- ให้นักศึกษาเขียนโปรแกรมเพื่อแสดงผลลัพธ์ Computer อยู่ข้างของจอภาพ และ Programming อยู่ขวาของจอภาพ



- ให้นักศึกษาเขียนโปรแกรมเพื่อแสดงผลลัพธ์ Programming ปรากฏอยู่ตรงตำแหน่งทั้ง 4 มุมของจอภาพ โดยใช้คำสั่ง printf เพียงคำสั่งเดียวเท่านั้นและห้ามใช้ \n



- เขียนโปรแกรมเพื่อแสดงผลค่าตัวเลขจำนวนจริง 3.14159265358979323846 โดยแสดงผลลัพธ์เป็นเลขศูนย์ 3,4,5,6,7,8,9 และ 10 ตำแหน่ง โดยแสดงผลคนละบรรทัดและให้จุดทศนิยมอยู่กลางบรรทัด (ตำแหน่งที่ 41)



## บทที่ 3 ชนิดของข้อมูลในภาษาซี คำสั่งรับข้อมูล และการคำนวณ

### วัตถุประสงค์การศึกษา

- 1) รู้จักและเข้าใจข้อมูลประเภทต่างๆ
- 2) สามารถสร้างตัวแปรและใช้งานตัวแปรได้อย่างถูกต้อง
- 3) สามารถแสดงข้อความอักขระของภาพและรับข้อมูลจากคีย์บอร์ดได้
- 4) สามารถใช้ Operator ทางคณิตศาสตร์ได้
- 5) เขียนโปรแกรมประมวลผลตัวอักษรได้

### 3.0 ตัวแปรในภาษาซี

การประกาศตัวแปรมี 2 ลักษณะได้แก่ ตัวแปรโภกบล (Global variable) และ ตัวแปรโลคอล (Local variable) ซึ่ง มีรูปแบบการประกาศตัวแปรที่เหมือนกัน แต่จะมีคุณสมบัติต่างกัน โดยจะเห็นได้ชัดเจนเมื่อมีการสร้างฟังก์ชันมาใช้งาน เป็นอย่างไรก็ตาม จะใช้งานเฉพาะตัวแปรนิดโลคอล โดยจะประกาศตัวแปรในส่วนเริ่มต้นของฟังก์ชัน

### 3.1 ชนิดของข้อมูลในภาษาซี

เป็นพารามิเตอร์ที่ใช้งานในภาษาซี เพื่อกำหนดลักษณะฟังก์ชันต้นแบบ และ ตัวแปรที่ใช้งานในโปรแกรม มี 4 ชนิด

void	ข้อมูลชนิดว่างเปล่า
int	ข้อมูลชนิดจำนวนเต็ม
float	ข้อมูลชนิดจำนวนทศนิยม
char	ข้อมูลชนิดอักขระ

#### 3.1.1 ข้อมูลชนิดว่างเปล่า (void)

void เป็นพารามิเตอร์ที่ใช้งานในส่วนของ ฟังก์ชันโพร์โตรไทร์ การสร้างและใช้งานฟังก์ชัน เพื่อแสดงให้รู้ว่าฟังก์ชันที่สร้างขึ้นมาไม่มีการส่ง หรือรับค่าจากการเรียกใช้งานฟังก์ชัน

#### 3.1.2 ข้อมูลชนิดจำนวนเต็ม (integer)

int เป็นพารามิเตอร์หลักที่ใช้กับข้อมูลชนิดจำนวนเต็ม โดยมีการใช้งาน 5 รูปแบบดังนี้

unsigned int	ข้อมูลชนิดจำนวนเต็มไม่คิดเครื่องหมายขนาด 2 bytes
short int	ข้อมูลชนิดจำนวนเต็มขนาด 2 bytes
int	ข้อมูลชนิดจำนวนเต็มขนาด 2 bytes หรือ 4 bytes (ลังเกตให้ดี ในคอมไฟเลอร์ขนาด 32bit เช่น MS-Visual studio 1.0 – 2008 เป็นต้น ตัวแปร int จะมีขนาด 4 byte และ ในคอมไฟเลอร์ขนาด 16 บิตตัวแปร เช่น Borland C 1.0 - 4.5 ตัวแปร int จะมีขนาด 2 byte)
unsigned long	ข้อมูลชนิดจำนวนเต็มไม่คิดเครื่องหมายขนาด 4 bytes
long	ข้อมูลชนิดจำนวนเต็มขนาด 4 bytes

#### 3.1.3 ข้อมูลชนิดจำนวนทศนิยม (float)

float เป็นพารามิเตอร์หลักที่ใช้กับข้อมูลชนิดจำนวนทศนิยมโดยมีการใช้งาน 3 รูปแบบดังนี้

float	ข้อมูลชนิดจำนวนทศนิยมขนาด 4 byte
-------	----------------------------------

<b>double</b>	ข้อมูลชนิดจำนวนทศนิยมขนาด 8 byte
<b>long double</b>	ข้อมูลชนิดจำนวนทศนิยมขนาด 10 byte

### 3.1.4 ข้อมูลชนิดอักขระ หรือตัวอักษร (character)

char เป็นพารามิเตอร์ที่ใช้งานเกี่ยวกับตัวอักษร และข้อความในภาษาซี โดยมีการกำหนดค่าอักขระโดยให้อยู่ในเครื่องหมาย single quote (...) เช่น 'C', 'o', 'm', '1' อักขระพิเศษบางตัวไม่สามารถกำหนดค่าให้ได้โดยตรง แต่ใช้ค่ารหัส ASCII เช่น อักขระควบคุมการแสดงผลการขึ้นต้นบรรทัดใหม่เช่น '\n' เป็นต้น โดยมีรูปแบบการใช้งาน 2 รูปแบบ

<b>unsigned char</b>	ข้อมูลชนิดอักขระไม่คิดเครื่องหมาย
<b>char</b>	ข้อมูลชนิดอักขระปกติ

### 3.1.5 ตารางสรุปชนิดข้อมูล

ชนิดของตัวแปร	ขนาด	ค่าต่ำสุด	ค่าสูงสุด
<b>unsigned char</b>	8 bit	0	255
<b>char</b>	8 bit	-128	127
<b>unsigned int</b>	16 bit	0	65,535
	32 bit	0	4,294,967,295
<b>short int</b>	16 bit	-32,768	32,767
<b>int</b>	16 bit	-32,768	32,767
	32 bit	-2,147,483,648	2,147,483,647
<b>unsigned long</b>	32 bit	0	4,294,967,295
<b>long</b>	32 bit	-2,147,483,648	2,147,483,647
<b>float</b>	32 bit	$3.4 \times 10^{-38}$	$3.4 \times 10^{38}$
<b>double</b>	64 bit	$1.7 \times 10^{-308}$	$1.7 \times 10^{308}$
<b>long double</b>	80 bit	$3.4 \times 10^{-4932}$	$3.4 \times 10^{4932}$

ขนาดจำนวนบิตของตัวแปร integer ขึ้นอยู่กับ รุ่นของ compiler โดย โปรแกรมคอมไฟเลอร์ เช่น TurboC 2.0-4.5 ขนาดของ Integer จะมีขนาด 16บิต หรือ 2 ไบท์ (ช่วงข้อมูล -32768 ถึง 32767) และโปรแกรมคอมไฟเลอร์ Visual 1.0 - Visual Studio 2008 (รุ่น 32บิต) ขนาดของ integer มีขนาด 32 บิต หรือ 4 ไบท์ (ช่วงข้อมูล -2147483648 ถึง 2147483647)

## 3.2 รูปแบบการประกาศตัวแปร

### 3.2.1 ประกาศตัวแปรแบบไม่กำหนดค่าเริ่มต้น

รูปแบบคำสั่ง

```
type var1; //รูปแบบที่1
type var1, var2, ..., varN; //รูปแบบที่2
```

type คือ ชนิดของข้อมูลที่จะกำหนดให้กับตัวแปร  
 var คือ ชื่อของตัวแปรที่จะตั้ง

ตัวอย่างการประกาศตัวแปร

```
int    number;           //ประกาศตัวแปรชนิด integer
int    a, b, c;
float  real;            //ประกาศตัวแปรชนิด float
float  point1, point2;
char   choice;          //ประกาศตัวแปรชนิด character
char   ch1, ch2;
```

การประกาศตัวแปรในลักษณะนี้เป็นการประกาศแบบไม่กำหนดค่าเริ่มต้น ดังนั้นค่าเริ่มต้นของตัวแปร อาจจะเป็นค่าใดๆได้ ขึ้นอยู่กับข้อมูลที่ค้างอยู่ในหน่วยความจำขณะนั้น

### 3.2.2 การประกาศตัวแปรแบบกำหนดค่าเริ่มต้น

รูปแบบคำสั่ง

```
type  var1 = value;      //รูปแบบที่1
      var1 = value1, var2 = value2; //รูปแบบที่2
```

type คือ ชนิดของข้อมูลที่จะกำหนดให้กับตัวแปร  
 var คือ ชื่อของตัวแปรที่จะตั้ง  
 value คือ ค่าของตัวแปรที่ต้องการกำหนดให้

ตัวอย่างการประกาศตัวแปร

```
int    number = 25;
int    a = 1, b = 2, c = 3;
float  real = 99.99;
float  point1 = 45.2, point2 = 30;
char   choice = 'a';
char   ch1 = 'o', ch2 = 'z';
```

### 3.2.3 หลักการตั้งชื่อตัวแปร

หลักการตั้งชื่อตัวแปรมีข้อกำหนดดังนี้

- ชื่อตัวแปรขึ้นต้นด้วยอักษร A-Z, a-z หรือ เครื่องหมาย “\_” (Underscore) เท่านั้น
- ภายในชื่อตัวแปรห้ามมีซ้ำว่าง
- ภายในชื่อตัวแปรประกอบด้วยอักษร A-Z, a-z, ตัวเลข 0-9 หรือ เครื่องหมาย “\_” เท่านั้น

4. การใช้อักษรตัวใหญ่และตัวเล็กมีความแตกต่างกัน
5. ห้ามใช้คำส่วน (Reserved word) เป็นชื่อตัวแปร
6. ควรตั้งชื่อตัวแปรให้สัมพันธ์กับค่าข้อมูลที่ใช้เก็บ

#### 3.2.3.1 คำส่วน (Reserved word)

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

ตัวอย่างการตั้งชื่อตัวแปร

//ตั้งชื่อตัวแปรถูกต้อง int _money; float salary_ot; char M11223_223; double GOTO_data;	//ตั้งชื่อตัวแปรผิด int \$money; float static; char 1M1223_223; double GOTO-data;
---	---

### 3.3 ตัวแปรชนิดข้อความ

ในภาษาซีจะไม่มีข้อมูลชนิดข้อความโดยเฉพาะ ซึ่งในการเขียนโปรแกรมส่วนใหญ่จำเป็นต้องมีการรับข้อมูลที่เป็นข้อความ เราสามารถใช้ตัวแปรข้อมูลชนิดอักขระหลายๆ ตัวมาใช้งานได้ในระดับหนึ่ง แต่ยังไม่สะดวกเมื่อข้อความมีความยาวมาก

ตัวอย่าง เช่น ต้องการใช้ข้อความว่า Hello สามารถใช้ตัวแปรชนิดอักขระ 5 ตัวแทน

```
char ch1='H',ch2='e',ch3='l',ch4='l',ch5='o';
```

ตัวแปรชนิดข้อความในภาษาซี คือ การนำอักษรมาเรียงต่อกัน ดังนั้นสามารถสร้างตัวแปรชนิดอักขระเรียงต่อ กันหลาย ๆ ตัว ให้เป็นตัวแปรชนิดแผลงคำดับ ทำให้สามารถใช้เก็บข้อมูลชนิดข้อความได้ โดยตัวแปรชนิดข้อความในภาษาซีจะอยู่ในเครื่องหมาย Double quote " "

#### 3.3.1 รูปแบบการประกาศตัวแปรชนิดข้อความ

รูปแบบคำสั่ง

char var1 [M1]; //รูปแบบที่1
char var1 [M1], var2 [M2]; //รูปแบบที่2

var คือ ชื่อตัวแปร

M คือ จำนวนของอักขระที่จะใช้เก็บบวกด้วย 1

การใช้ตัวแปรແຄວลำดับชนิดอักขระเป็นตัวแปรข้อความ ในภาษาซีกำหนดให้ว่าตัวสุดท้ายของตัวแปรແຄວลำดับคือ \0 สัญลักษณ์ \0 เเรียกว่า Null character

ตัวอย่าง

```
char str[15];
char first[20], last[20];
```

**str[15]**

**first[20]**

**last[20]**

### 3.3.2 รูปแบบการประกาศตัวแปรชนิดข้อความพร้อมกำหนดค่า

รูปแบบคำสั่ง

```
char var[M] = "?????"; //รูปแบบที่1
char var[M] = { '?', '?', '?', '?'}; //รูปแบบที่2
char var[] = "?????"; //รูปแบบที่3
```

var คือ ชื่อตัวแปร

M คือ จำนวนของอักขระที่จะใช้เก็บหากด้วย 1

? คือ อักขระที่จะกำหนดค่าให้ข้อความ จำนวน m-1 ตัว

### 3.3.3 รูปแบบการอ้างอิงในตัวแปรข้อความ

รูปแบบคำสั่ง

variable[N]

variable คือชื่อตัวแปร

N คือลำดับอักขระที่จะอ้างอิงในตัวแปรข้อความ

เริ่มนับอักขรตัวแรกเป็นตำแหน่งที่ 0

```
char subject[12] = "Programming";
```

**subject[12]**

P r o g r a m m i n g \0  
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]

**subject[0]** → 'P'

**subject[1]** → 'r'

**subject[2]** → 'o'

**subject[10]** → 'g'

### 3.4 การใช้งานตัวแปรร่วมกับคำสั่ง printf

ในคำสั่ง printf มีส่วนแสดงชนิดข้อมูล ซึ่งตัวแปรที่ใช้งานก็เป็นส่วนย่อยของข้อมูลชนิดต่าง ๆ รูปแบบคำสั่ง

```
printf (format-string, data-list);
```

จึงสามารถใช้ตัวแปรแทนในส่วนของ data-list ได้ เช่น

//แบบไม่ใช้ตัวแปร <pre>#include&lt;stdio.h&gt; int main() {     printf("GPA : %.2f",3.5);     return 0; }</pre>	//แบบใช้ตัวแปร <pre>#include&lt;stdio.h&gt; int main() {     float g = 3.5;      //ประกาศตัวแปรชนิด float     printf ("GPA : %.2f",g);     return 0; }</pre>
--	---

ตัวอย่าง ต้องการเขียนโปรแกรมแสดงคำว่า Programming จำนวน 5 บรรทัดที่จ่อคอมพิวเตอร์ โดยใช้ตัวแปรข้อความ

<pre>#include &lt;stdio.h&gt; int main() {     printf ("%s\n","Programming");     printf ("%s\n","Programming");     printf ("%s\n","Programming");     printf ("%s\n","Programming");     printf ("%s\n","Programming");     return 0; }</pre>	<pre>#include &lt;stdio.h&gt; int main() {     char name[20] = "Programming";     printf ("%s\n",name);     printf ("%s\n",name);     printf ("%s\n",name);     printf ("%s\n",name);     printf ("%s\n",name);     printf ("%s\n",name);     return 0; }</pre>
---	---

### 3.5 การรับข้อมูลจาก Keyboard ด้วย scanf

โปรแกรมโดยทั่วไปต้องมีการรับค่าข้อมูลจากผู้ใช้โปรแกรม เพื่อนำมาหาผลลัพธ์ตามกระบวนการทำงานของโปรแกรม หรือ ตามความต้องการของผู้ใช้งาน คำสั่งที่ใช้สำหรับการรับค่าในภาษาซีมีหลายคำสั่ง แต่ที่สามารถใช้งานได้ครอบคลุมและนิยมใช้กัน คือ คำสั่ง scanf

#### 3.5.1 รูปแบบคำสั่ง scanf

รูปแบบคำสั่ง

```
scanf(format-string, address-list);
```

format-string คล้ายในคำสั่ง printf แต่จะมีเฉพาะส่วนแสดงชนิดข้อมูล และอยู่ในเครื่องหมาย "

address-list คือ ตำแหน่งของตัวแปรที่ต้องการเก็บข้อมูลไว้ (การใช้งานตำแหน่งของตัวแปรจะใช้เครื่องหมาย & นำหน้าชื่อตัวแปร ยกเว้นตัวแปรชนิดข้อความ)

หมายเหตุ คำสั่ง scanf ต้องเรียกใช้ Preprocessor Directive #include<stdio.h>

ตัวอย่างการใช้คำสั่ง scanf

```
#include<stdio.h>
int main()
{
    float point;
    char name[20];
    printf ("Enter your name : ");
    scanf ("%s",name);
    printf ("Enter your point : ");
    scanf ("%f",&point);
    return 0;
}
```

ตัวอย่างการใช้คำสั่ง scanf

<pre>#include&lt;stdio.h&gt; int main() {     char first[20],last[20];      printf ("Enter your name and surname : ");     scanf ("%s %s",first,last);      printf ("Hi %s %s\nHow are you?",first,last);     return 0; }</pre>	<p>ผลลัพธ์</p> <p>Enter your name and surname : Peter Bravo Hi Peter Bravo How are you?</p>
---	---

### 3.5.2 การรับข้อมูลที่มีการเว้น (spacebar)

คำสั่ง scanf ไม่สามารถใช้รับข้อมูลที่มีการเว้นเพื่อเก็บในตัวแปรข้อมูลตัวเดียวได้ (เมื่อใช้ %s) เราสามารถใช้คำสั่ง scanf เพื่อให้รับข้อมูลที่มีการเว้นไปเก็บในตัวแปรชนิดข้อมูลได้ดังนี้

รูปแบบคำสั่ง

```
scanf ("%[^\\n]", string);
```

### โปรแกรม 3.1 คำสั่ง scanf

<pre>#include&lt;stdio.h&gt; int main() {     char name[40];     float gpa;     printf ("Enter your name : ");     scanf ("%[^\\n]",name);     printf ("Enter your GPA : ");     scanf ("%f",&amp;gpa);     printf ("Name : %s\\n",name);     printf ("Gpa : %f",gpa);     return 0; }</pre>	<pre>(Inactive C:\TCWIN45\BIN\NONAM Enter your name : Com eng Enter your GPA : 3.7 Name : Com eng Gpa : 3.700000</pre>
--	--

### โปรแกรม 3.2 คำสั่ง scanf

<pre>#include&lt;stdio.h&gt; int main() {     char name[40]; int date, month, year;     printf ("Enter your name &amp; surname : ");     scanf("%[^\\n]",name);     printf ("Enter your Birthday [d/m/y] : ");     scanf("%d/%d/%d",&amp;date,&amp;month,&amp;year);     printf ("%s birthday is %d %d %d",name,date,month,year);     return 0; }</pre>	<pre>(Inactive C:\TEMP\NONAME00.EXE) Enter your name &amp; surname : Jirasak Sittigorn Enter your Birthday [d/m/y] : 19/1/1988 JIRASAK SITTIGORN birthday is 19 1 1988</pre>
---	--

### 3.6 เครื่องหมายคำนวณทางคณิตศาสตร์

เครื่องหมาย	การทำงาน	ตัวอย่าง
+	บวก	<code>ans = a + b;</code>
-	ลบ	<code>ans = a - b;</code>
*	คูณ	<code>ans = a * b;</code>
/	หาร	<code>ans = a / b;</code>
%	ໄມຄູສັກ (modulo)	<code>ans = a % b;</code>

### โปรแกรม 3.3 แสดงการคำนวณ

```
#include <stdio.h>
int main()
{
    int a,b,ans;
    a = 10; b = 20;
    a+b = ans;           //error
    ans = a+b;          //OK
    printf("%d",ans);
    return 0;
}
```

การกำหนดค่าให้กับตัวแปรในภาษาซีใช้เครื่องหมาย = โดยการทำางจะนำค่าที่อยู่ทางขวาเมื่อ (จำนวน อักษร ข้อความ ค่า  
จากตัวแปร หรือผลลัพธ์จากฟังก์ชัน) ให้กับตัวแปรที่อยู่ทางซ้ายเมื่อ  
รูปแบบคำสั่ง

```
variable = value;
```

ตัวอย่างการแทนค่าลงในตัวแปร

```
num1 = 99;           n = num%10;
point = mid+final; ch = '9';
ans = pow(x,y);     str = "Com"; //Error
```

กรณีที่ตัวแปรเป็น string ไม่สามารถแทนข้อความด้วยเครื่องหมาย “=” ได้ ดังเช่นตัวอย่างด้านล่างนี้ วิธีการแก้ปัญหาให้ใช้ฟังก์ชัน strcpy (อ่านว่า string copy) ซึ่งต้องประกาศ #include <string.h>

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[10];
    str = "hello";           //Error
    strcpy(str,"hello");    //OK
    printf("%s",str);
    return 0;
}
```

### 3.6.1 การเพิ่มลดค่าตัวแปร

เครื่องหมาย	การทำงาน	ตัวอย่าง	ขั้นตอนการทำงาน	
++	เพิ่มค่าทีละ 1 (Increment)	<b>x++ ; ++x ;</b>	เพิ่มค่า x ขึ้น 1	
		<b>y = ++x ;</b>	เพิ่มค่า x ขึ้น 1	กำหนดค่าให้ y
		<b>y = x++ ;</b>	กำหนดค่าให้ y	เพิ่มค่า x ขึ้น 1
--	ลดค่าทีละ 1 (Decrement)	<b>x-- ; --x ;</b>	ลดค่า x ลง 1	
		<b>y = --x ;</b>	ลดค่า x ลง 1	กำหนดค่าให้ y
		<b>y = x-- ;</b>	กำหนดค่าให้ y	ลดค่า x ลง 1

### โปรแกรม 3.4 เพิ่ม/ลดค่าตัวแปร

```
#include <stdio.h>
int main()
{
    int a;
    a = 10;
    printf(" %d ",a++);
    printf(" %d ",++a);
    printf(" %d ",a+2);
    printf(" %d ",++a);    //สังเกตผลลัพธ์
    return 0;
}
```

### 3.6.2 เครื่องหมายแบบลดรูป

เครื่องหมาย	ตัวอย่างการใช้งาน	ตัวอย่างรูปแบบเต็ม
<b>+=</b>	<b>y += x ;</b>	<b>y = y + x ;</b>
<b>-=</b>	<b>y -= x ;</b>	<b>y = y - x ;</b>
<b>*=</b>	<b>y *= x ;</b>	<b>y = y * x ;</b>
<b>/=</b>	<b>y /= x ;</b>	<b>y = y / x ;</b>
<b>%=</b>	<b>y %= x ;</b>	<b>y = y % x ;</b>

### โปรแกรม 3.5 เพิ่ม/ลดค่าตัวแปร

#include <stdio.h> int main() { int a; a = 10; printf(" %d ", a++); printf(" %d ", ++a); printf(" %d ", a+=2); printf(" %d ", ++a); //สังเกตผลลัพธ์ return 0; }	//ผลลัพธ์ 10 12 14 15
---	--------------------------

### 3.6.3 ตัวดำเนินการ และตัวถูกดำเนินการ

Type 1	Operator	Type 2	Result	Exam	
<b>int</b>	+,-,*,/	<b>int</b>	<b>int</b>	$3*3$	9
				$19/2$	9
<b>int</b>	+,-,*,/	<b>float</b>	<b>float</b>	$3*3.0$	9.000000
				$19/2.0$	9.500000
<b>float</b>	+,-,*,/	<b>int</b>	<b>float</b>	$3.0*3$	9.000000
				$19.0/2$	9.500000
<b>float</b>	+,-,*,/	<b>float</b>	<b>float</b>	$3.0*3.0$	9.000000
				$19.0/2.0$	9.500000
<b>int</b>	%	<b>int</b>	<b>int</b>	$7\%4$	3
				$8\%4$	0

### 3.6.4 นิพจน์

ลำดับความสำคัญ	เครื่องหมาย
1	()
2	!, ++, --, (type)
3	*, /, %
4	+, -
5	<, <=, >, >=
6	==, !=
7	&&
8	
9	*=, /=, %=, +=, -=

หมายเหตุ: ไม่ควรใช้ == หรือ != กับข้อมูลประเภทหนึ่ง เช่น float double

### โปรแกรม 3.6 แสดงลำดับความสำคัญลักษณ์ทางคณิตศาสตร์

#include <stdio.h> int main() { int a = 10; printf(" %d ", 10*2*3-10/7); printf(" %d ", 10*2*(3-10)/7); printf(" %d ", 10*2>2+3); printf(" %d ", 10*2<2+3); return 0; }	//ผลลัพธ์ 58 -20 1 0
--	-------------------------

### 3.6.4 การเปลี่ยนชนิดของข้อมูล

ในภาษาซีมีบางคำสั่งที่ใช้งานได้กับเฉพาะข้อมูลบางชนิดเท่านั้น เช่น Modulo จะใช้งานได้เฉพาะข้อมูลชนิดจำนวนเต็ม แต่ไม่สามารถใช้กับข้อมูลชนิดจำนวนทศนิยมได้ รูปแบบการเปลี่ยนชนิดข้อมูล

รูปแบบคำสั่ง

```
(type) variable;
```

type คือ ชนิดของข้อมูลที่ต้องการ

variable คือ ชื่อของตัวแปรที่ต้องการนำมามาใช้

หมายเหตุ ไม่ได้ทำการเปลี่ยนตัวแปรเป็นชนิดอื่น แต่เปลี่ยนชนิดข้อมูลเพื่อนำไปใช้งาน

```
float gpa = 3.9; int num = 3;
printf ("gpa(f) :%-10f num(d) :%-10d\n", gpa, num);
printf ("gpa(d) :%-10d num(f) :%-10f\n", gpa, num);
printf ("%int(gpa(d) :%-10d", (int)gpa);
printf ("%float(num(f) :%-10f", (float)num);
```

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
gpa(f) :3.900000  num(d) :3
gpa(d) :0          num(f) :0.000000
(int)gpa(d) :3      (float)num(f) :3.000000
```

### 3.7 คำสั่งอื่นที่ใช้ แสดงผล และรับข้อมูล

putchar เป็นคำสั่งที่ใช้แสดงผลข้อมูลชนิดอักขระที่ลักษณะตัว โดยมีรูปแบบการใช้งานคำสั่งดังนี้

```
putchar(ch);
```

ch คือตัวแปรชนิดอักขระที่ต้องการแสดงผล

หมายเหตุ คำสั่ง putchar ต้องเรียกใช้ preprocessor Directive #include<stdio.h>

puts เป็นคำสั่งที่ใช้แสดงผลข้อมูลชนิดข้อความ โดยมีรูปแบบการใช้งานคำสั่งดังนี้

```
puts(str);
```

str คือตัวแปรชนิดข้อความที่ต้องการแสดงผล

หมายเหตุ คำสั่ง puts ต้องเรียกใช้ preprocessor Directive #include<stdio.h>

#### โปรแกรม 3.7 เรียนรู้ putchar, puts

```
char ch = 'A', str[] = "Computer";
putchar (ch);
putchar (' ');
putchar (str[1]);
putchar ('\n');
puts (str);
```

ผลลัพธ์

```
A
Computer
```

`getchar` เป็นคำสั่งที่ใช้รับข้อมูลชนิดอักขระจากผู้ใช้งานเพียงตัวเดียว โดยเมื่อป้อนข้อมูลแล้วต้องกด Enter คำสั่ง `getchar` มีรูปแบบการใช้งานคำสั่งดังนี้

```
ch = getchar();
```

ch คือตัวแปรชนิดข้อความที่ต้องการเก็บข้อมูลไว้

หมายเหตุ คำสั่ง `getchar` ต้องเรียกใช้ Preprocessor Directive `#include<stdio.h>`

`getch` เป็นคำสั่งที่ใช้รับข้อมูลชนิดอักขระจากผู้ใช้งานเพียงตัวเดียว โดยเมื่อป้อนข้อมูลแล้วโปรแกรมจะทำงานคำสั่งต่อไปทันที และจะไม่แสดงอักขระที่พิมพ์ไปคำสั่ง `getch` มีรูปแบบการใช้งานคำสั่งดังนี้

```
ch = getch();
```

ch คือตัวแปรชนิดข้อความที่ต้องการเก็บข้อมูลไว้

หมายเหตุ คำสั่ง `getch` ต้องเรียกใช้ Preprocessor Directive `#include<conio.h>`

`gets` เป็นคำสั่งที่ใช้รับข้อมูลชนิดข้อความจากผู้ใช้งาน โดยสามารถใส่ข้อมูลที่มีการเว้นช่องว่างภายในได้ คำสั่ง `gets` มีรูปแบบการใช้งานคำสั่งดังนี้

```
gets(str);
```

str คือตัวแปรชนิดข้อความที่ต้องการแสดงผล

หมายเหตุ คำสั่ง `gets` ต้องเรียกใช้ Preprocessor Directive `#include<stdio.h>`

### โปรแกรม 3.8 เรียกใช้ `getchar`, `getch`, `puts`

```
char ch1,ch2;
printf ("Enter Character 1 : ");
ch1 = getchar();
printf ("Enter Character 2 : ");
ch2 = getch();
puts ("\n***** Output *****");
printf ("Char 1 = %c\nChar 2 = %c",
ch1,ch2);
```

```
Enter Character 1 : J
Enter Character 2 :
*****
Output *****
Char 1 = J
Char 2 = O
```

### 3.8 คำถ้ามห้ายบท

1) ชนิดข้อมูลแบบข้อความเหมือนหรือแตกต่างจากชนิดข้อมูลแบบอักขระอย่างไร

2) จากตัวอย่างโปรแกรมต่อไปนี้ เมื่อ Run แล้วให้ผลลัพธ์ใด

```
#include <stdio.h>
int main()
{
    char name[20];
    int age;
    printf("Enter your name: "); scanf("%s",name);
    printf("Enter your age "); scanf("%d",age);
    printf("Your name is : %s\n Your age is : %d",name,age);
    return 0;
}
```

3) จากตัวอย่างโปรแกรมต่อไปนี้ เมื่อ Run แล้วให้ผลลัพธ์ใด

```
#include <stdio.h>
int main()
{
    printf("%d", (40/4*3+4*5/2));
    return 0;
}
```

## การทดลองที่ 3 การใช้งานคำสั่ง Input Output และการคำนวณต่างๆ

### วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้งานคำสั่งการแสดงผลต่างๆ
2. เพื่อให้นักศึกษาสามารถใช้งานคำสั่งการรับข้อมูลได้
3. เพื่อให้นักศึกษาเข้าใจหลักการเขียนโปรแกรมเบื้องต้น

### ทฤษฎี

คำสั่งในภาษา C สำหรับการสั่งให้คอมพิวเตอร์รับข้อมูลและแสดงผลข้อมูลมืออยู่หลายคำสั่ง แต่สำหรับผู้เริ่มต้นเขียนโปรแกรมแล้ว ควรเริ่มศึกษาคำสั่ง 2 คำสั่งคือคำสั่ง printf สำหรับการแสดงผลข้อมูลในหน้าจอคอมพิวเตอร์และคำสั่ง scanf สำหรับรับข้อมูลจากคีย์บอร์ด ในการทดลองนี้จะให้นักศึกษาลองใช้งานคำสั่งทั้งสองนี้

คำสั่ง                   printf

รูปแบบคำสั่ง       printf (format string, data list);

รายละเอียดการใช้งานคำสั่ง ดูรายละเอียดจากการทดลองที่ 2

คำสั่ง                   scanf

รูปแบบคำสั่ง       scanf (format string, address list);

รายละเอียดการใช้งานคำสั่ง scanf เป็นคำสั่งที่ใช้เพื่อรับข้อมูลจาก key board ประกอบด้วย 2 ส่วนได้แก่

- 1) format string : จะคล้ายกับคำสั่ง printf โดยจะอยู่ในเครื่องหมาย " " (Double quote) แต่จะมี เพียงลักษณะเดียว ได้แก่ รหัสรูปแบบค่าตัวแปร
- 2) address list : ส่วนที่เป็นชื่อตัวแปรต่างๆ ที่ต้องการนำค่าที่รับมาไปเก็บไว้ โดยที่ ถ้าตัวแปรที่ต้องการนำมาเก็บค่า เป็นชนิด จำนวนเต็ม (int), จำนวนจริง (float) หรือ อักขระ (char) ต้องมี & นำหน้าชื่อตัวแปร

### การคำนวณทางคณิตศาสตร์

ในการเขียนโปรแกรมสิ่งที่จะขาดไม่ได้เลยคือการคำนวณทางคณิตศาสตร์ และการคำนวณทางด้านตรรกศาสตร์สำหรับการคำนวณทางคณิตศาสตร์นั้น โปรแกรมเมอร์สามารถคำนวณโดยใช้เครื่องหมาย + , - , \* , / และเครื่องหมาย % โดยมีเครื่องหมาย ( และ ) เป็นเครื่องหมายที่ใช้ในการจัดกลุ่มของการคำนวณ

การคำนวณจะมีการคำนวณเครื่องหมายที่มีนัยสำคัญสูงกว่าก่อน โดยจัดนัยสำคัญของเครื่องหมายไว้ดังนี้

ตัวดำเนินการ	ความหมาย	ลำดับการคำนวณ
( )	Parenthesized expression	ซ้ายไปขวา
[ ]	Array subscript	
.	Member selection by object	
->	Member selection by pointer	
+ -	Unary + and -	ขวาไปซ้าย
++ --	Prefix increment and prefix decrement	
! ~	Logical NOT and bitwise complement	
&	Address-of	
sizeof	Size of expression or type	
(type)	Explicit cast to type such as (int) or (double)	
* / %	Multiplication and division and modulus	ซ้ายไปขวา
+ -	Addition and subtraction	
<< >>	Bitwise shift left and bitwise shift right	
< <=	Less than and less than or equal to	
> >=	Greater than and greater than or equal to	
== !=	Equal to and not equal to	
&	Bitwise AND	
^	Bitwise exclusive OR	
	Bitwise OR	
&&	Logical AND	

ในการคำนวนโปรแกรมเมอร์ต้องพิจารณาผลลัพธ์ของการคำนวนใน 2 ประเด็นคือ ผลลัพธ์ที่ได้มีค่าเท่าได้ และ ผลลัพธ์ที่ได้ เป็นชนิดอะไร ซึ่งโดยทั่วไปจะมีการสับสนระหว่างผลลัพธ์ที่ได้จากการคำนวนตัวเลขจำนวนเต็ม และตัวเลข จำนวนจริง ซึ่งเราสามารถสรุปชนิดของตัวเลขไว้ดังนี้

- 1) การคำนวนทางคณิตศาสตร์ระหว่างตัวแปรชนิด float กับตัวแปรชนิด float จะได้ผลลัพธ์เป็นตัวแปรชนิด float
- 2) การคำนวนทางคณิตศาสตร์ระหว่างตัวแปรชนิด float กับตัวแปรชนิด int จะได้ผลลัพธ์เป็นตัวแปรชนิด float
- 3) การคำนวนทางคณิตศาสตร์ระหว่างตัวแปรชนิด int กับตัวแปรชนิด int จะได้ผลลัพธ์เป็นตัวแปรชนิด int
- 4) การคำนวนทางคณิตศาสตร์ระหว่างตัวแปรชนิด int กับตัวแปรชนิด float จะได้ผลลัพธ์เป็นตัวแปรชนิด float

### ตัวอย่างการใช้งานคำสั่ง

1. เมื่อต้องการสั่งให้คอมพิวเตอร์แสดงคำว่า Computers and Programming ที่หน้าจอคอมพิวเตอร์สามารถเขียนคำสั่งดังนี้

```
printf("Computers and Programming");
```

2. เมื่อต้องการสั่งให้คอมพิวเตอร์แสดงผลลัพธ์เป็นชื่อวิชา Computers and Programming และให้คอมพิวเตอร์แสดงผลลัพธ์ในคำสั่งดามาให้อยู่ในบรรทัดใหม่ สามารถเขียนคำสั่งดังนี้

```
printf("Computers and Programming\n");
```

3. เมื่อต้องการสั่งให้คอมพิวเตอร์รับข้อมูลเป็นตัวเลขจำนวนเต็ม จะต้องมีการกำหนดค่าตัวแปรให้เป็นชนิดตัวเลขจำนวนเต็ม และใช้คำสั่ง scanf เพื่อรับข้อมูลมาเก็บไว้ในตัวแปรดังนี้

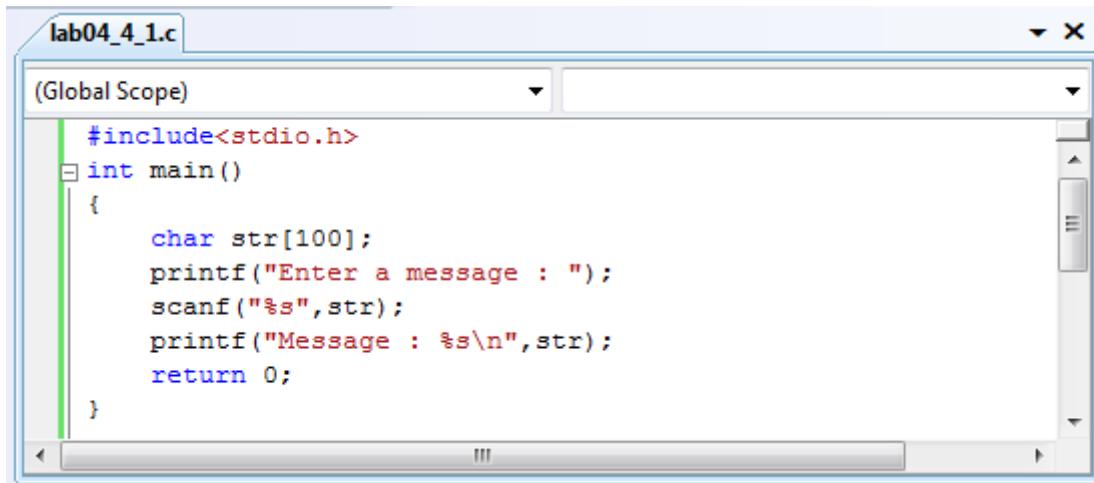
```
int a;  
scanf ("%d",&a);
```





3.2.5 เปลี่ยนคำสั่ง `scanf("%f",&num);` เป็น `scanf("%.2f",&num);` บันทึกผล


### การทดลองที่ 3.3 การรับข้อมูลด้วยตัวแปรข้อความ



```
#include<stdio.h>
int main()
{
    char str[100];
    printf("Enter a message : ");
    scanf("%s",str);
    printf("Message : %s\n",str);
    return 0;
}
```

3.3.1 ป้อนข้อความ Hello ผลลัพธ์ที่ได้คือ \_\_\_\_\_

3.3.2 ป้อนข้อความ Hello World ผลลัพธ์ที่ได้คือ \_\_\_\_\_

3.3.3 เปลี่ยน `scanf("%s",str);` เป็น `scanf("%[\r\n]",str);` และป้อนข้อความ Hello

ผลลัพธ์ที่ได้คือ \_\_\_\_\_

3.3.4 เปลี่ยน `scanf("%s",str);` เป็น `scanf("%[\r\n]",str);` และป้อนข้อความ Hello World!

ผลลัพธ์ที่ได้คือ \_\_\_\_\_

### การทดลองที่ 3.4 การคำนวณ

3.4.1 จะอธิบายขั้นตอนการทำงานของการคำนวณตัวเลขต่อไปนี้ ว่ามีการทำงานอะไรบ้าง จึงได้ผลลัพธ์ของการคำนวณ และได้ผลลัพธ์เป็นเท่าใด

1. <code>printf("%d\n", 4+5*2+5/4);</code> $\begin{aligned} 4+5*2+5/4 &= 4+10+5/4 \\ &= 4+10+1 \\ &= 14+1 \\ &= 15 \end{aligned}$	2. <code>printf("%d\n", 4+5*(2+5)/4);</code> <hr/> <hr/> <hr/> <hr/>
3. <code>printf("%d\n", (4+5*2+5)/4);</code> <hr/> <hr/> <hr/> <hr/>	4. <code>printf("%d\n", 4+5*(2+5/4));</code> <hr/> <hr/> <hr/> <hr/>
5. <code>printf("%d\n", (4+5)*(2+5/4));</code> <hr/> <hr/> <hr/> <hr/>	6. <code>printf("%f\n", (4+5)*(2+5.0/4));</code> <hr/> <hr/> <hr/> <hr/>

### 3.4.2 แจกโปรแกรมตัวอย่าง



```
#include<stdio.h>
int main()
{
    printf("%d\n", 2/3);
    return 0;
}
```

3.4.2.1 `printf("%d\n", 2/3);` ได้ผลลัพธ์เป็นเท่าใด \_\_\_\_\_

3.4.2.2 `printf("%f\n", 2/3);` ได้ผลลัพธ์เป็นเท่าใด \_\_\_\_\_

3.4.2.3 `printf("%d\n", 2.0/3);` ได้ผลลัพธ์เป็นเท่าใด \_\_\_\_\_

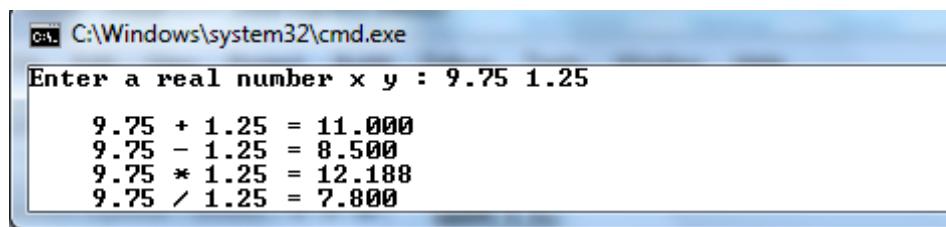
3.4.2.4 `printf("%f\n", 2.0/3);` ได้ผลลัพธ์เป็นเท่าใด \_\_\_\_\_

3.4.2.5 เหตุใดจึงเป็นเช่นนั้น \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

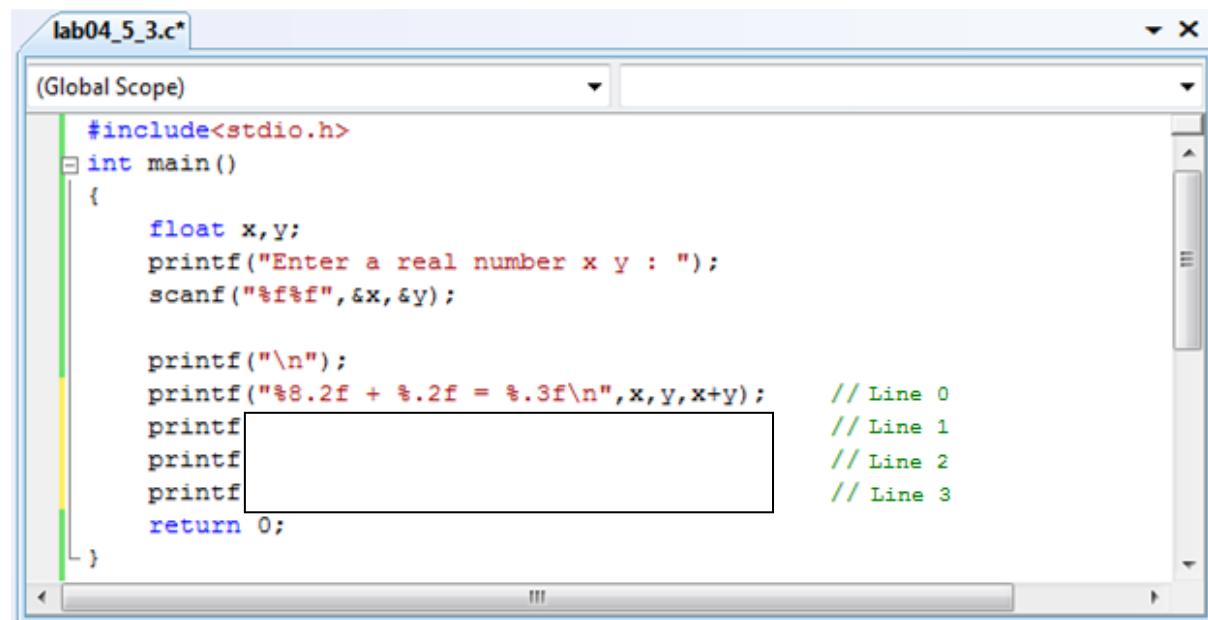
### การทดลองที่ 3.5 การรับข้อมูลจากคีย์บอร์ด

(ควรบอกผู้ใช้ทุกครั้งว่าให้ป้อนข้อมูลชนิดใด)

ให้นักศึกษาแก้ไขโปรแกรมให้แสดงผลลัพธ์ผลบวก ลบ คูณ และหาร ของเลขสองจำนวนนั้น โดยให้ผลลัพธ์ตรงกับตัวอย่าง



```
C:\Windows\system32\cmd.exe
Enter a real number x y : 9.75 1.25
 9.75 + 1.25 = 11.000
 9.75 - 1.25 = 8.500
 9.75 * 1.25 = 12.188
 9.75 / 1.25 = 7.800
```



```
#include<stdio.h>
int main()
{
    float x,y;
    printf("Enter a real number x y : ");
    scanf("%f%f", &x, &y);

    printf("\n");
    printf("%8.2f + %.2f = %.3f\n",x,y,x+y); // Line 0
    printf // Line 1
    printf // Line 2
    printf // Line 3
    return 0;
}
```

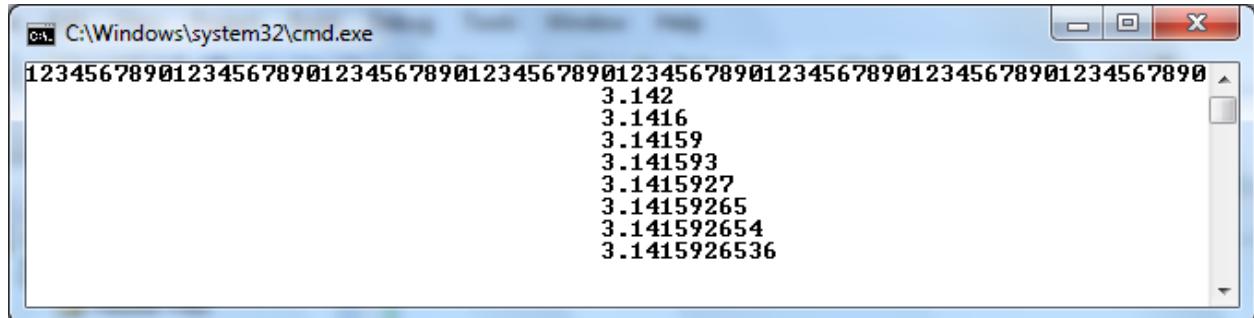
Code บรรทัด Line 1 คือ \_\_\_\_\_

Code บรรทัด Line 2 คือ \_\_\_\_\_

Code บรรทัด Line 3 คือ \_\_\_\_\_

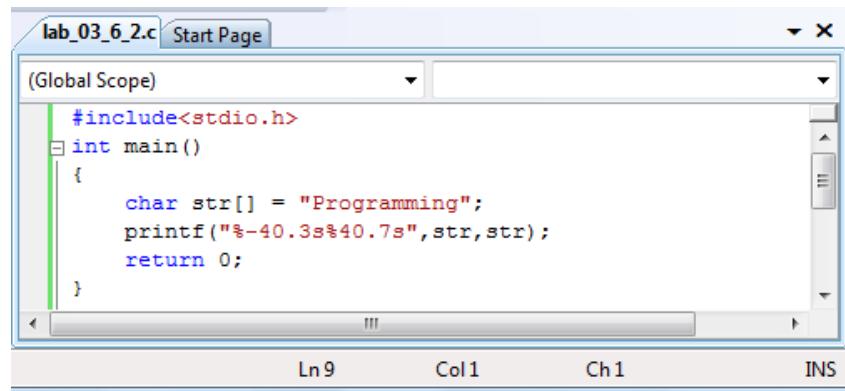
### การทดลองที่ 3.6 ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง

1. กำหนดให้  $\pi = 3.14159265358979323846$  โดยแสดงผลลัพธ์เป็นเลขศนิยม 3,4,5,6,7,8,9,และ 10 ตำแหน่ง โดยแสดงผลคนละบรรทัดและให้จุดศนิยมอยู่กลางบรรทัด (ตำแหน่งที่ 41)



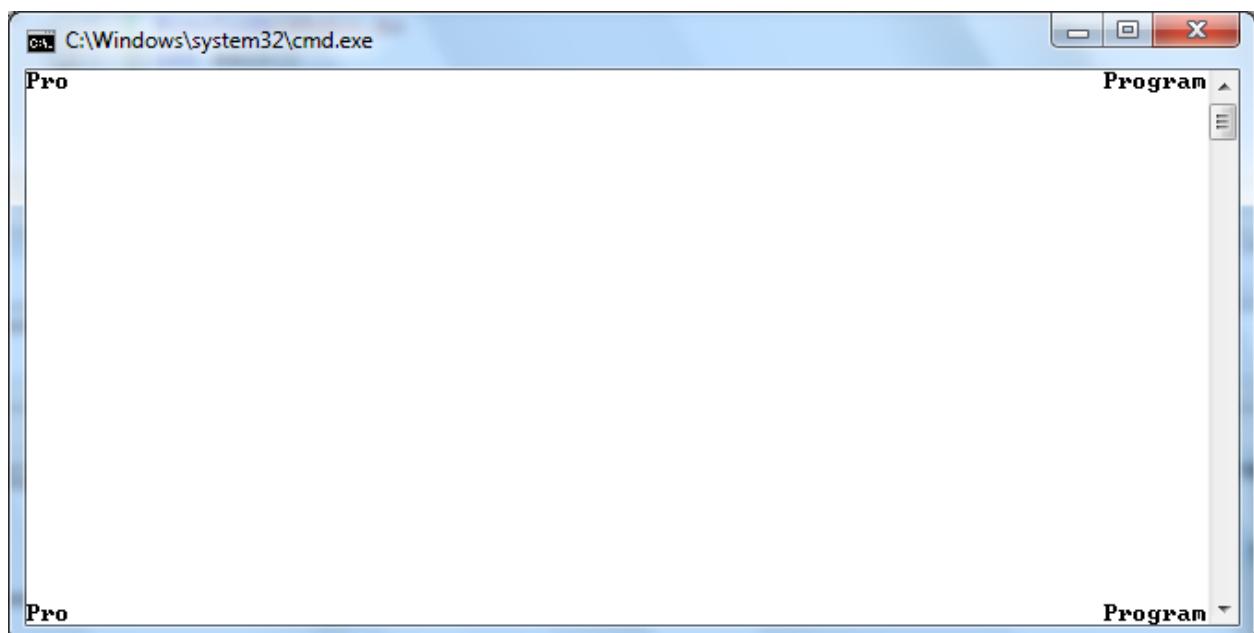
```
C:\Windows\system32\cmd.exe
1234567890123456789012345678901234567890123456789012345678901234567890
3.142
3.1416
3.14159
3.141593
3.1415927
3.14159265
3.141592654
3.1415926536
```

2. ให้นักศึกษาแก้ไขคำสั่ง printf ให้โปรแกรมแสดงผลลัพธ์ Pro ปรากฏอยู่ด้านซ้ายบรรทัดบนและล่างของจอภาพ และ Program ปรากฏตรงด้านขวาบรรทัดบนสุดและล่างสุดของจอภาพ (ห้ามเพิ่มคำสั่ง printf และห้ามใช้ \n)



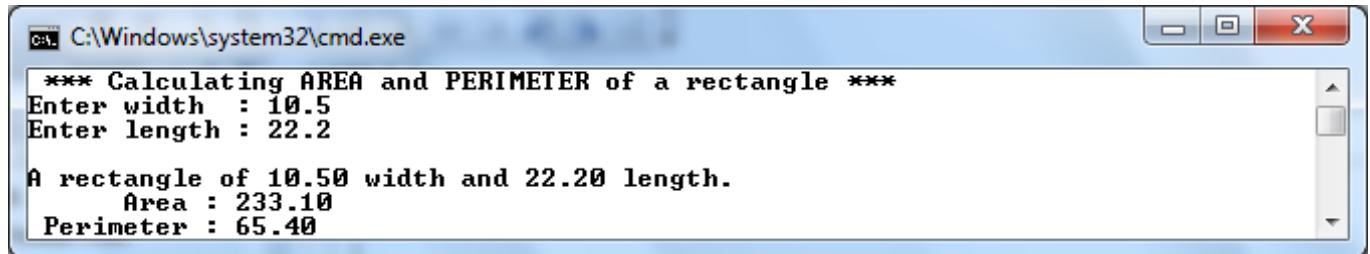
```
#include<stdio.h>
int main()
{
    char str[] = "Programming";
    printf("%-40.3s%40.7s",str,str);
    return 0;
}
```

ตัวอย่างการแสดงผล



```
C:\Windows\system32\cmd.exe
Pro
Program
Program
```

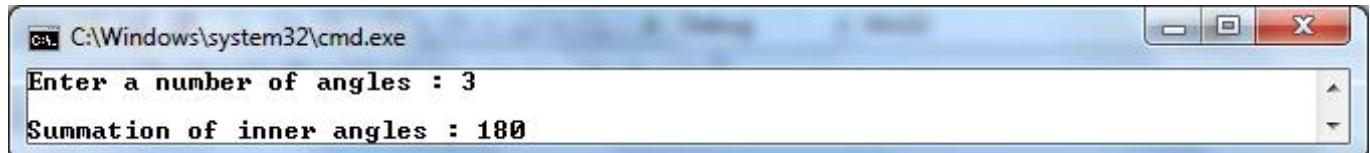
3. จงเขียนโปรแกรมคำนวณพื้นที่และเส้นรอบรูปของสี่เหลี่ยมผืนผ้า โดยเขียนโปรแกรมรับค่าตัวเลขด้านกว้างและยาวจากคีย์บอร์ด



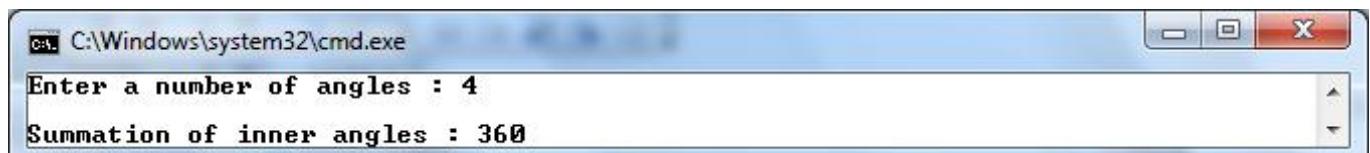
```
*** Calculating AREA and PERIMETER of a rectangle ***
Enter width : 10.5
Enter length : 22.2

A rectangle of 10.50 width and 22.20 length.
Area : 233.10
Perimeter : 65.40
```

4. เขียนโปรแกรมเพื่อหามุมภายในของรูป n เหลี่ยมใดๆ โดยรับค่า n จากผู้ใช้โปรแกรม  
 (Hint : มุมภายในสามเหลี่ยมคือ 180 องศา ห้าเหลี่ยมประกอบด้วยสามเหลี่ยมสองรูป)



```
C:\Windows\system32\cmd.exe
Enter a number of angles : 3
Summation of inner angles : 180
```

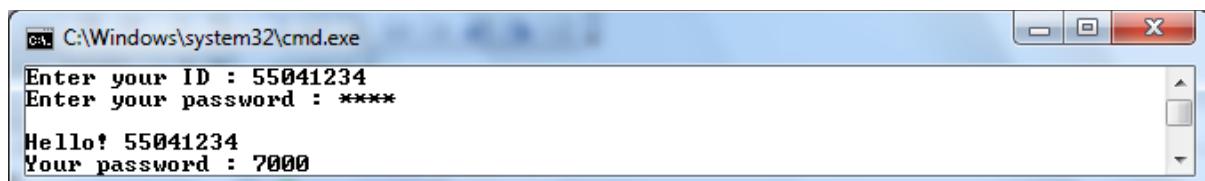


```
C:\Windows\system32\cmd.exe
Enter a number of angles : 4
Summation of inner angles : 360
```



```
C:\Windows\system32\cmd.exe
Enter a number of angles : 5
Summation of inner angles : 540
```

5. เขียนโปรแกรมรับ รหัสนักศึกษา และรหัสผ่าน 4 ตัวอักษร แล้วแสดงผลดังต่อไปนี้ กำหนดให้ประกาศตัวแปรได้สองตัว คือ ID และ char input[6] (Hint: รับข้อมูลด้วยคำสั่ง input[?] = getch() และแสดงผล '\*' ห้าครั้ง ตำแหน่งสุดท้าย input[5] ต้องเป็น null character)



```
C:\Windows\system32\cmd.exe
Enter your ID : 55041234
Enter your password : ****

Hello! 55041234
Your password : 7000
```

## บทที่ 4 ผังงาน และการเขียนโปรแกรมกำหนดเงื่อนไข

### วัตถุประสงค์การศึกษา

- 1) เขียนผังงานโปรแกรมจากโจทย์ปัญหาได้
- 2) เขียนโปรแกรมจากผังงานที่สร้างขึ้นได้
- 3) เขียนผังงานและโปรแกรมที่มีการกำหนดเงื่อนไขได้

### 4.1 การพัฒนาโปรแกรมคอมพิวเตอร์

การเขียนโปรแกรมที่มีความซับซ้อนมาก โปรแกรมเมอร์หรือผู้เขียนโปรแกรมจำเป็นต้องมีทักษะในการวิเคราะห์และพัฒนาทักษะเหล่านี้เริ่มต้นพัฒนาได้โดย เข้าใจกระบวนการวิเคราะห์และแก้ปัญหา กระบวนการหลักๆ มีดังนี้

ก่อนการพัฒนาโปรแกรม:

- 1) โปรแกรมเมอร์ต้องทำความเข้าใจกับปัญหา วิเคราะห์รูปแบบ สิ่งที่จำเป็น
- 2) โปรแกรมเมอร์ต้องคิดวางแผนวิธีการแก้ไขปัญหานั้น

ในขณะที่กำลังพัฒนาโปรแกรม:

- 1) พิจารณาถึงโครงสร้าง หรือชุดคำสั่งที่มีของภาษาคอมพิวเตอร์ ที่ใช้อยู่ ว่าคำสั่งมีการเรียกใช้ช้าๆ ซ้อนโครงสร้างมีช้าๆ หรือสามารถลดรูปกระบวนการได้ๆ ลงได้
- 2) ทำตามหลักการการพัฒนาโปรแกรมที่ดีตลอดเวลา

กระบวนการพัฒนาโปรแกรมมีสิ่งต่างๆ ที่ใช้ เช่น

อัลกอริทึม (algorithm) เป็นลักษณะคำสั่งซึ่งเขียนในรูปย่อ อาจเขียนเป็นประโยคที่สื่อความเข้าใจได้โดยง่าย ทำให้โปรแกรมเมอร์มองเห็นภาพกระบวนการทำงานของโปรแกรม

ผังงาน (flowchart) เป็นผังรูปสัญลักษณ์แสดงขั้นตอนกระบวนการทำงาน ผังงานช่วยให้โปรแกรมเมอร์เห็นภาพว่าส่วนงานใดมีความซับซ้อน

ผู้ร่วมเขียนโปรแกรมหรือนศ. ความเริ่มจากการศึกษาลองวัด flowchar ก่อนการเขียนโปรแกรมเพื่อให้เห็นภาพของกระบวนการทำงานของโปรแกรม

### อัลกอริทึม (Algorithm)

Problem

สามารถที่จะแก้ไขปัญหาเหล่านี้ได้โดยการทำงานตามชุดคำสั่งอย่างมีลำดับ

Algorithms

ขั้นตอนที่ระบุถึงวิธีการแก้ไขปัญหาหรือทำให้งานสำเร็จ

มีลำดับการทำงานที่แน่นอน

มีความชัดเจน

สามารถทำงานได้

มีจุดสิ้นสุดการทำงานที่ชัดเจน

Representation

วิธีการแปลง Algorithm เป็นภาษาคอมพิวเตอร์

Problem



Algorithms



Representation

ตัวอย่างอัลกอริทึมแสดงโปรแกรมทำงานคำนวณเลข

//algorithm กำหนดค่าเริ่มต้นเรียกใช้ กำหนดตัวแปรที่ใช้งาน รับค่าจากkeyboad เก็บในตัวแปร ถ้าตัวแปรมีค่าเป็น 10 แสดงคำว่า hello world ถ้าไม่ใช่ แสดงคำว่า bye	//โปรแกรม #include<stdio.h> int main() { int a; printf("Value ="); scanf("%d", &a); if(a==10) printf("hello"); else printf("bye"); return 0; }
--	--

### ขั้นตอนการคิด

- 1) ทำความเข้าใจกับปัญหา
- 2) พิจารณาแนวคิดถึงขั้นตอนการแก้ไขปัญหาที่มีอยู่ (ซึ่งอาจจะมีหลายวิธีในการแก้ไขปัญหานี้ ๆ)
- 3) พัฒนาโปรแกรมตามแนวคิดที่คิดไว้ตามขั้นตอนที่ 2
- 4) ประเมินความถูกต้องของโปรแกรม และลองพิจารณาถึงความสามารถที่จะใช้โปรแกรมนี้กับปัญหานี้ ๆ ได้หรือไม่

## 4.1 ผังงาน (Flowchart)

แผนภาพซึ่งแสดงลำดับขั้นตอนของการทำงาน โดยแต่ละขั้นตอนจะแสดงโดยใช้สัญลักษณ์ ซึ่งมีความหมายเบ่งอกกว่าขั้นตอนนั้นๆ มีลักษณะการทำงานแบบใด และในแต่ละขั้นตอนจะเขียนໂโยกันด้วยลูกศรเพื่อที่จะแสดงลำดับการทำงาน

### ประโยชน์ของผังงาน

- ช่วยให้สามารถทำความเข้าใจลำดับขั้นตอนการทำงานของโปรแกรมหรือระบบใดๆ ได้อย่างรวดเร็ว
- ช่วยแสดงลำดับขั้นตอนการทำงาน ทำให้สามารถเขียนโปรแกรมได้อย่างเป็นระบบไม่สับสน
- ช่วยการแก้ไขข้อผิดพลาดของโปรแกรมได้รวดเร็ว และช่วยให้ผู้พัฒนาโปรแกรมต่อสามารถทำงานได้ง่ายสะดวกขึ้น
- ง่ายแก่บุคคลภายนอกในการติดตามขั้นตอนของการปฏิบัติงาน

### ประเภทของผังงาน

- ผังงานระบบ (System Flowchart) เป็นผังงานซึ่งแสดงขอบเขต และลำดับขั้นตอนการทำงานของระบบหนึ่งๆ รวมทั้งแสดงรูปแบบของข้อมูลเข้า และข้อมูลออกว่าถูกรับเข้าหรือแสดงผลโดยผ่านสื่อประเภทใด เนื่องจากผังงานระบบเป็นแผนภาพที่แสดงถึงระบบโดยรวม ดังนั้นกระบวนการหรือโปรแกรมหนึ่งๆ อาจถูกแสดงเป็นเพียงขั้นตอนหนึ่งในผังงานระบบเท่านั้น
- ผังงานโปรแกรม (Program Flowchart) เป็นผังงานซึ่งแสดงลำดับขั้นตอนการทำงานของโปรแกรมหนึ่งๆ

### สัญลักษณ์ที่ใช้ในการเขียนผังงาน

สัญลักษณ์	ความหมาย
	แทนการเริ่มต้น หรือสิ้นสุดของผังงาน
	แทนการรับข้อมูลหรือป้อนข้อมูลเข้าทางแป้นพิมพ์
	แทนการแสดงผลลัพธ์บนหน้าจอภาพ
	แทนการทำหนดค่าหรือการคำนวณค่า
	แทนการเปรียบเทียบ
	แทนจุดต่อเนื่องที่อยู่หน้าเดียวกัน
	แทนจุดต่อเนื่องที่อยู่คนละหน้ากัน
	แทนการเรียกใช้ฟังก์ชันที่สร้างขึ้น

### หลักเกณฑ์ในการเขียนผังงาน

- 1) สัญลักษณ์ที่เข้ามาขนาดต่างๆกันได้ แต่จะต้องมีรูปร่างเป็นสัดส่วนตามมาตรฐาน
- 2) ทิศทางของลูกศรในผังงาน ควรจะมีทิศทางจากบนลงล่าง หรือจากซ้ายไปขวา
- 3) ผังงานควรมีความเรียบง่าย สะอาด พยามพยายามหลีกเลี่ยงการเขียนลูกศรที่ทำให้เกิดจุดตัด เพราะจะทำให้อ่านและทำความเข้าใจผังงานได้ยาก
- 4) ถ้าในผังงานมีการเขียนข้อความอธิบายใดๆ ควรทำให้สั้น gọnทั้งรัดและได้ใจความ อาจเขียนเป็นคำสั่งที่อยู่ในสัญลักษณ์ หรือใช้คำพูดแทน

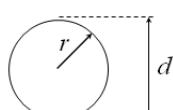
### แนวทางการสร้างผังงานสำหรับการเขียนโปรแกรม

- 1) การวิเคราะห์ข้อมูลเอาท์พุท หรือผลลัพธ์ (Output Analysis) วิเคราะห์ความต้องการของผู้ใช้ หรือผลลัพธ์จากโจทย์ปัญหา
- 2) การวิเคราะห์ข้อมูลอินพุท (Input Analysis) วิเคราะห์ข้อมูลที่ผู้ใช้ป้อน หรือข้อมูลที่โจทย์ให้มา
- 3) การวิเคราะห์กระบวนการทำงาน (Process Analysis) วิเคราะห์ขั้นตอนการทำงานที่ให้เด็มานา ซึ่งผลลัพธ์
- 4) การกำหนดตัวแปร (Variable Define) กำหนดตัวแปรที่ใช้งานการเขียนโปรแกรมเพื่อความถูกต้อง

### โปรแกรม 4.1 การสร้างโปรแกรมคำนวนหาพื้นที่วงกลม

เขียนผังงาน และโปรแกรมคำนวนหาพื้นที่วงกลม

- 1) Output Analysis ผลลัพธ์ที่ต้องการ คือ พื้นที่ของวงกลม
- 2) Input Analysis การคำนวนหาพื้นที่วงกลม จำเป็นต้องทราบขนาดของรัศมี (หรือเส้นผ่านศูนย์กลาง)



$$A = \pi r^2$$

### 3) Process Analysis

- รอรับค่ารัศมี (หรือเส้นผ่านศูนย์กลาง) จากผู้ใช้งาน
- คำนวนหาพื้นที่วงกลมจากสูตร
- แสดงผลค่าพื้นที่วงกลมออกทางหน้าจอ

### 4) Variable Define

radius : เป็นตัวแปรชนิดจำนวนทศนิยมสำหรับรับค่ารัศมี  
pi : เป็นตัวแปรชนิดจำนวนทศนิยมสำหรับเก็บค่า  
area : เป็นตัวแปรชนิดจำนวนทศนิยมสำหรับเก็บค่าพื้นที่

<p>5) วิเคราะห์งาน</p> <pre> graph TD     START([START]) --&gt; INPUT[รับค่ารัศมี radius]     INPUT --&gt; PI[คำนวณค่า pi = 22/7]     PI --&gt; CALCULATION[คำนวณพื้นที่ area = pi * radius * radius]     CALCULATION --&gt; OUTPUT([แสดงผลค่า area])     OUTPUT --&gt; END([END]) </pre>	<p>6) เขียนโปรแกรม</p> <pre>#include&lt;stdio.h&gt; int main() {     float radius, pi, area;     pi = 22.0/7; // pi = 3.14;     printf ("Enter Radius of Circular : ");     scanf ("%f",&amp;radius);     area = pi * radius * radius;     printf ("Area of Circular : %f",area);     return 0; }</pre>
---	---

## 4.2 การเขียนโปรแกรมแบบกำหนดเงื่อนไข

ในการเขียนโปรแกรมสำหรับงานส่วนใหญ่ จะเป็นต้องมีการทดสอบเงื่อนไขบางอย่างก่อน เพื่อตัดสินใจเลือกการทำงานของโปรแกรมในอันดับถัดไป

คำสั่งควบคุมการทำงานของโปรแกรมที่นิยมใช้มีอยู่ 2 คำสั่ง คือ คำสั่ง if และ คำสั่ง if ... else ซึ่งจะพิจารณาเลือกกระทำหรือไม่กระทำการพิสูจน์นิพจน์ว่าเป็น จริง หรือ เพ็จ โดยจะใช้ควบคู่กับเครื่องหมายเปรียบเทียบ และเครื่องหมายทางตรรกศาสตร์

#### 4.2.1 การเปรียบเทียบ

เครื่องหมาย	การเปรียบเทียบ	ตัวอย่าง
$=$	เท่ากัน	$x == y$
$!=$	ไม่เท่ากัน	$x != y$
$>$	มากกว่า	$x > y$
$\geq$	มากกว่าหรือเท่ากัน	$x \geq y$
$<$	น้อยกว่า	$x < y$
$\leq$	น้อยกว่าหรือเท่ากัน	$x \leq y$

ผลของการเปรียบเทียบจะได้ค่าจริง (ค่าที่ไม่ใช่ 0) หรือค่าเท็จ (ค่าที่เป็น 0)

การเปรียบเทียบ	ผลที่ได้	การเปรียบเทียบ	ผลที่ได้
$7 == 9$	False	$22 == 22$	True
$7 != 9$	True	$(3+5) != 8$	False
$8 > 8$	False	$9 > 7$	True
$8 \geq 8$	True	$7 \geq 9$	False
$(10+9) < 7$	False	$7 < (10+9)$	True
$4 \leq 3$	False	$3 \leq 4$	True

ไม่ควรใช้เครื่องหมายเท่ากับ  $=$  หรือไม่เท่ากับ  $!=$  สำหรับข้อมูลศูนย์

#### 4.2.2 เปรียบเทียบทางตระกศาสตร์

เครื่องหมาย	ความหมาย	ตัวอย่าง
$\&\&$	และ (and)	$x \&\& y$
$\ $	หรือ (or)	$x \  y$
!	ไม่ หรือ ตรงกันข้าม (not)	$!x$



AND

การดำเนินการ	ผลที่ได้
T && T	T
T && F	F
F && T	F
F && F	F

OR

การดำเนินการ	ผลที่ได้
T    T	T
T    F	T
F    T	T
F    F	F

Invert

การดำเนินการ	ผลที่ได้
! T	F
! F	T

ตัวอย่างการเขียนเงื่อนไข

กำหนดให้ int num1 = 10, num2 = 20, num3 = 30;

ถ้าเขียน

num1 == num2

ผลลัพธ์ เท็จ (false)

num1 &gt; num2

ผลลัพธ์ .....

num1 &gt; num2 &gt; num3

ผลลัพธ์ .....

(num1&lt;num2) &amp;&amp; (num2&lt;num3)

ผลลัพธ์.....

(num1&gt;num2) || (num1&gt;num3)

ผลลัพธ์.....

(num1&gt;num2) || (num2&lt;num3)

ผลลัพธ์.....

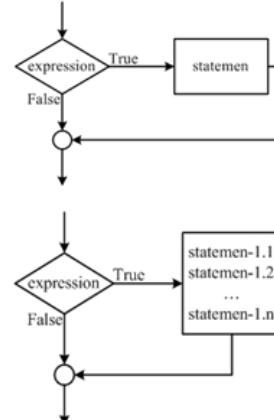
### 4.3 คำสั่งเงื่อนไข if

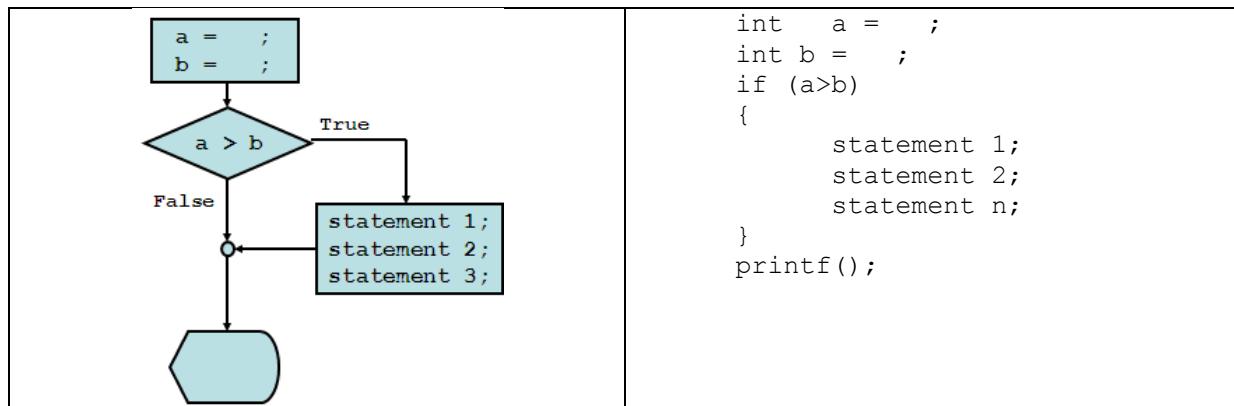
รูปแบบที่ 1

```
if( expression )
    statement1;
```

รูปแบบที่ 2 ใช้ปีกภาษาขยาย statement

```
if( expression )
{
    statement1.1;
    statement1.2;
    statement1.3;
}
```





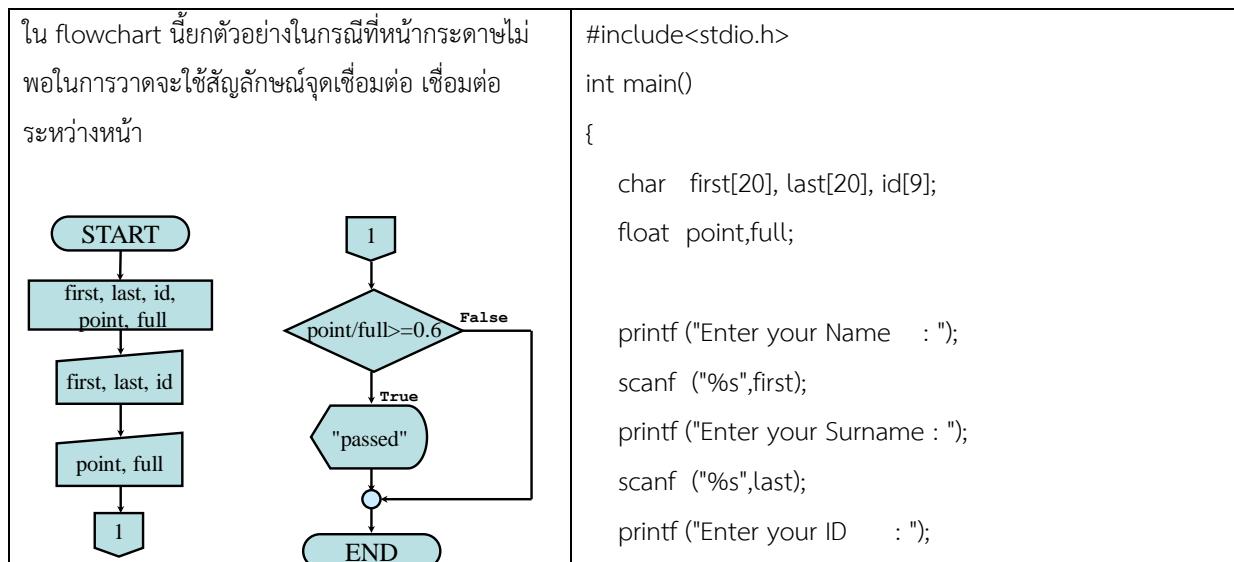
## โปรแกรม 4.2 โปรแกรมตรวจสอบค่าคะแนน

เขียนผังงานและโปรแกรมรับชื่อ นามสกุล รหัสนักศึกษา คะแนนสอบรวม และคะแนนเต็ม หากนักศึกษาสอบได้มากกว่า 60% ให้แสดงผลชื่อ นามสกุล รหัสนักศึกษา คะแนนสอบ และคะแนนเต็ม หากนักศึกษาสอบได้ไม่ถึง 60% ให้แสดงผลชื่อ-สกุล รหัสนักศึกษา คะแนนสอบ ผลสอบ เนื่องจากนักศึกษาสอบได้

เริ่มต้นโดย

- 1) Output analysis แสดงผลชื่อ-สกุล รหัสนักศึกษา คะแนนสอบ ผลสอบ
- 2) Input analysis ชื่อ / นามสกุล / รหัสนักศึกษา / คะแนนสอบ / คะแนนเต็ม
- 3) Process analysis
  - โปรแกรมรับชื่อ / นามสกุล / รหัสนักศึกษา / คะแนนสอบ / คะแนนเต็ม ตรวจสอบว่าคะแนนมากกว่าหรือเท่ากับ 60 % หรือไม่
  - ถ้าจริง แสดงผลชื่อ-สกุล รหัสนักศึกษา คะแนน และแสดงว่าสอบผ่าน
- 4) Variable define

first : ตัวแปรชนิดข้อความสำหรับเก็บชื่อขนาด 20  
 last : ตัวแปรชนิดข้อความสำหรับเก็บนามสกุลขนาด 20  
 id : ตัวแปรชนิดข้อความสำหรับเก็บรหัสนักศึกษานาด 9  
 point : ตัวแปรชนิดจำนวนทศนิยมสำหรับเก็บคะแนนรวม  
 full : ตัวแปรชนิดจำนวนทศนิยมสำหรับเก็บคะแนนเต็ม



```

scanf ("%s",id);
printf ("Enter your examination points : ");
scanf ("%f",&point);
printf ("Enter your total points      : ");
scanf ("%f",&full);
if ((point/full) >= 0.6)
{
    printf ("Name : %s %s\n",first,last);
    printf ("ID   : %s\n",id);
    printf ("Examination points : %f / %f\n",point,full);
    printf ("You passed, Congratulation\n");
}
return 0;
}

```

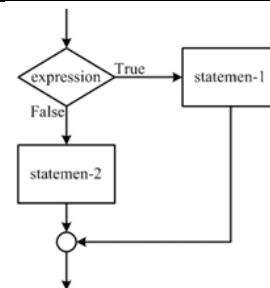
#### 4.4 คำสั่งเงื่อนไข if-else

รูปแบบที่ 1

```

if( expression )
    statement1;
else
    statement2;

```

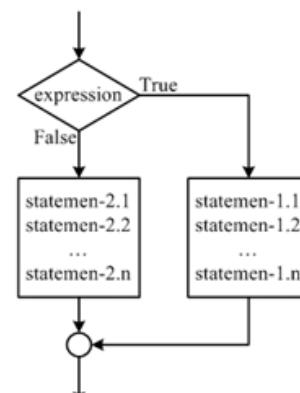


รูปแบบที่ 2 ใช้ปีกการขยาย statement

```

if( expression )
{
    statement1.1;
    statement1.2;
    statement1.3;
}
else
{
    statement2.1;
    statement2.2;
    statement2.3;
}

```



### โปรแกรม 4.3 หารเลข 2 จำนวน

จะเขียนผังงานและโปรแกรมหารเลข 2 จำนวน โดยโปรแกรมต้องตรวจสอบได้ว่าตัวหารเป็น "0" หรือไม่

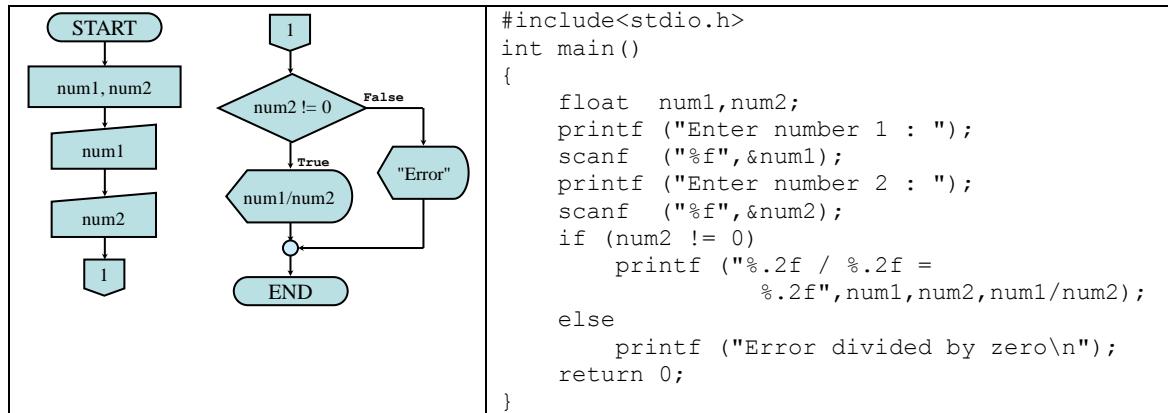
#### 1) Output analysis

แสดงผลหารของเลข 2 จำนวน

แสดงผลว่าไม่สามารถหารได้ เพราะตัวหารเป็นศูนย์

#### 2) Input analysis

ตัวตั้ง และตัวหาร



### 4.4 คำสั่งเงื่อนไข if-else ลักษณะผูกกัน

```
//รูปแบบที่ 1
if (expression-1)
    statement-1;
else if (expression-2)
    statement-2;
...
...
else if (expression-m)
    statement-m;
else
    statement-m+1;
```

```
//รูปแบบที่ 2
if (expression-1)
{
    statement-1.1;
    ...
    statement-1.n;
}
else if (expression-2)
{
    statement-2.1;
    ...
    statement-2.n;
}
...
...
else if (expression-m)
{
    statement-m.1;
    ...
    statement-m.n;
}
else
{
    statement-m+1.1;
    ...
    statement-m+1.n;
}
```

### โปรแกรม 4.4 แสดงเกรด A ถึง F

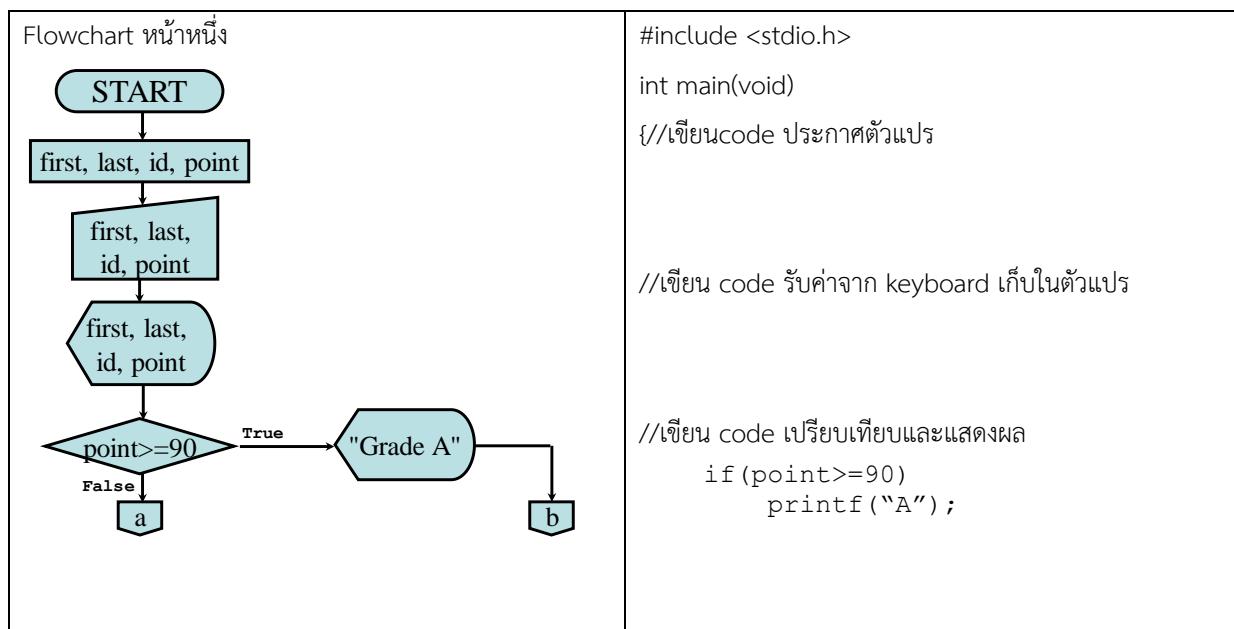
จงเขียนโปรแกรมสำหรับรับชื่อ นามสกุล รหัสนักศึกษา และคะแนนวิชา Computers and Programming เพื่อตรวจสอบว่าบุนเดิมศึกษาได้เกรดระดับใด โดยใช้เกณฑ์ดังนี้

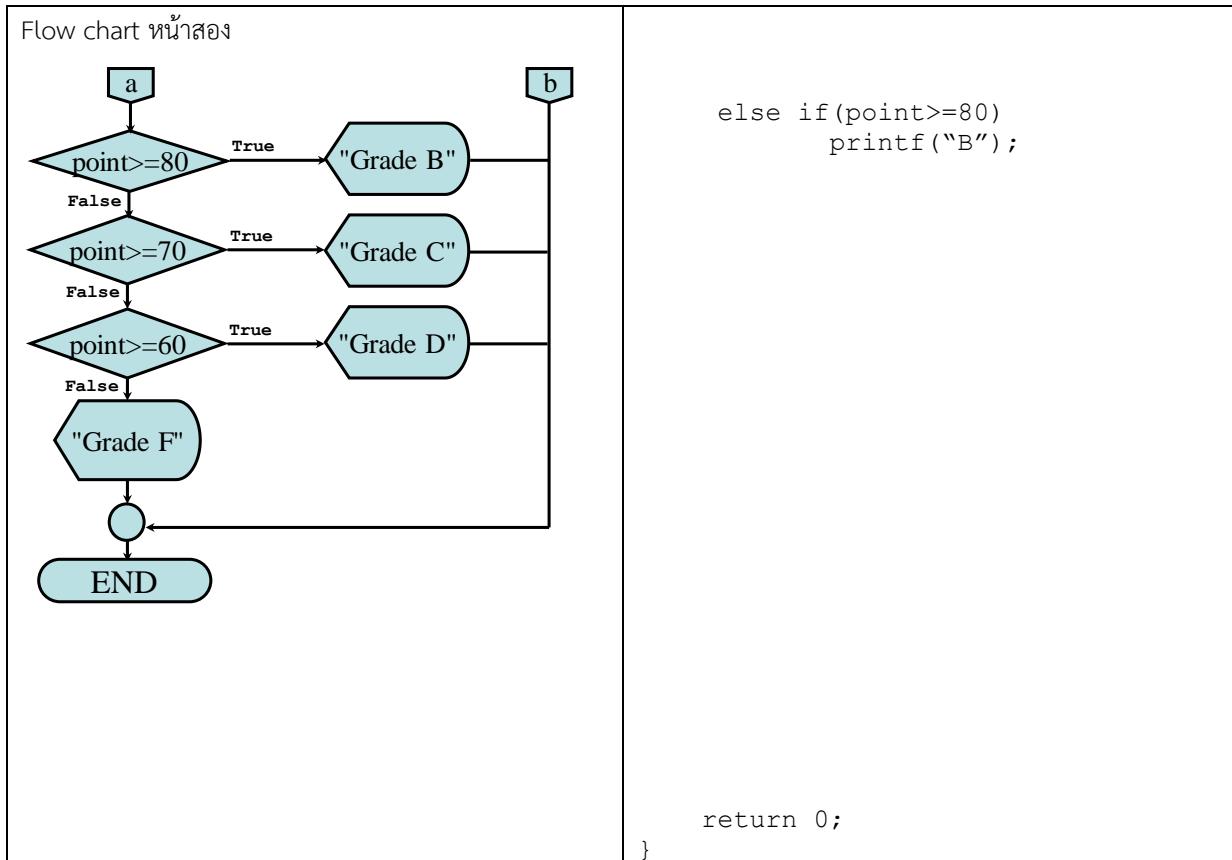
คะแนน 90 – 100	ได้เกรด A	คะแนน 80 – 89.99	ได้เกรด B
คะแนน 70 – 79.99	ได้เกรด C	คะแนน 60 – 69.99	ได้เกรด D
คะแนน 0 – 59.99	ได้เกรด F		

แสดงผลลัพธ์ ชื่อ สกุล รหัสนักศึกษา คะแนน และเกรด เริ่มต้นโดย

- 1) Output analysis แสดงชื่อ นามสกุล รหัสนักศึกษา คะแนน และเกรดที่ได้
- 2) Input analysis ชื่อ / นามสกุล / รหัสนักศึกษา / คะแนน
- 3) Process analysis
  - โปรแกรมรับชื่อ / นามสกุล / รหัสนักศึกษา / คะแนน
  - แสดงผลชื่อ-สกุล รหัสนักศึกษา และคะแนน
  - ตรวจสอบคะแนน
  - ถ้ามากกว่าหรือเท่ากับ 90 แสดงผลว่าได้เกรด A
  - ถ้ามากกว่าหรือเท่ากับ 80 แสดงผลว่าได้เกรด B
  - ถ้ามากกว่าหรือเท่ากับ 70 แสดงผลว่าได้เกรด C
  - ถ้ามากกว่าหรือเท่ากับ 60 แสดงผลว่าได้เกรด D
  - ถ้าไม่ตรงเงื่อนไขที่ผ่านมาทั้งหมด แสดงผลว่าได้เกรด F
- 4) Variable define

first : ตัวแปรชนิดข้อความสำหรับเก็บชื่อขนาด 20  
 last : ตัวแปรชนิดข้อความสำหรับเก็บนามสกุลขนาด 20  
 id : ตัวแปรชนิดข้อความสำหรับเก็บรหัสนักศึกษาขนาด 9  
 point : ตัวแปรชนิดจำนวนเต็มสำหรับเก็บคะแนน





#### 4.4 คำสั่ง switch case

<pre> switch (expression-1) { case constant-expr-1:     statement-1.1;     statement-1.2;     ...     statement-1.n;     break; case constant-expr-2:     statement-2.1;     statement-2.2;     ...     statement-2.n;     break; default:     statement-d.1;     statement-d.2;     ...     statement-d.n; } ...   </pre>	<p>//ตัวอย่างโปรแกรมแสดงการใช้งาน switch case</p> <pre> #include&lt;stdio.h&gt; int main() {     int d;     printf("Enter a number from 1 to 9: ");     scanf("%d", &amp;d);     switch (d)     {         case 1: puts("A stitch in time saves nine.");         break;         case 2: //กรณีติมคำสั่ง break;         case 6: //กรณีติมคำสั่ง break;         case 9: puts("Handsome is as handsome does.");         break;         default: puts("Very clever. Try again.");     }     return 0; }   </pre>
--	---

คำสั่ง

break คือ คำสั่งที่ใช้ออกจากเงื่อนไขเมื่อพบคำสั่งนี้

default คือ กรณีที่ไม่ตรงกับกรณีในคำสั่ง case ใด และสามารถมีได้เพียง 1 คำสั่ง default สำหรับแต่ละคำสั่ง switch

## 4.5 คำถ้ามห้ายบท

1) จงเขียนโปรแกรมเครื่องคิดเลขที่มีตัวอย่างผลการรันดังต่อไปนี้

Enter Num1 : 3

Enter Num2 : 6

Calculator Menu :

1. +

2. -

3. \*

4. /

5. %

Choose menu : 1

Ans: Num1 + Num2 = 9

2) จงเขียนโปรแกรมเพื่อแยกสาร 5 ชนิดซึ่งแต่ละชนิดมีคุณสมบัติดังต่อไปนี้

ชนิดที่ 1 มีคาร์บอนเป็นองค์ประกอบ, มี 5 อะตอม, เป็นกําชาด

ชนิดที่ 2 มีคาร์บอนเป็นองค์ประกอบ, มี 6 อะตอม, เป็นของเหลว

ชนิดที่ 3 มีไฮโดรเจนเป็นองค์ประกอบ, มี 6 อะตอม, เป็นกําชาด

ชนิดที่ 4 มีไฮโดรเจนเป็นองค์ประกอบ, มี 4 อะตอม, เป็นของแข็ง

ชนิดที่ 5 เป็นสารชนิดที่ 1 และมี ไฮโดรเจนเป็นองค์ประกอบ

3) มีส่วนของคำสั่ง switch อยู่ให้หาค่า x , y และ z หลังจากผ่านส่วนของคำสั่ง switch นี้ โดยกำหนดให้ x = 1, y = 0 และ

z = 0

```
switch(x%2)
{
    case 0 : x = 2; y = 3;
    case 1 : x = 4; break;
    default : y = 3; x = z;
}
```

## การทดลองที่ 4 การเขียนโปรแกรมแบบกำหนดเงื่อนไข

### วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจหลักการเขียนโปรแกรมกำหนดเงื่อนไขและสามารถใช้งานได้อย่างถูกต้อง

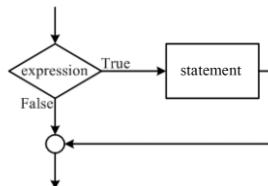
### ทฤษฎี

การกำหนดเงื่อนไขการทำงานต่างๆ ในโปรแกรมจะทำให้โปรแกรมมีความสามารถในการทำงานมากขึ้น ทั้งนี้ โปรแกรมเมอร์ต้องทำความเข้าใจกับตรรกศาสตร์เป็นอย่างมาก เนื่องจากเป็นส่วนที่มีความสำคัญสูงสุด สำหรับการเขียนโปรแกรมกำหนดเงื่อนไขในภาษา C นั้นมีคำสั่งที่สามารถกำหนดเงื่อนไขการทำงานอยู่ 2 คำสั่ง คือ if และ if – else

คำสั่ง                      if

รูปแบบคำสั่ง        if (expression) statement;

ผังงาน



รายละเอียดการใช้งานคำสั่ง

เป็นคำสั่งที่ใช้ในการกำหนดเงื่อนไขว่าจะให้โปรแกรมทำงาน Statement หรือไม่ โดย

ถ้า expression มีผลลัพธ์เป็น TRUE จะทำงาน Statement

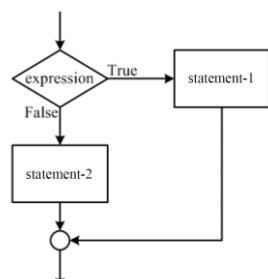
ถ้า expression มีผลลัพธ์เป็น FALSE จะไม่ทำงาน Statement

การใช้งานจึงใช้ในสถานการณ์ที่โปรแกรมเมอร์ต้องการให้โปรแกรม ทำ หรือ ไม่ทำ statement โดยโปรแกรมเมอร์ ต้องเป็นผู้กำหนดเงื่อนไขเอง

คำสั่ง                      if-else

รูปแบบคำสั่ง        if (expression) statement-1; else statement-2;

ผังงาน



รายละเอียดการใช้งานคำสั่ง

สำหรับคำสั่ง if – else จะมีความแตกต่างจากคำสั่ง if เล็กน้อย โดยคำสั่ง if – else จะมีการทำงานในลักษณะของ การเลือกทำ ระหว่าง Statement1 กับ Statement2 ดังนั้น หมายความว่าในคำสั่ง if – else นั้น จะต้องมีการทำงานของ Statement ใด Statement หนึ่งแน่นอน

### เทคนิคการใช้งานคำสั่ง

ในการเขียนโปรแกรมคำสั่ง if และ if – else นั้นจะมีรูปแบบการเขียนโปรแกรมที่ช่วยให้เราสามารถเขียนโปรแกรมได้ง่ายขึ้น และสามารถรองรับการกำหนดเงื่อนไขที่ซับซ้อนมากๆ ได้ โดยมีเทคนิค 2 เทคนิค คือ

#### 1. การเขียนโปรแกรมให้มีการทำงาน statement มากกว่า 1 statement

เนื่องจากรูปแบบของคำสั่ง if และ if – else อนุญาตให้มีการกำหนด statement ที่จะทำงานได้เพียง statement เดียวต่อคำสั่ง if หรือ if – else หนึ่งคำสั่ง แต่ในการเขียนโปรแกรมเราจะพบว่ามีบอยครั้งที่จะต้องมีการทำงานมากกว่า 1 statement เช่นกรณีการโอนเงินจะต้องมีการกำหนดเงื่อนไขคือถ้ามีการโอนเงิน ต้องมีการถอนจากบัญชีต้นทาง ไปฝ่ายบัญชีปลายทาง ซึ่งถ้าเงื่อนไขการโอนสมบูรณ์ เราต้องมีการทำงาน 2 การทำงานพร้อมๆ กัน ไม่สามารถแยกการการทำงานทั้งสองออกจากกันได้ การทำงานเมื่อมีการโอนเงินคือ

1. จำนวนเงินฝากบัญชี A = จำนวนเงินฝากบัญชี A – เงินโอน ;
2. จำนวนเงินฝากบัญชี B = จำนวนเงินฝากบัญชี B + เงินโอน ;

ซึ่งในเชิงการเขียนโปรแกรมโดยใช้คำสั่ง if แล้ว เราจะไม่สามารถปรับยอดบัญชีทั้งสองได้โดยใช้ Statement เพียง 1 Statement เท่านั้น จากตัวอย่างคำสั่ง

If (รายละเอียดการโอนเงินจากบัญชี A ไปยังบัญชี B สมบูรณ์)

    จำนวนเงินฝากบัญชี A = จำนวนเงินฝากบัญชี A – เงินโอน ;

    จำนวนเงินฝากบัญชี B = จำนวนเงินฝากบัญชี B + เงินโอน ;

ถ้าเขียนติดกันตามตัวอย่าง จะมีการทำงานที่ผิดพลาด นั่นคือ จะมีคำสั่ง 2 คำสั่งที่ทำงานไม่สัมพันธ์กัน คือ

If (รายละเอียดการโอนเงินจากบัญชี A ไปยังบัญชี B สมบูรณ์)

    จำนวนเงินฝากบัญชี A = จำนวนเงินฝากบัญชี A – เงินโอน ;

และคำสั่ง

    จำนวนเงินฝากบัญชี B = จำนวนเงินฝากบัญชี B + เงินโอน ;

เขียนติดๆ กัน ทำให้การทำงานผิดพลาด เพราะ ไม่ว่ารายละเอียดการโอนเงินจากบัญชี A ไปยังบัญชี B จะเป็น TRUE หรือ FALSE จำนวนเงินฝากในบัญชี B ก็จะถูกปรับยอดเพิ่มทันที ซึ่งผิดกระบวนการอย่างยิ่ง

ในภาษา C ได้กำหนดรูปแบบให้สามารถแก้ปัญหาดังกล่าวได้โดยการกำหนดให้ การเขียน Statement หลายๆ Statement ในวงเล็บปีกกา คอมไพล์เลอร์จะถือว่าเป็น 1 Statement นั่นคือ

{

    Statement;

    Statement;

.....

}

เราจึงสามารถเขียนเงื่อนไขการทำงานในการโอนเงินได้ดังนี้

```
If (รายละเอียดการโอนเงินจากบัญชี A ไปยังบัญชี B สมบูรณ์)
```

```
{
```

```
    จำนวนเงินฝากบัญชี A = จำนวนเงินฝากบัญชี A - เงินโอน ;
```

```
    จำนวนเงินฝากบัญชี B = จำนวนเงินฝากบัญชี B + เงินโอน ;
```

```
}
```

ดังนั้นการเขียนโปรแกรมในคำสั่ง if หรือ if – else ให้สามารถทำงานกับ statement มากกว่า 1 statement สามารถทำได้ดังรูปแบบต่อไปนี้

if Statement	if – else Statement
<pre>if (expression) {     statement-1.1;     statement-1.2;     ...     statement-1.n; }</pre>	<pre>if (expression) {     statement-1.1;     statement-1.2;     ...     statement-1.n; } else {     statement-2.1;     statement-2.2;     ...     statement-2.n; }</pre>

## 2 การเขียนโปรแกรมเพื่อกำหนดเงื่อนไขแบบซับซ้อน มากกว่า 1 ชั้น

เงื่อนไขบางเงื่อนไข จะเป็นเงื่อนไขที่ซับซ้อน ไม่สามารถเขียนใน expression เพียง expression เดียวได้โดยเฉพาะ การคำนวณทางคณิตศาสตร์ที่มีการทำงานในแต่ละช่วงเวลา หรือ การคิดคำนวณเป็นลำดับชั้น เช่นกรณีการคำนวณเกรดเฉลี่ยของนักศึกษา เป็นต้น

ในการเขียนโปรแกรมเพื่อกำหนดเงื่อนไขแบบซับซ้อนมากกว่า 1 ชั้น คอมไพเลอร์อนุญาตให้ทำได้เนื่องจากคอมไพล์ เลอร์รอมว่าคำสั่ง if หรือคำสั่ง if – else ก็เป็น statement เช่นเดียวกัน ดังตัวอย่าง

1. if ( expression ) statement\_A; else statement\_B;

2. เมื่อเราแทนค่า statement\_B ด้วย if ( expression ) statement\_C; else statement\_D; จะได้  
If ( expression1 ) statement\_A; else if ( expression2 ) statement\_C; else statement\_D;

3. เมื่อเราแทนค่า statement\_D ด้วย if ( expression3 ) statement\_E; จะได้  
If ( expression1 ) statement\_A; else if ( expression2 ) statement\_C; else if ( expression3 ) statement\_E; เป็นต้น

ดังนั้นเราสามารถเขียนคำสั่ง if – else ให้อยู่ในรูปแบบที่มีความซับซ้อน มีการคำนวณ expression ได้เป็นลำดับ ชั้น ได้ดังตัวอย่าง

```
If (expression_A) statement_A;
else if (expression_B) statement_B ;
else if (expression_C) statement_C ;
else .....
```

ตัวอย่างการใช้งาน

1. กรณีที่ต้องการเขียนคำสั่งเพื่อให้แสดงข้อความ “High Temperature” เมื่อตัวแปร temperature มากรกว่า 100 สามารถเขียนคำสั่งได้ดังนี้

```
If (temperature > 100) printf ("High temperature!!!\n");
```

2. กรณีที่ต้องการเขียนคำสั่งเพื่อให้แสดงรายงานความสัมพันธ์ระหว่างตัวแปร a กับตัวแปร b สามารถเขียนคำสั่งได้ดังนี้

```
If (a > b ) printf ("A is greater than B");
else if (a == b) printf ("A and B are equal");
else printf("B is greater than A");
```

3. ตัวอย่างการเขียนโปรแกรมที่มีการใช้ if – else ที่มีการคำนวณ expression หลายชั้นเข็น คำสั่งที่ใช้ในการคำนวณคะแนน เพื่อแสดงเกรดของนักศึกษา ดังโปรแกรมตัวอย่าง

```
#include <stdio.h>
Int main()
{
    int point ;
    printf ("Enter your point : ");
    scanf("%d",&point) ;
    if (point >= 80) printf("%d point is Grade A",point);
    else if (point >= 70) printf("%d point is Grade B",point);
    else if (point >= 60) printf("%d point is Grade C",point);
    else if (point >= 50) printf("%d point is Grade D",point);
    else printf("F");
    printf ("\n\nPress ENTER to end Program.");
    return 0;
}
```

**การทดลองที่ 4.1 จัดตัวอย่างการใช้งานคำสั่ง ให้นักศึกษาประยุกต์ใช้งานดังต่อไปนี้**

```
1 #include<stdio.h>
2 int main()
3 {
4     int a=97, b=46;
5     if(a>b)
6         printf ("%d , %d => max = %d\n",a,b,a);
7     else
8         printf ("%d , %d => max = %d\n",a,b,b);
9     return 0;
10 }
```

## 4.1.1 ผลลัพธ์ที่ได้คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## 4.1.2 บรรทัดที่ 4 เปลี่ยน int a=97,b=46; เป็น int a=30,b=46; ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## 4.1.3 บรรทัดที่ 5 ทดลองเปลี่ยน if(a&gt;b) เป็น if(b&lt;a) ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## 4.1.4 บรรทัดที่ 5 ทดลองเปลี่ยน if(a&gt;b) เป็น if(a&lt;b) ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

ในข้อนี้จะสังเกตว่า ผลลัพธ์ที่ได้ไม่ใช่ผลลัพธ์ที่ถูกต้อง แต่โปรแกรมนี้ คอมไพล์ผ่าน และสามารถรันได้ โดยไม่มีปัญหาแต่อย่างใด ข้อผิดพลาดประเภทนี้ เรียกว่า **Logical Error**

## การทดลองที่ 4.2 เครื่องคิดเลข การบวกและการลบ

```

1 #include<stdio.h>
2 int main()
3 {
4     float a, b;
5     int c;
6     printf("Enter 2 numbers : ");
7     scanf("%f%f", &a, &b);
8     putchar('\n');
9     printf("      1. +\n");
10    printf("      2. -\n");
11    printf("Select operator (1 or 2) : ");
12    scanf("%d", &c);
13    if(c==1)
14        printf("%.2f + %.2f = %.2f\n", a, b, a+b);
15    else
16        printf("%.2f - %.2f = %.2f\n", a, b, a-b);
17    return 0;
18 }
```

## 4.2.1 ใส่ข้อมูลตามตัวอย่าง

```
C:\Windows\system32\cmd.exe
Enter 2 numbers : 9 5
1. +
2. -
Select operator <1 or 2> : 1
```

ผลลัพธ์ที่ได้

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## 4.2.2 ใส่ข้อมูลตามตัวอย่าง

```
C:\Windows\system32\cmd.exe
Enter 2 numbers : 9 5
1. +
2. -
Select operator <1 or 2> : 2
```

ผลลัพธ์ที่ได้

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## การทดลองที่ 4.3 ความแตกต่างระหว่างคำสั่ง if และ if . . . else . . .

```
1 #include<stdio.h>
2 int main()
3 {
4     int a=10, b=5, c=3;
5     if (a>b)
6         printf("a>b ");
7     if (b>c)
8         printf("b>c ");
9     if (c>a)
10        printf("c>a ");
11    printf("\n\n");
12    if (a>b)
13        printf("a>b ");
14    else if (b>c)
15        printf("b>c ");
16    else if (c>a)
17        printf("c>a ");
18    printf("\n\n");
19 }
20 }
```

## 4.3.1 ผลลัพธ์ที่ได้คือ


4.3.2 ผลลัพธ์ที่ได้ ทั้งสองบรรทัดมีความแตกต่างกัน เพราะว่าอะไร

---

---

---

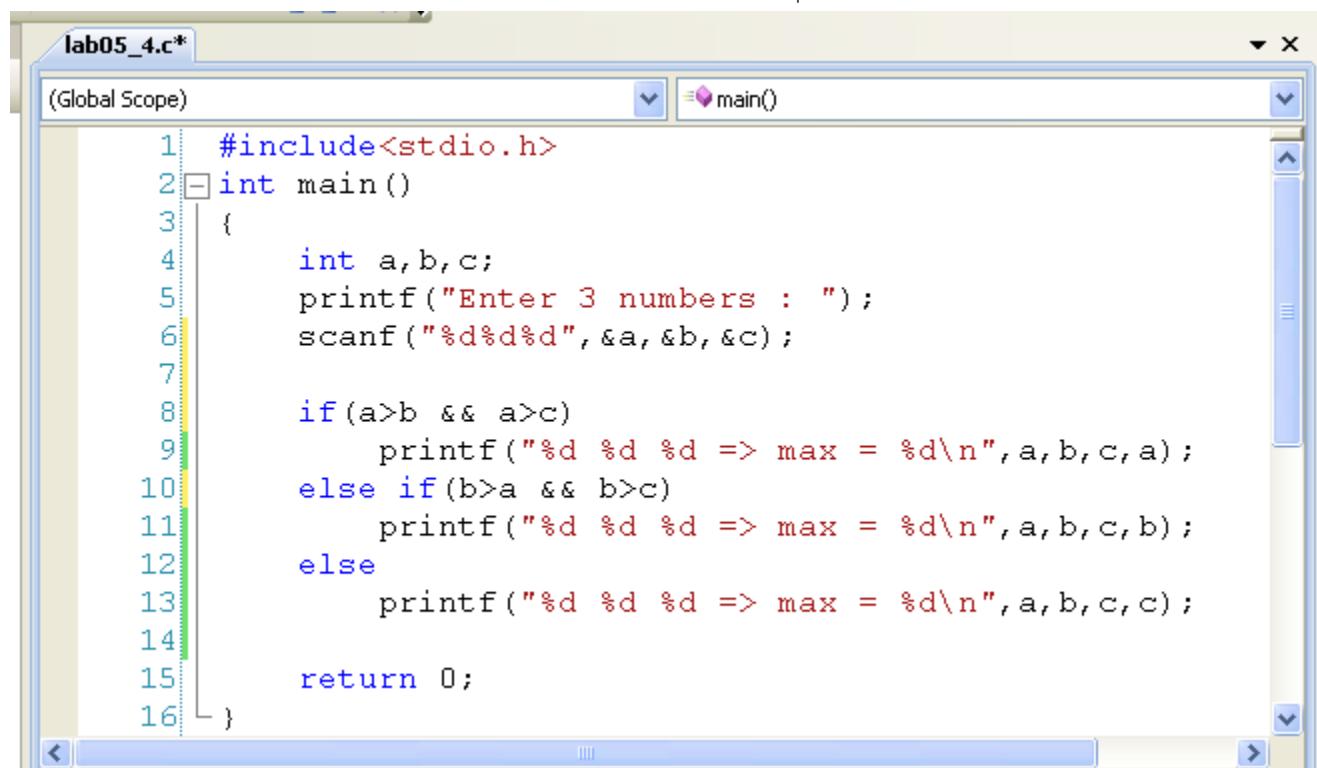
---

---

---

#### การทดลองที่ 4.4 การเปรียบเทียบตัวเลข 3 จำนวน

โปรแกรมต่อไปนี้ เป็นโปรแกรมรับเลขจำนวนเต็ม 3 จำนวนและหาตัวเลขที่มากที่สุด



```

lab05_4.c*
(Global Scope) main()
1 #include<stdio.h>
2 int main()
3 {
4     int a, b, c;
5     printf("Enter 3 numbers : ");
6     scanf("%d%d%d", &a, &b, &c);
7
8     if(a>b && a>c)
9         printf("%d %d %d => max = %d\n", a, b, c, a);
10    else if(b>a && b>c)
11        printf("%d %d %d => max = %d\n", a, b, c, b);
12    else
13        printf("%d %d %d => max = %d\n", a, b, c, c);
14
15    return 0;
16 }

```

4.4.1 ป้อนอินพุท 3 4 5 ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

4.4.2 ป้อนอินพุท 3 5 4 ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

4.4.3 ป้อนอินพุท 5 3 4 ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

4.4.4 ป้อนอินพุท 9 9 1 ได้ผลลัพธ์คือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

จะเห็นว่า ผลลัพธ์ที่ได้จากข้อ 4.4.1 – 4.4.3 เป็นผลลัพธ์ที่ถูกต้อง แต่ผลลัพธ์ที่ได้จาก 4.4.4 เป็นผลลัพธ์ที่ไม่ถูกต้อง เกิด Logical Error

4.4.5 เปลี่ยนบรรทัดที่ 8 เป็น if( $a >= b \&& a >= c$ ) และป้อนอินพุต 9 9 1 จะได้ผลลัพธ์ถูกต้องคือ

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

### การทดลองที่ 4.5 การเปรียบเทียบตัวเลขจำนวนทศนิยม

```

1 #include<stdio.h>
2 int main()
3 {
4     float a=1.9,b=0.4,c=1.5;
5     float x=1.5,y=0.125,z=1.375;
6     printf("a=% .2f \tor % .10f\n",a,a);
7     printf("b=% .2f \tor % .10f\n",b,b);
8     printf("c=% .2f \tor % .10f\n",c,c);
9     printf("a-b == c --> %d\n",a-b==c);
10    printf("x=% .2f \tor % .10f\n",x,x);
11    printf("y=% .2f \tor % .10f\n",y,y);
12    printf("z=% .2f \tor % .10f\n",z,z);
13    printf("x-y == z --> %d\n",x-y==z);
14
15    return 0;
16 }

```

บันทึกผลลัพธ์


4.5.1 ผลลัพธ์จากบรรทัดที่ 9 ได้ค่าเป็น \_\_\_\_\_ เป็นจริงหรือเท็จ \_\_\_\_\_

4.5.2 ผลลัพธ์จากบรรทัดที่ 13 ได้ค่าเป็น \_\_\_\_\_ เป็นจริงหรือเท็จ \_\_\_\_\_

4.5.3 จากการทดลองนี้ เครื่องหมาย == และ != ควรใช้เปรียบเทียบข้อมูลทศนิยมหรือไม่ \_\_\_\_\_

4.5.4 จากข้อ 4.5.3 เพราะอะไร

---



---



---



---

### การทดลองที่ 4.6 : ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง

- 1) จงเขียนโปรแกรมเพื่อรับค่าตัวเลขจำนวนเต็ม 3 จำนวน และแสดงผลลัพธ์ตัวเลขที่น้อยที่สุด (ดูแนวทางจากการทดลองที่ 4.4)

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 3 5 7
3 5 7 => min = 3
```

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 9 1 9
9 1 9 => min = 1
```

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 9 9 1
9 9 1 => min = 1
```

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 0 0 7
0 0 7 => min = 0
```

- 2) จงเขียนโปรแกรมเพื่อรับค่าตัวเลขจำนวนเต็ม 3 จำนวน และแสดงผลลัพธ์ตัวเลขกลางระหว่างเลขที่มากที่สุดกับเลขที่น้อยที่สุด (ดูแนวทางจาก การทดลองที่ 4.4)

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 3 5 7
3 5 7 => mid = 5
```

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 1 9 1
1 9 1 => mid = 1
```

```
C:\WINDOWS\system32\cmd.exe
Enter 3 numbers : 0 0 7
0 0 7 => mid = 0
```

- 3) จงเขียนโปรแกรมเพื่อรับค่าเก็บไว้ในตัวแปรเป็นตัวเลข 1 จำนวนแล้วแสดงผลลัพธ์ว่าตัวเลขที่รับเข้ามาเป็นเลขคู่หรือเลขคี่

```
C:\WINDOWS\system32\cmd.exe
Enter a number : 21
21 is ODD number.
```

```
C:\WINDOWS\system32\cmd.exe
Enter a number : 64
64 is EVEN number.
```

4) จากการทดลอง 4.2 ให้แก้ไขโปรแกรมให้สามารถเลือก + - \* / ได้ ดังตัวอย่าง

```
C:\Windows\system32\cmd.exe
Enter 2 numbers : 9 5
1. +
2. -
3. *
4. /
Select operator <1,2,3,4> : 4
9.00 / 5.00 = 1.80
```

```
C:\Windows\system32\cmd.exe
Enter 2 numbers : 9 5
1. +
2. -
3. *
4. /
Select operator <1,2,3,4> : 3
9.00 * 5.00 = 45.00
```

5) จงเขียนโปรแกรมเพื่อรับตัวเลขจำนวนเต็ม 3 จำนวน แล้วแสดงผลลัพธ์เป็นผลบวกของตัวเลขที่มากที่สุดกับตัวเลขที่น้อยที่สุด

```
C:\Windows\system32\cmd.exe
Enter 3 numbers : 1 2 3
1 2 3 => max+min = 4
```

```
C:\Windows\system32\cmd.exe
Enter 3 numbers : 1 9 7
1 9 7 => max+min = 10
```

```
C:\Windows\system32\cmd.exe
Enter 3 numbers : 9 3 7
9 3 7 => max+min = 12
```

```
C:\Windows\system32\cmd.exe
Enter 3 numbers : 9 1 1
9 1 1 => max+min = 10
```

6) จงเขียนโปรแกรมเพื่อรับค่าตัวเลขเป็นตัวเลข 3 จำนวน แล้วแสดงผลลัพธ์ว่าถ้าตัวเลขทั้งสาม เป็นความยาวของด้านของสามเหลี่ยม สามเหลี่ยมรูปนั้นจะเป็นสามเหลี่ยมนูนจากหรือไม่

- ด้านที่ยาวที่สุด ต้อง มีความยาวน้อยกว่า อีกสองด้านมากกัน
- สามเหลี่ยมนูนจาก ด้านยาวที่สุดยกกำลังสอง เท่ากับ ผลบวกของกำลังสองด้านประกอบมุมจาก

## บทที่ 5 การเขียนโปรแกรมแบบวนซ้ำ

### วัตถุประสงค์การศึกษา

- นักศึกษาเข้าใจกระบวนการทำงานแบบวนซ้ำ
- นักศึกษาสามารถเขียนโปรแกรมภาษาซีโดยใช้คำสั่งเพื่อให้คอมพิวเตอร์ทำงานแบบวนซ้ำได้

### 5.1 บทนำ

คำสั่งการวนซ้ำด้วย while

คำสั่งการวนซ้ำด้วย do - while

คำสั่งการวนซ้ำด้วย for

ทำไม่ต้องการเรียนโปรแกรมต้องมีการวนซ้ำ เพราะ เหตุการณ์ที่เกิดขึ้นหลายรอบ

เช่น โปรแกรมแสดง ซึ่อ 20 ครั้ง

เหตุการณ์ที่เกิดขึ้นหลายรอบ โดยมีการเปลี่ยนแปลงค่า หรือมีเงื่อนไข

เช่น แสดงผลเลข 0, 1, 2, ..., 10

แสดงผลรวมของ 1,3,5,7, ..., 99

แสดง ซึ่อ ไปเรื่อย ๆ จนกว่าค่า X จะมากกว่า 30

### โปรแกรม 5.1 ไม่มีการวนซ้ำ

จะเขียนผังงานและโปรแกรมเพื่อแสดงตัวเลข 0 - 10 ออกทางหน้าจอ

- |                     |                                      |
|---------------------|--------------------------------------|
| 1) Output analysis  | แสดงผลเลข 0, 1, 2, ..., 10           |
| 2) Input analysis   | ไม่มี                                |
| 3) Process analysis | โปรแกรมแสดงผลเลข 0, 1, 2, ..., 10    |
| 4) Variable define  | ไม่ใช้ (หรือใช้ count เพื่อเพิ่มค่า) |

```
#include<stdio.h>
#include<conio.h>
int main()
{
    printf ("0\t");
    printf ("1\t");
    printf ("2\t");
    printf ("3\t");
    printf ("4\t");
    ...
    ...
    printf ("10\t");
    return 0;
}
```

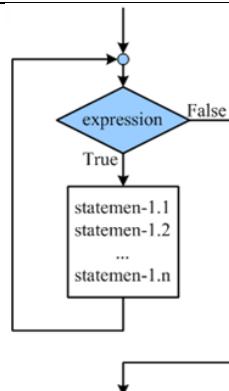
```
#include<stdio.h>
#include<conio.h>
int main()
{
    int count = 0;
    printf ("%d\t",count++);
    printf ("%d\t",count++);
    printf ("%d\t",count++);
    printf ("%d\t",count++);
    printf ("%d\t",count++);
    ...
    ...
    printf ("%d\t",count++);
    return 0;
}
```

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int count = 0;
    while (count <= 10) //คำสั่งวนซ้ำ
    {
        printf ("%d\t", count++);
    }
    return 0;
}
```

## 5.2 คำสั่ง while

```
//รูปแบบที่ 1
while(expression)
    statement-1;

//รูปแบบที่ 2 ใช้ปักกษาข่าย statement
while(expression)
{
    statement-1.1;
    statement-1.2;
    statement-1.3;
}
```

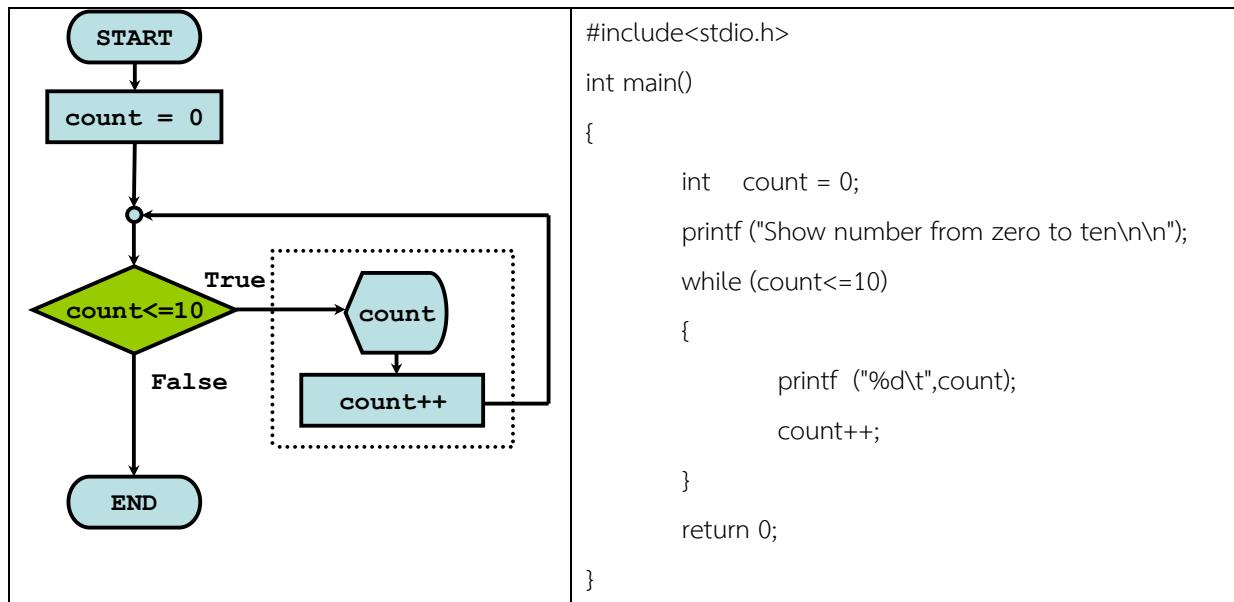


คำสั่งวนซ้ำแบบ while เป็นคำสั่งวนซ้ำที่มีการตรวจสอบเงื่อนไขก่อนการวนซ้ำ หากเงื่อนไขการวนซ้ำ แต่หากเป็นเท็จจะไม่เกิดการวนซ้ำ ซึ่งสามารถใช้ได้ในกรณีที่รู้จำนวนรอบ เช่น วนซ้ำตั้งแต่  $count = 1$  ถึง  $count = 5$  หรือในกรณีที่ไม่รู้จำนวนรอบที่แน่นอน เช่น ทำซ้ำเมื่อตัวแปร choice  $\neq 0$  โดยในการเขียนโปรแกรมควรระมัดระวังในกรณีที่เงื่อนไขในการตรวจสอบเป็นจริงเสมอ จะทำให้เกิดการวนซ้ำไม่รู้จบ (Infinity loop) โดยจะต้องเขียนให้มีทางออกจาก การวนซ้ำได้

### โปรแกรม 5.2 แสดงเลข 0..10 ด้วย while

จะเขียนผังงานและโปรแกรมเพื่อแสดงตัวเลข 0 - 10 ออกทางหน้าจอ

- |                     |  |
|---------------------|--|
| 1) Output analysis  | แสดงผลเลข 0, 1, 2, ..., 10             |
| 2) Input analysis   | ไม่มี                                  |
| 3) Process analysis | โปรแกรมแสดงผลเลข 0, 1, 2, ..., 10      |
| 4) Variable define  | count เป็นจำนวนเต็มเพื่อใช้นับจำนวนรอบ |



### โปรแกรม 5.3 หาผลรวมของเลขใช้ while

จะเขียนผังงานและโปรแกรมที่มีการควบคุมทิศทางแบบวนรอบโดยใช้คำสั่ง while เพื่อให้โปรแกรมทำการบวกเลขจำนวนเต็ม ตั้งแต่ 1 จนถึงค่าที่ผู้ใช้งานกำหนด

1) Output analysis ผลลัพธ์การบวกเลขจำนวนเต็ม ตั้งแต่ 1 ถึงค่าที่ผู้ใช้กำหนด

2) Input analysis ค่าที่ผู้ใช้งานป้อนเข้ามา

3) Process analysis

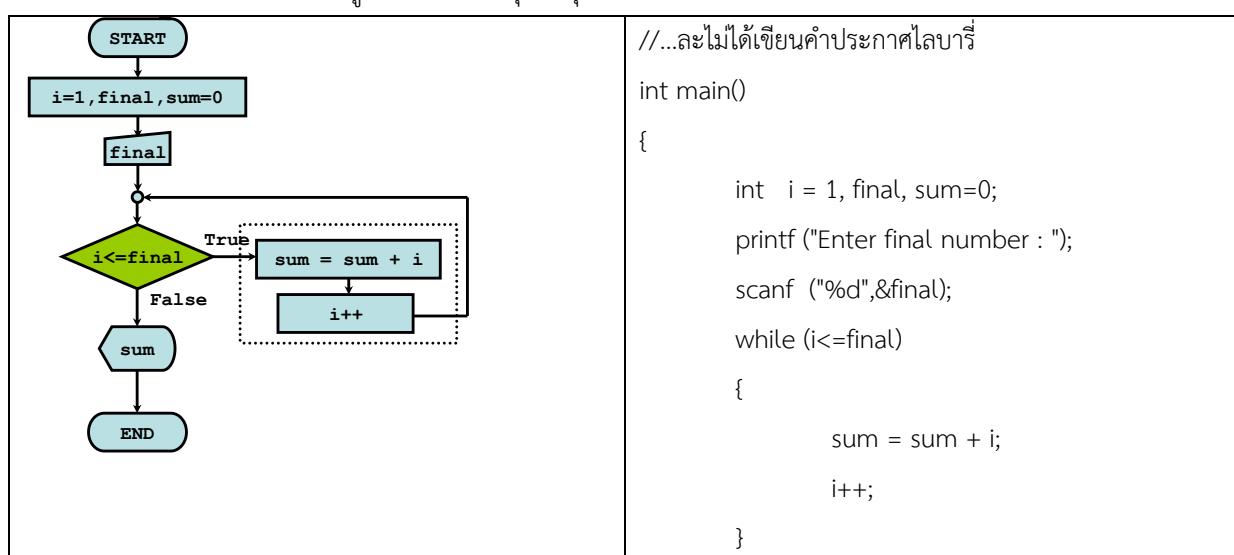
- โปรแกรมถามว่าผู้ใช้งานต้องการบวกเลขตั้งแต่ 1 ถึงเลขใด
- วนรอบแบบ while เพื่อบวกค่า
- แสดงผลลัพธ์ที่ได้

4) Variable define

sum = 0 ผลรวมของการบวก โดยเริ่มต้นมีค่าเท่ากับ 0

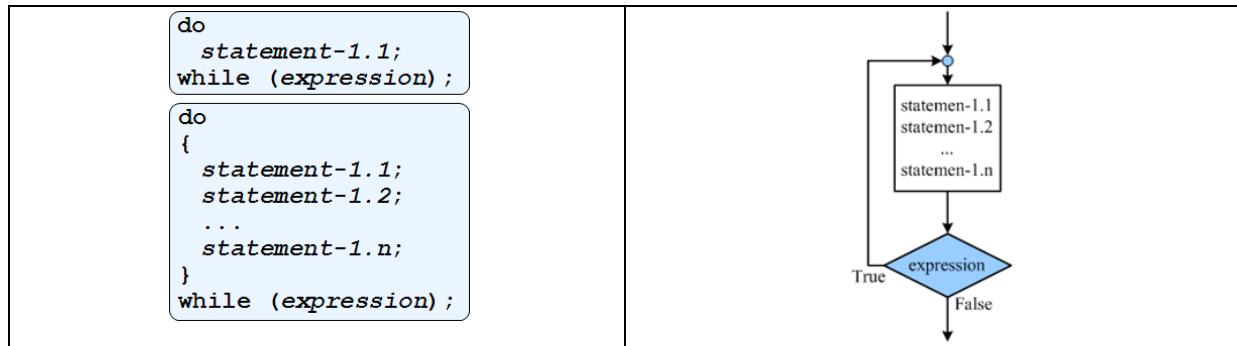
i = 1 ค่าที่นำเข้าไปบวกกับ sum ในแต่ละรอบ โดยรอบแรกค่า i มีค่าเท่ากับ 1 และมีค่าเพิ่มขึ้นรอบละ 1

final เพื่อรับค่าจากผู้ใช้ และกำหนดจุดสิ้นสุดของค่า i



	<pre> printf ("Sum = %d",sum); return 0; } </pre>
--	---

### 5.3 คำสั่ง do-while

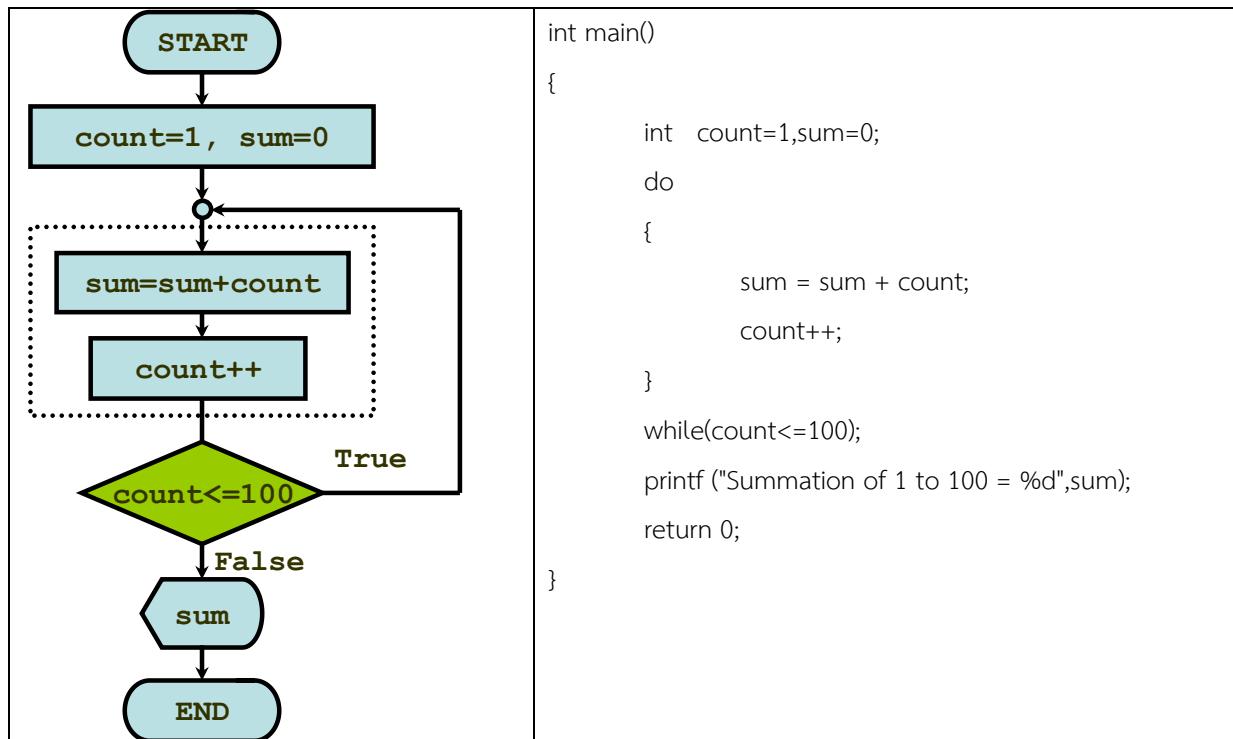


คำสั่งวนซ้ำแบบ do...while เป็นคำสั่งวนซ้ำที่ให้เข้าไปทำการวนซ้ำ ครั้งก่อนจะตรวจสอบเงื่อนไข หากเงื่อนไขการวนซ้ำเป็นจริงจึงจะเกิดการวนซ้ำครั้งถัดไป แต่หากเป็นเท็จจะไม่เกิดการวนซ้ำ

### โปรแกรม 5.4 หาผลรวม 1 ถึง 100 ใช้ do-while

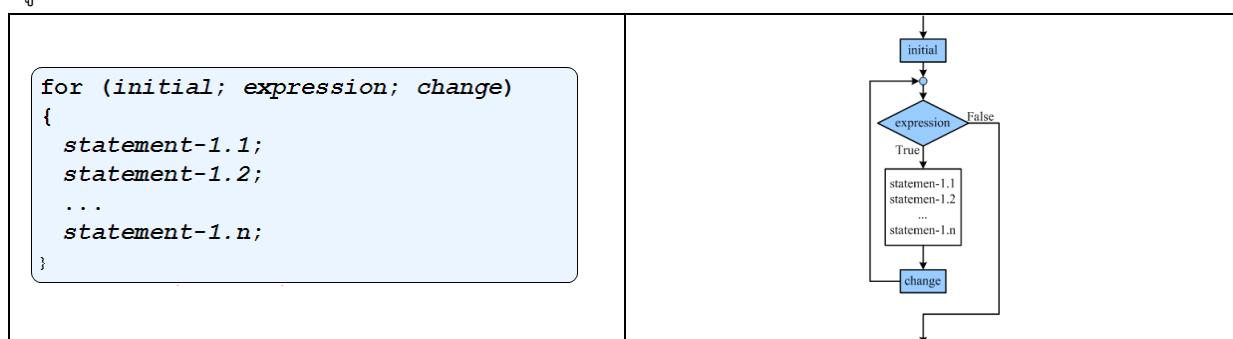
จงเขียนผังงานและโปรแกรมสำหรับรวมเลขจำนวนเต็ม ตั้งแต่ 1 – 100 โดยใช้คำสั่ง do-while

- 1) Output analysis      ผลรวมของเลขจำนวนเต็ม ตั้งแต่ 1 - 100
- 2) Input analysis      'ไม่มี'
- 3) Process analysis      โปรแกรมทำการบวกค่าเก็บไว้ในตัวแปรผลลัพธ์ และเพิ่มค่าจนถึง 100
- 4) Variable define
  - count      เป็นตัวแปรชนิดจำนวนเต็มเพื่อนับจำนวน
  - sum      เป็นจำนวนเต็มเพื่อกำบดค่าผลรวม



## 5.4 คำสั่ง for

รูปแบบ



คำสั่งวนซ้ำแบบ for มีกลไกการทำงานเหมือนคำสั่ง while ต่างกันตรงที่คำสั่ง for จะมีการประกาศส่วนประกอบที่ใช้ในการร่วมซ้ำ อยู่ 3 ส่วนดังนี้

- initial เป็นส่วนที่ใช้กำหนดค่าเริ่มต้นให้กับตัวแปร
- condition เป็นเงื่อนไขเพื่อพิจารณา
- change เป็นส่วนที่เปลี่ยนแปลงค่าตัวแปร

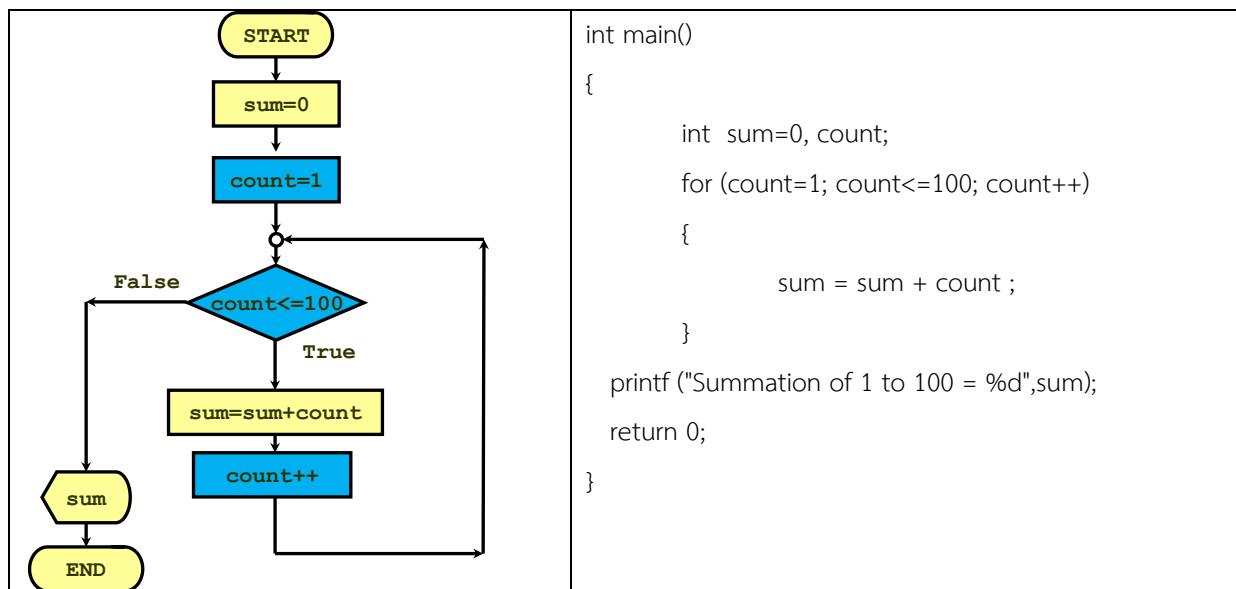
โดยจะมีการกำหนดส่วน initial ในครั้งแรกครั้งเดียว และจึงตรวจสอบเงื่อนไข เมื่อทำคำสั่ง statement และ จะทำคำสั่งในส่วน change จึงจะวนกลับไปตรวจสอบเงื่อนไขอีกครั้งหนึ่ง

## โปรแกรม 5.5 โปรแกรมหาผลรวม 1 ถึง 100 ใช้ for

จงเขียนผังงานและโปรแกรมสำหรับรวมเลขจำนวนเต็ม ตั้งแต่ 1 – 100 โดยใช้คำสั่ง for

- 1) Output analysis ผลรวมของเลขจำนวนเต็ม ตั้งแต่ 1 - 100
- 2) Input analysis ไม่มี
- 3) Process analysis โปรแกรมทำการบวกค่าเก็บไว้ในตัวแปรผลลัพธ์ และเพิ่มค่าจนถึง 100
- 4) Variable define
 

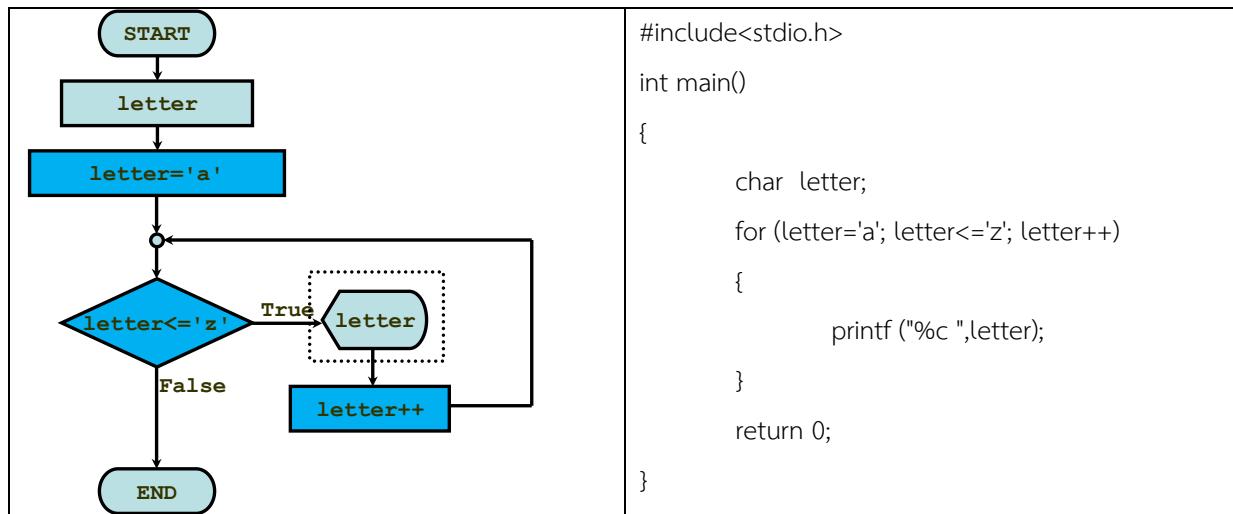
count	เป็นตัวแปรชนิดจำนวนเต็มเพื่อนับจำนวน
sum	เป็นจำนวนเต็มเพื่อกำกับค่าผลรวม



## โปรแกรม 5.6 แสดงผล a - z ใช้ for

จงเขียนผังงานและโปรแกรมสำหรับแสดงผลอักษร a – z ออกทางจอภาพ โดยใช้คำสั่ง for

- 1) Output analysis แสดงผล a – z ทางจอภาพ
- 2) Input analysis ไม่มี
- 3) Process analysis โปรแกรมทำการวนรอบเพื่อแสดงผลอักษรตั้งแต่ a – z โดยการเพิ่มค่าตัวแปรชั้นครั้งละ 1 (ดูตาราง ASCII Code)
- 4) Variable define letter เป็นตัวแปรชนิดอักขระ

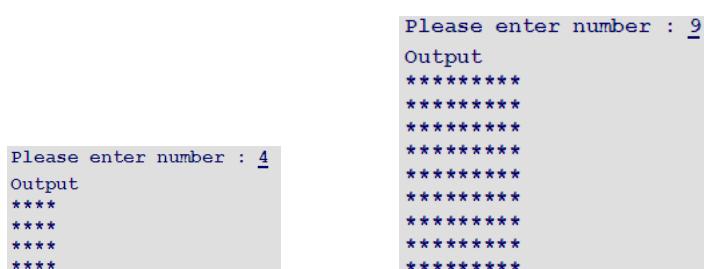


```

#include<stdio.h>
int main()
{
    char letter;
    for (letter='a'; letter<='z'; letter++)
    {
        printf ("%c ",letter);
    }
    return 0;
}
  
```

### โปรแกรม 5.7 แสดงผลรูปสี่เหลี่ยม ใช้ for

จงเขียนผังงานและโปรแกรมแสดงผลรูปสี่เหลี่ยมขนาด  $n \times n$  โดยโปรแกรมจะรอรับจำนวนเต็มจากผู้ใช้งาน ดังตัวอย่าง



- 1) Output analysis ผลตัวเลข เป็นรูปสี่เหลี่ยมจัตุรัสขนาดเท่ากับจำนวนตัวเลขที่รับเข้ามา
- 2) Input analysis เลขจำนวนเต็มที่ผู้ใช้ป้อนเข้ามา
- 3) Process analysis

โปรแกรมรอรับค่าจำนวนเต็มจากผู้ใช้งาน

โปรแกรมวนรอบเพื่อทำการแสดง '\*' เป็นรูปสี่เหลี่ยมจัตุรัส

บรรทัดที่ 1 แสดงผล '\n' และแสดงผล '\*' จำนวนเท่ากับค่าที่รับมา

บรรทัดที่ 2 แสดงผล '\n' และแสดงผล '\*' จำนวนเท่ากับค่าที่รับมา

...

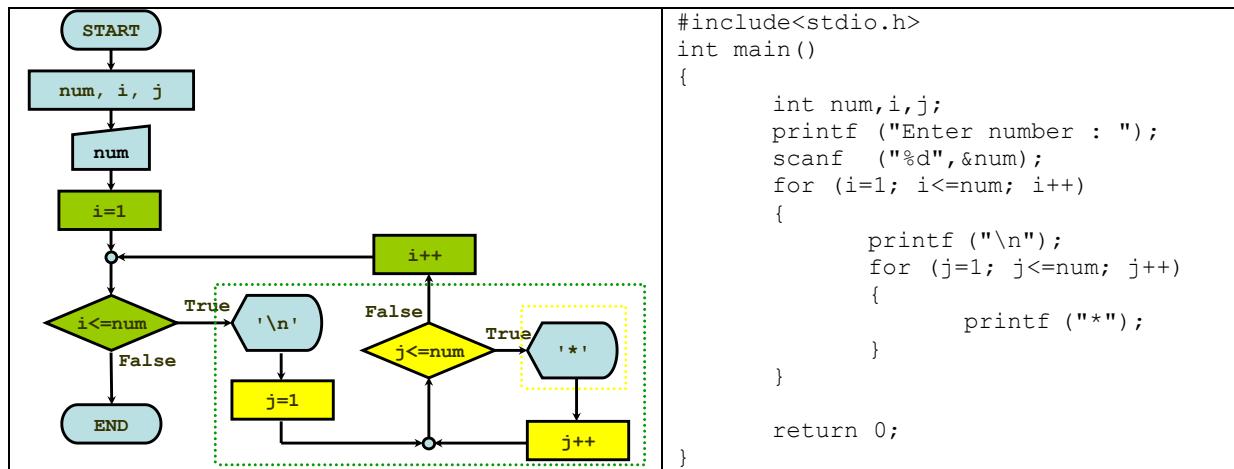
บรรทัดที่  $n$  แสดงผล '\n' และแสดงผล '\*' จำนวนเท่ากับค่าที่รับมา

- 4) Variable define

num เป็นจำนวนเต็มเพื่อใช้เก็บค่าตัวเลขที่ผู้ใช้ป้อน

i เป็นจำนวนเต็มเพื่อใช้นับจำนวนบรรทัด

j เป็นจำนวนเต็มเพื่อใช้นับจำนวน '\*'



## 5.4 คำถ้ามหัยบท

- 1) จงเขียนผังงาน และโปรแกรมคำนวนดอกเบี้ย โดยโปรแกรมรับ เงินต้น (บาท) และดอกเบี้ยต่อปี (%) จำนวนเงินที่ผ่อนชำระต่อเดือน และแสดงผลว่าต้องใช้เวลา กี่ปีในการผ่อนชำระ และจ่ายดอกเบี้ยทั้งหมดคิดเป็นเท่าใด
- 2) จงเขียนโปรแกรมแสดงรหัสแอสกี ตั้งแต่ 33 ถึง 55

Decimal	ASCII
33	!
34	"
35	#
...	
55	7

- 3) จงเขียนโปรแกรมรับตัวเลขเพื่อมาคำนวนหาผลลบหากกำลังสอง จนกระทั่งตัวเลขที่รับเข้ามามีค่าเป็น 0

Enter a number : 2

Enter a number : -5

Enter a number : 0

Result : 29

- 4) ข้อใดเป็นโปรแกรมที่รันไม่รู้จบ (Infinite loop) เมื่อกำหนด int i=0;

- 4.1 for(i=0; i>0; i++) printf("%d",i);
- 4.2 for(i=0; i%2!=0; i += 2) puts("a");
- 4.3 while(i<7) printf("%d",i--);
- 4.4 do {  
    i+=3;  
} while(i%3==0);

## การทดลองที่ 5 การเขียนโปรแกรมแบบวนซ้ำ

### วัตถุประสงค์

1. เพื่อให้นักศึกษาทดลองเขียนโปรแกรมเรียกใช้คำสั่งวนซ้ำ
2. เพื่อให้นักศึกษาสามารถเขียน และเข้าใจหลักการเขียนโปรแกรมแบบวนซ้ำ

### ทฤษฎี

การเขียนโปรแกรมแบบวนซ้ำจะเป็นการเขียนโปรแกรมที่ช่วยให้โปรแกรมเมอร์ไม่จำเป็นต้องมีการพิมพ์ข้อมูลหลาย ๆ บรรทัด สามารถลดขนาดของไฟล์ลงได้มาก และสามารถเพิ่มความเร็วในการทำงานต่างๆ ได้ สำหรับการเขียนโปรแกรมแบบวนซ้ำนั้นจะมีคำสั่งที่เกี่ยวข้องทั้งหมด 3 คำสั่งด้วยกัน คือคำสั่ง for , while และคำสั่ง do ... while ซึ่งคำสั่งทั้งสามคำสั่งสามารถทำงานทดแทนกันได้

เนื่องจากการเขียนโปรแกรมแบบวนซ้ำมีคำสั่งที่เกี่ยวข้องอยู่ 3 คำสั่ง ในการเลือกใช้คำสั่งให้ตรงตามการทำงานของผู้ใช้งานจะทำให้การเขียนโปรแกรมทำได้อย่างสะดวกมากขึ้น โดยการเลือกใช้งานคำสั่งต่างๆ สามารถยืดหยุ่นได้

- 1) เมื่อผู้ใช้งานทราบจำนวนของการวนซ้ำที่แน่นอน ทราบว่าต้องเริ่มที่ค่าเท่าใด จบที่ค่าเท่าใด หรือทราบว่าต้องทำงานนั้นๆ กี่รอบ ควรใช้คำสั่ง for
- 2) เมื่อผู้ใช้งานไม่ทราบจำนวนการทำงานที่แน่นอน แต่ทราบเฉพาะเงื่อนไขการทำงานว่าการทำงานแบบวนซ้ำ ทำงานเมื่อเงื่อนไขเป็นอย่างไร ถ้าไม่ตรงตามเงื่อนไขที่กำหนดไว้ จะหยุดการทำงาน ควรใช้คำสั่ง while
- 3) เมื่อผู้ใช้งานไม่ทราบจำนวนการทำงานที่แน่นอน แต่ทราบเฉพาะเงื่อนไขการทำงานว่าการทำงานแบบวนซ้ำ จะทำงานจนกว่าจะเกิดเงื่อนไขใดขึ้น จึงหยุดทำงาน ควรใช้คำสั่ง while

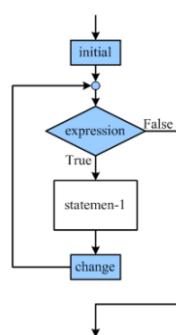
คำสั่ง

or

รูปแบบคำสั่ง

for (initial; expression; change) statement;

ผังงาน



รายละเอียดการใช้งานคำสั่ง

คำสั่ง for เป็นคำสั่งให้ทำซ้ำหรือวนรอบ เมื่อเงื่อนไขสอดคล้องหรือ เงื่อนไขเป็นจริง จะทำงานในขอบข่ายที่กำหนดไว้ และจะเลิกกระทำซ้ำหรือวนรอบ เมื่อเงื่อนไขไม่สอดคล้องหรือ เงื่อนไขเป็นเท็จ เช่น ถ้าต้องการให้พิมพ์คำว่า Engineer จำนวน 10 ครั้ง เราสามารถเขียนคำสั่งได้ดังนี้

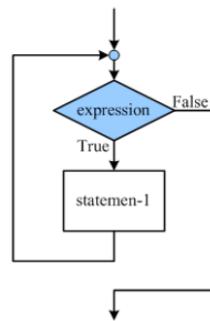
For ( i=1 ; i<=10 ; i++ ) printf("Engineer\n");

จากรูปแบบคำสั่งเราจะได้ความสัมพันธ์จากตัวอย่างดังนี้

Initial	คือ $i = 1$
Expression	คือ $i < 10$
Change	คือ $i++$
Statement	คือ <code>printf ("Engineer\n")</code>

จากผังงานเราจะได้รูปแบบและขั้นตอนการทำงานคือ ก่อนจุดเริ่มการทำงานแบบวนซ้ำ จะมีการตั้งค่า  $i$  มีค่าเท่ากับ 1 หลังจากนั้นจึงมีการตรวจสอบว่าค่า  $i$  น้อยกว่าหรือเท่ากับ 10 หรือไม่ ถ้า  $i$  น้อยกว่าหรือเท่ากับ 10 เป็นจริงจะแสดงผลคำว่า Engineer หนึ่งครั้ง และวิจัยมีการเพิ่มค่า  $i$  ที่ลิ๊ง 1 และกลับไปพิจารณาที่จุดเริ่มของการทำงานแบบวนซ้ำอีกครั้ง การทำงานจะเป็นไปอย่างนี้เรื่อยๆ จนกว่าค่า  $i$  จะมากกว่า 10 ค่า expression จะเป็น False และจบการทำงานของคำสั่งนี้

คำสั่ง	<code>while</code>
รูปแบบคำสั่ง	<code>while (expression) statement;</code>
ผังงาน	



รายละเอียดการใช้งานคำสั่ง

คำสั่ง while เป็นคำสั่งให้ทำซ้ำ หรือวนรอบเมื่อเงื่อนไขสอดคล้องหรือเงื่อนไขเป็นจริง สำหรับ while จะมีความแตกต่างจาก for คือจะไม่มีการกำหนดค่าเริ่มต้นต่างๆ ก่อนการทำงาน การทำงานวนซ้ำจะทำงานในขอบข่ายที่กำหนดได้แล้วและจะเลิกทำซ้ำ หรือวนรอบเมื่อเงื่อนไขไม่สอดคล้องหรือผลลัพธ์ของเงื่อนไขเป็นเท็จ เช่นต้องการให้มีการรับค่าข้อมูลจากคีย์บอร์ดเป็นตัวเลขจำนวนเต็ม 1 จำนวน แล้วโปรแกรมจะแสดงค่าตัวเลขเป็นสองเท่า ไปเรื่อยๆ โดยตัวเลขมากที่สุดที่จะแสดงจะไม่เกิน 30000 จะได้ผลลัพธ์ดังนี้

```

scanf ("%d",&i);
while (i<30000) { printf ("%d\n",i); i=i*2; }
  
```

จากรูปแบบคำสั่งเราจะได้ความสัมพันธ์จากตัวอย่างดังนี้

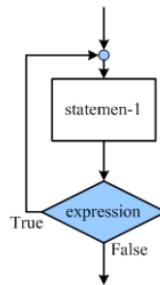
```

Expression i<30000
statement { printf ("%d\n",i); i=i*2; }
  
```

จากโปรแกรมตัวอย่าง ก่อนที่จะมีการทำงานคำสั่ง while จะมีการรับค่าจากคีย์บอร์ดโดยคำสั่ง `scanf` เก็บข้อมูลไว้ที่ ตัวแปร  $i$  จากผังงานเราจะได้รูปแบบและขั้นตอนการทำงานคือ หลังจากจุดเริ่มต้นการทำงานแบบวนซ้ำ จะมีการตรวจสอบ expression ว่าค่า  $i$  ที่รับจากคีย์บอร์ดนั้น มีค่าน้อยกว่า 30000 หรือไม่ ถ้าเป็นจริงตามนั้นจะเริ่มทำงาน statement คือการแสดงผลลัพธ์ค่า  $i$  และเพิ่มค่า  $i$  เป็นสองเท่า และวิจัยกลับไปทำงานที่จุดเริ่มต้นการทำงานแบบวนซ้ำอีกครั้งหนึ่ง การทำงานจะเป็นอย่างนี้เรื่อยๆ จนกว่าค่า  $i$  จะไม่น้อยกว่า 30000

จากการทำงานจะพบว่าใน statement ของคำสั่ง while จะมีการเปลี่ยนแปลงค่าตัวแปรที่ส่งผลกับ expression อยู่ทุกคราวของการทำงาน ถ้าไม่มีการเปลี่ยนแปลงค่าตัวแปรที่ส่งผลกับ expression จะทำให้เกิด infinite loop ได้ ซึ่ง การศึกษาต้องระมัดระวังเมื่อใช้คำสั่ง while ด้วย

คำสั่ง	do....while
รูปแบบคำสั่ง	do statement; while (expression);
ผังงาน	



#### รายละเอียดการใช้งานคำสั่ง

คำสั่ง do... while เป็นคำสั่งที่ให้ทำซ้ำ หรือวนรอบการทำงานเมื่อเงื่อนไขเป็นจริง สำหรับ do... while จะมีความแตกต่างจาก while คือจะมีการตรวจสอบ expression หลังจากที่มีการทำงาน statement เรียบร้อยแล้ว การกำหนดค่าเริ่มต้นมักจะถูกกำหนดก่อนที่จะเรียกใช้คำสั่ง do... while และจะมีการเปลี่ยนแปลงค่าที่ส่งผลกับ expression ใน statement นั้นๆการทำงานวนซ้ำจะทำงานในขอบข่ายที่กำหนดไว้และจะเลิกทำซ้ำหรือวนรอบเมื่อเงื่อนไขไม่สอดคล้องหรือผลลัพธ์ของเงื่อนไขเป็นเท็จ เช่นต้องการให้มีการวนรับค่าตัวเลขไปเรื่อยๆ จนกว่าค่าที่ได้รับจะมีค่าเป็น 0 จะได้ผลลัพธ์ของโปรแกรมดังนี้

```

do
{
    scanf("%d",&i);
    printf("Input = %d\n",i);
} while (i != 0);
  
```

จากรูปแบบคำสั่งเราจะได้ความสัมพันธ์จากตัวอย่างดังนี้

```

Expression i != 0
Statement { scanf("%d",&i); printf("Input = %d\n",i); }
  
```

จากโปรแกรมตัวอย่าง หลังจากจุดเริ่มต้นการทำงานแบบบวนซ้ำ จะมี statement ในการรับตัวเลขจากคีย์บอร์ดแล้วแสดงผลลัพธ์ตัวเลขออกมานทางหน้าจอ แล้วจึงมีการตรวจสอบค่า expression ว่ามีค่าเป็น 0 หรือไม่ ถ้า expression เป็นจริงจะกลับไปเริ่มต้นการทำงานที่จุดเริ่มต้นการทำงานแบบบวนซ้ำอีกครั้ง แต่ถ้าตรวจสอบค่า expression แล้วมีค่าเป็นเท็จ จะออกจากการทำงานแบบบวนซ้ำทันที

#### เทคนิคการใช้งานคำสั่ง

การใช้งานคำสั่งน้ำหนัก มักใช้ประโยชน์ในสองกรณีเท่านั้น

1. ลดจำนวนการพิมพ์ข้อมูล โดยอาศัยการทำงานซ้ำๆ นั้น แทนการพิมพ์คำสั่งซ้ำๆ กัน
2. เรียกใช้ตัวแปรที่มีการปรับเปลี่ยนไปเรื่อยๆ ภายในการทำงานวนซ้ำในแต่ละรอบ

เนื่องจากรูปแบบของคำสั่ง for , while และ do ... while นั้นมีส่วนที่เป็น statement เพียง statement เดียวอยู่ในรูปแบบของคำสั่ง ดังนั้นเราสามารถใช้หลักการของคอมไฟล์อิร์ 2 ข้อ มาใช้ในการสร้างรูปแบบการเขียนโปรแกรมที่มีความซับซ้อนสูงขึ้นได้ เช่นเดียวกับชุดคำสั่งกำหนดเงื่อนไข (if, if – else)

- 1) การรวม statement หลายๆ statement ให้กลายเป็น statement เดียวด้วยการใช้เครื่องหมายวงเล็บปีกกา { }
- ล้อมรอบกลุ่มของ statement ที่ต้องการให้คอมไฟล์อิร์แปลความหมายเป็น statement เดียว
- 2) การแทนค่า statement ด้วยคำสั่งของตัวมันเอง ทำให้เกิดลักษณะการทำงานเป็น ลูป ซ้อน ลูป ทำให้สามารถเขียนโปรแกรมที่ซับซ้อนมากขึ้นได้

### ตัวอย่างการใช้งาน

1. ถ้าต้องการพิมพ์ตัวเลข 1-20 สามารถใช้คำสั่ง for ดังนี้

```
for (i=1;i<=20;i++) printf("%d\n",i);
```

2. ถ้าต้องการพิมพ์เลขคี่ตั้งแต่ 1-19 สามารถใช้คำสั่ง for ดังนี้

```
for (i=1;i<=20;i=i+2) printf("%d\n",i);
```

3. ถ้าต้องการเขียนโปรแกรมเพื่อรับค่าตัวเลข 1 จำนวนแล้วแสดงค่าผลบวกตั้งแต่ 1 จนถึงค่าที่รับเข้ามาสามารถใช้คำสั่ง for ดังนี้

```
int a,i,sum
int main()
{
    scanf("%d",&a);
    sum = 0;
    for (i=1; i<=a ; i++) sum = sum + i;
    printf("%d\n",i);
    return 0;
}
```

## การทดลองที่ 5.1 while loop

ให้นักศึกษาทำการทดลองโปรแกรมต่อไปนี้ แล้วบันทึกผลและตอบคำถาม

```

lab06_1_1.c Start Page

(Global Scope)

1 #include<stdio.h>
2 int main()
3 {
4     int n=0;
5     while(n<=30)
6     {
7         printf("%4d",n);
8         n=n+2;
9     }
10    printf("Loop has finished. n=%\n\n",n);
11    return 0;
12 }

```


- 1) ตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_  
ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_
- 2) โปรแกรมมีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร n คือ \_\_\_\_\_
- 3) ในบรรทัดที่ 8 ถ้าเปลี่ยน  $n=n+2$ ; เป็น  $n=n+3$ ; จะมีการวนลูปกี่ครั้ง \_\_\_\_\_  
ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_ หลังจากจบลูปค่าของตัวแปร n คือ \_\_\_\_\_
- 4) ในบรรทัดที่ 8 ถ้าเปลี่ยน  $n=n+2$ ; เป็น  $n=n+4$ ; จะมีการวนลูปกี่ครั้ง \_\_\_\_\_  
ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_ หลังจากจบลูปค่าของตัวแปร n คือ \_\_\_\_\_
- 5) ในบรรทัดที่ 8 ถ้าเปลี่ยน  $n=n+2$ ; เป็น  $n++$ ; จะมีการวนลูปกี่ครั้ง \_\_\_\_\_  
ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_ หลังจากจบลูปค่าของตัวแปร n คือ \_\_\_\_\_

## การทดลองที่ 5.2 การหาผลรวมของเลขตั้งแต่ 1 ถึงค่าที่รับเข้ามา while loop

lab06\_1\_2.c\* Start Page

(Global Scope)

```

1 #include<stdio.h>
2 int main()
3 {
4     int num,sum=0,i=1;
5     printf("Enter last number : ");
6     scanf("%d",&num);
7     while(i<=num)
8     {
9         sum = sum + i;
10        i++;
11    }
12    printf("Summation from 1 to %d = %d\n",num,sum);
13    return 0;
14 }
```

1. ใส่ค่า 10 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

2. ใส่ค่า 25 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

3. ใส่ค่า 50 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

## การทดลองที่ 5.3 do . . . while loop

```

1 #include <stdio.h>
2 int main()
3 {
4     int i=1, num;
5     printf("Enter a positive number : ");
6     scanf("%d", &num);
7     do {
8         printf("%4d", i);
9         i++;
10    } while (i<=num);
11    printf("\n\n");
12
13    return 0;
14 }
15

```

1. ใส่ค่า 7 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

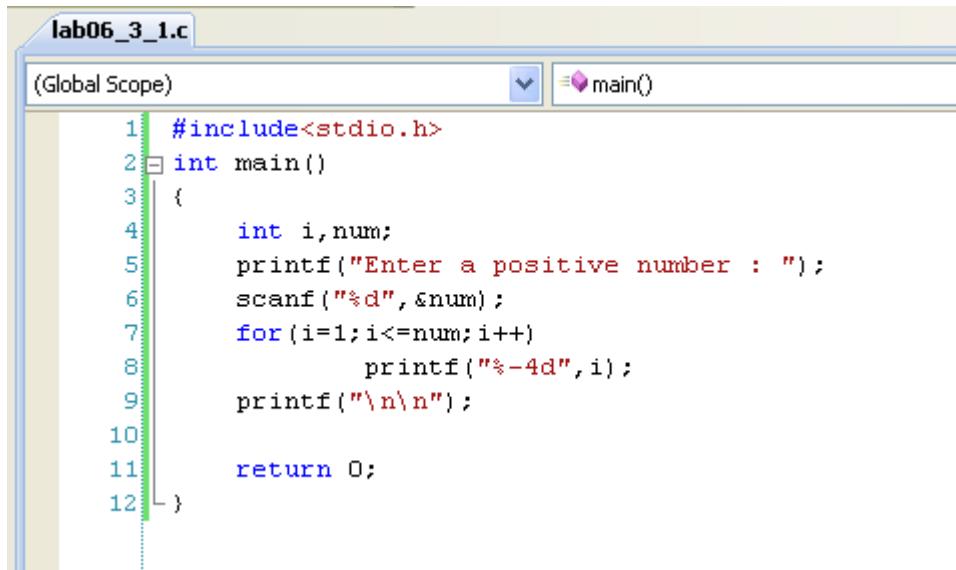
2. ใส่ค่า 12 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

### การทดลองที่ 5.4 for loop



```

1 #include <stdio.h>
2 int main()
3 {
4     int i, num;
5     printf("Enter a positive number : ");
6     scanf("%d", &num);
7     for(i=1; i<=num; i++)
8         printf("%-4d", i);
9     printf("\n\n");
10
11 }
12

```

1. ใส่ค่า 6 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

2. ใส่ค่า 11 ผลลัพธ์ที่ได้คือ


ตัวแปรตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_

ค่าเริ่มต้นคือ \_\_\_\_\_ ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_

โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_ หลังจากจบลูป ค่าของตัวแปร i คือ \_\_\_\_\_

### การทดลองที่ 5.5 Infinite Loop (ลูปที่รันไม่รู้จบ)

lab06\_4\_1.c\* Start Page

(Global Scope)

```

1 #include<stdio.h>
2 int main()
3 {
4     char ch;
5     for(ch=0;ch>=0;ch++)
6         printf("%4d",ch);
7     printf("\nLoop has finished. ch=%d\n",ch);
8     return 0;
9 }
```

- 1) ตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_
- 2) ค่าเริ่มต้นคือ \_\_\_\_\_
- 3) ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_
- 4) โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_
- 5) หลังจากจบลูป ค่าของตัวแปร ch คือ \_\_\_\_\_

lab06\_4\_2.c\* Start Page

(Global Scope)

```

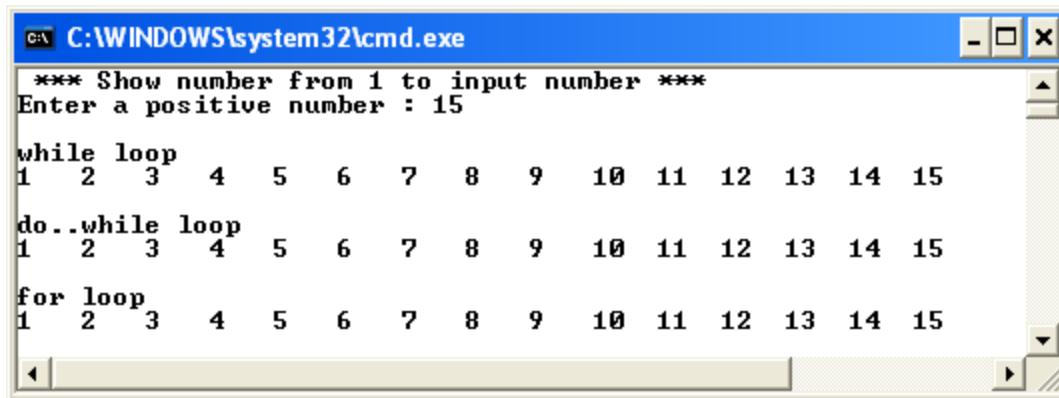
1 #include<stdio.h>
2 int main()
3 {
4     char ch;
5     for(ch=0;ch%2==0;ch+=2)
6         printf("%4d",ch);
7     printf("\nLoop has finished. ch=%d\n",ch);
8     return 0;
9 }
```

- 1) ตัวแปรที่ใช้ควบคุมคือ \_\_\_\_\_
- 2) ค่าเริ่มต้นคือ \_\_\_\_\_
- 3) ค่าสุดท้ายก่อนจบลูปคือ \_\_\_\_\_
- 4) โปรแกรมนี้มีการวนลูปกี่ครั้ง \_\_\_\_\_
- 5) หลังจากจบลูป ค่าของตัวแปร ch คือ \_\_\_\_\_

**หมายเหตุ** Infinite loop คือ loop ที่ทำงานต่อไปเรื่อย ๆ ไม่รู้จบ ในการทดลองนี้ ถ้าทำงานเกิน 1 นาทีถือว่าเป็น Infinite loop

## การทดลองที่ 5.6 ให้นักศึกษาเขียนโปรแกรมต่อไปนี้ แสดงผลตามตัวอย่าง

- 1) จงเขียนโปรแกรมรับจำนวนเต็ม 1 จำนวน แล้วแสดงผล ตัวเลขตั้งแต่เลข 1 ถึง จำนวนที่รับมา โดยใช้ while loop, do.. while loop และ for loop



```

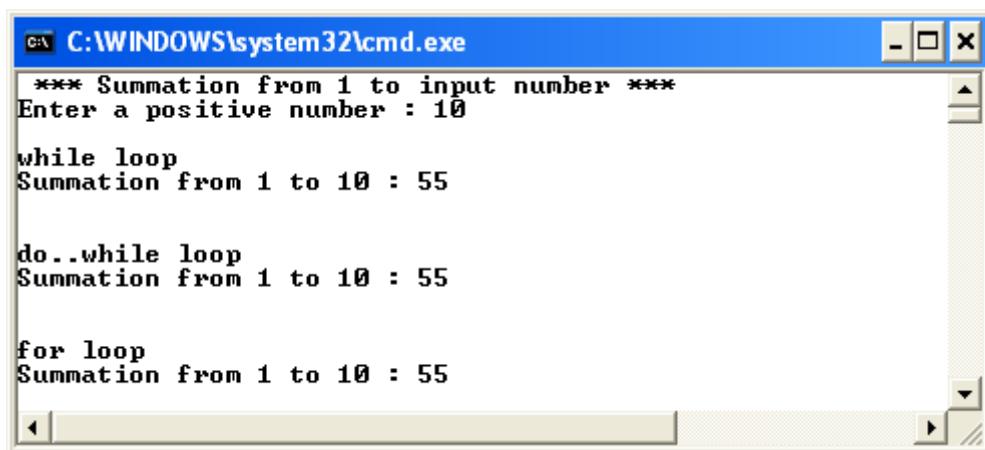
C:\WINDOWS\system32\cmd.exe
*** Show number from 1 to input number ***
Enter a positive number : 15

while loop
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

do..while loop
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

for loop
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  
```

- 2) จงเขียนโปรแกรมรับจำนวนเต็ม 1 จำนวน แล้วแสดงผล ผลบวกตัวเลขตั้งแต่เลข 1 ถึง จำนวนที่รับมา โดยใช้ while loop, do.. while loop และ for loop



```

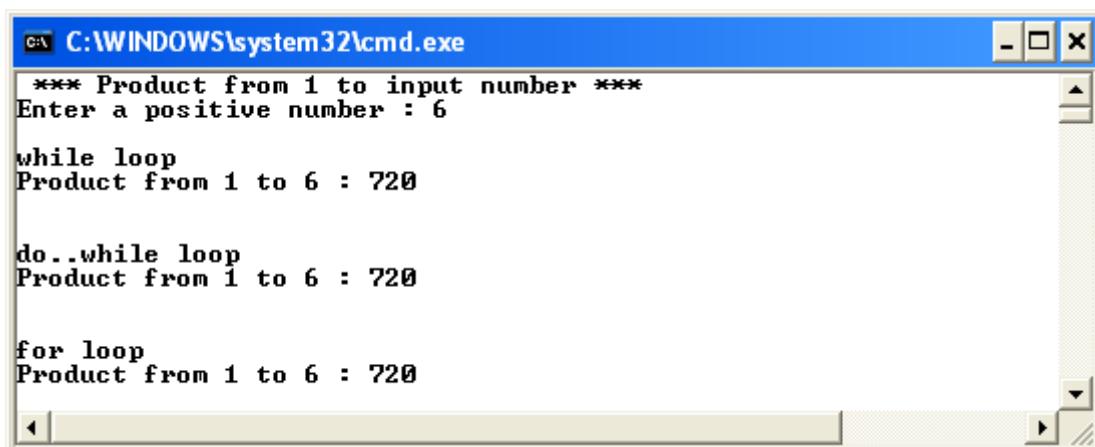
C:\WINDOWS\system32\cmd.exe
*** Summation from 1 to input number ***
Enter a positive number : 10

while loop
Summation from 1 to 10 : 55

do..while loop
Summation from 1 to 10 : 55

for loop
Summation from 1 to 10 : 55
  
```

- 3) จงเขียนโปรแกรมรับอินพุทเป็นตัวเลขจำนวนเต็ม 1 จำนวนแล้วคำนวณหาผลคูณของตัวเลขตั้งแต่ 2 จนถึงตัวเลขที่รับเข้า โดยใช้ while loop, do.. while loop และ for loop



```

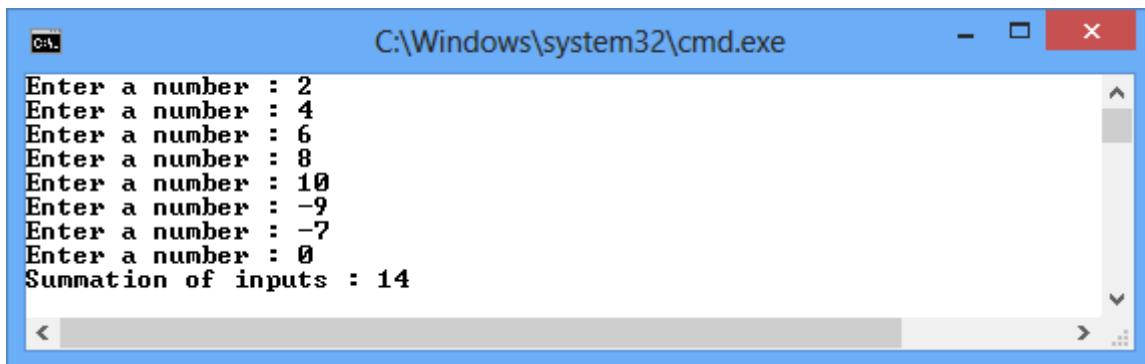
C:\WINDOWS\system32\cmd.exe
*** Product from 1 to input number ***
Enter a positive number : 6

while loop
Product from 1 to 6 : 720

do..while loop
Product from 1 to 6 : 720

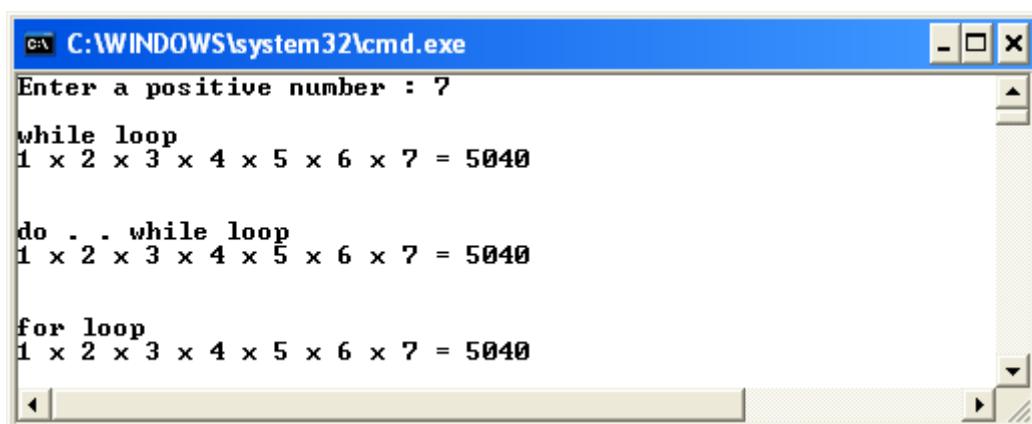
for loop
Product from 1 to 6 : 720
  
```

- 4) จงเขียนโปรแกรมเพื่อรับตัวเลขไปเรื่อยๆ จนกว่าจะรับค่าตัวเลข 0 และแสดงผลลัพธ์รวมของตัวเลขที่รับค่าทั้งหมด



```
C:\Windows\system32\cmd.exe
Enter a number : 2
Enter a number : 4
Enter a number : 6
Enter a number : 8
Enter a number : 10
Enter a number : -9
Enter a number : -7
Enter a number : 0
Summation of inputs : 14
```

- 5) จงเขียนโปรแกรมรับอินพุทเป็นตัวเลขจำนวนเต็ม 1 จำนวนแล้วคำนวณหาผลคูณของตัวเลขตั้งแต่ 2 จนถึงตัวเลขที่รับเข้า โดยใช้ while loop, do.. while loop และ for loop และแสดงผลตามตัวอย่าง



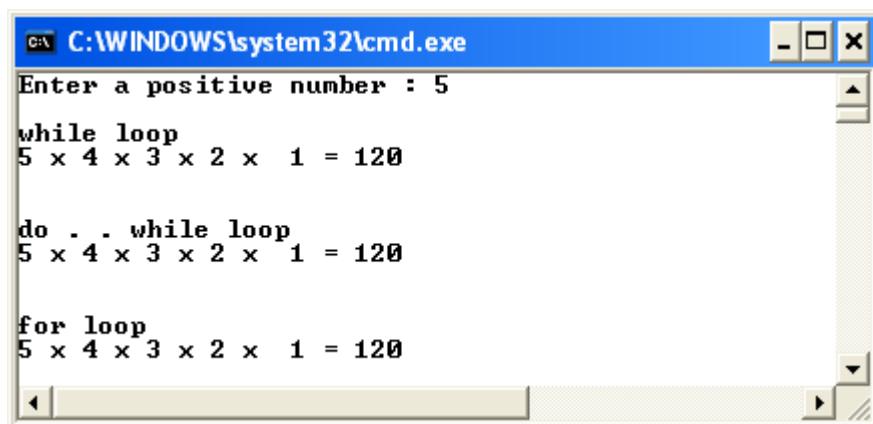
```
C:\WINDOWS\system32\cmd.exe
Enter a positive number : 7

while loop
1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040

do . . . while loop
1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040

for loop
1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040
```

- 6) จงเขียนโปรแกรมรับอินพุทเป็นตัวเลขจำนวนเต็ม 1 จำนวนแล้วคำนวณหาผลคูณของตัวเลขตั้งแต่ 2 จนถึงตัวเลขที่รับเข้า โดยใช้ while loop, do.. while loop และ for loop และแสดงผลตามตัวอย่าง



```
C:\WINDOWS\system32\cmd.exe
Enter a positive number : 5

while loop
5 x 4 x 3 x 2 x 1 = 120

do . . . while loop
5 x 4 x 3 x 2 x 1 = 120

for loop
5 x 4 x 3 x 2 x 1 = 120
```

## บทที่ 6 การเขียนโปรแกรมแบบวนซ้ำ และกำหนดเงื่อนไข

### วัตถุประสงค์การศึกษา

- 1) ทบทวนเรื่องการเขียนโปรแกรมแบบวนซ้ำ และผังงาน
- 2) สามารถเขียนโปรแกรมภาษาซีให้มีทำงานแบบวนซ้ำได้ และกำหนดเงื่อนไขร่วมกันได้

### 6.1 ตัวอย่างการประยุกต์ใช้คำสั่งวนซ้ำ กับเงื่อนไข เพื่อแก้ปัญหาต่างๆ

#### โปรแกรม 6.1 โปรแกรมแสดงสูตรคูณแม่ 2

จะเขียนผังงานและโปรแกรมแสดงสูตรคูณแม่ 2

- 1) Output analysis สูตรคูณแม่ 2
- 2) Input analysis ไม่มี
- 3) Process analysis

โปรแกรมวนรอบเพื่อแสดงสูตรคูณแม่ 2 แสดงอยู่ในรูป  $2 * \text{num} = 2*\text{num}$

$$2 * 1 = 2$$

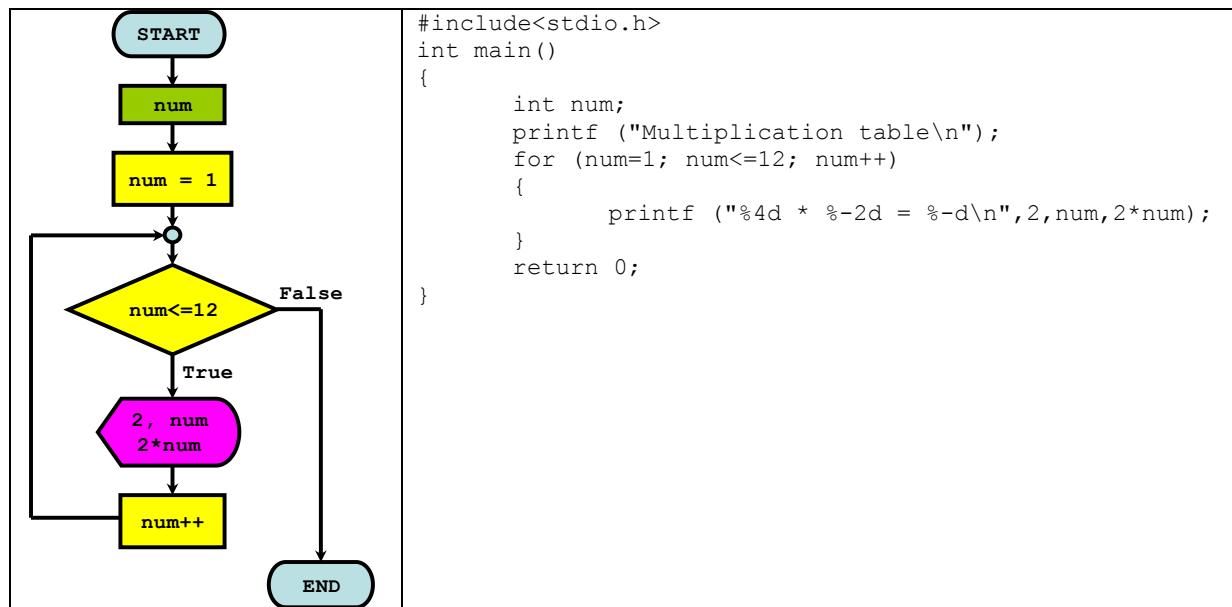
$$2 * 2 = 4$$

...

$$2 * 12 = 24$$

- 4) Variable define

num เป็นจำนวนเต็มเพื่อนับค่า 1 - 12



### หลักการเลือกใช้ for , while , do-while

- ใช้ for ในกรณีที่ทราบจำนวนรอบของการวนซ้ำ
- ใช้ while ในกรณีที่ต้องคิดเงื่อนไขก่อนการทำงานแบบวนซ้ำ
- ใช้ do-while ในกรณีที่ต้องคิดเงื่อนไขการทำงานภายหลังจากทำงานไปแล้วครั้งหนึ่ง

### การเขียนโปรแกรมวนรอบ และกำหนดเงื่อนไข

- การเขียนโปรแกรมที่มีความซับซ้อนมากขึ้น จำเป็นต้องอาศัยการเขียนโปรแกรมแบบวนรอบร่วมกับการเขียนโปรแกรมแบบมีเงื่อนไข
- โดยการเพิ่มเงื่อนไขการทำงานในส่วนของการวนรอบ หรือมีการตรวจสอบเงื่อนไขว่าจะให้โปรแกรมมีการวนรอบอย่างไร

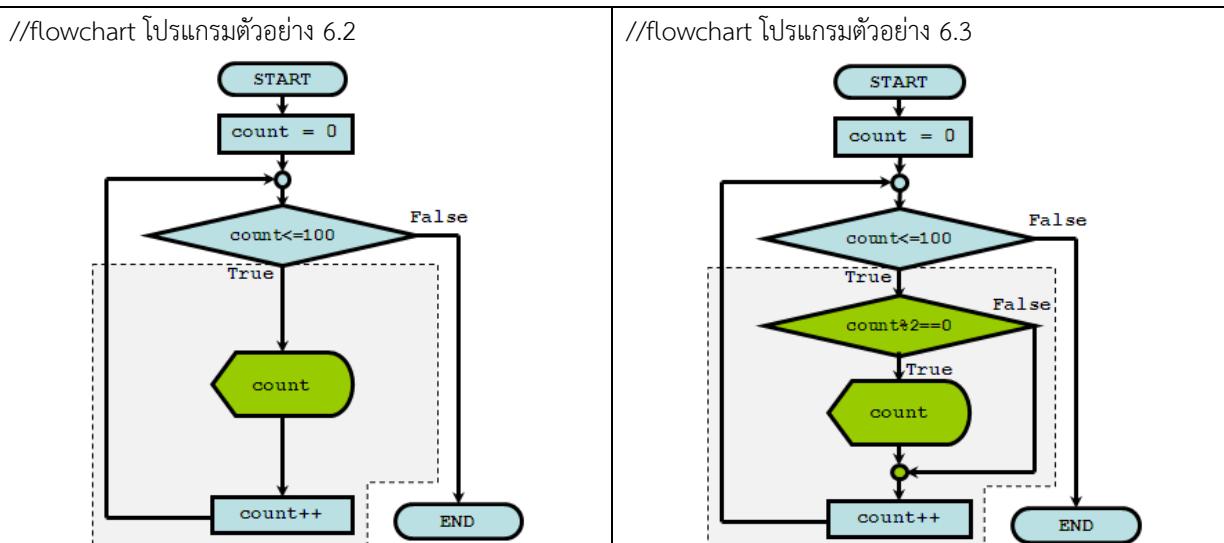
### โปรแกรม 6.2 โปรแกรมแสดงเลข 0 – 100 ใช้ while

```
#include<stdio.h>
int main()
{
    int count = 0;
    printf ("Show number from 0 to 100\n\n");
    while (count<=100)
    {
        printf ("%d ",count);
        count++;
    }
    return 0;
}
```

### โปรแกรม 6.3 โปรแกรมแสดงเลขคู่ 0 – 100 ใช้ while

```
#include<stdio.h>
int main()
{
    int count = 0;
    printf ("Show even number from 0 to 100\n\n");
    while (count<=100)
    {
        if (count%2 == 0) //ใช้เงื่อนไข ควบคุมให้แสดงผลเฉพาะเลขคู่
            printf ("%d ",count);
        count++;
    }
    return 0;
}
```

แผนผังเปรียบเทียบการทำงานระหว่างโปรแกรมตัวอย่างที่ 6.2 กับ 6.3



## โปรแกรม 6.4 โปรแกรมตรวจสอบจำนวนสระ ใช้ for

จะเขียนผังงาน และโปรแกรมเพื่อรับอักษรตัวเล็กมา 10 ตัว และตรวจสอบว่ามีอักษรที่เป็นสระกี่ตัว และไม่ใช่สระกี่ตัว

- 1) Output analysis จำนวนอักษรที่เป็นสระ และไม่ใช่สระ
- 2) Input analysis อักษรที่ผู้ใช้ป้อนมากำหนด 10 ตัว
- 3) Process analysis โปรแกรมทำงานแบบวนรอบ เพื่อรับค่าจำนวนอักษร แล้วตรวจสอบว่าเป็นสระ หรือไม่ และนับจำนวนไว้ จนครบ 10 ตัว
- 4) Variable define
  - vowel เป็นจำนวนเต็มเพื่อใช้นับจำนวนสระ
  - alphabet เป็นจำนวนเต็มเพื่อใช้นับจำนวนที่ไม่ใช่สระ
  - count เป็นจำนวนเต็มเพื่อใช้นับว่าครบ 10 ตัวหรือไม่
  - letter เป็นอักษรเพื่อรับตัวอักษร

<pre> graph TD     START([START]) --&gt; Init[vowel=0, alphabet=0, count, letter]     Init --&gt; CountInit[count = 0]     CountInit --&gt; ReadLetter[letter]     ReadLetter --&gt; Cond{count &lt; 10}     Cond -- True --&gt; IsVowel{letter is vowel}     IsVowel -- True --&gt; VowelInc[vowel++]     IsVowel -- False --&gt; AlphabetInc[alphabet++]     VowelInc --&gt; Output{vowel alphabet}     AlphabetInc --&gt; Output     Output --&gt; End([END])     End --&gt; CountInit     </pre>	<pre> #include&lt;stdio.h&gt; #include&lt;conio.h&gt;  int main() {     int vowel=0,alphabet=0,count;     char letter;     for (count=0; count&lt;10; count++)     {         printf ("\nEnter letter a-z : ");         letter = getche();         if ((letter=='a')  (letter=='e')  (letter=='i'))               (letter=='o')  (letter=='u'))             vowel++;         else             alphabet++;     }     printf ("\n***Result***\n");     printf ("Vowel (a,e,i,o,u) = %d\n",vowel);     printf ("Other letter = %d",alphabet);     return 0; } </pre>
--	--

## โปรแกรม 6.5 โปรแกรมแสดงผลรูปสี่เหลี่ยมกลวง

จะเขียนผังงานและโปรแกรมแสดงผลรูปสี่เหลี่ยมขนาด  $n \times n$  โดยโปรแกรมจะรอรับจำนวนเต็มจากผู้ใช้งาน ดังตัวอย่าง

```

Please enter number : 9
Output
*****
*   *
*   *
*   *
*   *
*   *
*****
```

```

Please enter number : 4
Output
*****
*   *
*   *
*   *
*****
```

1) Output analysis ผลตัวเลข เป็นรูปสี่เหลี่ยมจัตุรัสขนาดเท่ากับจำนวนตัวเลขที่รับเข้ามา โดยเว้นช่องว่างตรงกลาง

2) Input analysis เลขจำนวนเต็มที่ผู้ใช้ป้อนเข้ามา

3) Process analysis

โปรแกรมรอรับค่าจำนวนเต็มจากผู้ใช้งาน

โปรแกรมวนรอบเพื่อทำการแสดง '\*' เป็นรูปสี่เหลี่ยมจัตุรัส

โดยพิจารณาว่าส่วนที่อยู่ในกรอบให้แสดงเป็นช่องว่าง

บรรทัดที่ 1 แสดงผล '\n' และผล '\*' เฉพาะตำแหน่งขอบ ที่เหลือแสดงผล ''

บรรทัดที่ 2 แสดงผล '\n' และผล '\*' เฉพาะตำแหน่งขอบ ที่เหลือแสดงผล ''

.....

บรรทัดที่ n แสดงผล '\n' และผล '\*' เฉพาะตำแหน่งขอบ ที่เหลือแสดงผล ''

4) Variable define

num เป็นจำนวนเต็มเพื่อใช้เก็บค่าตัวเลขที่ผู้ใช้ป้อน

i เป็นจำนวนเต็มเพื่อใช้นับจำนวนบรรทัด

j เป็นจำนวนเต็มเพื่อใช้นับจำนวนอักษรในบรรทัด

//วัด flowchart	#include<stdio.h> int main() {     int num,i,j;     char space=' ';     printf ("Enter number : ");     scanf ("%d",&num);     for (i=1; i<=num; i++)     {         printf ("\n");         for (j=1; j<=num; j++)         {             if (i==1    i==num    j==1                    j==num)                 printf ("*");             else                 printf ("%c",space);         }     }     return 0; }
-----------------	---

## โปรแกรม 6.6 โปรแกรมตู้ ATM

จะเขียนผังงาน และโปรแกรมการจ่ายเงินของตู้ ATM โดยให้ผู้ใช้กรอกจำนวนเงินเข้ามาแล้วโปรแกรมจะตรวจสอบว่าถูกต้องหรือไม่โดยมีเงื่อนไขว่า ATM จะจ่ายเงินให้ไม่เกิน 20000 และต้องมากกว่า 100 ซึ่งโปรแกรมจะต้องคำนวณว่าต้องจ่ายเป็นธนบัตรมูลค่าต่างๆอย่างลักษณะนี้ไปโดยกำหนดให้ในเครื่อง ATM มีธนบัตรมูลค่าต่างๆดังนี้

ธนบัตรใบละ 1000

ธนบัตรใบละ 500

ธนบัตรใบละ 100

1) Output analysis แสดงผลลัพธ์ของจำนวนธนบัตรมูลค่าต่างๆ

2) Input analysis จำนวนเงินที่ผู้ใช้ต้องการกรอก

3) Process analysis

- โปรแกรมทำงานแบบวนรอบ เพื่อรับค่าจำนวนเงินที่ถูกต้อง
- ถ้าจำนวนเงินถูกต้องให้คำนวณว่ามีธนบัตรใดจำนวนเท่าไหร่
- ตรวจสอบว่าจำนวนเงินคร่าวมีธนบัตรมูลค่า 1000 กี่ใบ แสดงผลและหักจำนวนเงินที่จ่ายธนบัตรมูลค่า 1000 ฿
- ตรวจสอบว่าจำนวนเงินที่เหลือคร่าวมีธนบัตรมูลค่า 500 กี่ใบ แสดงผลและหักจำนวนเงินที่จ่ายธนบัตรมูลค่า 500 ฿
- ตรวจสอบว่าจำนวนเงินที่เหลือคร่าวมีธนบัตรมูลค่า 100 กี่ใบ แสดงผล

4) Variable define

money เป็นจำนวนเต็มสำหรับเก็บค่าเงิน

amount เป็นจำนวนเต็มสำหรับเก็บจำนวนธนบัตรที่จ่ายออกไป

//วาด flowchart	#include<stdio.h> int main() { int money, amount; printf ("Enter money : "); scanf ("%d",&money); while ((money > 20000)    (money%100!=0)) { printf ("Sorry, please enter money : "); scanf ("%d",&money); } printf ("\nTotal bank note\n"); if (money/1000!=0) { amount = money/1000; money = money%1000; printf ("Banknote 1000 : %d\n",amount); } if (money/500!=0) { amount = money/500; money = money%500; printf ("Banknote 500 : %d\n",amount); } if (money/100!=0) { amount = money/100; money = money%100; printf ("Banknote 100 : %d\n",amount); } }
-----------------	---

## โปรแกรม 6.7 โปรแกรมคำนวณค่าอาหาร

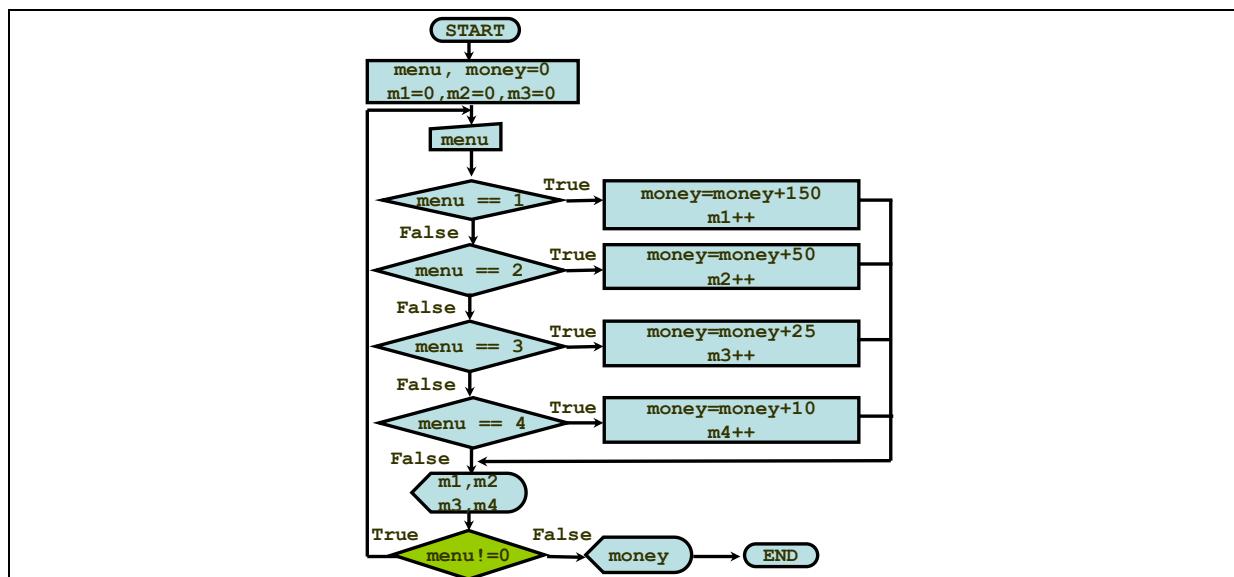
จะเขียนผังงานและโปรแกรม เมนูรับรายการอาหารจากลูกค้าเพื่อคำนวณราคากลางๆทั้งหมด โดยกำหนดให้โปรแกรมมีรายการดังนี้

- |                    |       |
|--------------------|-------|
| 1. Pizza           | 150 ฿ |
| 2. Hamburger       | 50 ฿  |
| 3. Sandwich        | 25 ฿  |
| 4. Water           | 10 ฿  |
| 0. Calculate money |       |

โดยทุกครั้งที่เลือกเมนูรายการอาหารจะแสดงจำนวนอาหารที่สั่งไปแล้วด้วย

- 1) Output analysis แสดงผลจำนวนรายการอาหาร และราคา
- 2) Input analysis จำนวนรายการอาหารที่ผู้ซื้อเลือก
- 3) Process analysis โปรแกรมทำงานแบบวนรอบ เพื่อแสดงรายการอาหาร และรับข้อมูลรายการอาหารที่เลือก แล้วรวมผลของการแต่ละชนิด และราคารวม
- 4) Variable define

- |             |  |
|-------------|--|
| menu        | เป็นจำนวนเต็มสำหรับรายการอาหาร         |
| m1,m2,m3,m4 | เป็นจำนวนเต็มสำหรับจำนวนรายการอาหาร    |
| money       | เป็นจำนวนหนอนิยมสำหรับเก็บจำนวนเงินรวม |
| i           | เป็นจำนวนเต็มเพื่อควบคุมการเว้นบรรทัด  |



```
#include<stdio.h>
int main()
{
    int menu,m1=0,m2=0,m3=0,m4=0,i;
    float money=0;
    printf ("Welcome to Restaurant\n");
    do
    {
```

```

printf ("1. Pizza      150 B\n");
printf ("2. Hamburger  50 B\n");
printf ("3. Sandwich   25 B\n");
printf ("4. Water       10 B\n");
printf ("0. Calculate money\n");
printf ("\nSelect menu : ");
scanf ("%d",&menu);
for (i=0;i<34;i++)
printf ("\n");
if (menu==1)
{
    money = money + 150;      m1++;
}
else if (menu==2)
{
    money = money + 50;      m2++;
}
else if (menu==3)
{
    money = money + 25;      m3++;
}
else if (menu==4)
{
    money = money + 10;      m4++;
}
printf ("\n\nU have :\n");
printf ("Pizza      %d * 150 : %d\n",m1,m1*150);
printf ("Hamburger  %d * 50 : %d\n",m2,m2*50);
printf ("Sandwich   %d * 25 : %d\n",m3,m3*25);
printf ("Water      %d * 10 : %d\n\n",m4,m4*10);
}
while (menu!=0);
printf ("\nTotal payment = %.2f\n",money);
return 0;
}

```

## 6.2 คำถ้ามห้ายบท

- 1) จะเขียนโปรแกรม รับตัวเลข 1 2 3 เข้ามา ถ้ารับเป็นเลข 1 ให้พิมพ์คำว่า Hello ถ้าเป็นเลข 2 ให้พิมพ์คำว่า Thank you ถ้ารับเป็นเลข 3 ให้พิมพ์คำว่า Good bye และออกจากโปรแกรม ถ้าเป็นเลขอื่น ให้พิมพ์คำว่า Sorry ตัวอย่างโปรแกรม

```

Enter a number : 1
Hello
Enter a number : 2
Thank you
Enter a number : 9
Sorry
Enter a number : 0
Sorry
Enter a number : 3
Good bye

```

- 2) จะเขียนโปรแกรมรับตัวเลข ระหว่าง 2 ถึง 25 และแสดงสูตรคูณของมา ถ้าตัวเลขที่รับเข้ามาไม่อยู่ในช่วงที่กำหนด ให้ผู้ใช้ป้อนค่าเข้ามาใหม่

```
Enter a number : 31
Enter a number : 4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
.....
.....
4 * 11 = 44
4 * 12 = 48
```

3) จงเขียนโปรแกรมรับข้อความเข้ามาหนึ่งข้อความ แล้วนำมำแสดงผลบรรทัดละ 10 ตัวอักษร

```
Enter a sentence :
You are the wind beneath my wings.
```

```
Result :
You are th
e wind ben
eath my wi
ngs.
```

## การทดลองที่ 6 การเขียนโปรแกรมการวนรอบ และกำหนดเงื่อนไข

### วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจการใช้ คำสั่งการวนรอบ และกำหนดเงื่อนไข ร่วมกัน
2. เพื่อให้นักศึกษาสามารถใช้ คำสั่งการวนรอบ และกำหนดเงื่อนไข เพื่อเขียนโปรแกรมที่ซับซ้อนได้

### การทดลองที่ 6.1

จะเขียนโปรแกรมรับตัวเลข 1 ตัว และแสดงผลลัพธ์ว่าค่าที่รับมาเป็นจำนวนเฉพาะหรือไม่

### การทดลองที่ 6.2

จะเขียนโปรแกรมรับค่า 2 ค่า และ แสดงจำนวนเฉพาะที่อยู่ระหว่างค่าที่รับมา โดยเรียงจากน้อยไปมากเสมอ

### การทดลองที่ 6.3

จะเขียนโปรแกรมรับตัวเลข 1 ตัวแล้วให้คอมพิวเตอร์วัดรูปเครื่องหมาย \* เป็นสี่เหลี่ยมจัตุรัสที่มีด้านเท่ากับตัวเลขที่รับเข้ามา ดังตัวอย่าง เมื่อป้อนตัวเลข 5 จะได้ผลลัพธ์ดังนี้

```
*****
*****
*****
*****
*****
```

### การทดลองที่ 6.4

จะเขียนโปรแกรมรับตัวเลข 1 ตัวเพื่อแสดงผล เป็นรูปสี่เหลี่ยมขนาดเท่ากับจำนวนตัวเลขที่รับเข้ามา โดยมีพื้นหลังเป็น \* และ - สลับกันไปเรื่อยๆ เช่นป้อนตัวเลข 5 จะได้ผลลัพธ์ดังนี้

```
*-*-*-
-*-*-*-
*-*-*-
-*-*-*-
*-*-*-
```

### การทดลองที่ 6.5

ให้นักศึกษาเขียนโปรแกรมรับค่าตัวเลข ตัวแล้วให้คอมพิวเตอร์วัดรูปเครื่องหมาย \* เป็นปริภูมิที่มีความสูงเท่ากับตัวเลขที่รับเข้ามาดังตัวอย่าง เมื่อป้อนตัวเลข 5 จะได้ผลลัพธ์ดังนี้

\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

### การทดลองที่ 6.6

ให้นักศึกษาเขียนโปรแกรมรับค่าตัวเลข 1 ตัว แล้วให้แสดงผลลัพธ์เป็นรูปผังเสื่อเช่น

<b>Input = 4</b>	<b>Input = 3</b>
* *	* *
** **	** **
*** ***	*****
*****	** **
*** ***	*
** **	
*	*

## บทที่ 7 ตัวแปรແຄວลำดับ Array

### วัตถุประสงค์การศึกษา

1. เข้าใจหลักการของตัวแปรແຄວลำดับ
2. สามารถเขียนโปรแกรมใช้งานตัวแปรແຄວลำดับได้อย่างถูกต้อง

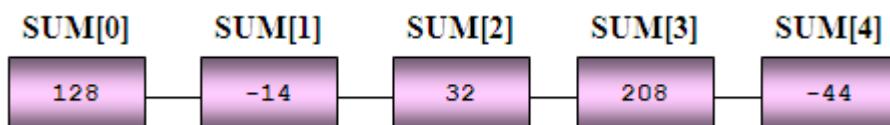
หากต้องการเขียนโปรแกรมเพื่อรับรหัสนักศึกษา และคะแนนสอบกลางภาควิชา Computers and Programming ของนักศึกษาห้อง 1 – 10

```
char id0001[9],id0002[9],id0003[9],...,  
      ...,id1158[9],id1159[9];  
float point0001,point0002,point0003,...,  
      ...,pint158,point1159;  
scanf ("%s",id0001);  
scanf ("%f",&point0001);  
...  
scanf ("%s",id1159);  
scanf ("%f",&point1159);
```

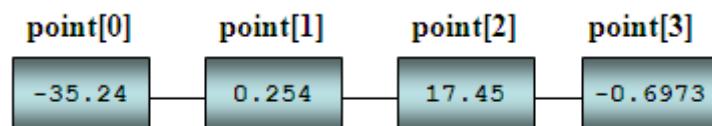
### 7.1 ตัวแปรແຄວลำดับ Array

ตัวแปรอาร์เรย์ คือ ตัวแปรที่สามารถเก็บข้อมูลได้เป็นชุด โดยสร้างตัวแปรขึ้นมาเพียงตัวเดียว (ตัวแปร 1 ตัว จัดเก็บข้อมูลได้หลายค่า) แต่ข้อมูลนั้นต้องเป็นชนิดเดียวกัน

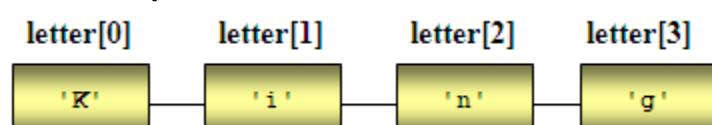
ตัวอย่าง ตัวแปรอาร์เรย์ชื่อ SUM เก็บข้อมูลชนิดจำนวนเต็มขนาด 5 อีลิเมนท์ (element)



ตัวแปรอาร์เรย์ชื่อ point สำหรับเก็บข้อมูลชนิดตัวเลขทศนิยมขนาด 4 อีลิเมนต์



ตัวแปรอาร์เรย์ชื่อ letter สำหรับเก็บข้อมูลชนิดอักษรขนาด 4



#### ลักษณะตัวแปรແຄວลำดับ

ตัวแปรอาร์เรย์ 1 มิติ ตัวแปรอาร์เรย์ที่เก็บข้อมูลได้เพียงແຄວเดียว (ใช้ลำดับในการอ้างถึงข้อมูล)



อาร์เรย์ 2 มิติตัวแปรอาร์เรย์ที่เก็บข้อมูลโดยใช้ แถว(Row) และหลัก(Column) ในการอ้างถึงข้อมูล และอาร์เรย์ 3 มิติ ตัวแปรอาร์เรย์ที่เก็บข้อมูลลักษณะของลูกบาศก์ โดยใช้ แถว(Row) หลัก(Column) และลึก (Deep) ใน การอ้างถึงข้อมูล



## 7.2 อาร์เรย์ 1 มิติ

รูปแบบคำสั่ง

```
type array-name [n]
```

type เป็นชนิดของตัวแปรที่จะสร้างขึ้น  
int ประกาศตัวแปรเป็นชนิดจำนวนเต็ม  
float ประกาศตัวแปรเป็นชนิดทศนิยม  
char ประกาศตัวแปรเป็นชนิดอักขระ  
array-name เป็นชื่อของตัวแปรอาร์เรย์  
n เป็นขนาดของตัวแปรอาร์เรย์

### 7.2.1 การกำหนดค่าเริ่มต้นให้อาร์เรย์

รูปแบบคำสั่ง

```
type array_name [n] = {value1, value2, ..., value};
```

value-1, value-2, ..., value-n เป็นข้อมูลที่จะทำการกำหนดให้กับตัวแปรແລ厝ลำดับ โดยจะต้องเป็นข้อมูลชนิดเดียวกับ type ที่กำหนด

ตัวอย่างการประกาศอาร์เรย์

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

```

int number[3] = {23, -186, 43};
float value_2[5]={0.98,43.213,-3.47,52.08,-0.987};
char vowel[5] = {'a','e','i','o','u'};
char name[9] = {'E','n','g','i','n','e','r','\0'};
return 0;
}

```

## 7.2.2 การอ้างอิงข้อมูลอาร์เรย์

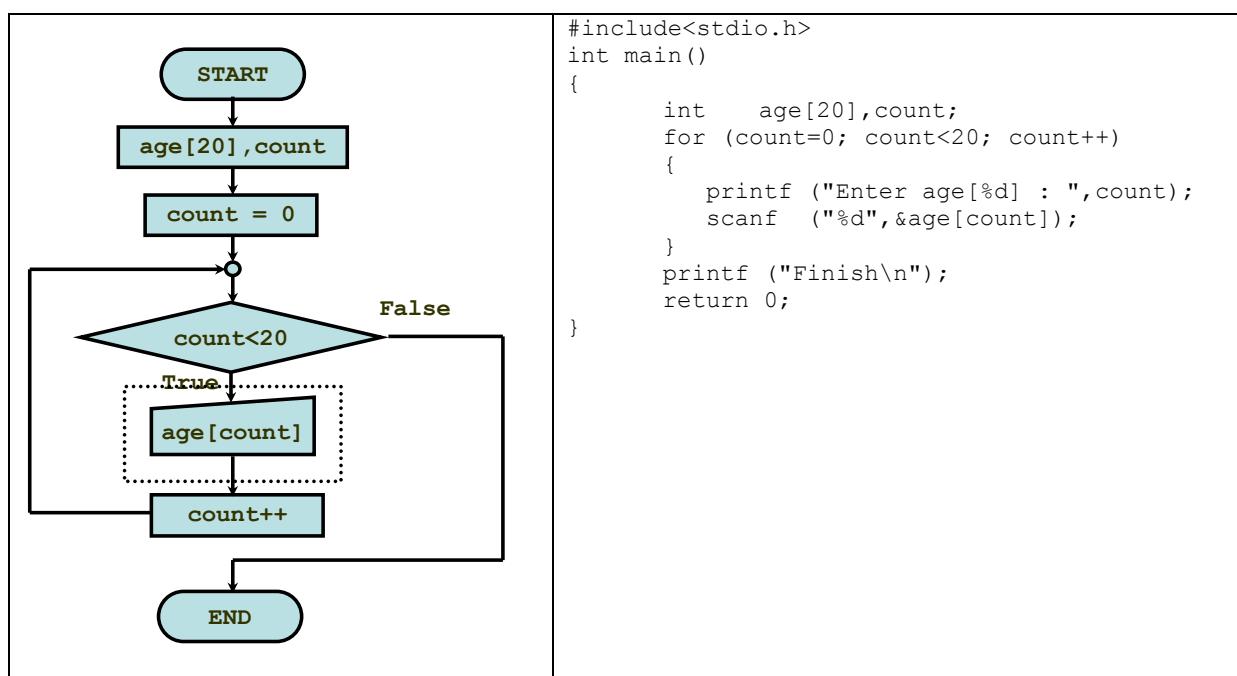
ตัวอย่างการอ้างอิงข้อมูลในอาร์เรย์

#include<stdio.h>	//ผลลัพธ์
#include<conio.h>	
int main()	
{	
int year[5] = {2001,2542,1999,2000,2521};	1999
printf ("%d\n",year[2]);	2521
printf ("%d\n\n",year[4]);	
year[0] = 2545;	2545
printf ("%d\n",year[0]);	
}	

## โปรแกรม 7.1 โปรแกรมรับอายุของคน 20 คน

จงเขียนผังงานและโปรแกรมเพื่อเก็บอายุของผู้ใช้งานจำนวน 20 คนโดยเก็บข้อมูลอายุในตัวแปรชนิดอาร์เรย์

- 1) Output analysis ไม่มี
- 2) Input analysis อายุที่ผู้ใช้งานป้อนเข้ามา
- 3) Process analysis สร้างตัวแปรແควลำดับขนาด 20 เพื่อเก็บค่าอายุ  
โปรแกรมวนรอบเพื่อรับค่าจากผู้ใช้งาน
- 4) Variable define age[20] เป็นตัวแปรແควลำดับชนิดจำนวนเต็มขนาด 20 เพื่อใช้เก็บค่าอายุ  
count เป็นตัวแปรชนิดจำนวนเต็มเพื่อใช้นับจำนวนรอบ



## โปรแกรม 7.2 โปรแกรมวิเคราะห์ส่วนสูงของคน ก คน

จะเขียนผังงานและโปรแกรม เพื่อรับจำนวนนักศึกษาในห้อง หลังจากนั้น ให้โปรแกรมรับส่วนสูงของคน ก คน แล้ววิเคราะห์ว่ามีนักศึกษาในห้องมีส่วนสูงช่วงต่างๆ จำนวนกี่คน และคำนวณส่วนสูงเฉลี่ย แล้วแสดงผลส่วนสูงของนักศึกษาทั้งหมด

Group A	Group B	Group C	Group D
0 - 160	161 - 170	171 - 180	181 - 200

### 1) Output analysis

- จำนวนนักศึกษาที่สูงแต่ละช่วง
- ส่วนสูงของนักศึกษาเฉลี่ยในห้อง
- ส่วนสูงของนักศึกษาทั้งหมด

### 2) Input analysis จำนวนนักศึกษาทั้งหมด และส่วนสูงของแต่ละคน

### 3) Process analysis

- โปรแกรมรับจำนวนนักศึกษา
- วนรอบเพื่อรับส่วนสูงเท่ากับจำนวนนักศึกษา
- วนรอบเพื่อตรวจสอบช่วงส่วนสูงของนักศึกษาและหาผลรวมส่วนสูงของนักศึกษาทุกคน
- คำนวณหาค่าเฉลี่ย

### 4) Variable define

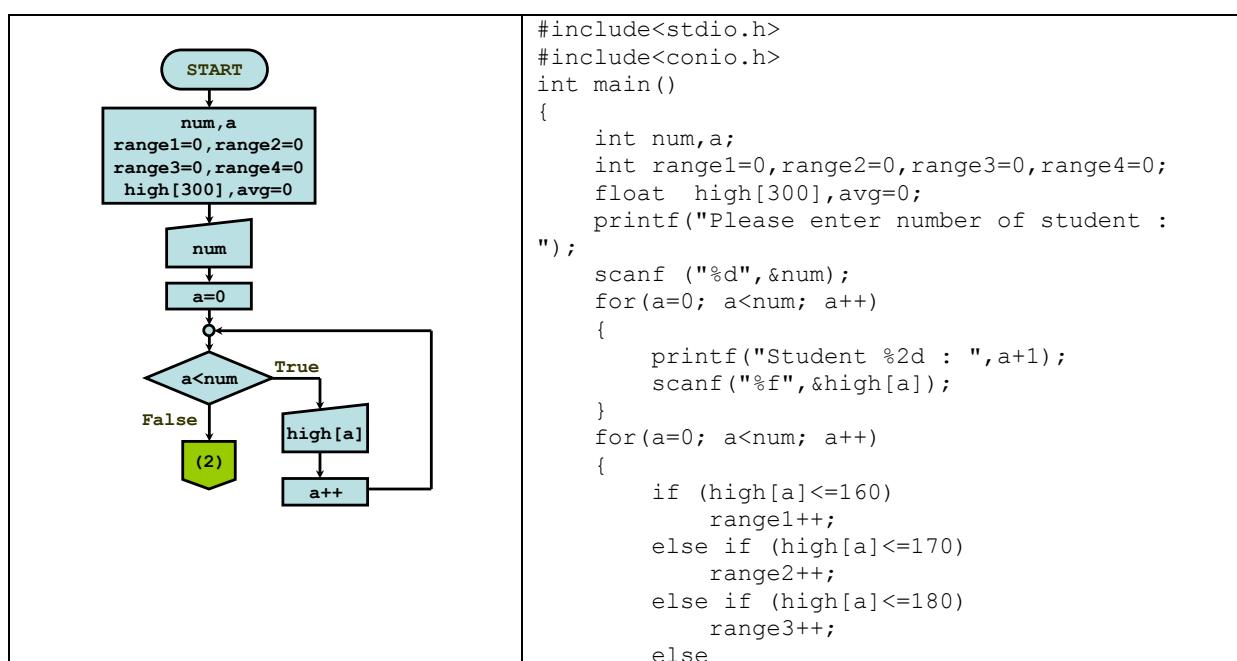
num เป็นจำนวนเต็มเพื่อกีบจำนวนนักศึกษา

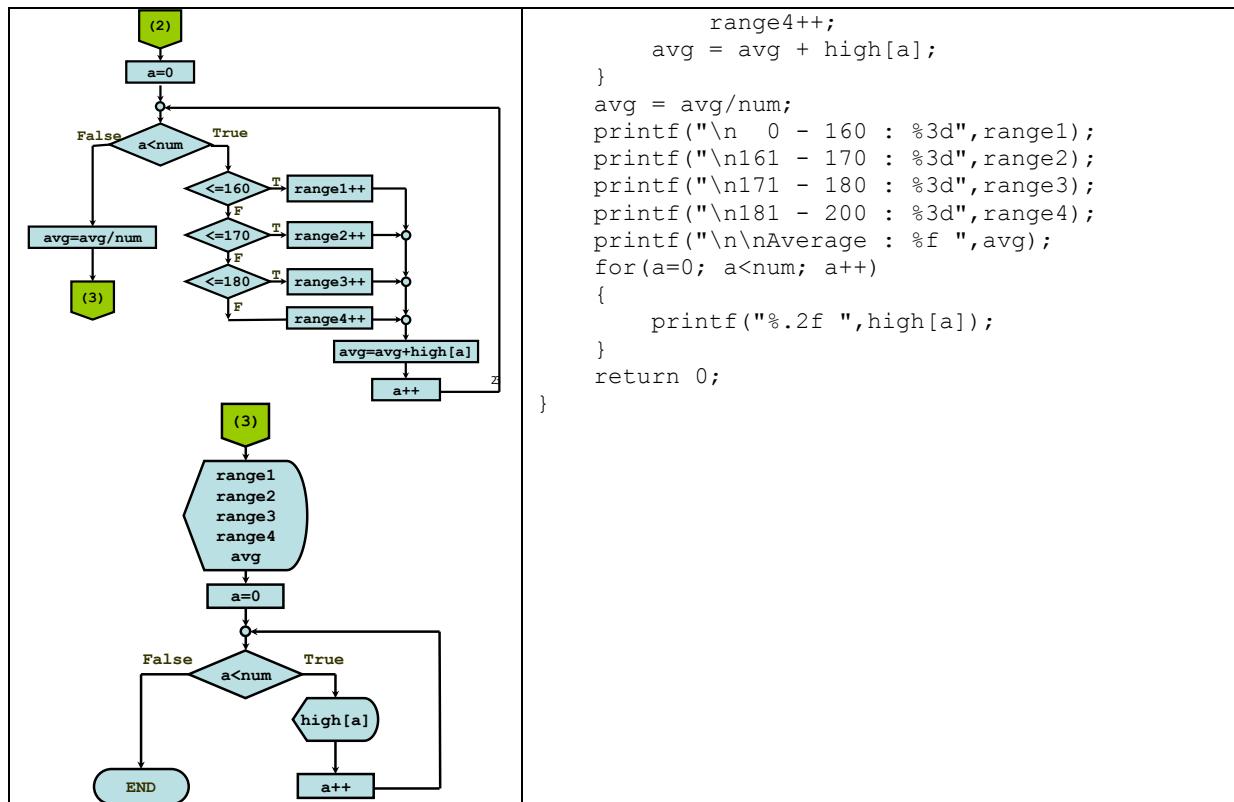
a เป็นจำนวนเต็มเพื่อตรวจตำแหน่งตัวแปร และนับรอบ

range1=0, range2=0, range3=0, range4=0 เป็นจำนวนเต็มสำหรับเก็บค่าจำนวนนักศึกษาแต่ละช่วง

high[300] เป็นตัวแปรແຄவ์ดับชนิดศนิยมเพื่อกีบส่วนสูง

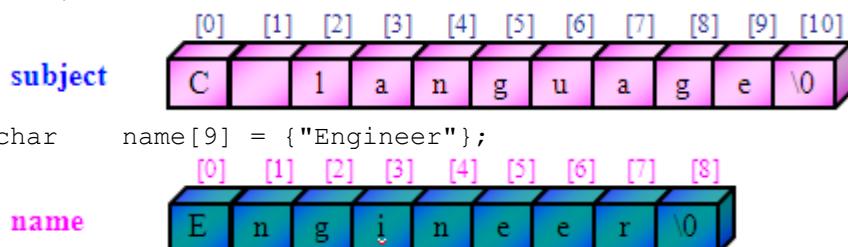
avg = 0 เป็นจำนวนทศนิยมเพื่อกีบค่าผลรวมและค่าเฉลี่ย





### 7.3 ตัวแปรและเก็บข้อมูล

char subject[11] = {"C language"};  
หรือ  
char subject[11] = {'C', ' ', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', '\0'};



ตัวอย่างโปรแกรมเก็บข้อมูลและแสดงผล

```
#include<stdio.h>
int main()
{
    char sentence[22] = "Welcome to my country";
    char word[9] = {'T', 'h', 'a', 'i', 'l', 'a', 'n', 'd', '\0'};
    char not_word[4] = {'l', 'o', 'v', 'e'};
    printf ("Message1 = %s\n", sentence);
    printf ("Message2 = %s\n", word);
    printf ("Message3 = %s\n", not_word);
    return 0;
}
```

//ผลลัพธ์  
Message1 = Welcome to my country  
Message2 = Thailand  
Message3 = loveThailand

### 7.4 อาร์เรย์ 2 มิติหรือมากกว่า

รูปแบบคำสั่ง

```
type   array-name [n] [m];
```

- type เป็นชนิดของตัวแปรที่จะสร้างขึ้น
- int ประกาศตัวแปรเป็นชนิดจำนวนเต็ม
- float ประกาศตัวแปรเป็นชนิดทศนิยม
- char ประกาศตัวแปรเป็นชนิดอักขระ
- array-name เป็นชื่อของตัวแปรอาร์เรย์
- n เป็นจำนวนแคลวของตัวแปรอาร์เรย์
- m เป็นจำนวนคอลัมน์ของตัวแปรอาร์เรย์

#### 7.4.1 การกำหนดค่าเริ่มต้นให้อาร์เรย์ 2 มิติ

รูปแบบคำสั่งการกำหนดค่าให้อาร์เรย์ 2 มิติ

```
type array-name [n] [m] = {value1,value2,...,valueN};
```

value-1-1, value-1-2, ..., value-1-n, ..., ..., value-n-m เป็นข้อมูลที่จะทำการกำหนดให้กับตัวแปรແລ厝ำดับ โดยจะต้องเป็นข้อมูลชนิดเดียวกับ type ที่กำหนด

ตัวอย่างการกำหนดค่าเริ่มต้นและการอ้างค่าในอาร์เรย์ 2 มิติ

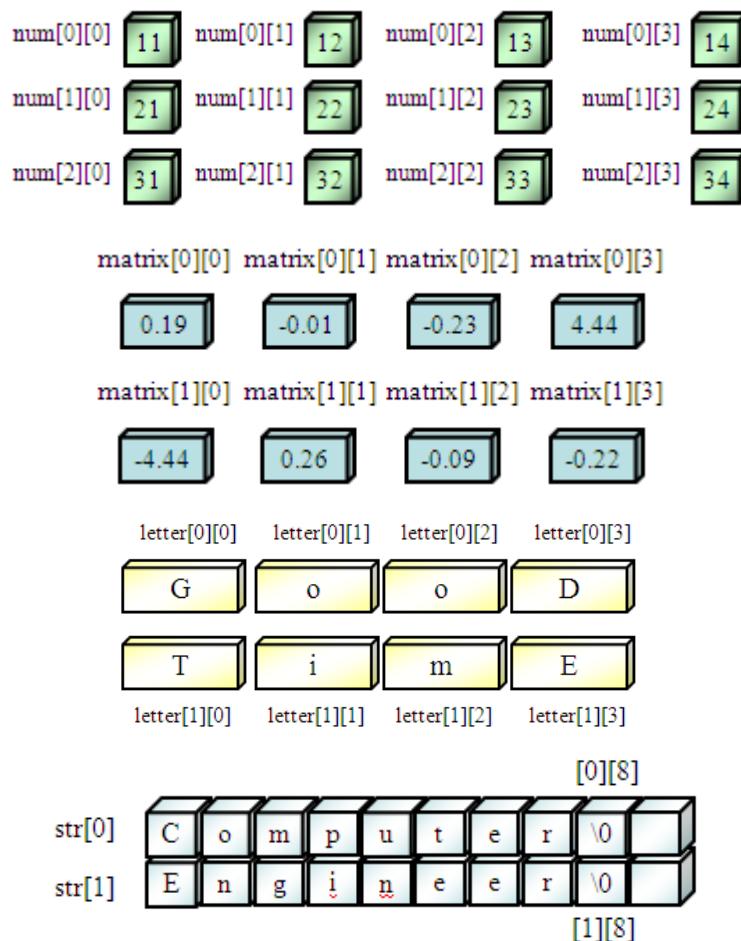
##### ลักษณะการเขียนประกาศแบบที่ 1

```
int num[3][4] = { 11, 12, 13, 14,
                  21, 22, 23, 24,
                  31, 32, 33, 34 };
float matrix[2][4] = { 0.19, -0.01, -0.23, 4.44,
                      -4.44, 0.26, -0.09, -0.22 };
char letter[2][4] = { 'G', 'o', 'o', 'D',
                      'T', 'i', 'm', 'E' };
char str[2][10] = {"Computer",
                   "Engineer" };
```

##### ลักษณะการเขียนประกาศแบบที่ 2 ในกรณีที่อาร์เรย์มีขนาดยาวกว่าบรรทัดที่เขียน

```
int num[3][4] = { 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34 };
float matrix[2][4] = { 0.19, -0.01, -0.23, 4.44, -4.44, 0.26, -0.09, -0.22 };
char letter[2][4] = {'G', 'o', 'o', 'D', 'T', 'i', 'm', 'E'};
char str[2][10] = {"Computer", "Engineer" };
```

การอ้างค่าตำแหน่งดังรูปด้านล่าง



### โปรแกรม 7.3 โปรแกรมรับข้อมูลและแสดงผลลัพธ์แบบเมทริกซ์

จะเขียนโปรแกรมเพื่อรับค่าจำนวนเต็มในรูปแบบเมทริกซ์โดยเก็บค่าไว้ในตัวแปร แவล่าดับ ขนาด  $3 \times 3$  แล้วแสดงผลเมทริกซ์

<pre>#include&lt;stdio.h&gt; #include&lt;conio.h&gt; int main() {     int matrix[3][3],r,c;     for(r=0; r&lt;3; r++)     {         for(c=0; c&lt;3; c++)         {             printf("Enter numbers [%d] [%d] : ",r,c);             scanf("%d",&amp;matrix[r][c]);         }     }     printf ("\n*** Matrix ***\n");     for (r=0; r&lt;3; r++)     {         for(c=0; c&lt;3; c++)         {             printf ("%5d ",matrix[r][c]);         }     }     printf ("\n"); } return 0;</pre>	<p>//ผลลัพธ์ของโปรแกรม //รับค่า</p> <pre>Enter numbers [0] [0] : 1 Enter numbers [0] [1] : 2 Enter numbers [0] [2] : 3 Enter numbers [1] [0] : 4 Enter numbers [1] [1] : 5 Enter numbers [1] [2] : 6 Enter numbers [2] [0] : 7 Enter numbers [2] [1] : 8 Enter numbers [2] [2] : 9</pre> <p>//แสดงเมทริกซ์</p> <pre>***Matrix***    1   2   3    4   5   6    7   8   9</pre>
---	---

## โปรแกรม 7.4 หาผลรวมในเมทริกซ์

จากตัวแปรແຄລາດັບທີ່ກຳນົດໄທ້ ຈະເຂື້ອນໂປຣແກຣມຫາພວກຫຼຸມຂອງຈຳນວນໃນແຕ່ລະຫັກ ແລະພວກຫຼຸມຂອງຈຳນວນໃນແຕ່ລະແກວ ໂດຍເກີບຄ່າພວກຫຼຸມໄວ້ໃນຕັ້ງແປຣ `row[ ]`, `column[ ]`

ກຳນົດ `int num[3][4] = {1, 2, 3, 4, 2, 3, 4, 5, 3, 4, 5, 6};`

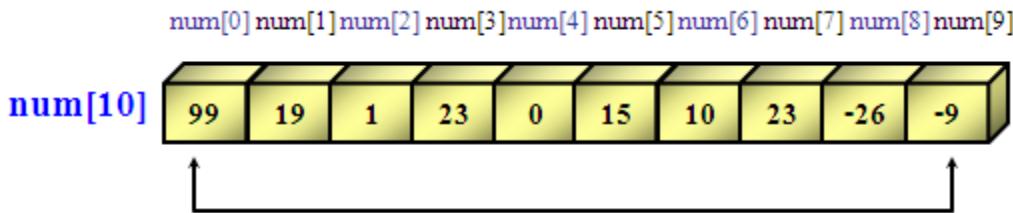
<pre>#include&lt;stdio.h&gt; int main() {     int num[3][4] = { 1, 2, 3, 4,                       2, 3, 4, 5,                       3, 4, 5, 6 };     Int r,c,row[3]={0,0,0},column[4]={0,0,0,0};      /* Display Matrix */     printf ("*** Show Matrix ***\n\n");     for (r=0; r&lt;3; r++)     {         for(c=0; c&lt;4; c++)             printf ("%5d ",num[r][c]);         printf ("\n\n");     }      /* Summation Matrix */     for (r=0; r&lt;3; r++)         for(c=0; c&lt;4; c++)         {             row[r] = row[r] + num[r][c];             column[c] = column[c] + num[r][c];         }      /* Display Summation */     printf ("\n\n");     for (r=0; r&lt;3; r++)         printf ("sum of row [%d] = %d\n",r,row[r]);     for (c=0; c&lt;4; c++)         printf("sum of column [%d] = %d\n",c,column[c]);     return 0; }</pre>	<p>//ຜລລັບພື້ນ</p> <p>*** Show Matrix ***</p> <table border="0"> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> <p>Sum of row[0] = 10      Sum of row[1] = 14      Sum of row[2] = 18      Sum of column[0] = 6      Sum of column[1] = 9      Sum of column[2] = 12      Sum of column[3] = 15</p>	1	2	3	4	2	3	4	5	3	4	5	6
1	2	3	4										
2	3	4	5										
3	4	5	6										

## โปรแกรม 7.5 ນັບຈຳນວນຂໍ້ຄວາມ

ຈະເຂື້ອນຜັງຈານ ແລະໂປຣແກຣມ ເພື່ອຮັບຂໍ້ຄວາມເຂົ້າມາ ຕຽບສອບວ່າມີທີ່ໜູນດັກປະໂຍດ

<pre> graph TD     START([START]) --&gt; Init[/count=0, c=0, message[100]/]     Init --&gt; Message[/message/]     Message --&gt; Cond{message[c] != '\0'}     Cond -- False --&gt; END([END])     Cond -- True --&gt; PointQu{message[c] == POINT    message[c] == QU}     PointQu -- True --&gt; CountInc[/count++/]     CountInc --&gt; Cpp[/c++/]     PointQu -- False --&gt; Cpp     Cpp --&gt; Cond     Cpp --&gt; END     END --&gt; Output[/Your message has %d sentence(s). \n, count/] </pre>	<pre>#include&lt;stdio.h&gt; int main() {     char message[100];     int count=0,c=0;     printf ("*****\n");     printf ("* Program for count SENTENCE *\n");     printf ("*****\n");     printf ("\n(One sentence must have '.' or '?')\n");     printf ("Enter Message : ");     gets(message);     while (message[c]!='\0')     {         if (message[c]=='.'    message[c]=='?')             count++;         c++;     }     printf ("\n\nYour message has %d sentence(s). \n",count);     return 0; }</pre>
---	---

## 7.5 การสลับค่าในอาร์เรย์



ตัวอย่าง code

```
temp = num[0]
num[0] = num[9];
num[9] = temp;
```

### โปรแกรม 7.6 หาค่ามากที่สุด

จะเขียนโปรแกรมเพื่อรับจำนวน 10 จำนวน เพื่อหาค่ามากที่สุดในจำนวนที่รับมา โดยการย้ายค่ามากที่สุดไปปั๊ว์ Array ที่ตำแหน่งสุดท้าย

<p>num[0] num[1] num[2] num[3] num[4] num[5] num[6] num[7] num[8] num[9]</p> <p><b>num[10]</b> [ 0   19   1   23   99   15   10   23   -26   -9 ]</p> <p><b>num[10]</b> [ 0   19   1   23   99   15   10   23   -26   -9 ]</p> <p><b>num[10]</b> [ 0   1   19   23   99   15   10   23   -26   -9 ]</p> <p><b>num[10]</b> [ 0   1   19   23   15   99   10   23   -26   -9 ]</p> <p><b>num[10]</b> [ 0   1   19   23   15   10   23   -26   -9   99 ]</p>	<pre>#include&lt;stdio.h&gt; #define SIZE 10 int main() {     int num[SIZE], temp, n;     for (n=0; n&lt;SIZE; n++)     {         printf("Enter num[%d] : ", n+1);         scanf("%d", &amp;num[n]);     }     for (n=0; n&lt;SIZE-1; n++)     {         if (num[n]&gt;num[n+1])         {             temp = num[n+1];             num[n+1] = num[n];             num[n] = temp;         }     }     printf ("The maximum number = %d", num[SIZE-1]);     return 0; }</pre>
---	--

## 7.6 คำถาวรห้ายบท

- 1) กำหนดให้ Matrix A มีขนาด  $3 \times 3$  ดังนี้

$$A = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 2 & 3 \\ 2 & 4 & 5 \end{bmatrix}$$

จะเขียนโปรแกรมหา Diagonal matrix ของ A โดยให้นำผลที่ได้ใส่ลงใน Array A

$$\text{Diagonal matrix of } A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

2) จากโจทย์ข้อที่ 1) จงเขียนโปรแกรมหา Transpose ของ Matrix A โดยให้นำผลมาใส่ใน Matrix A

$$\text{Transpose of } A = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 4 \\ -1 & 3 & 5 \end{bmatrix}$$

3) จาก Matrix A ในข้อที่ 1 จงเขียนโปรแกรมเพื่อแสดงผลคูณของ  $A \times A$

$$A \times A = \begin{bmatrix} (1)(1) + (0)(-1) + (-1)(2) & (1)(0) + (0)(2) + (-1)(4) & (1)(-1) + (0)(3) + (-1)(5) \\ (-1)(1) + (2)(-1) + (3)(2) & (-1)(0) + (2)(2) + (3)(4) & (-1)(1) + (2)(3) + (3)(5) \\ (2)(1) + (4)(-1) + (5)(2) & (2)(0) + (4)(2) + (5)(4) & (2)(-1) + (4)(3) + (5)(5) \end{bmatrix}$$

ผลลัพธ์

$$A \times A = \begin{bmatrix} -1 & -4 & -6 \\ 3 & 16 & 22 \\ 8 & 28 & 35 \end{bmatrix}$$

## การทดลองที่ 7 ตัวแปรແຄວลำดับ

### วัตถุประสงค์

1. เพื่อให้นักศึกษาทดลองเขียนโปรแกรมโดยใช้ตัวแปรແຄວลำดับ
2. เพื่อให้นักศึกษาสามารถใช้งานตัวแปรແຄວลำดับได้
3. เพื่อให้นักศึกษาเขียนโปรแกรมโดยประยุกต์ใช้ความรู้ที่ผ่านมากับตัวแปรແຄວลำดับได้

### ทฤษฎี

ในการเขียนโปรแกรมตามโจทย์ที่ได้เรียนมาในบทก่อนหน้านี้ จะเป็นโจทย์ที่มีรูปแบบการกำหนดตัวแปรที่ลักษณะเด่นนั้น ซึ่งในการใช้งานจริงการเขียนโปรแกรมโดยกำหนดตัวแปรที่ลักษณะข้อจำกัดในการเขียนโปรแกรมอยู่สองประเด็นใหญ่ๆ ประเด็นแรกคือกรณีที่ต้องเก็บข้อมูลเป็นชุดจะเกิดความยุ่งยากในการเขียนโปรแกรมมาก เช่น ข้อมูลความสูงของนักศึกษาจำนวน 50 คน ถ้าต้องเขียนโปรแกรมในลักษณะนี้โดยกำหนดตัวแปรที่ลักษณะเดียวกัน จะทำให้ต้องเขียนโปรแกรมที่มีการทำงานซ้ำซ้อนกันหลายบรรทัดโดยไม่สามารถใช้การเขียนโปรแกรมแบบวนซ้ำมาช่วยให้เขียนโปรแกรมได้ง่ายขึ้นได้เลย

อีกข้อจำกัดหนึ่งที่สำคัญมาก นั่นคือในกรณีที่เขียนโปรแกรมแล้วต้องมีการเก็บข้อมูลหรือรับข้อมูลที่โปรแกรมเมอร์ไม่ทราบว่าเตรียมข้อมูลที่แท้จริงเป็นเท่าใด จำเป็นต้องมีการกำหนดจำนวนตัวแปรในการเก็บข้อมูลเพื่อไว้ก่อน ซึ่งในกรณีนี้โปรแกรมเมอร์จะไม่ทราบได้เลยว่าจะต้องทำงานกับตัวแปรใดแต่ละตัวอย่างไร จึงไม่สามารถเขียนโปรแกรมในลักษณะนี้ได้

ทางออกสำหรับปัญหาใหญ่สองปัญหาดังกล่าวคือการกำหนดตัวแปรเป็นกลุ่ม ที่เราเรียกว่า Array ซึ่ง Array จะเป็นชุดของข้อมูลชนิดเดียวกัน เรียกว่า 1 มิติ ตามรูปแบบที่ผู้ใช้งานกำหนด โดยโปรแกรมเมอร์สามารถกำหนดขนาดรวมถึงลักษณะ มิติของ Array ที่ต้องการได้ อีกทั้งโปรแกรมเมอร์จะสามารถอ้างอิงค่าที่เก็บไว้ใน Array ได้อย่างง่ายดาย Array คือหน่วยเก็บข้อมูลที่มีลักษณะเป็นกลุ่มของข้อมูลที่เรียงต่อกัน โดยอาจเป็นการเรียงต่อกันเป็นแบบ 1 มิติ เรียงกันเป็นแบบ 2 มิติ หรือมีองค์ประกอบซึ่งกันและกัน หรือจะมากกว่านั้นก็ได้ Array จะอนุญาตให้มีการเก็บข้อมูลเพียงชนิดเดียวเท่านั้นในกลุ่มข้อมูล ทั้งหมด นั่นคือเราจะไม่สามารถเก็บข้อมูลที่มีชนิดต่างกันได้เลยใน Array จะต้องใช้การเก็บข้อมูลในรูปแบบอื่นๆ แทน

สำหรับผู้เริ่มต้นเขียนโปรแกรม ควรเริ่มต้นจากการใช้งาน Array 1 มิติ และ Array 2 มิติ ซึ่งเป็น Array ที่มีขนาดเล็ก การใช้งานไม่ซับซ้อน แต่เป็นพื้นฐานในการเขียนโปรแกรมที่ซับซ้อนและต้องใช้ข้อมูลหลายๆ มิติในอนาคตได้สำหรับการใช้งาน Array นักศึกษาควรมีความเข้าใจและสามารถทำงานต่างๆ กับ Array ตั้งต่อไปนี้ได้

1. การสร้าง Array ที่มีขนาด และชนิดข้อมูลตามที่ต้องการได้
2. การนำข้อมูลเข้าเก็บใน Array และการนำข้อมูลใน Array ออกมายังงาน

### การสร้าง Array ที่มีขนาด และชนิดข้อมูลตามที่ต้องการได้

การสร้าง Array จะนั้น จะมีลักษณะคล้ายกับการกำหนดค่าตัวแปรต่างๆ แต่จะแตกต่างกันเล็กน้อยตรงที่ Array ต้องมีการกำหนดขนาดของ Array ด้วยทุกครั้งโดยรูปแบบการสร้าง Array สำหรับเก็บข้อมูลจะมีรูปแบบการประกาศดังนี้

Array 1 มิติ : type array-name [n]

Array 2 มิติ : type array-name [r][c]

โดย type คือชนิดของตัวแปร เช่น int , float , char เป็นต้น

array\_name คือชื่อของตัวแปร array โดยข้อกำหนดต่างๆ ของชื่อ array จะเหมือนกับ

ข้อกำหนดของชื่อตัวแปรทุกประการ

n คือ ตัวเลขขนาดของ array 1 มิติ ที่ได้กำหนดไว้

r คือ ตัวเลข จำนวนแพร ของ array 2 มิติ

c คือ ตัวเลข จำนวนหลักของ array 2 มิติ

### ตัวอย่างการใช้งาน

- ตัวอย่างการประกาศ array ที่ใช้ในการเก็บความสูงของนักศึกษา 50 คน เราจะต้องมีการกำหนดค่าอุปกรณ์ของ array จำนวน 50 จำนวน โดยเราจะกำหนด Array ชื่อ height เก็บข้อมูลความสูงของนักศึกษาทั้ง 50 คนได้ดังนี้

```
int height[50];
```

- ตัวอย่างการประกาศ array 2 มิติที่ใช้ในการเก็บข้อมูลตัวเลขขนาด  $40 \times 50$  ช่อง สามารถประกาศตัวแปรดังนี้

```
int data[40][50];
```

### การนำข้อมูลเข้าเก็บใน Array และการนำข้อมูลใน Array ออกมายังงาน

ในการอ้างอิงข้อมูลใน Array เพื่อดึงข้อมูลออกมาใช้งานหรือเก็บข้อมูลลงไว้ใน Array จะสามารถใช้งานโดยการเขียนชื่อตัวแปร ตามด้วยตำแหน่งของช่องที่ต้องการดึงข้อมูลออกมา โดยช่องแรกของ Array จะมีหมายเลข 0

### ตัวอย่างการกำหนด Array 1 มิติ : int a[9];

รูปแบบข้อมูลที่ถูกสร้างขึ้น



ในแต่ละช่องจะสามารถบรรจุข้อมูลตัวเลขชนิด int จำนวน 9 ช่อง

การใส่ข้อมูลเลข 1 ลงไประบบแรก สามารถทำได้โดยใช้คำสั่ง

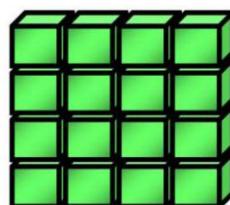
```
a[0] = 1;
```

การนำเอาค่าตัวเลขในช่องแรก มาแสดงผลในหน้าจอ สามารถทำได้โดยใช้คำสั่ง

```
printf ("%d",a[0]);
```

### ตัวอย่างการกำหนด Array 2 มิติ : int b[4][4];

รูปแบบข้อมูลที่ถูกสร้างขึ้น



ในแต่ละช่องจะสามารถบรรจุข้อมูลตัวเลขชนิด int จำนวน 16 ช่อง แบ่งเป็น 4 แถว แต่ละแถวมี 4 ช่องการใส่ข้อมูลเลข 5 ลงไประบบแรกที่ 3 หลักที่ 4 สามารถทำได้โดยใช้คำสั่ง

```
b[2][3] = 5;
```

การนำเอาค่าตัวเลขในในแถวที่ 3 หลักที่ 4 มาแสดงผลในหน้าจอ สามารถทำได้โดยใช้คำสั่ง

```
printf ("%d", b[2][3]);
```

โดยเราจะสังเกตเห็นว่า โดยวิธีการเขียนโปรแกรมแล้วการอ้างอิง Array ก็คือการเขียนตัวแปรรرمดาเท่านั้นเพียงแค่เพิ่มเติมตัวเลขแสดงตำแหน่ง (index) ของค่าข้อมูลนั้นๆ ใน Array เท่านั้น การเขียนโปรแกรมกับ Array จึงไม่ได้แตกต่างจากการเขียนโปรแกรมโดยทั่วไปเลย

### ตัวอย่างการใช้งาน

1. การเขียนโปรแกรมสร้าง Array เก็บตัวเลขจำนวนเต็ม 10 จำนวน โดยเก็บค่าตัวเลข 1-10 ตามลำดับทำได้ดังนี้

```
#include<stdio.h>
int main()
{
    int i,data[10];
    for (i=0;i<10;i++)
    {
        data[i]=i+1;
        printf("data[%d] = %d\n",i,data[i]);
    }
    return 0;
}
```

2. การเขียนโปรแกรมสร้าง Array เก็บตัวเลข 10 ตัวโดยรับค่าจากคีย์บอร์ด และแสดงผลลัพธ์จากตัวเลขจากหลังมาหน้า

```
#include<stdio.h>
int main()
{
    int i,data[10];
    for (i=0;i<10;i++)
    {
        printf("input number : ");
        scanf("%d",&data[i]);
    }
    printf("reverse data :");
    for (i=9;i>=0;i--)
    {
        printf("%d ",data[i]);
    }
    return 0;
}
```

### การทดลองที่ 7.1

จากตัวอย่างการใช้งานคำสั่ง ให้นักศึกษาประยุกต์ใช้งานดังต่อไปนี้

1. การเขียนโปรแกรมสร้าง Array เก็บตัวเลขจำนวนเต็ม 10 จำนวน โดยเก็บค่าตัวเลข 10-1 ตามลำดับ
2. การเขียนโปรแกรมสร้าง Array เก็บตัวเลข 10 ตัวโดยรับค่าจากคีย์บอร์ด และแสดงผลลัพธ์จากตัวเลขจากหลังมาหน้า โดยแสดงผลเฉพาะตัวเลขที่เป็นเลขคู่เท่านั้น
3. การเขียนโปรแกรมสร้าง Array เก็บตัวเลข 10 ตัวโดยรับค่าจากคีย์บอร์ด และแสดงผลลัพธ์จากตัวเลขจากหลังมาหน้า โดยแสดงผลเฉพาะลำดับที่เป็นเลขคี่เท่านั้น

### การทดลองที่ 7.2

ให้นักศึกษาทดลองเขียนโปรแกรมต่อไปนี้แล้วสังเกตผลลัพธ์ขณะ compile โปรแกรม หรือรันโปรแกรม และตอบคำถาม

1. ให้นักศึกษาทดสอบการทำงานเมื่อมีการอ้างอิงข้อมูลในตำแหน่งเกินจากที่ได้กำหนดไว้ โดยกำหนด int

a[10]; (ให้ Array a เก็บชนิดข้อมูลชนิด int จำนวน 10 จำนวน) แต่ทดสอบในโปรแกรมโดยใช้คำสั่งกำหนดค่า  
a[15] = 100; (กำหนดค่า Array a ตำแหน่งที่ 15 ให้เก็บชนิดข้อมูล int ที่มีค่า 100)

ผลลัพธ์ที่เกิดขึ้น

2. ให้นักศึกษาทดสอบการทำงานเมื่อมีการใส่ข้อมูลลงใน Array โดยที่ข้อมูลที่กำหนดให้มี type ไม่ตรงกับ type ของ Array โดยกำหนด `int a[10];` (ให้ Array a เก็บชนิดข้อมูลชนิด int จำนวน 10 จำนวน) แต่ทดสอบใน โปรแกรมโดยใช้คำสั่งกำหนดค่า `a[3]=5.5 ;` (กำหนดค่า Array a ตำแหน่งที่ 3 ให้เก็บชนิดข้อมูล float ที่มีค่า 5.5) ผลลัพธ์ที่เกิดขึ้น
- 
- 

### การทดลองที่ 7.3

ให้นักศึกษาเขียนโปรแกรมต่อไปนี้

1. ให้นักศึกษาเขียนโปรแกรมรับค่าเมตริกซ์ขนาด  $3 \times 3$  จำนวน 2 ตัว และคำนวณหาผลคูณของเมตริกซ์ทั้งสอง และพิมพ์ผลลัพธ์ออกมา ดังตัวอย่าง

```

input:
Enter the matrix A:
1 2 3
4 5 6
7 8 9

input:
Enter the matrix B:
9 8 7
6 5 4
3 2 1

output:
The matrix C = A * B is:
30 24 18
84 69 54
138 114 90

```

2. ให้นักศึกษาเขียนโปรแกรมเพื่อรับตัวเลข 10 ตัวตามลำดับ และแสดงผลลัพธ์ว่าตัวเลขใดที่มีตัวเลขข้างเคียงเป็นเลขคี่ ดังตัวอย่าง

```

Data : 1 3 6 7 8 9 4 5 6 6
Result : 6 8 4

Data : 2 1 4 9 5 8 5 3 1 7
Result : 4 8 3 1

```

3. ให้นักศึกษาเขียนโปรแกรมเพื่อรับข้อมูล String จำนวน 1 ชุด และแสดงผลลัพธ์โดยเปลี่ยนตัวอักษรตัวพิมพ์เล็กให้กลายเป็นตัวอักษรตัวพิมพ์ใหญ่ และเปลี่ยนตัวอักษรตัวพิมพ์ใหญ่ให้กลายเป็นตัวอักษรตัวพิมพ์เล็ก

4. ให้เขียนโปรแกรมรับจำนวนเต็ม 10 จำนวนเก็บไว้ในตัวแปรแล้วคำนวณ และเรียงค่าจากมากไปหาน้อยแล้วแสดงผลลัพธ์ที่ได้ออกทางจอภาพดังตัวอย่าง

```

input : 99 -89 77 196 21 0 -99 55 66 19
output : 196 99 77 66 55 21 19 0 -89 -99

```

## บทที่ 8 ตัวแปรโครงสร้าง Structure

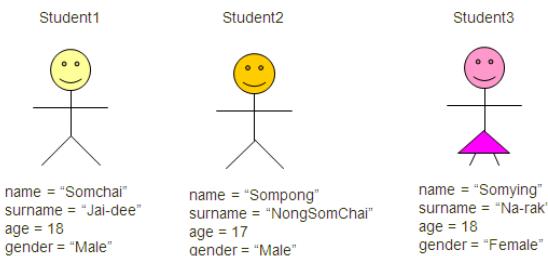
### วัตถุประสงค์การศึกษา

- 1) ศึกษารการตัวแปรโครงสร้าง
- 2) เขียนโปรแกรมที่มีการใช้ตัวแปรโครงสร้างเก็บข้อมูลได้

### 8.0 ตัวแปรโครงสร้าง

ตัวแปรโครงสร้างมีคุณสมบัติในการจัดกลุ่มข้อมูล เช่น ข้อมูลนักเรียนประกอบด้วยชื่อ นามสกุล อายุ เกรด ทั้งหมด เป็นข้อมูลที่รวมอยู่ในนักเรียนหนึ่งคน ภายใต้ตัวแปรโครงสร้างหนึ่งตัวสามารถมีพารามิเตอร์หรือตัวแปรได้หลายๆตัว และมีชนิดของตัวแปรแตกต่างกัน

ตัวอย่าง นักเรียน(student) ประกอบไปด้วย ชื่อ(name), นามสกุล(surname), อายุ(age) และ เพศ (gender)

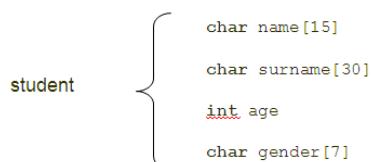


หากไม่ใช้ตัวแปรโครงสร้าง เราจะต้องเขียนโปรแกรมเก็บข้อมูลของนักเรียนทั้ง 3 คนดังนี้

```
#include<stdio.h>
main( )
{
    char student1_name[15], student2_name[15], student3_name[15];
    char student1_surname[30], student2_surname[30], student3_surname[30];
    int student1_age, student2_age, student3_age;
    char student1_gender[7], student2_gender[7], student3_gender[7];
}
```

จะได้เกิดขึ้น หากเราต้องเขียนโปรแกรมรองรับนักเรียน 2,000 คน? โปรแกรมจะยาวมาก และทำให้เขียนโปรแกรมยากลำบาก

โครงสร้างข้อมูล (Structure) คือการกำหนดตัวแปรชนิดใหม่ขึ้นมาใช้งาน โดยนำตัวแปรพื้นฐานในภาษา C มารวมกันเป็นโครงสร้างของตัวแปรชนิดใหม่ เช่น



## 8.1 การประกาศตัวแปรโครงสร้าง

//รูปแบบที่ 1	//รูปแบบที่ 2
<pre>struct name {     type var_1;     type var_2;     .....     type var_n; };  struct name struct_var;</pre>	<pre>struct name {     type var_1;     type var_2;     .....     type var_n; } struct_var;</pre>

ตัวอย่าง ประกาศตัวแปรโครงสร้างใช้เก็บข้อมูลนศ. ซึ่งประกอบด้วย ชื่อ นามสกุล อายุ และเพศ

//รูปแบบที่ 1	//รูปแบบที่ 2
<pre>struct student {     char name[15];     char surname[30];     int age;     char gender[7]; };  struct student st1,st2;</pre>	<pre>struct student {     char name[15];     char surname[30];     int age;     char gender[7]; } st1,st2;</pre>

## 8.2 การอ้างถึงตัวแปรในโครงสร้าง

การกำหนดหรืออ่านค่าของตัวแปร ภายในโครงสร้างทำได้ โดยมีรูปแบบดังนี้

struct\_var.var\_name

การเข้าถึงตัวแปรแต่ละตัวจะใช้ "." นำหน้า ตามด้วยชื่อตัวแปร เช่น

	name	count	price
pr1	pr1.name	pr1.count	pr1.price
pr2	pr2.name	pr2.count	pr2.price
	pr1.name		
	pr1.count		
	pr2.price		
	pr2.count		เป็นต้น

## 8.3 การกำหนดข้อมูลให้ตัวแปรโครงสร้าง

วิธีการกำหนดข้อมูลให้กับตัวแปรที่เป็นสมาชิกของโครงสร้าง ทำได้ในลักษณะแบบเดียวกับตัวแปรปกติ เช่น

pr1.count = 10;

pr1.price = 30;

แต่ในกรณีที่ตัวแปรเป็นชนิดข้อความ เช่น

pr1.name = "Joy" /\* error \*/

ไม่สามารถกำหนดค่าให้ได้ ต้องใช้คำสั่ง strcpy( ) เช่น

strcpy(pr1.name, "Joy") /\*pass\*/

หมายเหตุ: strcpy() เป็นคำสั่งคัดลอกข้อความ เมื่อใช้งานจำเป็นต้อง #include <string.h>

### โปรแกรม 8.1 ประกาศตัวแปรโครงสร้าง

```
#include<stdio.h>
#include<conio.h>
int main()
{
    struct income {
        float salary;
        float bonus;
        int      age;
    };
    struct income people1;
    people1.salary = 16000;
    people1.bonus = 40000;
    people1.age = 23;
    return 0;
}
```

### โปรแกรม 8.2 กำหนดค่าให้ตัวแปรโครงสร้าง

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    struct letter {
        char name[20];
        char surname[30];
        char message[50];
    } first;
    printf ("Enter name : ");
    scanf ("%s", first.name);
    printf ("Enter surname : ");
    scanf ("%s", first.surname);
    strcpy (first.message,"Hi");
    printf ("%s",first.message);
    printf ("\n%s",first.name);
    printf (" %s",first.surname);
    return 0;
}
```

### โปรแกรม 8.3 โปรแกรมเก็บข้อมูลหนังสือ

จะเขียนโปรแกรมสำหรับเก็บข้อมูลของหนังสือ 1 เล่ม โดยข้อมูลของหนังสือประกอบด้วยคือ ชื่อหนังสือ, ราคา และส่วนลด (คงที่ 10 %) โดยกำหนดให้ใช้เก็บข้อมูลแบบโครงสร้าง โปรแกรมจะแสดงผลการทำงานดังนี้

```
Enter book name : Programming in TurboC
Enter book price : 200

Book : Programming in TurboC
Price : 200.00
Discount 10 percent : 20.00
Total price : 180.00
```

```
#include<stdio.h>
#include<conio.h>
int main()
{
    struct book {
        char name[50];
        float price;
        float discount;
    } book1;

    printf ("Enter book name : ");
    gets (book1.name);
    printf ("Enter book price : ");
    scanf ("%f",&book1.price);
    book1.discount = 0.1*book1.price;
    printf ("\n\n\nBook : %s\n",book1.name);
    printf ("Price : %.2f\n",book1.price);
    printf ("Discount 10 percent : %.2f\n"
           "Total price : %.2f",
           book1.discount,
           book1.price - book1.discount);

    return 0;
}
```

## 8.4 การกำหนดค่าเริ่มต้นให้โครงสร้าง

เราสามารถกำหนดค่าเริ่มต้นให้แก่โครงสร้างได้ โดยการกระทำคล้ายกับการกำหนดค่าเริ่มต้นให้แก่ตัวแปรทั่วไป แต่ตัวแปรโครงสร้างจะต้องกำหนดค่าเริ่มต้นให้ตามลำดับของตัวแปรที่ประกาศไว้

```
#include<stdio.h>
#include<conio.h>
int main()
{
    struct book {
        char name[50];
        float price;
    } b1 = {"harry",120};
    struct book b2 = {"Using C",150};
    struct book b3 = {150, "Pascal"}; //เกิด error
    return 0;
}
```

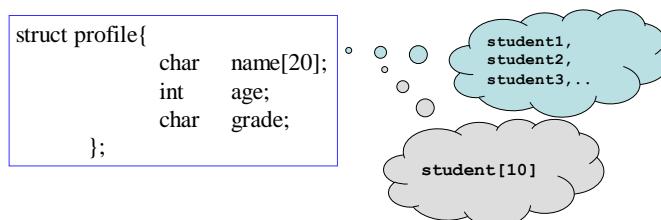
	name char[50]	price float
b1	harry	120
b2	Using C	150
b3	150	Pascal

## 8.5 ประกาศตัวแปรโครงสร้างแบบอาร์เรย์

กรณีที่ต้องใช้ตัวแปรชนิดโครงสร้างซ้ำๆ กันหลายๆ ตัว เราสามารถใช้คุณสมบัติของอาร์เรย์ กับโครงสร้างได้ เรียกว่า

Structure Array

ตัวอย่างเช่นต้องการตัวแปร student1, student2, ..., student10 จะสามารถกำหนดเป็นตัวแปรโครงสร้างเป็น อาร์เรย์ คือ student[10] เป็นต้น



//ตัวอย่างประกาศโครงสร้างแบบไม่ใช้อาร์เรย์  struct profile student1, student2, student3, student4, student5, student6, student7, student8, student9, student10;	//ตัวอย่างประกาศโครงสร้างแบบใช้อาร์เรย์  struct profile student[10];
--	--

รูปแบบการประกาศโครงสร้างแบบอาร์เรย์

//รูปแบบที่ 1  #include<stdio.h> int main() { struct profile{ char name[20]; int age; char grade; }; struct profile student[10]; return 0; }	//รูปแบบที่ 2  #include<stdio.h> int main() { struct profile{ char name[20]; int age; char grade; } student[10]; return 0; }
--	---

### โปรแกรม 8.4 เก็บข้อมูลนศ.จำนวน 10 คน

จะเขียนโปรแกรมเก็บข้อมูลนักศึกษาจำนวน 10 คน โดยมีรายละเอียดประกอบด้วย ข้อมูลประกอบด้วย ชื่อ กับ อายุ รับข้อมูลนักศึกษาจาก keyboard โปรแกรมจะต้องใช้ structure array เมื่อป้อนข้อมูลเสร็จ โปรแกรมจะค้นหาคน哪กเรียนที่ อายุมากกว่า 20 ปี และซื้อ ออกจอภาพ หน้าจอของโปรแกรมเมื่อทำงานแสดงดังนี้

Student[0] name:joy age:12	Student[6] name:tu age:25
Student[1] name:boy age:20	Student[7] name:tee age:34
Student[2] name:jo age:23	Student[8] name:bat age:44
Student[3] name:pat age:21	Student[9] name:phon age:33
Student[4] name:ple age:13	jo,23 pat,21 tu,25 tee,34 bat,44 phon,33
Student[5] name:tom age:11	

#include<stdio.h> #include<conio.h> #include<string.h> int main() { int i; struct profile{ char name[20]; int age; } s[10]; for(i=0;i<10;i++) { printf("Student[%d]\n", i); }
--

```

        printf("\t name:");
        scanf("%s", s[i].name);
        printf("\t age:");
        scanf("%d", &s[i].age);
    }
    for(i=0;i<10;i++)
        if(s[i].age > 20)
            printf("\n %s,%d", s[i].name, s[i].age);
    return 0;
}

```

## 8.6 โครงสร้างข้อมูลโครงสร้าง (Nest structure)

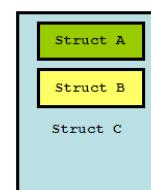
เราสามารถสร้างโครงสร้างข้อมูลโครงสร้างได้ เรียกว่า Nest structure เช่นประกาศโครงสร้าง C โดยภายในโครงสร้าง C ประกอบด้วยโครงสร้าง A และ B มีลักษณะดังรูป

```
//ตัวอย่างการประกาศโครงสร้างข้อมูลโครงสร้าง
struct A{
    ...
};

struct B{
    ...
};

struct C{
    struct A data_a;
    struct B data_b;
    ...
};
```

//รูปแบบของโครงสร้างข้อมูลโครงสร้าง



ตัวอย่างประกาศโครงสร้างข้อมูลโครงสร้าง เพื่อเก็บข้อมูลที่อยู่

```

struct address{
    int num;
    int moo;
    char road[20];
    char district[20];
    char province[20];
};

struct phone{
    char home[10];
    char mobile[10];
};

struct student{
    char name[20];
    char surname[20];
    char id[9];
    struct address
add;
    struct phone tel;
};
```

//ลักษณะของโครงสร้างที่ช้อนกัน

name	surname	id	add					tel	
			num	moo	road	district	province	home	mobile

## โปรแกรม 8.5 โปรแกรมเก็บข้อมูลสถานศึกษา

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    struct address{
        int add;
        int moo;
        char road[20];
        char district[20];
        char province[20];
    };
    struct university{
        char name[70];
        struct address place;
    };
    struct university king;
    strcpy (king.name,"King Mongkut's Institute of Technology Ladkrabang");
    king.place.add = 3;
    king.place.moo = 2;
    strcpy(king.place.road,"Chalongkrung");
    strcpy(king.place.district,"Ladkrabang");
    strcpy(king.place.province,"Bangkok");

    printf ("%s\n",king.name);
    printf ("Address : %d Moo %d, %s Rd.\n",
           king.place.add,king.place.moo,king.place.road);
    printf ("%s, %s",king.place.district,
           king.place.province);
    return 0;
}
```

```
//ผลลัพธ์
King Mongkut's Institute of Technology Ladkrabang
Address: 3 Moo 2, Chalongkrung Rd.
Ladkrabang, Bangkok
```

## 8.7 คำถ้ามท้ายบท

- 1) ประการโครงสร้างต่อไปนี้ แล้วพบว่าคำสั่งเรียกใช้โครงสร้างคำสั่งได้ไม่ถูกต้อง

struct product {     char name[15];     int count;     float price; };  struct product pr1, pr2;	คำสั่งข้อใดไม่ถูกต้อง <ol style="list-style-type: none"> <li>1. pr1.count = 10;</li> <li>2. pr2.count = 30;</li> <li>3. pr1.price = 130.5;</li> <li>4. pr2.price = 240.75;</li> <li>5. pr1.name = "Programming";</li> </ol>
--	---

2) ประกาศโครงสร้างต่อไปนี้ แล้วพิจารณาคำสั่งเรียกใช้โครงสร้างคำสั่งใดไม่ถูกต้อง

<pre>struct profile {     char name[30];     float salary; };  struct profile a[] ={"John",3000, "Tiger",2000,"Lisa", 5000};</pre>	จากส่วนของโปรแกรมด้านบน ข้อใดไม่ถูกต้อง <ol style="list-style-type: none"> <li>1. a[0].name คือ “John”</li> <li>2. a[0].salary + a[1].salary มีค่าเท่ากับ 5000</li> <li>3. a[1].salary มีค่าเท่ากับ 2000</li> <li>4. a เป็นตัวแปรແควาลำดับ ขนาด 6</li> <li>5. a ใช้พื้นที่ 102 ไบท์</li> </ol>
--	--

3) การประกาศตัวแปรข้อใดใช้พื้นที่รวมได้หน่วยความจำ 12 ไบท์

1. struct data{int item; double price[3]} A;
2. char i[3][3]; int a\_str;
3. struct data{char name[8]; float money;} A;
4. int i[3][4]; char a\_str;
5. ไม่มีข้อใดถูกต้อง

## การทดลองที่ 8 โครงสร้างข้อมูล

### วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจการทำงานของตัวแปรโครงสร้าง
2. เพื่อให้นักศึกษาสามารถเรียกใช้งานโครงสร้างได้ถูกต้อง
3. เพื่อให้นักศึกษาสามารถใช้งานตัวแปรแคลบชนิดโครงสร้างได้

### ทฤษฎี

ในการเขียนโปรแกรมสำหรับการใช้งานเฉพาะด้านต่างๆ โปรแกรมเมอร์พบปัญหาอย่างมากเนื่องจาก ตัวแปรมาตรฐานที่ภาษา C กำหนดไว้ให้ ไม่สามารถใช้งานได้โดยตรง การใช้ข้อมูลโดยที่ภาษา C กำหนดไว้เพียงตัวเดียวไม่สามารถบ่งบอกรายละเอียดของวัตถุที่มีรายละเอียดมากได้ เช่นในการเขียนโปรแกรมระบบเบียน ต้องมีการเก็บข้อมูลของนักเรียน 1 คน โดยข้อมูลที่ต้องใช้ในการประมวลผลสำหรับนักเรียน 1 คนได้แก่ ชื่อ นามสกุล รหัสนักศึกษา วันเกิดที่อยู่ โรคประจำตัว ฯลฯ ซึ่งการใช้ชนิดข้อมูลที่ข้อมูลโดยเดียว เพียงตัวเดียวชื่อ string ก็ไม่สามารถเก็บข้อมูลที่จำเป็นได้ทั้งหมด

ในภาษา C จึงมีกระบวนการที่ให้โปรแกรมเมอร์สามารถสร้าง ชนิดข้อมูลได้เอง โดยมีการนำเอาชนิดข้อมูลที่มีอยู่แล้วมาประกอบกันให้อยู่ในรูปแบบของโครงสร้างข้อมูลใหม่ (structure) เช่น ถ้าต้องมีการเก็บข้อมูลเกรดเฉลี่ยของนักศึกษา 1 คน โดยข้อมูลที่ต้องนำมาใช้คือ รหัสนักศึกษา ชื่อ – นามสกุล และเกรดเฉลี่ย ซึ่งเมื่อพิจารณาจากโจทย์แล้วภาษา C ไม่ได้มีชนิดข้อมูลสำหรับทำงานนี้เลย ถ้าต้องการเก็บข้อมูลทั้งหมดนี้ในชนิดข้อมูลเพียงชนิดเดียวโปรแกรมเมอร์จะต้องสร้างชนิดข้อมูลขึ้นมาเอง โดยต้องนำชนิดข้อมูล string จำนวน 3 ตัว สำหรับเก็บข้อมูลรหัสนักศึกษา ชื่อนักศึกษา และนามสกุล ตามลำดับ ประกอบกับชนิดข้อมูล float สำหรับเก็บข้อมูลเกรดเฉลี่ย กล่าวเป็นชนิดข้อมูลชนิดใหม่ ที่หมายความกับการเขียนโปรแกรมในลักษณะนี้

การใช้งาน structure ในภาษา C นั้น มีการทำงานที่นักศึกษาต้องสามารถใช้งานให้เป็น 2 ส่วนคือ

1. สร้างโครงสร้างข้อมูลที่ต้องการ
2. สามารถใช้งานโครงสร้างที่สร้างขึ้นได้

### การสร้างโครงสร้างข้อมูลที่ต้องการ

ในการสร้างโครงสร้างข้อมูลสามารถสร้างได้โดยใช้คำสั่ง struct ตรงตำแหน่ง local variable หรือ ตำแหน่ง global variable สำหรับคำสั่ง struct มีรูปแบบคำสั่งดังนี้

//รูปแบบที่ 1	//รูปแบบที่ 2
<pre>struct name {     type var-1;     type var-2;     ...     type var-n; };</pre>	<pre>struct name {     type var-1;     type var-2;     ...     type var-n; }variable;</pre>

struct name เป็นชื่อโครงสร้างที่ต้องการสร้างขึ้น

– type var-1, type var-2, ... ,type var-n เป็นส่วนของโครงสร้างที่ต้องการสร้างขึ้น

variable เป็นชื่อของตัวแปรชนิดโครงสร้างที่มีลักษณะโครงสร้างเป็น name ที่ประกอบด้วย var-1, var-2, ...

ตัวอย่างการสร้าง struct ที่มีรายละเอียดการเก็บข้อมูลนักศึกษา

```
struct student{
    char name[20];
    char id[9];
    int age ;
    float gpa;
    char gender;
};
```

การใช้งานโครงสร้างที่สร้างขึ้น

เนื่องจาก struct เป็นวิธีการที่ให้โปรแกรมเมอร์สามารถกำหนดชนิดตัวแปรได้ ดังนั้นการใช้งานคำสั่ง struct จึงมี การใช้งานเหมือนกับการกำหนดชนิดตัวแปรให้กับตัวแปรตัวใดตัวหนึ่ง นอกเหนือไปนี้โปรแกรมเมอร์ต้องสามารถใช้งานตัวแปรที่มี ชนิดข้อมูลเป็น struct ที่สร้างขึ้นมาแล้วได้ โดยการใช้งานตัวแปรที่ถูกสร้างขึ้นนั้น โปรแกรมเมอร์ต้องสามารถอ้างอิงข้อมูล ในโครงสร้างส่วนต่างๆ ได้

ในการอ้างอิงส่วนต่างๆ ของตัวแปรชนิดโครงสร้าง สามารถอ้างอิงได้โดยใช้ชื่อตัวแปร (variable) ตามด้วย เครื่องหมายจุด (.) และตามด้วย ส่วนของโครงสร้างต่างๆ (var-1, var-2, ..., var-n) เช่น variable.var-1

การกำหนดค่าให้ตัวแปรโครงสร้าง

การกำหนดค่าให้กับตัวแปรโครงสร้างสามารถกำหนดได้ 2 แบบ

- 1) การกำหนดค่าเมื่อเริ่มประกาศตัวแปรชนิดโครงสร้าง เป็นการกำหนดค่าเมื่อประกาศตัวแปร กำหนดค่าด้วยการใส่ ค่าที่ต้องการกำหนดลงในเครื่องหมายปีกกา {} โดยเรียงตามลำดับส่วนของโครงสร้างข้อมูล
- 2) การกำหนดค่าเมื่อหลังประกาศตัวแปรชนิดโครงสร้าง เป็นการกำหนดค่าให้กับส่วนต่างๆ ของตัวแปรโครงสร้าง เมื่อประกาศกำหนดค่าตัวแปรปกติ

ตัวอย่าง การกำหนดค่าเมื่อเริ่มประกาศตัวแปรชนิดโครงสร้าง

```
struct student{
    char name[20];
    char id[9];
    int age;
    float gpa;
    char gender;
};
struct student st = {"kmitl","51010000",17,3.33,'m'};
```

ตัวอย่าง การกำหนดค่าเมื่อหลังประกาศตัวแปรชนิดโครงสร้าง

<pre>struct student{     char name[20];     char id[9];     int age;     float gpa;     char gender; };</pre>	<pre>struct student st; strcpy (st.name,"kmitl") strcpy (st.id,"51010000") age = 17; gpa = 3.33; gender = 'm';</pre>
---	--

Array of Structure

ในการทำงานกับ Structure นั้นโดยทั่วไปเรามักจะไม่มีการใช้งาน Structure เพียงอย่างเดียว มักจะมีการใช้งานควบคู่กับ Array โดยสร้างตัวแปรที่มีลักษณะเป็น Array of Structure เช่น กรณีที่เราต้องการเก็บข้อมูลเกรดเฉลี่ยของนักศึกษาแต่ละ คนในกลุ่มนักศึกษาจำนวน 10 คน ก่อนอื่นจะต้องมีการสร้างโครงสร้างข้อมูลสำหรับเก็บข้อมูลของนักศึกษา 1 คนก่อน เช่น

```
struct student{
    char ID[10];
    char Name[50];
    float gpa;
};
```

หลังจากนั้นจึงสร้างตัวแปร array ขึ้นมาโดยกำหนด type เป็น struct student ดังนี้

```
struct student std[10];
```

เราจะได้ array ที่ข้อมูลแต่ละช่องคือ struct student ในการอ้างอิงข้อมูลใน struct ของ array สามารถทำได้โดยการกำหนดค่าให้กับโครงสร้างอย่างแต่ละตัวดังตัวอย่าง การกำหนดค่า gpa ให้กับนักเรียนคนแรก สามารถทำได้โดยการใช้คำสั่ง std[0].gpa = 3.40; เป็นต้น

### การทดลองที่ 8.1

- ให้นักศึกษาออกแบบโครงสร้างข้อมูลสำหรับเก็บข้อมูลนักศึกษาสำหรับงานทะเบียนนักศึกษา โดยสามารถเก็บรหัสนักศึกษา ชื่อ นามสกุล เพศ วันเดือนปีเกิดได้ โดยสามารถเก็บรหัสนักศึกษา ชื่อวิชา หน่วยกิต และเกรด
- ให้นักศึกษาออกแบบโครงสร้างข้อมูลสำหรับการคำนวนเกรดเฉลี่ยของนักศึกษาโดยสามารถเก็บรหัสวิชา หน่วยกิต และเกรด
- ให้นักศึกษาออกแบบโครงสร้างข้อมูลสำหรับการเก็บข้อมูลรายนิต์ของกรรมการขันส่ง โดยสามารถเก็บรหัสผู้ตัดสิน ชื่อหนังสือ ชื่อผู้แต่ง สำนักพิมพ์ ISBN
- ให้นักศึกษาออกแบบโครงสร้างข้อมูลสำหรับเก็บข้อมูลหนังสือในห้องสมุด โดยสามารถเก็บ หมวดหนังสือ เลขเรียกหนังสือ ชื่อหนังสือ ชื่อผู้แต่ง สำนักพิมพ์ ISBN

### การทดลองที่ 8.2

- ให้นักศึกษาเขียนโปรแกรมรับข้อมูลนักศึกษาโดยใช้โครงสร้างที่ได้ออกแบบไว้
- ให้นักศึกษาเขียนโปรแกรมเพื่อรับค่าชื่อวิชา เกรด และหน่วยกิตของวิชาต่างๆ จำนวนมากที่สุด 10 วิชา โดยใช้โครงสร้างข้อมูลสำหรับคำนวนเกรดเฉลี่ยของนักศึกษาที่ได้ออกแบบไว้ใน การทดลองที่ 8.1 หลังจากนั้นให้คำนวนเกรดเฉลี่ยที่ได้แสดงผลในหน้าจอ (Hint : ใช้ Array of structure เพื่อเก็บข้อมูลที่ละ 10 วิชา)
- ให้นักศึกษาเขียนโปรแกรมเพื่อรับค่าข้อมูลนักศึกษา 5 คนและผลการเรียน 5 วิชาของนักศึกษาแต่ละคนพร้อมทั้งให้แสดงผลลัพธ์คือคะแนนสูงสุด, ต่ำสุดของแต่ละวิชา ชื่อนักศึกษาที่ได้คะแนนสูงสุดของแต่ละวิชา และชื่อนักศึกษาที่ได้คะแนนรวมสูงสุด (สร้างโครงสร้างข้อมูลใหม่โดยประกอบจากโครงสร้างข้อมูลในข้อ 8.1)

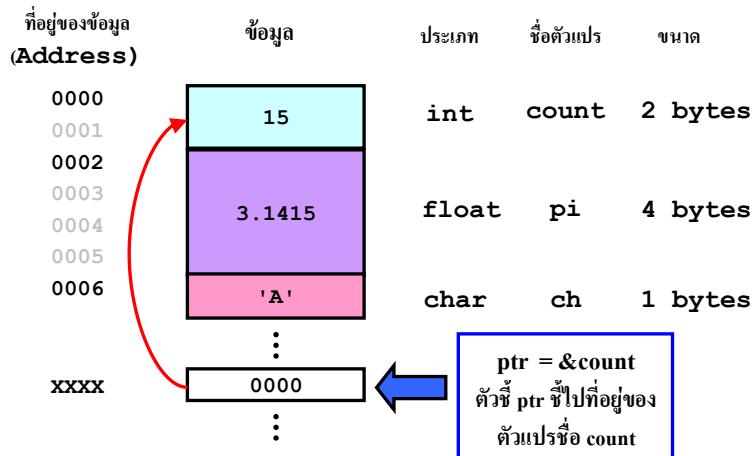
4. ให้นักศึกษาเขียนโปรแกรมสำหรับเก็บข้อมูลหนังสือในห้องสมุด โดยสามารถเก็บชื่อหนังสือ ชื่อผู้แต่ง และสำนักพิมพ์ โดยแต่ละพิลเด้มีขนาดไม่เกิน 50 byte โดยโปรแกรมมีความสามารถดังนี้
  - a. แสดงเมนูทั้งหมด 4 เมนู คือ เพิ่มข้อมูลหนังสือ ลบข้อมูลหนังสือ แสดงข้อมูลหนังสือ และออกจากโปรแกรม
  - b. การเพิ่มข้อมูลหนังสือสามารถเพิ่มข้อมูลได้มากที่สุด 20 เล่ม โดยให้ผู้ใช้งานกรอกรายละเอียดของหนังสือทั้งหมด
  - c. การแสดงรายการหนังสือสามารถแสดงผลรายละเอียดหนังสือทั้งหมด ที่เก็บข้อมูลไว้ได้
  - d. การลบข้อมูลหนังสือ สามารถลบจากเลขลำดับหนังสือที่ต้องการลบข้อมูล
5. ให้นักศึกษาออกแบบโครงสร้างข้อมูลเพื่อเก็บข้อมูลประวัติการยืมและคืนหนังสือของนักศึกษา โดยออกแบบโครงสร้างข้อมูลที่เชื่อมต่อโครงสร้างข้อมูลที่สร้างไว้แล้วในข้อ 8.1 พร้อมทั้งเขียนโปรแกรมทดสอบการทำงาน

## บทที่ 9 ตัวแปรชี้ตัวแหน่ง (Pointer)

### วัตถุประสงค์การศึกษา

- 1) เพื่อศึกษาตัวแปรประเภทตัวชี้
- 2) เพื่อนำตัวแปรประเภทตัวชี้ไปใช้งานได้ถูกต้อง

### 9.1 โครงสร้างของหน่วยความจำและตัวชี้



#### 9.1.1 การประกาศตัวแปร pointer

```
type *pointer_name;
```

type      คือ      ชนิดของตัวแปรประเภทตัวชี้ (pointer)  
 \*          คือ      เครื่องหมายแสดงว่าเป็นตัวแปรประเภทตัวชี้  
 pointer\_name    คือ      ชื่อของตัวแปรประเภทตัวชี้

#### ตัวอย่างการใช้ตัวชี้

```

int      * ptr_int;      /* pointer to integer */
float     * ptr_float;     /* pointer to float */
char     * ptr_char;     /* pointer to char */
  
```

### 9.1.2 ตัวดำเนินการ (Reference Operator) “&”

ตัวดำเนินการ & ส่งกลับเลขที่อยู่ (Address) ของวัตถุ

<pre>int count; int *ptr; ptr = &amp;count //ให้ ptr จี้ไปที่ count</pre>	<p>//รูปแสดงลักษณะการใช้ของ pointer ในหน่วยความจำ</p> <pre>xxxx      int *ptr;</pre>
---	--

ตัวอย่างการประกาศตัวแปร pointer พร้อมกำหนดให้จี้ไปยังตัวแปรที่ต้องการ

<pre>//ประกาศตัวแปร int count = 15;           //ขนาดของ count 2byte float pi = 3.1415;        //ขนาดของ pi 4byte char ch = 'A';            //ขนาดของ ch 1byte  //ประกาศ pointer และให้จี้ไปยังตัวแปร char *ptr_char = &amp;ch; //ขนาดของ ptr_char 4byte float *ptr_float = &amp;pi; //ขนาดของ ptr_float 4byte int *ptr_int = &amp;count; //ขนาดของ ptr_int 4byte</pre>	<p>//รูปแสดงลักษณะการใช้ของ pointer ในหน่วยความจำ</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0000</td><td>15</td></tr> <tr><td>0002</td><td>3.1415</td></tr> <tr><td>0006</td><td>'A'</td></tr> </table>	0000	15	0002	3.1415	0006	'A'
0000	15						
0002	3.1415						
0006	'A'						

### 9.1.3 ตัวดำเนินการเชิงอ้อม (Indirect Operator) “\*”

เป็นตัวดำเนินการเชิงอ้อม (Indirection) หรือ กลับการอ้างอิง (Dereferencing)

<pre>int count = 15, y, z[10]; int *ptr;           // ptr เป็นตัวจี้ int ptr = &amp;count;       // ptr จี้ไปที่ count y = *ptr;          // y มีค่า 15 *ptr = 0;           // count มีค่า 0 ptr = &amp;z[0];         // ptr จี้ไปที่ z[0]</pre>	<p>ตัวแหน่งที่ xxxx</p> <pre>xxxx      int count; xxxx      int *ptr;</pre>
--	---

## 9.2 การแสดงตัวชี้ด้วยฟังก์ชัน printf

ฟังก์ชัน printf สามารถแสดงตำแหน่งที่อยู่ (address) ได้โดยใช้ เครื่องหมาย

%p เพื่อแสดงตำแหน่งเป็นเลขฐานสิบหก

%n เพื่อแสดงตำแหน่งเป็นเลขฐานสิบ ผลลัพธ์ที่ได้จะอยู่ในรูปแบบ XXXX:YYYY หรือ XXXX ขึ้นอยู่กับ memory model ที่ใช้

```

1 #include <stdio.h>
2 int main()
3 {
4     int i = 10;
5     int *p;
6     p = &i;
7     printf("%p %d\n", &i, i);
8     printf("%p %p %d\n", &p, p, *p);
9     printf("-----\n");
10    printf("%u %d\n", &i, i);
11    printf("%u %u %d\n", &p, p, *p);
12    return 0;
13 }

```

C:\WINDOWS\system32\cmd.exe  
0012FF60 10  
0012FF54 0012FF60 10  
1245024 10  
1245012 1245024 10  
Press any key to continue .

### โปรแกรม 9.1 โปรแกรมแสดงข้อมูลผ่านทาง pointer

<pre>#include&lt;stdio.h&gt; int main() {     char letter = 'D';     int num = 19;     float point = 26.09;     char *pt_letter;     int *pt_num;     float *pt_point;     pt_letter = &amp;letter;     pt_num = &amp;num;     pt_point = &amp;point;     printf("Address of letter = %p \n",pt_letter);     printf("Address of num = %p \n",pt_num);     printf("Address of point = %p \n",pt_point);     return 0; }</pre>	<p>//รูปแสดงลักษณะการซื้อของ pointer ในหน่วยความจำ</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>letter</th> <th>num</th> <th>point</th> </tr> </thead> <tbody> <tr> <td>D</td> <td>19</td> <td>26.09</td> </tr> <tr> <td>0000</td> <td>0002</td> <td>0004</td> </tr> </tbody> </table> <p>pt_letter pt_num pt_point</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>0000</th> <th>0002</th> <th>0004</th> </tr> </thead> <tbody> <tr> <td>????</td> <td>????</td> <td>????</td> </tr> </tbody> </table> <p>//ผลลัพธ์</p> <p>Address of letter = 0000  Address of num = 0002  Address of point = 0004</p>	letter	num	point	D	19	26.09	0000	0002	0004	0000	0002	0004	????	????	????
letter	num	point														
D	19	26.09														
0000	0002	0004														
0000	0002	0004														
????	????	????														

### โปรแกรม 9.2 โปรแกรมการใช้ referencing และ dereferencing

<pre> int main() {     int     num1 = 113, num2;     float   price1 = 4.85;     char    hint1 = 'J', hint2;     int     *pt_num;     float   *pt_price;     char    *pt_hint;     pt_num  = &amp;num1;     pt_price = &amp;price1;     pt_hint  = &amp;hint1;     num2   = *pt_num;     hint2  = *pt_hint;     printf ("Variable num1 = %d \n", num2);     printf ("Variable price1 = %f \n", *pt_price);     printf ("Variable hint2 = %c \n", hint2);     return 0; } </pre>	<p>//รูปแสดงลักษณะการซื้อ pointer ในหน่วยความจำ</p> <p>num1 price1 hint1</p> <table border="1"> <tr> <td>113</td> <td>4.85</td> <td>J</td> </tr> </table> <p>--x-- --y-- --z--</p> <p>pt_num pt_price pt_hint</p> <p>--x-- --y-- --z--</p> <p>num2 hint2</p> <table border="1"> <tr> <td>113</td> <td>J</td> </tr> </table> <p>//ผลลัพธ์</p> <p>Variable num1 = 113 Variable price1 = 4.850000 Variable hint2 = J</p>	113	4.85	J	113	J
113	4.85	J				
113	J					

### โปรแกรม 9.3 โปรแกรมการใช้ referencing และ dereferencing แบบ 2

<pre> #include&lt;stdio.h&gt;  int main() {     int     a;     int     *aPtr;     a = 7;     aPtr = &amp;a;      //ให้ aPtr ซื้อ a     printf ("The address of a is %p\n"             "The value of aPtr is %p\n\n", &amp;a, aPtr);     printf ("The value of a is %d\n"             "The value of *aPtr is %d\n\n", a, *aPtr);     printf ("Proving that * and &amp; are complements of each other.\n"             "&amp;*aPtr = %p\n*&amp;aPtr = %p\n",             &amp;*aPtr, *&amp;aPtr);     return 0; } </pre>	<p>FFF4 7 a ??? FFF4 aPtr</p> <p>//ผลลัพธ์</p> <p>The address of a is FFF4 The value of aPtr is FFF4  The value of a is 7 The value of *aPtr is 7  Proving that * and &amp; are complements of each other. &amp;*aPtr = FFF4 *&amp;aPtr = FFF4</p>
---	--

### 9.3 Pointer ชื่อ pointer

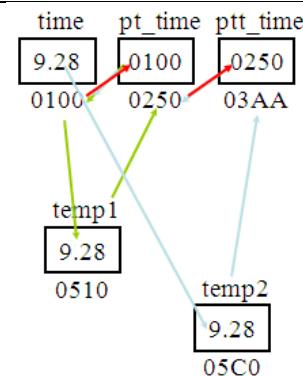
รูปแบบคำสั่ง

```
type    **ptt_name;
```

Type	คือ	ชนิดของตัวแปร พอยน์เตอร์
**	คือ	เครื่องหมายที่แสดงว่าเป็นตัวแปร พอยน์เตอร์ชื่อพอยน์เตอร์
ptt_name	คือ	ชื่อของตัวแปร พอยน์เตอร์ชื่อพอยน์เตอร์

ตัวอย่าง Pointer ชื่อ pointer

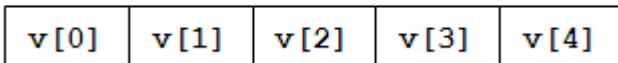
```
float    time = 9.28;
float    *pt_time;
float    **ptt_time; //ประกาศ pointer ชื่อ pointer
pt_time = &time;
ptt_time = &pt_time;
float    temp1;
temp1 = *pt_time;
float    temp2;
temp2 = **ptt_time; //ดึงค่าที่ pointer ชี้
```



### 9.4 ตัวชี้และแคลว์ลามดับ

ตัวชี้และแคลว์ลามดับในภาษาซีนั้น มีความใกล้ชิดกันอย่างมาก การประกาศ `float v[5]` เป็นการกำหนดแคลว์ลามดับ v ขนาด 5 นั่นคือกลุ่มของวัตถุติดกัน 5 ชิ้นมีชื่อว่า v[0], v[1], v[2], v[3], v[4]

3000 3004 3008 300C 3010



การประกาศ `float *vPtr` และกำหนดให้ vPtr ชี้ไปยังตัวแปรแคลว์ลามดับ v สามารถทำได้สองวิธีคือ

//วิธีที่ 1	vPtr = v;	
//วิธีที่ 2	vPtr = &v[0];	

### 9.4.1 การย้าย Pointer ที่ชี้ตัวแปรอาร์เรย์ ด้วย += และ -=

ตัวชี้สามารถนำมารวนทางเลขคณิตได้ ดังนี้ เพิ่ม (++, increment) หรือลด (--, decrement) จำนวนเต็มสามารถบวกกับตัวชี้ได้ (+ หรือ +=) หรือลบกับตัวชี้ได้ (- หรือ -=) ตัวชี้ตัวหนึ่งสามารถลบกับตัวอีกตัวหนึ่งได้

เมื่อบวกหรือลบจำนวนเต็มกับตัวชี้แล้ว ค่าของตัวชี้มีได้เพิ่มหรือลดลงตามตัวเลขจำนวนนั้น ค่าของตัวชี้เพิ่มหรือลดตามตัวเลขจำนวนนั้นคูณกับขนาดของวัตถุที่ตัวชี้นั้นชี้อยู่ ขนาด (ไบท์) ขึ้นกับประเภทของข้อมูลที่ใช้ในวัตถุนั้น

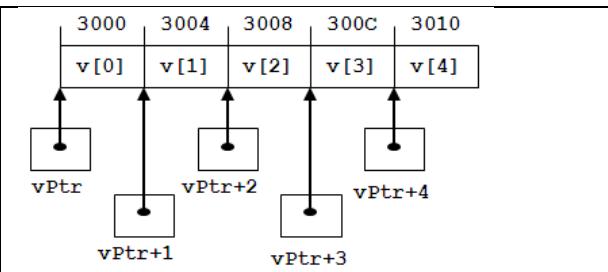
ตัวอย่าง (กำหนดขนาดของวัตถุ ชนิด float คือ 4 ไบท์)

<pre>//เริ่มย้าย Pointer vPtr += 2;           //   vPtr = vPtr+(2*4)                      // or vPtr = 3000+(2*4)</pre>	<p>//ก่อนย้าย Pointer</p> <p>//หลังย้าย Pointer</p>
---	---

<pre>//เริ่มย้าย Pointer vPtr -= 2;           //   vPtr = vPtr-(2*4)                      // or vPtr = 3008-(2*4)</pre>	<p>//ก่อนย้าย Pointer</p> <p>//หลังย้าย Pointer</p>
---	---

### 9.4.2 การย้าย Pointer ที่ชี้ตัวแปรอาร์เรย์ ด้วย + และ -

จากรูปด้านขวาจะเห็นความสัมพันธ์ระหว่างตัวชี้กับอาร์เรย์  
ในการอ้างอิงตำแหน่งเมื่อกำหนดให้  
  
vPtr = v;  
การอ้างอิงตำแหน่งข้อมูลของตัวแปรตัวชี้สามารถใช้เลข  
ตำแหน่งอ้างอิงได้ดังรูป



### 9.4.3 การคำนวณจำนวน element ด้วย pointer

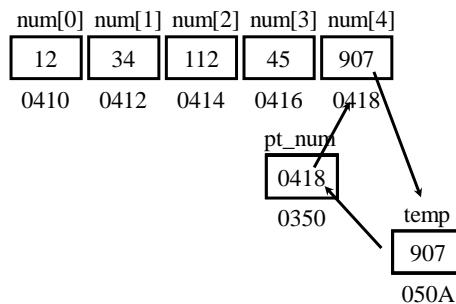
<pre>//pointer ตัวแรก vPtr = &amp;v[0];      //   vPtr = 3000 //pointer ตัวที่สอง v2Ptr = &amp;v[2];     // or v2Ptr = 3008 //คำนวณจำนวน element ของ array ที่ pointer ทั้งสองชี้ x = v2Ptr - vPtr; // x = จำนวน element</pre>	
--	--

ค่าที่ x ได้รับคือจำนวนหน่วย (element) ของตัวแปรແ霎ลำดับนับจาก vPtr ถึง vPtr2 ในกรณีนี้คือ 2

### โปรแกรม 9.4 โปรแกรมแสดงการใช้ pointer กับ array

```

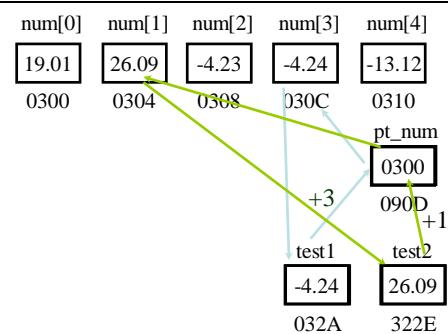
int      num[5] = {12, 34, 112, 45, 907};
int      *pt_num;
pt_num = &num[1];
pt_num = &num[4];
int      temp;
temp = *pt_num;
    
```



### โปรแกรม 9.5 โปรแกรมแสดงการใช้ pointer กับ array

```

float    *pt_num;
pt_num = num;
float    test1, test2;
test1 = *(pt_num+3);
test2 = *(pt_num+1);
    
```



## 9.4 ความสัมพันธ์ระหว่าง Pointer กับ Array

เห็นได้ว่าตัวชี้และตัวแปรແ胄ດลำดับมีความสัมพันธ์กัน และสามารถใช้แทนกันได้เกือบทุกรณี

### 9.4.1 กรณีที่ 1 ใช้ array ปกติ

```

int main()
{
    int i, offset, b[] = {10, 20, 30, 40};
    int *bPtr = b; //ให้ bPtr ชี้ b

    printf("Array b printed with:\n"
           "Array subscript notation\n");
    for (i=0; i<=3; i++)
        printf("b[%d] = %d\n", i, b[i]); //สังเกตที่นี่
    return 0;
}
    
```

//ผลลัพธ์  
Array b printed with:  
Array subscript notation  
b[0] = 10  
b[1] = 20  
b[2] = 30  
b[3] = 40

#### 9.4.2 กรณีที่ 2 ใช้ array เขียนแบบ pointer

<pre>int main() {     int i, offset, b[] = {10, 20, 30, 40};     int *bPtr = b; //ให้ bPtr ชี้ b      printf("Pointer/offset notation where\n"            "the pointer is the array name\n");     for (offset = 0; offset&lt;=3 ; offset++)         printf("%*(b + %d) = %d\n", offset, *(b + offset)); //สังเกตที่นี่     return 0; }</pre>	<p>//ผลลัพธ์</p> <p>Pointer/offset notation where The pointer is the array name</p> <p><math>*(b + 0) = 10</math></p> <p><math>*(b + 1) = 20</math></p> <p><math>*(b + 2) = 30</math></p> <p><math>*(b + 3) = 40</math></p>
--	---

#### 9.4.3 กรณีที่ 3 ใช้ pointer เขียนแบบ array

<pre>int main() {     int i, offset, b[] = {10, 20, 30, 40};     int *bPtr = b; //ให้ bPtr ชี้ b      printf("Pointer subscript notation\n");     for (i=0 ; i&lt;=3 ; i++)         printf("bPtr[%d] = %d\n", i, bPtr[i]); //สังเกตที่นี่     return 0; }</pre>	<p>//ผลลัพธ์</p> <p>Pointer subscript notation</p> <p><math>bPtr[0] = 10</math></p> <p><math>bPtr[1] = 20</math></p> <p><math>bPtr[2] = 30</math></p> <p><math>bPtr[3] = 40</math></p>
---	--

#### 9.4.4 กรณีที่ 4 ใช้ pointer ปกติ

<pre>int main() {     int i, offset, b[] = {10, 20, 30, 40};     int *bPtr = b; //ให้ bPtr ชี้      printf("Pointer/offset notation\n");     for (offset=0; offset&lt;=3; offset++) </pre>	<p>//ผลลัพธ์</p> <p>Pointer/offset notation</p> <p><math>*(bPtr + 0) = 10</math></p> <p><math>*(bPtr + 1) = 20</math></p> <p><math>*(bPtr + 2) = 30</math></p> <p><math>*(bPtr + 3) = 40</math></p>
--	---

```
printf(*(bPtr + %d) = %d\n", offset, *(bPtr + offset)); //สังเกต
return 0;
}
```

## 9.5 ตัวชี้กับตัวแปรโครงสร้าง

เราสามารถใช้งานสมาชิกของตัวแปรแบบโครงสร้างได้อยู่สองวิธี

- ใช้เครื่องหมายจุด (struc\_var.struc\_member) (structure member operator, or dot operator)
- ใช้เครื่องหมาย -> เมื่อเป็นตัวชี้ไปยังตัวแปรแบบโครงสร้าง เราจะใช้ structure pointer operator (struc\_Ptr->struc\_member)

หากต้องการพิมพ์ชื่อนักศึกษา สามารถทำได้สองวิธีคือ

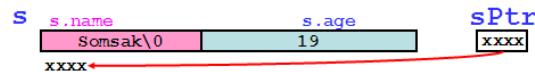
```
printf("%s", s.name);
```

```
printf("%s", sPtr->name);
```

### ตัวอย่าง

```
struct student{
    char name[40];
    int age ;
};

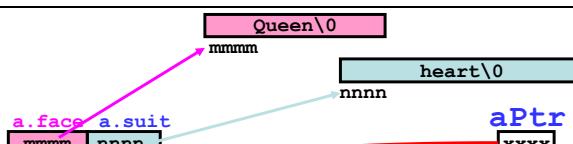
struct student s;
struct student *sPtr;
strcpy(s.name, "Somsak");
s.age = 19;
sPtr = &s;
```



### ตัวอย่าง

```
struct card {
    char *face;
    char *suit;
};

struct card a;
struct card *aPtr;
```

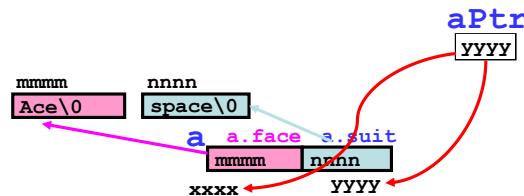


## ตัวอย่าง

```

int main(){
    struct card {
        char *face; // 2,3,..,9,J,Q,K,A
        char *suit; // space, heart, diamond, club
    };
    struct card a; struct card *aPtr;
    a.face = "Ace"; a.suit = "spade"; aPtr = &a;
    printf( "%s%s%s\n%s%s%s\n%s%s%s\n",
            a.face , " of " ,a.suit,
            aPtr->face , " of " , aPtr->suit,
            (*aPtr).face, " of " , (*aPtr).suit );
    return 0;
}

```



//ผลลัพธ์  
Ace of spade  
Ace of spade  
Ace of spade

โปรแกรม 9.6 โปรแกรมเก็บข้อมูลนศ. 10 คน

จะเขียนโปรแกรมเก็บข้อมูลนักศึกษาจำนวน 10 คน โดยมีรายละเอียดดังนี้

- ข้อมูลประกอบด้วย ชื่อกับ อายุ
- รับข้อมูลนักศึกษาจากคีย์บอร์ด
- โปรแกรมจะต้องใช้ตัวชี้ที่ป้อนข้อมูลประเภทโครงสร้าง
- เมื่อป้อนข้อมูลเสร็จ โปรแกรมจะค้นหาคนเรียนที่อายุมากกว่า 20 ปี และแสดงชื่อ ของจบภาค

```

#include<stdio.h>
#include<conio.h>
int main()
{
    struct profile{
        char name[20];
        int age;
    } s[10];

    int i;
    struct profile *sPtr;

    sPtr = s; //sPtr ชี้ที่ s[0]
    //รับค่า
    for (i=0; i<10; i++)

```

## ผลลัพธ์

Student [0]	name:iov	age:12
Student [1]	name:bov	age:20
Student [2]	name:jo	age:23
Student [3]	name:pat	age:21
Student [4]	name:pile	age:13
Student [5]	name:tom	age:11
Student [6]	name:tu	age:25
Student [7]	name:tee	age:34
Student [8]	name:bat	age:44
Student [9]	name:pho	age:33

jo,23  
pat,21  
tu,25  
tee,34  
bat,44  
phon,33

```

{
    printf("Student # %d\n\tName :", i+1 );
    scanf("%s",sPtr->name);
    printf("\tAge:");
    scanf("%d",&(sPtr->age));
    sPtr++;
}

sPtr -= 10; //ย้าย sPtr กลับ
//แสดงผล
for (i=0; i<10; i++)
{
    if ((sPtr->age) > 20)
        printf("\n%s, %d",sPtr->name,sPtr->age);
    sPtr++;
}
return 0;
}

```

## 9.6 คำตามท้ายบท

- 1) จงสร้างอาร์เรย์ A เป็นอาร์เรย์ขนาด  $3 \times 3$  เก็บข้อมูลตัวเลข 1-9 และแสดงผลลัพธ์ตัวเลขในอาร์เรย์โดยใช้การอ้างอิงค่าตัวเลขโดยใช้ Pointer เท่านั้น
- 2) กำหนดให้ x และ y เป็นตัวแปรแบบทศนิยม จงเขียนโปรแกรม ทำการสลับค่า x และ y โดยใช้ pointer เท่านั้น
- 3) ให้ S เป็น Structure ที่เก็บข้อมูลตัวเลข 3 ตัว จงเขียนโปรแกรมให้เก็บข้อมูลตัวเลขตัวที่ 1 และ 2 ลงใน Structure และหยอดรวมลงในตัวเลขตัวที่ 3 โดยใช้การอ้างอิงข้อมูลโดย Pointer เท่านั้น

## การทดลองที่ 9 ตัวชี้ (Pointer)

### วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจตัวแปรประเภทตัวชี้และการคำนวณต่าง ๆ ของตัวชี้
2. เพื่อให้นักศึกษาเข้าใจความสัมพันธ์ระหว่างตัวชี้กับตัวแปรและลำดับ

### ทฤษฎี

ในการทำงานของคอมพิวเตอร์ นอกจากระบบทองทรานซิสเตอร์แล้ว ยังมีมาตรวัดข้อมูล และปริมาณข้อมูลแล้ว สิ่งสำคัญอีกอันหนึ่งที่คอมพิวเตอร์จำเป็นต้องทราบคือ ตำแหน่งที่เก็บข้อมูล ในการกำหนดว่าข้อมูลตัวแปรต่างๆ จะเก็บไว้ ณ ตำแหน่งไหนในหน่วยความจำ จะเป็นหน้าที่ของระบบปฏิบัติการ สำหรับโปรแกรมเมอร์แล้ว เราจะทราบค่าตำแหน่งของตัวแปรต่างๆ ที่ระบบปฏิบัติการกำหนดตำแหน่งไว้ให้แล้วได้ โดยการใช้เครื่องหมาย & นำหน้าตัวแปรนั้นๆ ซึ่งนักศึกษาเคยเห็นรูปแบบการใช้งานลักษณะนี้แล้วตั้งแต่คำสั่ง scanf และจากคำสั่ง scanf("%d",&a); ที่ได้เรียนมาแล้ว โดยการทำงานของคำสั่งนี้ คือการสั่งให้คอมพิวเตอร์รับข้อมูลจากคีย์บอร์ดแล้วนำมาตีความ เป็นตัวเลขจำนวนเต็ม เก็บไว้ ณ ตำแหน่งของตัวแปร a ซึ่งเราใช้สัญลักษณ์ &a แทนตำแหน่งของตัวแปร a

เนื่องจาก ตำแหน่งของตัวแปรหรือข้อมูลต่างๆ ถือว่าเป็นข้อมูลอีกชนิดหนึ่งที่โปรแกรมเมอร์ จำเป็นต้องใช้งาน เช่นกัน ภาษา C จึงมีการกำหนดค่าชนิดข้อมูลชนิดหนึ่ง ที่ใช้ในการเก็บ ตำแหน่งของตัวแปรหรือข้อมูลไว้ นั่นคือ ตัวชี้ (pointer)

ตัวชี้เป็นตัวแปรที่บันทึกตำแหน่งของหน่วยความจำ แทนที่จะบันทึกค่าหนึ่ง ๆ เช่นตัวแปรประเภทอื่น ๆ ตัวแปรประเภทตัวชี้บันทึกตำแหน่งของหน่วยความจำ(Memory Address) จากล่าဂได้ว่าตัวแปรประเภทตัวชี้อ้างอิงถึงข้อมูลอย่างอ้อม (indirect reference) การประกาศตัวแปรประเภทตัวชี้ ประกาศโดย

Type \* pointer\_name;

type คือชนิดของตัวแปรที่ตัวชี้จะอ้างอิงไปถึง โดยที่

int ประกาศตัวแปรที่ตัวชี้จะอ้างอิงไปถึงเป็นประเภทจำนวนเต็ม

float ประกาศตัวแปรที่ตัวชี้จะอ้างอิงไปถึงเป็นประเภทศนยิม

char ประกาศตัวแปรที่ตัวชี้จะอ้างอิงไปถึงเป็นประเภทอักขระ

\* เครื่องหมายแสดงว่าเป็นตัวแปรประเภทตัวชี้

Pointer\_name ชื่อของตัวแปรประเภทตัวชี้

เมื่อประกาศตัวแปรประเภทตัวชี้แล้ว ตัวชี้นั้นจะอ้างอิงไปถึงตัวแปรได้โดยกำหนดตำแหน่งของตัวแปรให้กับตัวชี้ ลักษณะ & (Address Operator) ในภาษาซี จะให้ผลลัพธ์คือตำแหน่งที่อยู่ของตัวแปรนั้น ตัวอย่างเช่น

```
int y = 5 /* ประกาศตัวแปร y เป็นประเภทจำนวนเต็ม และมีค่าเริ่มต้น 5 */
```

```
int *yPtr /* ประกาศตัวแปร yPtr เป็นตัวชี้ที่จะอ้างอิงไปยังตัวแปรประเภทจำนวนเต็ม */
```

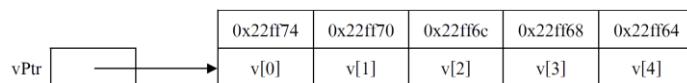
```
yPtr = &y /* ประกาศให้ yPtr อ้างอิงไปถึงตำแหน่งของ y */
```

Variable	Address	Memory
y	0x22ff74	5
yPtr	0x22ff70	0x22ff74

จากตารางข้อมูลเราจะเห็นว่าตัวแปร yPtr จะมีการเก็บข้อมูลตำแหน่งของตัวแปร y หรือความสัมพันธ์ระหว่าง yPtr กับ y คือ ตัวแปร yPtr เป็น pointer ซึ่งตัวแปร y เมื่อตัวซึ้งอิงไปถึงตัวแปรได้ตัวแปรหนึ่งแล้ว เราสามารถใช้สัญลักษณ์ \* (indirect operator หรือ Dereferenceoperator) เพื่อจะอ้างถึงข้อมูล ณ ตำแหน่งที่ตัวแปรนั้นเก็บอยู่ ดังนั้นเมื่อใช้ \*yPtr จะได้ผลลัพธ์ที่อ้างอิงถึงในกรอบนี้คือ

จำนวนเต็ม 5

ในภาษาซีนั้นตัวแปรประเภทตัวชี้และแคล้มดับมีความใกล้ชิดกันมาก เช่น เมื่อประกาศ int v[5] จะมีการจ่องเนื้อที่สำหรับข้อมูลประเภท Integer (ขนาด 4 ไบต์) ขึ้นมาห้าตำแหน่งติดกัน สมมติให้ตำแหน่งเริ่มต้นอยู่ที่ตำแหน่ง 0x22ff74 และแต่ละตัวมีขนาดสองไบต์ จะมีการจัดเก็บในหน่วยความจำดังต่อไปนี้



ตัวแปรประเภทตัวชี้ในกรณีข้างต้นนี้ จะถูกประกาศโดยใช้คำสั่ง int \*vPtr; และกำหนดให้ตัวชี้นี้ ซึ่งเป็นตำแหน่งเริ่มต้นของตัวแปรแคล้มดับโดยใช้วิธีโดยวิธีหนึ่งดังต่อไปนี้

```
vPtr = v; /* วิธีที่ 1 */
vPtr = &v[0] /* วิธีที่ 2 */
```

เมื่อใช้วิธีแรกตัวชี้จะซึ่งปะ榜ตำแหน่งเริ่มต้นคือตำแหน่งของ v[0] เสมอ แต่เมื่อใช้วิธีที่สอง เราสามารถให้ตัวชี้นั้นซึ่งเป็นตำแหน่งใดๆ ในตัวแปรแคล้มดับได้ ตั้งนั้นควรจะพิจารณาให้ถ้วนถี่ว่า ณ ขณะนั้น ตัวชี้นี้ที่ตำแหน่งใดอยู่การใช้งานข้อมูลที่เก็บไว้ในตัวแปรแคล้มดับนั้น มีอยู่สี่วิธีตามตัวอย่างต่อไปนี้ สมมติให้มีตัวแปรประเภทตัวชี้และแคล้มดับตัวอย่างข้างต้น และกำหนดให้ตัวชี้ซึ่งปะ榜ตำแหน่งแรกของตัวแปรแคล้มดับ โดยคำสั่งใดคำสั่งหนึ่งในสองวิธีที่กล่าวไปแล้ว การเข้าถึงข้อมูลของตัวแปรแคล้มดับตำแหน่งที่ n สามารถทำได้โดย

1. v[n] เป็นการใช้งานข้อมูลแบบแคล้มดับตามแบบปกติ
2. \*(v+n) เป็นการใช้งาน Dereference operator (\*) โดยเริ่มจากตำแหน่งเริ่มต้นที่ระบุโดย v และเพิ่ม(หรือลด)ไปอีก n หน่วย สังเกตได้ว่าวิธีการนี้ใช้ตัวแปรแบบแคล้มดับแบบตัวชี้
3. vPtr[n] เป็นการใช้ตัวแปรประเภทตัวชี้โดยระบุตำแหน่งที่ห่างไปจากตำแหน่งเริ่มต้น n หน่วย สังเกตได้ว่าวิธีการนี้ใช้ตัวชี้ เสมือนว่าเป็นตัวแปรแคล้มดับ
4. \*(vPtr+n) เป็นการใช้ข้อมูลแบบตัวชี้ตามปกติ คือใช้ Derefernce operator (\*) โดยเริ่มจากตำแหน่งที่ระบุโดยตัวชี้ (vPtr) ที่ห่างออกไปอีก n หน่วย

การคำนวณทางคณิตศาสตร์ของตัวชี้ เป็นอีกหัวข้อหนึ่งซึ่งนักศึกษาพึงใช้ด้วยความระมัดระวัง การเพิ่มหรือลดค่าของตัวชี้ด้วยจำนวนเต็มได ๆ จะเป็นการเพิ่มหรือลดค่าที่ตัวชี้นั้นเก็บอยู่คุณด้วยขนาดของวัตถุที่ตัวชี้นั้นซื้อยู่ เช่น เมื่อเป็นตัวชี้ที่ซึ่งปะ榜ตัวเลขจำนวนเต็ม (Integer) ซึ่งมีขนาด 4 ไบต์ การเพิ่มหรือลดจะเป็นการทวีคูณของขนาด (ในกรณีนี้คือ 4 ไบต์) เช่น หากว่า vPtr ซึ่งปะ榜ตำแหน่ง 3000 เมื่อใช้คำสั่ง vPtr++ แล้ว vPtr จะเพิ่มไปอีกหนึ่งหน่วย (ในกรณีนี้คือ 4 ไบต์) ตำแหน่งใหม่ที่ vPtr ซึ่งปะ榜เป็นตำแหน่งที่ 3000 + (1 หน่วย \* 4 ไบต์) คือตำแหน่ง 3004 ผลลัพธ์ของการคำนวณทางคณิตศาสตร์ก็

เช่นเดียวกัน นั่นก็คือจะได้เป็นหน่วยของวัตถุในกรณีที่ตัวชื่นนี้ไปยังข้อมูลประเภทโครงสร้าง การใช้ข้อมูลในโครงสร้างนั้นทำได้โดยใช้เครื่องหมาย ->(Structure pointer operator) คือใช้รูปแบบ StructurePointer->StructureMember

## การทดลองที่ 9.1

เขียนโปรแกรมต่อไปนี้และสังเกตผลลัพธ์

```
#include <stdio.h>
int main(void)
{
    char letter = 'D';
    int num = 19 ;
    float point = 26.09;
    char *pt_letter;
    int *pt_num;
    float *pt_point;
    pt_letter = &letter;
    pt_num = &num;
    pt_point = &point;
    printf("Address of letter = %p \n", pt_letter);
    printf("Address of num = %p \n", pt_num);
    printf("Address of point = %p \n", pt_point);
    return 0;
}
```

1) บันทึกผลลัพธ์ของโปรแกรม

---

2) ให้นักศึกษาใช้คำสั่ง printf แล้วใช้ format string ต่างๆ ต่อไปนี้

%p สำหรับสั่งให้คำสั่ง printf แสดงค่าตำแหน่งในหน่วยความจำ

%c สำหรับสั่งให้คำสั่ง printf แสดงตัวอักษร

%f สำหรับสั่งให้คำสั่ง printf แสดงค่าตัวเลขทศนิยม

%d สำหรับสั่งให้คำสั่ง printf แสดงค่าตัวเลขจำนวนเต็ม

แล้วเติมค่าต่างๆ ต่อไปนี้ในตารางให้สมบูรณ์ (สำหรับค่าที่ถูกอ้างอิง ให้เติมเฉพาะช่องที่ตัวแปรเป็นตัวชี้เท่านั้น)

ตัวแปร (variable)	ตำแหน่งที่อยู่ (&Variable)	ค่าที่อยู่ (value)	ค่าที่ถูกอ้างอิง (dereference value)
letter			
num			
point			
pt_letter			
pt_num			
pt_point			

## การทดลองที่ 9.2

เขียนโปรแกรมต่อไปนี้ แล้วบันทึกผล

```
#include <stdio.h>
void main(void)
{
    int v[5] = { 10, 20, 30, 40, 50};/* int v[] = { 10, 20, 30, 40, 50} */
    int *v1Ptr, *v2Ptr;
    v1Ptr = &v[1];
    v2Ptr = &v[2];
    printf("The address of v1Ptr in Hex is %p and in DEC is %u\n",v1Ptr,v1Ptr);
    printf("The address of v2Ptr in Hex is %p and in DEC is %u\n",v2Ptr,v2Ptr);
    printf("The result of v2Ptr-v1Ptr is %d\n", v2Ptr-v1Ptr);
    printf("Size of integer is %d\n", sizeof(int));
    return 0;
}
```

1. ตำแหน่งของ v1Ptr คือ \_\_\_\_\_ (ฐานสิบหก) และ \_\_\_\_\_ (ฐานสิบ)
2. ตำแหน่งของ v2Ptr คือ \_\_\_\_\_ (ฐานสิบหก) และ \_\_\_\_\_ (ฐานสิบ)
3. ผลลัพธ์ของ v2Ptr – v1Ptr คือ \_\_\_\_\_
4. ขนาดของตัวแปรประเภท integer มีขนาด \_\_\_\_\_ ไบต์

## การทดลองที่ 9.3

เขียนโปรแกรมต่อไปนี้ แล้วบันทึกผล

```
int main(void)
{
    int i, offset, b[] = {10, 20, 30, 40};
    int *bPtr = b;
    printf("Array b printed with: \n", "Array Subscript notation \n");
    for (i=0; i<=3; i++)
        printf("b[%d] = %d \n", i, b[i]);
    printf("Pointer/offset notation where\nthe pointer is the array name\n");
    for (offset=0; offset<=3; offset++)
        printf("*(b+%d) = %d\n", offset, *(b+offset));
    printf("Pointer subscript notation\n");
    for (i=0; i<=3; i++)
        printf("bPtr[%d] = %d\n", i, bPtr[i]);
    printf("Pointer/offset notation\n");
    for (offset=0; offset<=3; offset++)
        printf("*(bPtr+ %d) = %d\n", offset, *(bPtr+offset));
    return 0;
}
```

บันทึกผล

---



---

## การทดลองที่ 9.4

```
#include <stdio.h>
struct card
{
    char *face; /* 2, 3, 4, ..., J, Q, K, A */
    char *suit; /* Spade, Heart, Diamond, Club */
};

int main(void)
{
    struct card aCard;
    struct card *cardPtr;
    aCard.face = "Ace";
    aCard.suit = "Spade";
    cardPtr = &aCard;
    printf("%s%s\n%s%s\n%s%s\n",
        aCard.face, " of ", aCard.suit,
        cardPtr->face, " of ", cardPtr->suit,
        (*cardPtr).face, " of ", (*cardPtr).suit);
    return 0;
}
```

บันทึกผล

ให้นักเรียนทดสอบความเข้าใจเรื่องการใช้งานตัวชี้โดยตอบคำถามดังต่อไปนี้

1. ตัวแปรประเภทตัวชี้เป็นตัวแปรที่บันทึก \_\_\_\_\_ ของตัวแปรตัวอื่น
  2. สัญลักษณ์ \_\_\_\_\_ ใช้เพื่อแสดงตำแหน่ง (Address) ของตัวแปร
  3. สัญลักษณ์ \_\_\_\_\_ ใช้เพื่อแสดงข้อมูล ณ ตำแหน่งที่ตัวชี้นั้นชี้อยู่
  4. การประกาศตัวแปรและลำดับแบบ double ที่มีชื่อว่า numbers มีขนาด 10 หน่วย โดยใช้มีค่าเริ่มต้นเป็น 0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9 ทำได้โดยใช้ \_\_\_\_\_
- 
5. ประกาศตัวชี้ชื่อว่า nPtr โดยให้ชี้ไปยังตัวแปรและลำดับในข้อข้างต้น โดยให้ชี้ไปที่ตัวแปรและลำดับตัวแรกสามารถทำได้สองวิธีคือ
  6. จากการทดลองจะเห็นได้ว่าความสามารถใช้งานข้อมูลและลำดับได้ถึงสี่วิธี ทั้งจากการอ้างอิงจากชื่อตัวแปรและลำดับหรือจากตัวชี้ ถ้าหากต้องการให้แสดงข้อมูลทั้งสิบจำนวนจากตัวแปรและลำดับนั้นความสามารถแสดงข้อมูลนั้นได้อย่างไรบ้าง
  7. สมมติว่า nPtr ชี้อยู่ที่ตำแหน่งของตัวแปรและลำดับ โดยให้ตำแหน่งแรกของตัวแปรและลำดับอยู่ที่ตำแหน่ง FA00 และขนาดของตัวแปรนิด double มีขนาด 8 ไบต์ ตำแหน่งที่อยู่ (Address) เมื่อเราอ้างอิง nPtr + 8 คือ \_\_\_\_\_ และค่าที่เก็บอยู่ ณ ตำแหน่งนั้นคือ \_\_\_\_\_
  8. สมมติว่า nPtr ชี้อยู่ที่ตำแหน่ง [ 5 ] ตำแหน่งที่อยู่ หรือ ค่าที่บันทึกอยู่ใน nPtr คือ \_\_\_\_\_ และถ้าหากใช้คำสั่ง nPtr -= 4 แล้ว ตำแหน่งที่อยู่ใหม่คือ \_\_\_\_\_ ข้อมูลที่บันทึกอยู่ ณ ตำแหน่งนั้นคือ \_\_\_\_\_

9. สมมติว่า nPtr ชี้อยู่ที่ตำแหน่ง [ 5 ] ถ้าหากว่าเราใช้อ้างอิง nPtr [0] ตำแหน่งที่อยู่คือ \_\_\_\_\_ ข้อมูลที่บันทึก ณ ตำแหน่งนั้นคือ \_\_\_\_\_  
 nPtr [2] ตำแหน่งที่อยู่คือ \_\_\_\_\_ ข้อมูลที่บันทึก ณ ตำแหน่งนั้นคือ \_\_\_\_\_  
 nPtr [-2] ตำแหน่งที่อยู่คือ \_\_\_\_\_ ข้อมูลที่บันทึก ณ ตำแหน่งนั้นคือ \_\_\_\_\_

## คำถามพิเศษ

- รับค่าจำนวนตัวเลขศูนย์ 10 จำนวน แล้วบันทึกในตัวแปรແຄวาลำดับ จากนั้นให้รับค่ามาอีกหนึ่งค่าจากผู้ใช้งาน พัฒนาโปรแกรมเพื่อค้นหาเลขจำนวนนั้น หากพบจำนวนนั้น ให้แสดงตำแหน่ง และค่านั้น ๆ ออกที่จอภาพ หากไม่พบให้แสดงคำว่า “Cannot find the matched data” จากนั้นให้ถามว่าผู้ใช้ต้องการค้นหาอีกหรือไม่ โดยรับอักขระ Y หรือ N ให้โปรแกรมทำงานจนกว่าผู้ใช้กด N ทั้งนี้อนุญาตให้ใช้ตัวแปรແຄวาลำดับสำหรับเก็บข้อมูลทั้งสิบจำนวน จากนั้นให้ตัวชี้ไปที่ตำแหน่งเริ่มต้นของตัวแปรແຄวาลำดับ แล้วไม่ให้ใช้ตัวแปรແຄวาลำดับอีก ให้ใช้ได้แต่ตัวชี้เท่านั้น
- ให้สร้างตัวแปรประเภทโครงสร้าง โดยให้รับข้อมูลเกี่ยวกับนักศึกษาจำนวน 5 คน โดยให้บันทึก ชื่อและนามสกุล รหัสประจำตัวนักศึกษา หมายเลขโทรศัพท์ที่ให้ติดต่อ เกรดของวิชาฟิสิกซ์ (Physics) คณิตศาสตร์(Mathematics) เคมี (Chemistry) ชีววิทยา (Biology) โดยรับเป็นเกรด A, B, C, D, หรือ F จากนั้นให้โปรแกรมค้นหา โดยโปรแกรมจะรับรหัสประจำตัวนักศึกษา หากพบ ให้แสดงข้อมูล ชื่อและนามสกุล รหัสประจำตัวหมายเลขอโทรศัพท์ เกรดรายวิชา และคะแนนเฉลี่ย (GPA) ของนักศึกษานั้นหากไม่พบ ให้แสดงคำว่า “ Searchnot found ” ให้โปรแกรมทำไปเรื่อยจนกว่าผู้ใช้จะป้อนรหัสนักศึกษาเป็น 0 จึงออกจากโปรแกรม ทั้งนี้ให้เก็บແປรແຄวาลำดับที่เป็นโครงสร้างนั้น แล้วไม่ให้ใช้ตัวแปรແຄวาลำดับอีก ให้ใช้ได้แต่ตัวชี้เท่านั้น
- ทำเหมือนข้อ 2 แต่ให้เพิ่มข้อมูลที่อยู่ โดยให้เป็นโครงสร้างอีกชุดหนึ่ง โดยให้มีสมาชิกของโครงสร้างนั้นเป็นบ้านเลขที่ (Num) ถนน (Road) ตำบล (Subdistrict) อำเภอ (District) จังหวัด (Province) และรหัสไปรษณีย์(ZIPcode) และให้เพิ่มข้อมูลหมายเลขติดต่อ เป็นอีกหนึ่งโครงสร้าง โดยให้มีสมาชิกของโครงสร้างนั้นเป็นหมายเลขอโทรศัพท์ ปกติ (LandlineNumber) หมายเลขโทรศัพท์เคลื่อนที่ (MobileNumber) และหมายเลขอารสาร (FaxNumber) ดังนั้นข้อมูลของนักศึกษาแต่ละคนจะเป็นโครงสร้างซ้อนโครงสร้างอยู่ ทั้งนี้การใช้ข้อมูลตำแหน่งต่าง ๆ ให้ใช้เฉพาะตัวชี้เท่านั้น เมื่อันข้อกำหนดเดิม
- เขียนโปรแกรมเพื่อรับข้อมูลตัวเลขศูนย์สามจำนวน แล้วส่งเลขศูนย์ทั้งสามนี้ไปยังฟังก์ชันย่ออย โดยให้ฟังก์ชันย่ออย สลับตำแหน่งของตัวเลขทั้งสองนี้ (ศึกษาเรื่องการส่งข้อมูลแบบ Call by reference) โดยใช้ตัวเลขที่แรกไปอยู่ที่ตำแหน่งที่สอง ตัวเลขตำแหน่งที่สองไปอยู่ตำแหน่งที่สาม ตัวเลขที่สามไปอยู่ตำแหน่งแรก
- ทำเหมือนข้อ 4 แต่ให้รับข้อมูลเป็นข้อความหรือประโยคแทน
- เขียนโปรแกรมเพื่อตรวจสอบว่าเป็นคำหรือข้อความสมมาตร (Palindromes) กันหรือไม่ ทั้งนี้ให้ใช้เฉพาะตัวชี้ และการจัดการต่าง ๆ ของตัวชี้เท่านั้น คำหรือข้อความสมมาตรหรือ Palindromes คือคำ/ข้อความครึ่งซ้ายของคำ/ข้อความ และครึ่งขวาของคำ/ข้อความสมมาตรกัน เช่น ABBA หรือ ABCBA หรือ MADAM หรือ “A TOYOTA” หรือ “DNA LAND” หรือ DETARTRATED หรือ “WONTON? NOT NOW” เป็นต้น

## บทที่ 10 พังก์ชัน (Function)

### วัตถุประสงค์การศึกษา

- 1) เพื่อศึกษาการเขียนพังก์ชัน
- 2) เพื่อศึกษาการนำพังก์ชันมาช่วยในการพัฒนาโปรแกรม

### 10.1 พังก์ชันคืออะไร

ภายในโปรแกรมคอมพิวเตอร์ที่เขียนขึ้นมา เมื่อมีจำนวนบรรทัด และความซับซ้อนมากขึ้น นักพัฒนาจะพบปัญหาคือโปรแกรมมีความยาว ไม่สามารถมองภาพของงานในลักษณะเป็นกลุ่มได้โดยง่าย หากเราวิเคราะห์การทำงานของโปรแกรม จะพบว่ามีส่วนของงานที่ซ้ำๆ กัน เช่น ส่วนแสดงผล ส่วนรับอินพุต และอาจจะพบอีกว่าในโปรแกรมมีการเรียกใช้ส่วนแสดงผล ส่วนรับอินพุตซ้ำๆ กันหลายครั้ง ดังนั้นวิธีการลดความซับซ้อน ลดจำนวนบรรทัดของโปรแกรม ก็จะนำเอาราคาการทำงานที่ซ้ำซ้อน การทำงานที่สามารถจัดกลุ่มงานได้ แยกออกจากเป็นโปรแกรมย่อยๆ ซึ่งเราเรียกว่า พังก์ชัน

พังก์ชัน คือ ชุดของการทำงาน หรือกลุ่มคำสั่งของโปรแกรม ที่ถูกเขียนขึ้นเพื่อโปรแกรมเมอร์สามารถเรียกใช้งานได้ โดยง่าย ลดความซ้ำซ้อน และแยกส่วนประกอบของโปรแกรมได้โดยง่าย

การพัฒนาโปรแกรมกับพังก์ชัน พบปัญหาหลักคือ ที่เกิดขึ้นคือโปรแกรมที่ซับซ้อน เพราะ

- โปรแกรมเมอร์ไม่สามารถทราบการทำงานของระบบโดยละเอียดได้เช่น ไม่ทราบกระบวนการส่งข้อมูลผ่านเครือข่าย แต่ต้องเขียนโปรแกรมเพื่อเชื่อมต่อระบบเครือข่าย
- โปรแกรมเมอร์ไม่สามารถทราบขั้นตอนการทำงานของคอมพิวเตอร์ทั้งหมดได้ เช่น ทำอย่างไรตัวอักษรจะปรากฏในหน้าจอภาพได้
- โปรแกรมบางโปรแกรมมีการทำงานที่ซับซ้อน และการทำงานซับซ้อนนั้นถูกเรียกใช้งานบ่อยครั้ง เช่น การหาผลลัพธ์ทางวิทยาศาสตร์ การวิเคราะห์ข้อมูลขนาดใหญ่ เป็นต้น

#### วิธีการแก้ไข

- เพื่อให้โปรแกรมเมอร์สามารถทำงานได้โดยไม่จำเป็นต้องทราบรายละเอียดการทำงานทั้งหมดของระบบ จะให้โปรแกรมเมอร์ที่ทราบการทำงานโดยละเอียดของกระบวนการต่างๆ จะเขียนชุดคำสั่งในรูปแบบของ พังก์ชัน เลี้ยงๆ กันไป ให้โปรแกรมเมอร์อื่นๆ ได้ใช้งาน
- โปรแกรมเมอร์สามารถเรียกใช้พังก์ชันโดยทราบเพียงวิธีการใช้งาน และผลลัพธ์ที่เกิดขึ้นหลังจากเรียกใช้งานพังก์ชัน เท่านั้น
- เช่น โปรแกรมเมอร์ที่ไม่ทราบว่าทำอย่างไรตัวอักษรจะปรากฏหน้าจอ สามารถใช้คำสั่ง printf ได้โดย โปรแกรมเมอร์จะทราบเพียงแค่ การเรียก printf จะทำให้มีตัวอักษรปรากฏหน้าจอได้เท่านั้น

#### ข้อดีของพังก์ชัน

- 1) ทำให้โปรแกรมเมอร์สามารถพัฒนาโปรแกรมได้โดยง่าย โดยโปรแกรมเมอร์ไม่จำเป็นต้องทราบว่าการทำงานของพังก์ชันทำงานอย่างไรทราบเพียงผลลัพธ์ของการทำงานเท่านั้น
- 2) โปรแกรมเมอร์สามารถเขียนโปรแกรมให้มีการทำงานที่ซับซ้อนได้ โดยไม่จำเป็นต้องเขียนโปรแกรมส่วนที่ซับซ้อนนั้น หลายๆ ครั้ง
- 3) โปรแกรมเมอร์สามารถออกแบบแบบพังก์ชันได้ตามความต้องการของโปรแกรมเมอร์

## 10.2 ประเภทของฟังก์ชัน

ฟังก์ชันแบ่งออกเป็น 2 ประเภท

- 1) ฟังก์ชันไลบรารีมาตรฐาน (Standard Library function)
- 2) ฟังก์ชันที่สร้างขึ้นเอง (User Defined function)

### 10.2.1 ฟังก์ชันไลบรารีมาตรฐาน

ฟังก์ชันไลบรารีมาตรฐาน (Standard Library Function) เป็นฟังก์ชันที่มีอยู่แล้วเก็บไว้ใน Library ในการใช้งาน ต้อง include directives ก่อน

Directive คือสารบัญของกลุ่มฟังก์ชัน เช่น stdio.h , conio.h , string.h , math.h เป็นต้น

การ include directives จะเป็นเหมือนการประกาศให้กับ compiler ทราบว่าจะใช้คำสั่ง ในกลุ่มของ directive นั้นๆ เช่น การใช้คำสั่ง sin() ซึ่งอยู่ใน math.h จะต้องมีบรรทัด include math.h เสมอ ดังตัวอย่าง

ตัวอย่างการเรียกใช้ฟังก์ชันมาตรฐาน จากไลบรารี math.h

```
#include<stdio.h>
#include<conio.h>
#include<math.h>      //Include math.h
int main()
{
    double rad = -1.0;
    do {
        printf ("Sine of %f is %f\n", rad, sin(rad)); //คำนวณค่า sin ด้วยฟังก์ชัน sin()
        rad += 0.1;
    } while (rad <= 1.0);
    return 0;
}
```

การเรียกใช้ Standard Library Function มีขั้นตอนดังนี้

- 1) ทราบว่าโปรแกรมที่เขียนต้องการการทำงานอะไร
- 2) การทำงานดังกล่าวคือฟังก์ชันชื่ออะไร
- 3) ทราบ directive ที่เป็นสารบัญของคำสั่ง
- 4) Include directive นั้นๆ
- 5) เรียกใช้ฟังก์ชันในโปรแกรม

ฟังก์ชันการคำนวณทางคณิตศาสตร์

ฟังก์ชันทางคณิตศาสตร์เก็บอยู่ใน Library ชื่อ math.h เวลาเรียกใช้จะจำเป็นต้องประกาศ #include <math.h>

sin(var);	//คำนวณค่า sin
cos(var);	
tan(var);	
sqrt(var);	//คำนวณรากกำลังสอง (square root)
pow(var1,var2);	//คำนวณเลขยกกำลังโดย var1 ยกกำลังด้วย var2
log(var);	//คำนวณ logarithm ฐานธรรมชาติ
log10(var);	//คำนวณ logarithm ฐานสิบ
exp(var);	//คำนวณเลขยกกำลัง (exponential)

fabs(var);	//คำนวณ absolute value ของเลขศนิยม (floating point)
abs(var);	//คำนวณ absolute value ของเลขจำนวนเต็ม (integer)

### โปรแกรม 10.1 โปรแกรมคำนวณ sin,cos,tan,sqrt,pow,log,log10,exp และ fabs

#include<stdio.h>	//ผลลัพธ์
#include<math.h>	
int main()	
{	
float a = 3.14;	0.002
float b = -123.456;	-1.000
float c = 2.7;	-0.002
int d = -2000;	1.772
printf ("% .3f \n", sin(a));	21.964
printf ("% .3f \n", cos(a));	1.144
printf ("% .3f \n", tan(a));	0.497
printf ("% .3f \n", sqrt(a));	23.104
printf ("% .3f \n", pow(a,c));	123.456
printf ("% .3f \n", log(a));	
printf ("% .3f \n", log10(a));	
printf ("% .3f \n", exp(a));	
printf ("% .3f \n", fabs(b));	
printf ("%d \n", abs(d));	2000
return 0;	
}	

ฟังก์ชันสำหรับอักขระและข้อความ

ไฟล์ header => string.h

ctype.h

//ฟังก์ชันที่ใช้ประจำในไฟล์ string.h	
strcpy(str1, str2);	
strcat(dest1, src2);	
strcmp(dest1, src2);	
strcmpl(str1, str2);	
strlen(str);	

//ฟังก์ชันที่ใช้ประจำในไฟล์ ctype.h	
-------------------------------------	--

tolower(ch);

toupper(ch);

## โปรแกรม 10.2 โปรแกรมใช้ strcpy, strcat, strcmp, strncmp, strlen, tolower และ toupper

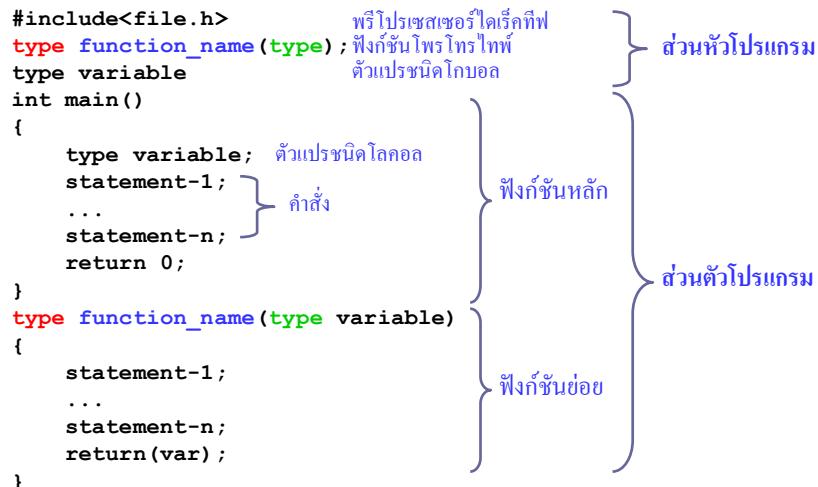
<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; #include &lt;ctype.h&gt;  int main(void) {     char string[10];     char *str = "Abc";      //strcpy     strcpy(string, str);     printf("%s\n", string);      //strcat     strcat(string, "Def");     printf("%s\n", string);      //strcmp and strncmp     printf("%d\n", strcmp(str, "abc"));     printf("%d\n", strncmp(str, "abc"));      //strlen     printf("%d\n", strlen(string));      //tolower and toupper     string[0] = tolower(string[0]);     string[1] = toupper(string[1]);     printf("%s\n", string);     return 0; }</pre>	<pre>//ผลลัพธ์  Abc AbcDef -1 0 6 aBcDef</pre>
--	--

### 10.2.2 ฟังก์ชันที่ผู้ใช้งานสร้างขึ้นเอง (User-defined function)

เนื่องจาก Standard Library Function ทั้งหมด เป็นฟังก์ชันมาตรฐานที่มีเฉพาะการทำงานพื้นฐานต่างๆ เท่านั้น หากต้องการฟังก์ชันที่มีการทำงานเฉพาะกิจ โปรแกรมเมอร์ต้องเขียนฟังก์ชันขึ้นมาเอง

ข้อกำหนดพื้นฐานของ User-defined Function

- 1) ต้องมีการประกาศ function prototype ที่ต้นโปรแกรมเสมอ จึงจะเรียกใช้งาน function นั้นๆ ได้ (เป็นการบอก Compiler ว่าคำสั่งดังกล่าวคือฟังก์ชัน ไม่ใช่ syntax error)
- 2) ต้องมีการเขียนฟังก์ชันตามโครงสร้างที่ได้ประกาศไว้ใน function prototype เท่านั้น



### 10.3 รูปแบบโครงสร้างของฟังก์ชัน

ໂປຣໂຕໄທປັບປຸງກໍ່ຂັ້ນ (Function Prototype)

ຮູບແບບຄໍາສົ່ງ

```
type function_name(type1, type2, ..., typeN);
```

ເປັນການປະກາສການໃຊ້ຈານພັບປຸງກໍ່ຂັ້ນທີ່ອີ່ຍໍ່ຫລັ້ງ main()

type ດືວ່າ ຂີດຂອງພັບປຸງກໍ່ຂັ້ນ ວ່າພັບປຸງກໍ່ຂັ້ນທີ່ທໍາການສ້າງຈະສ່າງຂໍ້ມູນນິດໄດ້ກຳລັບ

function\_name ດືວ່າ ຂໍ້ອີ່ຍໍ່ຫລັ້ງພັບປຸງກໍ່ຂັ້ນທີ່ຈະສ້າງຂຶ້ນ

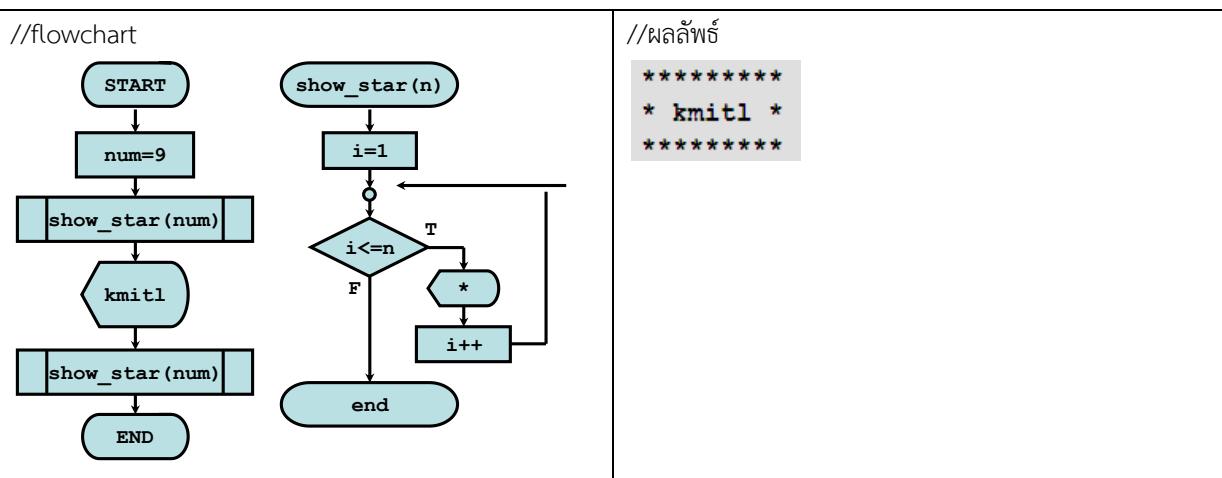
type-n ດືວ່າ ຂີດຂອງຂໍ້ມູນທີ່ຈະສ່າງໃຫ້ພັບປຸງກໍ່ຂັ້ນ

ການເຂື່ອນໂປຣແກຣມໂດຍມີພັບປຸງກໍ່ຂັ້ນທີ່ສ້າງຂຶ້ນເອງມີ 2 ຮູບແບບ

- 1) ສ້າງພັບປຸງກໍ່ຂັ້ນ ກ່ອນ ພັບປຸງກໍ່ຂັ້ນຫລັກ ພັບປຸງກໍ່ຂັ້ນຫລັກສາມາດເຮັດວຽກໃຊ້ຈານພັບປຸງກໍ່ຂັ້ນທີ່ສ້າງຂຶ້ນໄດ້
- 2) ສ້າງພັບປຸງກໍ່ຂັ້ນ ຫລັງ ພັບປຸງກໍ່ຂັ້ນຫລັກ ຕ້ອງປະກາສ Function Prototype ກ່ອນເພື່ອໃຫ້ພັບປຸງກໍ່ຂັ້ນຫລັກຮັກວ່າມີພັບປຸງກໍ່ຂັ້ນທີ່ສ້າງຂຶ້ນ

<pre>//ສ້າງພັບປຸງກໍ່ຂັ້ນກ່ອນພັບປຸງກໍ່ຂັ້ນຫລັກ #include&lt;file.h&gt; type variable <b>type function_name(type variable)</b> {     statement-1;     ...     statement-n;     return(var); } int <b>main()</b> {     type variable;     statement-1;     ...     statement-n;     return 0; }</pre>	<pre>//ສ້າງພັບປຸງກໍ່ຂັ້ນຫລັງພັບປຸງກໍ່ຂັ້ນຫລັກ #include&lt;file.h&gt; <b>type function_name(type)</b>; type variable <b>int main()</b> {     type variable;     statement-1;     ...     statement-n;     return 0; } <b>type function_name(type variable)</b> {     statement-1;     ...     statement-n;     return(var); }</pre>
---	--

#### ໂປຣແກຣມ 10.3 ໂປຣແກຣມແສດງ kmitl ໃນກຣອບລື່ເທົ່າຍົນ



<pre>//สร้างฟังก์ชันก่อนฟังก์ชันหลัก #include&lt;stdio.h&gt; #include&lt;conio.h&gt; void show_star (int n) {     int i;     for (i=1;i&lt;=n;i++)         putchar('*'); } int main() {     int num=9;     show_star(num);     printf ("\n* kmitl *\n");     show_star(num);     return 0; }</pre>	<pre>//สร้างฟังก์ชันหลังฟังก์ชันหลัก #include&lt;stdio.h&gt; #include&lt;conio.h&gt; void show_star (int); int main() {     int num=9;     show_star(num);     printf ("\n* kmitl *\n");     show_star(num);     return 0; } void show_star (int n) {     int i;     for (i=1;i&lt;=n;i++)         putchar('*'); }</pre>
--	--

## 10.4 การรับค่าและส่งค่าจากฟังก์ชัน

รูปแบบการรับค่าเข้าฟังก์ชัน และรูปแบบการส่งค่าออกจาฟังก์ชัน จัดได้สี่แบบคือ

- 1) ฟังก์ชันที่ไม่มีการรับส่งค่า
- 2) ฟังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ
- 3) ฟังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ
- 4) ฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ

### 10.4.1 ฟังก์ชันที่ไม่มีการรับส่งค่า

เป็นฟังก์ชันที่ไม่มีการรับค่าเข้ามาในฟังก์ชัน และไม่มีการส่งค่ากลับออกไปจากฟังก์ชัน

<pre>#include&lt;stdio.h&gt; #include&lt;conio.h&gt; void function_name(void); int main() {     function_name();     return 0; }  void function_name() {     statement-1;     statement-2;     ...     statement-n; }</pre>	<pre>//ตัวอย่างโปรแกรม  #include &lt;stdio.h&gt; void show(void) {     printf("hello"); }  int main(void) {     show();     show();     return 0; }</pre>
---	---

### 10.4.2 ฟังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ

เป็นฟังก์ชันที่มีการรับค่าเข้ามาในฟังก์ชัน แต่ไม่มีการส่งค่ากลับออกไปจากฟังก์ชัน

<pre>#include&lt;stdio.h&gt; #include&lt;conio.h&gt; void func_name(type); int main() { }</pre>	<pre>//ตัวอย่างโปรแกรม  #include &lt;stdio.h&gt; void show(int i,float f) {</pre>
---	---

<pre> ... func_name(varX); ... return 0; } void func_name(type varY) {     statement-1;     statement-2;     ...     statement-n; } </pre>	<pre> printf("%f", i+f); }  int main(void) {     show(22,3.14);     show(10,12.1);     return 0; } </pre>
--	---

#### 10.4.3 ฟังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ

เป็นฟังก์ชันที่มีการรับค่าเข้ามาในฟังก์ชัน และมีการส่งค่ากลับออกไปจากฟังก์ชัน

<pre> #include&lt;stdio.h&gt; #include&lt;conio.h&gt; type func_name(type); int main() {     ...     var = func_name(varX);     ...     return 0; } type func_name(type varY) {     statement-1;     statement-2;     ...     statement-n;     return(varZ); } </pre>	<pre> //ตัวอย่างโปรแกรม #include &lt;stdio.h&gt; float show(int i,float f) {     printf("%f",i+f);     return i*f; }  int main(void) {     float a;     a = show(22,3.14);     printf("%f",a);     printf("%f",show(10,12.1));     return 0; } </pre>
---	---

#### 10.4.4 ฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ

เป็นฟังก์ชันที่ไม่มีการรับค่าเข้ามาในฟังก์ชัน แต่มีการส่งค่ากลับออกไปจากฟังก์ชัน

<pre> #include&lt;stdio.h&gt; type func_name(void); int main() {     ...     var = func_name();     ...     return 0; } type func_name() {     statement-1;     statement-2;     ...     statement-n;     return(varZ); } </pre>	<pre> //ตัวอย่างโปรแกรม #include &lt;stdio.h&gt; int getdata() {     int k;     printf("enter value");     scanf("%d",&amp;k);     return k; }  int main(void) {     int a;     a = getdata();     printf("%d",a);     printf("%d",getdata());     return 0; } </pre>
--	---

### โปรแกรม 10.4 โปรแกรมคำนวณ *degree radian*

```
#include<stdio.h>
#include<math.h>
#define PI 3.14
float deg_rad(float); /*Function Prototype*/
int main()
{
    float deg,rad;
    printf ("Enter Degree : ");
    scanf ("%f",&deg);
    rad = deg_rad(deg); //printf ("%f->%f\n",deg,rad);
    printf ("sin(% .2f) = % .3f\n",deg,sin(rad));
    printf ("cos(% .2f) = % .3f\n",deg,cos(rad));
    printf ("tan(% .2f) = % .3f\n",deg,tan(rad));
    return 0;
}
float deg_rad(float num)
{
    float ans;
    ans = num * PI / 180;
    return(ans);
}
```

## 10.5 ตัวแปรและขอบเขตการใช้งานฟังก์ชัน

ขอบเขตของตัวแปรในโปรแกรมแบ่งขอบเขตได้สองระดับคือ

- 1) ตัวแปร global เป็นตัวแปรที่ฟังก์ชันใดก็สามารถเรียกใช้ได้โดยจะประกาศสร้างตัวแปรต่อจาก พิริโอเดตเซอร์ ไดเรคทีฟ
- 2) ตัวแปร local เป็นตัวแปรที่สามารถเรียกใช้ได้ภายในเฉพาะฟังก์ชันที่ประกาศสร้างตัวแปรนั้นโดยจะประกาศสร้าง ตัวแปรภายในแต่ละฟังก์ชัน

ตัวอย่าง ประกาศตัวแปร num1 เป็น global ชนิด integer

<pre>#include&lt;stdio.h&gt; #include&lt;conio.h&gt; int num1; void test(void); /*Function Prototype*/ int main() {     num1 = 19;     printf ("line1 (main) : num1 = %d\n",num1);     test();     printf ("line2 (main) : num1 = %d\n",num1);     return 0; } void test() {     num1 = 26;     printf ("line1 (test) : num1 = %d\n",num1); }</pre>	<pre>//ผลลัพธ์ line1 (main) : num1 = 19 line1 (test) : num1 = 26 line2 (main) : num1 = 26</pre>
---	---

ตัวอย่าง ประกาศตัวแปร num1 เป็น local ชนิด integer

```
#include<stdio.h>
#include<conio.h>
void test(void);      /*Function Prototype*/
int main()
{
    int      num1;
    num1 = 19;
    printf ("line1 (main) : num1 = %d\n",num1);
    test();
    printf ("line2 (main) : num1 = %d\n",num1);
    return 0;
}
void test()
{
    int      num1;
    num1 = 26;
    printf ("line1 (test) : num1 = %d\n",num1);
}
```

//ผลลัพธ์

```
line1 (main) : num1 = 19
line1 (test) : num1 = 26
line2 (main) : num1 = 19
```

## 10.6 การส่งค่าตัวแปร

การส่งค่าตัวแปรมีสองแบบคือ

- 1) การส่งค่าที่เก็บอยู่ในตัวแปรให้กับฟังก์ชัน (pass by value)
- 2) การส่งค่า Address ของตัวแปรให้กับฟังก์ชัน (pass by reference)

### 10.6.1 Pass by value

เป็นการส่งค่าที่เก็บอยู่ในตัวแปรเข้าสู่ฟังก์ชัน การเปลี่ยนแปลงค่าต่างๆ ของพารามิเตอร์จะไม่เปลี่ยนแปลงค่าของตัวแปรในโปรแกรมหลัก ลักษณะของ Pass by Value Function จะไม่ส่งค่า Address (หน้าตัวแปรจะไม่มีเครื่องหมาย \*)

ตัวอย่าง

```
int add(int a, int b)
void draw_line(int count)
```

### 10.6.2 Pass by reference

เป็นการส่งค่า Address ของตัวแปรเข้าสู่ฟังก์ชัน การเปลี่ยนแปลงค่าต่างๆ ของพารามิเตอร์จะส่งผลไปยังตัวแปรในโปรแกรมหลัก ลักษณะของ Pass by Reference Function จะส่งค่า Address (หน้าตัวแปรจะมีเครื่องหมาย \* หน้าตัวแปรเสมอ) ตัวอย่าง

```
int max(int *a, int *b)
void increase(int *count)
```

### โปรแกรม 10.5 โปรแกรมแสดง pass by value และ pass by reference

```
//Pass by value
void func(int va)
{
    va = va+1;
    printf ("In function la = %d\n",va);
}

//Pass by reference
void func2(int *pa)
{
    *pa = *pa +1;
    printf ("In function *pa = %d\n",*pa);
}

int main()
{
    int x;
    x = 10;
    printf ("Before call function x = %d\n",x);
    func(x);           //pass by value
    printf ("After call function x = %d\n",x);
    printf ("\n\n");
    x = 10;
    printf ("Before call function x = %d\n",x);
    func2(&x);         //pass by reference
    printf ("After call function x = %d\n",x);
return 0;
}
```

//ผลลัพธ์

```
Before call function x = 10
In Function la = 11
After call function x = 10
```

```
Before call function x = 10
In Function *pa = 11
After call function x = 11
```

### โปรแกรม 10.6 โปรแกรมใช้ pass by value

```
#include <stdio.h>
#include <conio.h>
int a;
//global
void func(int x)
{
    x=10;
    printf("x = %d\n",x);
}

int main()
{
    int b;
    a = 3;
    b = 5;
    func(b);
    printf ("a = %d\n",a);
    printf ("b = %d\n",b);
    return 0;
}
```

//ผลลัพธ์

```
x = 10
a = 3
b = 5
```

### โปรแกรม 10.7 โปรแกรมใช้ pass by reference

<pre>#include &lt;stdio.h&gt; #include &lt;conio.h&gt; int a; void func(int *x) {     *x=10;     printf("x = %d\n", *x); }  int main() {     int b;     a = 3;     b = 5;     func(&amp;b);     printf ("a = %d\n", a);     printf ("b = %d\n", b);     return 0; }</pre>	//ผลลัพธ์ x = 10 a = 3 b = 10
---	--

### โปรแกรม 10.8 โปรแกรมรีจิก global variable

<pre>#include &lt;stdio.h&gt; #include &lt;conio.h&gt; int a; void func(int x) {     a=10;     printf("x = %d\n", x); }  int main() {     int b;     a = 3;     b = 5;     func(b);     printf ("a = %d\n", a);     printf ("b = %d\n", b);     return 0; }</pre>	//ผลลัพธ์ x = 5 a = 10 b = 5
---	---------------------------------------

## 10.7 คำถ้ามท้ายบท

1) โปรแกรมเก็บข้อมูลสินค้า 1 ชนิด ภายในโปรแกรมประกอบด้วย พังก์ชันต่างๆดังนี้

พังก์ชันใส่ข้อมูลสินค้า      getdata()  
 พังก์ชันเพิ่มจำนวนสินค้า 10 ชิ้น      add10()  
 พังก์ชันลดจำนวนสินค้า 10 ชิ้น      sub10()  
 และพังก์ชันแสดงจำนวนรายละเอียดภายในสินค้า

โปรแกรมนี้ยังไม่สมบูรณ์ดี ให้นักศึกษาเขียนคำประกาศพังก์ชัน getdata() และ add10 และคำสั่งภายในให้สมบูรณ์

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

struct product {
    char name[15];
    int number;
    float price;
};

struct product getdata(void) {
    struct product x;
    strcpy(x.name, "Chewing gum");
    x.number = 100;
    x.price = 25.5;
    return x;
}

void showdata(.....)
{
    .....
}

void sub10(struct product &x) {
    if (x.number > 10)
        x.number -= 10;
}

void add10(.....)
{
    x.number += 10;
}

int main(void)
{
    int i;
    struct product y;
    y = getdata();
    showdata(y);
    add10(y);
    add10(y);
    showdata(y);
    sub10(y);
    showdata(y);
    return 0;
}
```

2) ให้นักศึกษาอธิบายการทำงานของโปรแกรมนี้ พิรอดมกับชี้ให้เห็นว่าจุดผิดของโปรแกรมคือจุดใด

Hint: มีที่ผิด 1 จุด

```
#include <stdio.h>
#include <string.h>
void callnum(int i,char y) {
    char data [10][8] = {"zero","one","two","three","four",
                        "five","six","seven","eight","nine"};
    strcpy(y,data[i]);
}

int main(void) {
    char ch[10];
    int i;
    int telno[] = {0,2,7,3,7,3,0,0,0};
    for (i=0;i<9;i++) {
        callnum(telno[i],ch);
        printf("%s ",ch);
    }
    return 0;
}
```

3) โปรแกรมต่อไปนี้แสดงผลเลขอะไรมากว่า ใช้เวลาคิดไม่เกิน 10นาที เมื่อคิดเสร็จแล้วให้ลองพิมพ์ code ดังกล่าวลงคอมพิวเตอร์ และตรวจสอบคำตอบ

```
#include <stdio.h>
int i;
struct data {
    int i;
    int j;
}W;

void fa(int *a, int b);
void fb(int c, int *d);
void fc(struct data x, struct data *y, struct data z);
int main(void) {
    int j=0;
    int i = 15;
    struct data X,Y,Z;
    X.i = 10;
    X.j = 20;
fa(&i,j);
    printf("\n i=%d, j=%d",i,j);
    fb(i,&j);
    printf("\n i=%d, j=%d",i,j);
    fc(X,&Y,Z);
    printf("\n W.i=%d,W.j=%d",W.i,W.j);
    printf("\n X.i=%d, X.j=%d",X.i,X.j);
    printf("\n Y.i=%d, Y.j=%d",Y.i,Y.j);
    printf("\n Z.i=%d, Z.j=%d",Z.i,Z.j);
    return 0;
}
void fa(int *a, int b) {
    i = 4;
    *a = *a+3;
    b = i+2;
}
void fb(int c,int *d) {
    c += i;
    *d += i;
}

void fc(struct data x,struct data *y, struct data z) {
    y->i = x.j++;
    y->j = y->i + (++x.i);
    z.i = x.j;
    z.j = y->j;
    W.i = y->i + (++x.j);
    W.j = y->j + z.i;
}
```

## การทดลองที่ 10 พังก์ชัน

### วัตถุประสงค์

1. เพื่อให้นักศึกษาทดลองเขียนโปรแกรมโดยใช้พังก์ชัน
2. เพื่อให้นักศึกษาเพื่อให้นักศึกษาเข้าใจเรื่องพังก์ชัน และการทำงานของคำสั่งพื้นฐานต่างๆ

### ทฤษฎี

ในการเขียนโปรแกรมบางครั้งต้องมีการใช้ชุดคำสั่งแบบเดียวกันซ้ำ ๆ หลายครั้งในหลาย ๆ จุดของตัวโปรแกรม หรืออาจต้องการแบ่งโปรแกรมออกเป็นส่วน ๆ ตามรูปแบบการทำงาน ดังนั้นเพื่อเป็นการลดจำนวนชุดคำสั่งที่ต้องใช้ เพื่อความสะดวกในการเขียนโปรแกรม และเพื่อความง่ายในการแก้ไขและบำรุงรักษา ภาษาโปรแกรมจึงเปิดโอกาสให้ผู้เขียนโปรแกรมสามารถนำเอาชุดคำสั่งที่อาจใช้ซ้ำได้นั้นไปเขียนรวมกันไว้เป็นพวง ๆ เพื่อให้ง่ายต่อการต่อการเรียกใช้ซึ่งชุดคำสั่งดังกล่าวจะถูกเรียกว่าพังก์ชัน โดยพังก์ชันในภาษาซีมีด้วยกัน 2 ประเภท

1. พังก์ชันมาตรฐาน – ไลบรารีพังก์ชัน (Standard Library Function)
2. พังก์ชันที่สร้างขึ้นเอง (User-defined Function)

การทดลองนี้จะให้นักศึกษาทำความเข้าใจเกี่ยวกับการใช้งานพังก์ชันที่สร้างขึ้นเอง สำหรับการประกาศพังก์ชันที่สร้างขึ้นเอง นั้น มีการประกาศในสองรูปแบบด้วยกันคือ

2.1 การสร้างพังก์ชัน ก่อนพังก์ชันหลัก ซึ่งต้องเขียนพังก์ชันที่จะใช้งานก่อนเรียกใช้พังก์ชัน เพื่อให้โปรแกรมได้รู้จักพังก์ชันก่อน

2.2 การสร้างพังก์ชัน หลังพังก์ชันหลัก สามารถเขียนพังก์ชันหลังจากพังก์ชันหลักได้ (main function) แต่ต้องประกาศ โพรโตไทป์พังก์ชัน (Function Prototype) ก่อน พังก์ชันหลักโดยโปรแกรมเมอร์สามารถใช้งานได้ทั้งสองรูปแบบ สำหรับพังก์ชันที่โปรแกรมเมอร์ได้สร้างขึ้นเองนั้นสามารถแบ่งออกได้เป็น 4 กลุ่มหลักตามการรับส่งค่าของพังก์ชัน ดังนี้

1. พังก์ชันที่ไม่มีการรับส่งค่า
2. พังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ
3. พังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ
4. พังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ

พังก์ชันที่ไม่มีการรับส่งค่า คือพังก์ชันที่ทำงานแบบตัวตัว ทุกๆ ครั้งที่เรียกใช้งานพังก์ชันนี้จะมีการทำงานในรูปแบบเดิมๆ ทุกๆ ครั้ง เนื่องจากไม่มีการรับค่าพารามิเตอร์ใดๆ และหลังจากการทำงานพังก์ชันนี้แล้ว จะไม่มีการส่งผลลัพธ์กลับไปยังผู้เรียกใช้งานพังก์ชัน ตัวอย่างเช่น void printmenu(void);

พังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ คือพังก์ชันทำงานโดยขึ้นอยู่กับผู้ใช้งานพังก์ชันจะป้อนพารามิเตอร์อะไรดังนั้น การทำงานของพังก์ชันในลักษณะนี้จะยืดหยุ่น แต่ภายหลังจากที่พังก์ชันทำงานเรียบร้อยแล้ว จะไม่มีการส่งผลลัพธ์การทำงานกลับไปยังผู้เรียกใช้งาน เช่นเดียวกับแบบแรก ตัวอย่างเช่น void print\_message(int m\_id);

พังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ คือพังก์ชันที่มีการทำงานขึ้นอยู่กับผู้ใช้งานพังก์ชันจะป้อนพารามิเตอร์อะไร การทำงานลักษณะนี้มีความยืดหยุ่นสูง นอกจากรับส่งค่าผลลัพธ์การทำงานกลับไปยังผู้ใช้งานพังก์ชันด้วย ตัวอย่างเช่น int area(int width, int height);

ฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับคือฟังก์ชันที่ส่งข้อมูลกลับโดยไม่ต้องอาศัยพารามิเตอร์ใดๆ ก็ตามทั้งเช่น

```
int time(void)
```

#### การส่งค่าพารามิเตอร์เข้าสู่ฟังก์ชัน

1. การส่งค่าแบบ pass by value คือการส่งข้อมูลที่ตัวแปรนั้นๆ เก็บอยู่ ส่งให้ฟังก์ชันทำงาน โดยการทำงานของฟังก์ชันจะไม่มีผลกระทบต่อค่าที่เก็บในตัวแปร เนื่องจากส่งเฉพาะข้อมูลไปยังฟังก์ชันเท่านั้น
2. การส่งค่าแบบ pass by reference คือการส่งค่าตำแหน่งของตัวแปรนั้นๆ ส่งให้ฟังก์ชันทำงาน โดยการทำงานของฟังก์ชันจะมีผลกระทบต่อค่าที่เก็บในตัวแปร เนื่องจากการทำงานภายในฟังก์ชันจะไปกระทําภัยตัวแปรที่ตำแหน่งนั้นๆ โดยตรง

ชนิดของตัวแปร

1. ตัวแปร global เป็นตัวแปรที่ถูกประกาศตรงตำแหน่งก่อนการประกาศฟังก์ชันหลัก ตัวแปร global จะเป็นตัวแปรที่สามารถอ้างถึงได้จากทุกๆ ตำแหน่งในโปรแกรม
2. ตัวแปร local เป็นตัวแปรที่ถูกประกาศภายในฟังก์ชันต่างๆ ซึ่งจะสามารถอ้างอิงได้ภายในเครื่องหมาย { และ } ของฟังก์ชันนั้นๆ เท่านั้น

#### การทดลองที่ 10.1

1. ให้นักศึกษาเขียนโปรแกรมเพื่อรับค่าจำนวนเต็ม 2 จำนวน และส่งค่าไปในฟังก์ชัน

```
int findmax(int, int);
```

เป็นฟังก์ชันที่รับค่าจำนวนเต็มมา แล้วส่งค่าที่มีค่ามากกว่ากลับ

2. นำค่าที่ได้จากฟังก์ชัน findmax มาแสดงผล

#### การทดลองที่ 10.2

ให้นักศึกษา แก้ไขโปรแกรมในการทดลองที่ 10.1 โดยให้รับตัวเลขมา 3 จำนวน และแสดงผลค่าที่มากที่สุด โดยใช้ฟังก์ชัน findmax (ห้ามแก้ไขตัวฟังก์ชัน)

#### การทดลองที่ 10.3

ให้นักศึกษา เขียนโปรแกรมรับค่ามา 2 จำนวน โดยโปรแกรมจะแสดงเลขจำนวนเฉพาะตั้งแต่ค่าแรกจนถึงค่าที่ 2 ที่รับเข้ามา โดยให้สร้างฟังก์ชัน

```
void primecheck(int);
```

เป็นฟังก์ชันที่รับค่าจำนวนเต็มมาตรวจสอบว่าเป็นจำนวนเฉพาะหรือไม่ ถ้าใช่ให้แสดงผลค่านั้นในรูปแบบ printf ("%4d", var)

#### การทดลองที่ 10.4

ให้นักศึกษาเขียนโปรแกรมเพื่อรับข้อความมา 1 ข้อความ และหาว่าในข้อความที่รับมามีความยาวเท่าใดและมีสระ กี่ตัว

## การทดลองที่ 10.5

ให้นักศึกษาแก้ไขโปรแกรมจากการทดลองที่ 10.4 โดยการสร้างฟังก์ชัน

```
int countstr(char[]);
```

เป็นฟังก์ชันที่รับข้อความเข้ามาเพื่อตรวจสอบว่ามีความยาวเท่าใด แล้วส่งค่าความยาวกลับฟังก์ชันหลัก

```
int countvowel(char[],int);
```

เป็นฟังก์ชันที่รับข้อความ และค่าความยาวของข้อความเข้ามาเพื่อตรวจสอบว่ามีสรุจจำนวนเท่าใด แล้วส่งค่ากลับฟังก์ชันหลัก

## การทดลองที่ 10.6

พิจารณาโปรแกรมต่อไปนี้

```
#include <stdio.h>
void swap (int *, int *);
int main(void)
{
    int a = -5, b=10;
    printf("Before swapping\n");
    printf("The value of a is %d, and the value of b is %d\n",a,b);
    swap(&a,&b);
    printf("After swapping\n");
    printf("The value of a is %d, and the value of b is %d\n",a,b);
    return 0;
}

void swap (int *aptr, int *bptr)
{
    int temp;
    temp = *bptr;
    *bptr = *aptr;
    *aptr = temp;
}
```

จากฟังก์ชัน swap ที่มีการส่งค่าตัวแปรแบบ pass by reference ให้นักศึกษาอธิบายการเปลี่ยนแปลงค่าของตัวแปร a และ b ว่ามีการเปลี่ยนแปลงค่าอย่างไร และเหตุใดจึงเป็นเช่นนั้น

## การทดลองที่ 10.7

พิจารณาโปรแกรมต่อไปนี้

```
#include <stdio.h>
int cubeByValue(int);
void cubeByReference(int *);
int main(void)
{
    int number =5;
    printf("The original value of number is %d\n", number);
    number = cubeByValue(number);
    printf("The new value of number is %d\n", number);
    cubeByReference(&number);
    printf("The new value of number is %d\n", number);
    return 0;
}

int cubeByValue(int n)
{
    return n*n*n;
}
```

```
void cubeByReference(int *nPtr)
{
    *nPtr = *nPtr * *nPtr * *nPtr;
}
```

ให้นักศึกษาพิจารณาความแตกต่างระหว่างฟังก์ชัน cubeByValue กับ cubeByReference ว่าการทำงานมีความแตกต่างกันอย่างไร ให้ผลลัพธ์แตกต่างกันหรือไม่ อย่างไร

---



---

### การทดลองที่ 10.8

โปรแกรมต่อไปนี้ ทำงานอะไร

```
#include <stdio.h>
int mystery1 (char *s);
int main (void)
{
    char string[80];
    printf("Enter a string: ");
    scanf("%s", string);
    printf("%d\n", mystery1(string));
    return 0;
}

int mystery1 (char *s)
{
    int x;
    for (x=0; *s != '\0'; s++)
    {
        x++;
    }
    return x;
}
```

ผลลัพธ์ของโปรแกรมอธิบาย

---



---

### การทดลองที่ 10.9

1. จงเขียนฟังก์ชันในการคำนวณค่า Det ของเมตริกขนาด 3x3 พร้อมทั้งเขียนโปรแกรมเพื่อทดสอบการทำงานของฟังก์ชัน
2. จงสร้างฟังก์ชันให้ผลลัพธ์ตัวอักษร A-F เมื่อป้อนคะแนน โดยใช้เกณฑ์คะแนนของวิชา Principles of Computer Programming พร้อมทั้งเขียนโปรแกรมเพื่อทดสอบการทำงาน
3. จักสมการ  $a(x) = b(x-1) + a(x-1)$  และ  $b(x) = a(x-2) + b(x-2)$  เมื่อ  $a(0) = b(0) = 0$  ,  $a(1) = b(1) = 1$  จงเขียนโปรแกรมรับค่า  $n$  และแสดงผลลัพธ์  $a(n)$  (หมายเหตุ: ควรศึกษาการเขียนโปรแกรมแบบ recursion เพื่อช่วยแก้ปัญหา)
4. กระต่ายตัวหนึ่งสามารถขึ้นบันไดโดยการกระโดดได้ทีละ 1 ขั้นหรือ 2 ขั้น จงเขียนโปรแกรมเพื่อหาจำนวนรูปแบบที่เป็นไปได้ทั้งหมดในการกระโดดขึ้นบันได  $N$  ขั้นของกระต่ายตัวนี้ (หมายเหตุ: ควรศึกษาการเขียนโปรแกรมแบบ recursion เพื่อช่วยแก้ปัญหา)

## บทที่ 11 ไฟล์ข้อมูล

### วัตถุประสงค์การศึกษา

- 1) เพื่อศึกษาการเขียนโปรแกรมเก็บข้อมูลลงไฟล์
- 2) เพื่อศึกษาคำสั่งที่ใช้ในการเก็บข้อมูลลงไฟล์

### 11.1 ไฟล์ข้อมูล

ไฟล์ข้อมูลถูกสร้างขึ้นเพื่อเก็บรวบรวมข้อมูลต่างๆเข้าไว้ด้วยกัน ไฟล์ข้อมูลมักจะถูกเก็บอยู่ในหน่วยความจำภายในอก เช่น Hard Disk, CD หรือ DVD เป็นต้น

เมื่อคอมพิวเตอร์ทำการอ่าน (read) ข้อมูลจะถูกสำเนาจากอุปกรณ์ภายนอกเข้ามาอย่างหน่วยความจำภายใน เมื่อ คอมพิวเตอร์ทำการบันทึก (write) ข้อมูลจะถูกสำเนาจากหน่วยความจำภายในไปยังอุปกรณ์ภายนอก

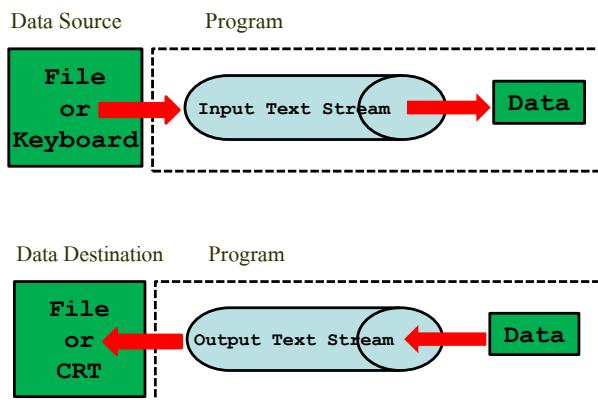
ในการเคลื่อนย้ายข้อมูลนี้จะมีการใช้พื้นที่หน่วยความจำที่กันเอาไว้สำหรับงานนี้เป็นพิเศษ ซึ่งเรียกว่า buffer เป็นองจากการอ่านหรือนำข้อมูลจากอุปกรณ์ภายนอกนั้น ในบางครั้งจะได้จำนวนข้อมูลที่มากกว่าที่จะป้อนเข้าโปรแกรมในครั้งเดียว

ในทางกลับกันเมื่อโปรแกรมต้องการบันทึกข้อมูลลงハードดิสก์ buffer จะทำหน้าที่ในการพักข้อมูลไว้จนกว่าจะมี ข้อมูลจำนวนที่มากพอที่จะบันทึกลงハードดิสก์

การทำงานที่เกี่ยวข้องกับ buffer จะดูแลโดยโปรแกรมพิเศษที่เราเรียกว่า device drivers ซึ่งโปรแกรมนี้จะให้ มาโดยผู้สร้างอุปกรณ์ต่างๆ หรือมาพร้อมกับ Operating System

โปรแกรมภาษา C จะมองข้อมูลที่ส่งมาจากอุปกรณ์ (e.g. keyboard) หรือไฟล์ว่าเป็นสายของข้อมูล (data stream) และเมื่อโปรแกรมต้องการส่งข้อมูลให้กับอุปกรณ์ (e.g. CRT) หรือไฟล์ก็จะส่งเป็นสายข้อมูลไปให้

ในระหว่างการรับส่งข้อมูลกับ buffer หากมีข้อผิดพลาดเกิดขึ้นก็จะทำให้มีข้อมูลบางส่วนตกค้างอยู่ใน buffer



ภาษา C จะแบ่ง data stream ออกเป็น 2 ประเภท คือ

- 1) **สายข้อมูลตัวอักษร (Text Stream)** เป็นสายข้อมูลตัวอักษรที่มีนุษย์สามารถอ่านได้ ข้อมูลจะอยู่ในรูปแบบของ รหัส ASCII (ตัวอักษร) เช่น A, B,...,Z, a, b,...,z, 0,1,..., 9 อักษรพิเศษ โดยที่สายข้อมูลตัวอักษรจะแบ่งออกเป็น บรรทัดๆ แต่ละบรรทัดปิดท้ายด้วยตัวรหัส \n
- 2) **สายข้อมูลในนารี (Binary Stream)** เป็นสายข้อมูลชนิดต่างๆ เช่น intger, real, complex ซึ่งข้อมูลต่างๆเหล่านี้ จะเก็บอยู่ในรูปแบบที่คอมพิวเตอร์จะเข้าใจ เช่น 2's Complement

File เป็นสิ่งที่มีชื่อและ Operating System รู้จัก

Stream เป็นสิ่งที่มีชื่อเหมือนกัน โดยที่สตรีมถูกสร้างโดยโปรแกรม

ในการประมวลผลหรือทำงานเกี่ยวกับไฟล์ จะต้องมีการเชื่อมโยงระหว่าง stream และ file เข้าด้วยกัน

ขั้นตอนการทำงานในการประมวลผลเกี่ยวกับไฟล์ ประกอบด้วย 4 ขั้นตอนหลัก คือ

- 1) ทำการสร้าง file stream pointer ตัวอย่างคำสั่ง File\* fp;
- 2) ทำการเปิดไฟล์โดยการเขียน file name เข้ากับชื่อของ stream name ตัวอย่างคำสั่ง fp = fopen("c:\\MyFile.DAT","r");
- 3) ทำการอ่านหรือบันทึกข้อมูลลงไฟล์ผ่านทางช่อง stream ตัวอย่างคำสั่ง fscanf(fp,"%d %d %d",&date,&month,&year);
- 4) ทำการปิดไฟล์ ตัวอย่างคำสั่ง fclose(fp);

## 11.2 เปิดไฟล์

รูปแบบการเปิดเท็กซ์ไฟล์

รูปแบบคำสั่ง

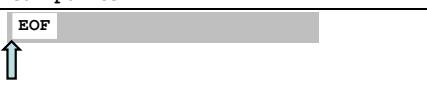
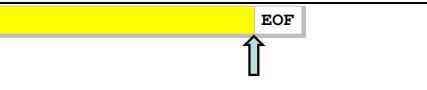
```
FILE *fp;
fp = fopen ("name", "mode");
```

FILE คือ ประเภทข้อมูลแบบโครงสร้างที่กำหนดไว้ให้แล้วใน stdio.h

*	คือ เครื่องหมายที่แสดงว่าเป็นตัวแปร Pointer
fp	คือ ชื่อตัวแปรพอยน์เตอร์ไปยังไฟล์สตรีม (file stream pointer)
name	คือ ชื่อของไฟล์ที่ทำการเปิด
mode	คือ โหมดของการจัดการกับไฟล์

หมายเหตุ ชื่อของ file stream pointer ที่ระบบกำหนดให้แล้วใน stdio.h ได้แก่ stdin (standard input stream), stdout (standard output stream) และ stderr (standard error stream)

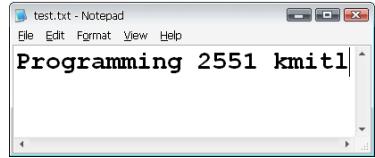
### Mode ของการจัดการกับไฟล์

“r”	เป็นการเปิดเท็กซ์ไฟล์เพื่ออ่าน (Read) ถ้าไม่มีไฟล์อยู่ก็จะเกิดสถานะ fail	
“w”	เป็นการสร้างไฟล์เพื่อเขียน (Write) ถ้าไฟล์นั้นมีอยู่แล้วจะลบข้อมูลเดิมทั้งหมด ถ้าไม่มีไฟล์นั้นจะทำการสร้างเท็กซ์ไฟล์ใหม่	
“a”	เป็นการ เปิด หรือ สร้าง ไฟล์เพื่อเขียนเพิ่ม (Append) ถ้าไฟล์นั้นมีอยู่แล้วจะเป็นการเขียนข้อมูลต่อท้าย ถ้าไม่มีไฟล์นั้นจะทำการสร้างเท็กซ์ไฟล์ใหม่	

“r+”	เป็นการเปิดไฟล์เพื่อ Update (อ่านและเขียนข้อมูล) ถ้าไม่มีไฟล์อยู่ก็จะเกิดสถานะ fail ตัวชี้ตำแหน่งจะซึ้งไปที่ตำแหน่งเริ่มต้นของไฟล์	
“w+”	เป็นการสร้างไฟล์เพื่อ Update (อ่านและเขียนข้อมูล) ถ้าไฟล์นั้นมีอยู่แล้วจะลบข้อมูลเดิมทั้งหมด ถ้าไม่มีไฟล์นั้นจะทำการสร้างเท็กซ์ไฟล์ใหม่เพื่อ Update	
“a+”	เป็นการเปิดหรือสร้างไฟล์เพื่อ Update (อ่านและเขียนข้อมูล) ถ้าไฟล์นั้นมีอยู่แล้วจะเขียนต่อท้าย ข้อมูลเดิม ตัวชี้ตำแหน่งจะซึ้งไปยังตำแหน่งท้ายสุด (EOF: End Of File)	

### โปรแกรม 11.1 โปรแกรมเปิด text file

จะเขียนโปรแกรมเพื่อเปิดไฟล์ข้อมูล โดยถ้าไฟล์สามารถเปิดได้จะแสดงคำว่า “can open file” หากไฟล์ไม่สามารถเปิดได้แสดงคำว่า “cannot open file” ใช้โปรแกรม Notepad สร้างไฟล์ข้อมูล ดังรูปด้านล่าง บันทึกเป็นชื่อ test.txt เก็บไว้ที่ C:\Temp\

#include<stdio.h>  #include<conio.h>   int main() {  FILE *fp;  fp = fopen("test.txt","r"); //สังเกตระบุการ directory if (fp == NULL) { printf("Cannot open file\n"); return 0; } else printf("Can open file\n"); return 0; }	//สร้างไฟล์ข้อมูลด้วย notepad และ save ไว้ที่ c:\temp\test.txt    //หากระบุ directory เป็น test.txt ผลลัพธ์ได้ Cannot open file  //หากระบุ directory เป็น c:/temp/test.txt ผลลัพธ์ได้ Can open file
--	---

### 11.3 ปิดไฟล์

รูปแบบคำสั่ง

```
fclose (fp);
```

fp คือ ตัวแปรพอยน์เตอร์ที่ใช้ไปยังไฟล์สตรีม  
 ผลลัพธ์ของฟังก์ชัน fclose() หากสามารถปิดไฟล์ได้สำเร็จจะ return ค่า 0 กลับ แต่หากไม่สำเร็จจะ return ค่า EOF (โดยปกติคือค่า -1)

### โปรแกรม 11.2 โปรแกรมเปิดและปิด text file

<pre>#include&lt;stdio.h&gt; #include&lt;conio.h&gt; int main() {     FILE *fp;     fp = fopen("C:/Temp/test.txt","r");     if (fp == NULL)     {         printf("Cannot open file\n");         return 0;     }     printf("Can open file\n");     if (!fclose(fp))         printf("\nClose file\n");     return 0; }</pre>	//ผลลัพธ์  Can open file  Close file
---	--

### 11.4 อ่านไฟล์รูปแบบ Text

รูปแบบคำสั่ง

```
fscanf(fp, "format", &variable);
```

fp คือตัวแปรพอยน์เตอร์ที่ใช้ไปยังไฟล์สตรีม  
 format คือ การกำหนดรูปแบบในการอ่านข้อมูลขึ้นมาจากไฟล์ เช่น อ่านข้อมูลเป็นชนิด int หรือ double  
 variable คือ ตัวแปรที่เก็บค่าข้อมูลจากไฟล์ (จะต้องกำหนดแอดเดรสของตัวแปรโดยใช้สัญลักษณ์ & เสมอ)

### โปรแกรม 11.3 โปรแกรมอ่านข้อมูลจาก text file (string หนึ่งตัว)

จะเขียนโปรแกรมเพื่อเปิดไฟล์ข้อมูล เพื่ออ่านข้อมูล และแสดงผล ใช้โปรแกรม Notepad สร้างไฟล์ข้อมูล

บันทึกเป็น test.txt เก็บไว้ที่ C:\Temp\ (คล้ายกับตัวอย่างโปรแกรมที่ 11.1)

<pre>#include&lt;stdio.h&gt; int main() {     FILE *fp;     char str[20];     fp = fopen("C:/Temp/test.txt","r");     if (fp == NULL)     {         printf("Cannot open file\n");         return 0;     }     printf("Can open file\n");     fscanf(fp,"%s",str);           //สังเกต     if (!fclose(fp))         printf("\nClose file\n");     printf("%s",str);     return 0; }</pre>	//ผลลัพธ์  Can open file  Close file  Programming
---	---

### โปรแกรม 11.4 โปรแกรมอ่านข้อมูลจาก text file (string หลายตัว)

<pre>#include&lt;stdio.h&gt; int main() {     FILE *fp;     char str1[20],str2[10],str3[10];     fp = fopen("C:/Temp/test.txt","r");     if (fp == NULL)     {         printf("Cannot open file\n");         return 0;     }     printf("Can open file\n");     fscanf(fp,"%s%s%s",str1,str2,str3); //สังเกต     if (!fclose(fp))     {         printf("\nClose file\n");     }     printf("%s %s %s",str1,str2,str3); }</pre>	//ผลลัพธ์ Programming 2551 kmitl
--	-------------------------------------

### โปรแกรม 11.5 โปรแกรมอ่านข้อมูลจาก text file (character)

<pre>#include&lt;stdio.h&gt; int main() {     FILE *fp;     char ch1,ch2,ch3;     fp = fopen("C:/Temp/test.txt","r");     if (fp == NULL)     {         printf("Cannot open file\n");         return 0;     }     printf("Can open file\n");     fscanf(fp,"%c%c%c",&amp;ch1,&amp;ch2,&amp;ch3); //สังเกต     if (!fclose(fp))     {         printf("\nClose file\n");     }     printf("%c%c%c",ch1,ch2,ch3);     return 0; }</pre>	//ผลลัพธ์ Pro
--	------------------

### โปรแกรม 11.6 โปรแกรมอ่านข้อมูลจาก text file (string กับ int)

<pre>#include&lt;stdio.h&gt; int main() {     FILE *fp;     char str1[20],str2[10]; int num;     fp = fopen("C:/Temp/test.txt","r");     if (fp == NULL)     {         printf("Cannot open file\n");         return 0;     }</pre>	//ผลลัพธ์ Programming 2551 kmitl
--	-------------------------------------

```

printf("Can open file\n");
fscanf(fp,"%s%d%s",str1,&num,str2); //สั่งเกต
if (!fclose(fp))
{
    printf("\nClose file\n");
}
printf("%s %d %s",str1,num,str2);
return 0;
}

```

## 11.5 เขียนไฟล์รูปแบบ text file

รูปแบบคำสั่ง

```
fprintf (fp, "format-string", value);
```

fp	คือตัวแปรพอยน์เตอร์ที่ใช้ไปยังไฟล์สตรีม
format	คือ การกำหนดรูปแบบของข้อมูลที่จะเขียนลงไฟล์
variable	คือ ค่าที่ต้องการเขียนลงไฟล์ (ไม่ต้องมี & )

### โปรแกรม 11.7 โปรแกรมรับชื่ออายุ 3 คน เก็บลงไฟล์

จะเขียนโปรแกรม เพื่อรับข้อมูล ชื่อ และอายุของคน 3 คน เก็บข้อมูลใน C:\Temp\new.txt นำข้อมูลจากไฟล์มา

แสดงผล

```

#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *fp;
    char name[20],ch;    int age,count=0;
    fp = fopen("C:/Temp/new.txt","w");
    if (fp == NULL)
    {
        printf("Cannot open file\n");
        return 0;
    }
    printf("Can open file (write)\n");
/*input data in program & write on file*/
    while(count<3)
    {
        printf ("Enter name : ");
        scanf ("%s",name);
        printf ("Enter age : ");
        scanf ("%d",&age);
        fprintf (fp,"%-20s %d\n",name,age);
        count++;
    }
    if (!fclose(fp))
        printf("\nClose file\n");
    fp = fopen("C:/Temp/new.txt","r");
    if (fp == NULL)
        printf("Cannot open file\n");
    return 0;
}

```

```

    }
    printf("Can open file (read)\n");
    do
    {
        ch = getc(fp);
        printf("%c",ch);
    } while (ch!=EOF);
    if (!fclose(fp))
        printf("\nClose file\n");
    return 0;
}

```

## 11.6 เขียน array หรือ struct ลงไฟล์รูปแบบ binary

รูปแบบคำสั่ง

```
int fwrite (ptr, size, items, stream);
```

ptr คือ ตำแหน่งของตัวแปร struct หรือ array ในหน่วยความจำ (address) ที่ต้องการจะเขียนลงไฟล์

size คือ ขนาดของตัวแปร struct หรือ array

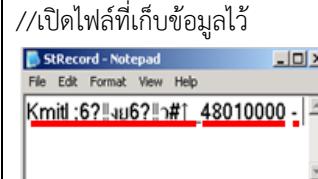
items คือ จำนวนของ struct หรือ array ที่ต้องการจะเขียนลงไฟล์

stream คือ ตัวแปรพอยน์เตอร์ที่ป้ายไฟล์สตรีม

fwrite จะ return จำนวน bytes ที่ได้เขียนลงในไฟล์ และจะ return 0 เมื่อไม่สามารถเขียนข้อมูลลงไฟล์ได้

### โปรแกรม 11.8 โปรแกรมเก็บข้อมูลจาก struct ลงไฟล์

<pre> int main() {     FILE *fptr;     struct student {         char name[20];         char ID[9];         int age;      } s;     fptr = fopen("StRecord.txt", "w");     printf("Name : "); scanf("%s",s.name);     printf("ID : ");   scanf("%s",s.ID);     printf("Age : "); scanf("%d",&amp;s.age);     fwrite(&amp;s, sizeof(struct student), 1, fptr);     fclose(fptr);     return 0; } </pre>	//ป้อนข้อมูลจาก keyboard Name : Kmitl ID : 48010000 Age : 45_
--	--



## 11.7 อ่าน array หรือ struct จากไฟล์รูปแบบ binary

รูปแบบคำสั่ง

```
int fread(dest, size, items, stream);
```

dest คือ ตัวแหน่งของตัวแปร struct หรือ array ที่ใช้เก็บค่าที่อ่านจากไฟล์

size คือ ขนาดของตัวแปร struct หรือ array ที่ต้องการอ่านเข้ามา

items คือ จำนวนของ struct หรือ array ที่ต้องการจะอ่านจากไฟล์

stream คือ ตัวแปรพอยน์เตอร์ที่ปั้นไปยังไฟล์สตรีม

fread จะ return ค่าเป็นจำนวน bytes ที่อ่านได้ และจะ return 0 มันทำการอ่านจนถึงตำแหน่งสุดท้ายของไฟล์

### โปรแกรม 11.9 โปรแกรมอ่านข้อมูลจากไฟล์เก็บลง struct

```
int main()
{
    FILE *fptr;
    struct student
    {
        char name[20];
        char ID[9];
        int age;
    }s;
    fptr = fopen("StRecord.txt", "r");
    if (fptr == (FILE *)NULL) printf("Cannot open file\n");
    else
        while (fread(&s,sizeof(struct student),1,fptr)!=0)
        {
            printf("Name: %s\n", s.name);
            printf("ID: %s and Age: %d\n", s.ID,s.age);
            printf("-----\n");
        }
    fclose(fptr);
    return 0;
}
```

//ผลลัพธ์

Name: Kmitl

ID: 48010000 and Age: 45

## 11.8 ย้าย file pointer

รูปแบบคำสั่ง

```
int fseek(stream, offset, where);
```

stream คือ ตัวแปรพอยน์เตอร์ที่ปั้นไปยังไฟล์สตรีม

offset คือ ระยะที่ต้องการให้ตัวชี้ตำแหน่งในไฟล์เคลื่อนไปในแต่ละครั้ง

where เป็นกำหนดตำแหน่งของตัวชี้ตำแหน่งในไฟล์ มีได้ 3 ค่าคือ

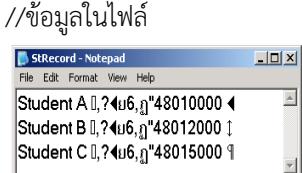
0: ตัวชี้ตำแหน่งในไฟล์เริ่มต้นที่ตำแหน่งแรกสุดของไฟล์

1: ตัวชี้ตำแหน่งในไฟล์เริ่มต้นที่ตำแหน่งปัจจุบัน

2: ตัวชี้ตำแหน่งในไฟล์เริ่มต้นที่ตำแหน่งสุดท้ายของไฟล์

fseek จะ return 0 ถ้าสามารถเลื่อนตัวชี้ตำแหน่งในไฟล์ได้สำเร็จ และจะ return ค่า int ใด ๆ ที่ไม่เท่ากับ 0 เมื่อดำเนินการตาม parameter ไม่สำเร็จ

### โปรแกรม 11.10 โปรแกรมอ่านข้อมูลจากไฟล์เก็บลง struct ใช้ fseek

<pre> FILE *fptr; int noffset; struct student {     char name[20];     char ID[9];     int age; } s; fptr = fopen("StRecord.txt", "w"); strcpy (s.name,"Student A"); strcpy (s.ID,"48010000"); s.age = 17; fwrite(&amp;s, sizeof(struct student), 1, fptr); strcpy (s.name,"Student B"); strcpy (s.ID,"48012000"); s.age = 18; fwrite(&amp;s, sizeof(struct student), 1, fptr); strcpy (s.name,"Student C"); strcpy (s.ID,"48015000"); s.age = 20; fwrite(&amp;s, sizeof(struct student), 1, fptr); fclose(fptr); fptr = fopen("StRecord.txt", "r"); noffset = 0 * sizeof(struct student); //สังเกต if (fseek( fptr, noffset, 0) == 0 ) {     if (fread(&amp;s,sizeof(struct student),1,fptr) != 0)     {         printf("Name: %s\n", s.name);         printf("ID: %s and Age: %d\n", s.ID,s.age);         printf("-----\n");     } } fclose(fptr); </pre>	 <p>//ข้อมูลในไฟล์</p> <p>//(1) ย้าย file pointer โดยกำหนด //noffset = 0 * sizeof(struct student); //ได้ผลลัพธ์</p> <p>Name: Student A ID: 48010000 and Age: 17</p> <p>//(2) ย้าย file pointer โดยกำหนด //noffset = 1 * sizeof(struct student); //ได้ผลลัพธ์</p> <p>Name: Student B ID: 48012000 and Age: 18</p> <p>//(3) ย้าย file pointer โดยกำหนด //noffset = 2 * sizeof(struct student); //ได้ผลลัพธ์</p> <p>Name: Student C ID: 48015000 and Age: 20</p>
---	---

### 11.9 คำสั่งอื่นเกี่ยวกับไฟล์

การอ่านและเขียนข้อมูลตัวอักษร

```

var = fgetc(fp); //var = getc(fp);
fputc (ch,fp); //putc (ch,fp);
คำสั่ง getchar() และ putchar() ใช้ได้กับสตรีม stdin และ stdout เท่านั้น

```

การอ่านและเขียนข้อมูลที่เป็นข้อความ

```
fgets(var,length,fp);
fputs (str,fp);
การอ่านจะอ่านข้อความขนาดไม่เกิน length-1
```

การอ่านและเขียนข้อมูลตามชนิดของข้อมูล

```
fscanf (fp,"format",&variable);
fprintf (fp,"format",value);
```

การอ่านและเขียนข้อมูลจาก Array ชี้โดย ptr

```
fread(*ptr,size,items,*stream);
fwrite(*ptr,size,items,*stream);
```

## 11.10 คำถ้ามหัยบท

1) จะเขียนโปรแกรมเพื่ออ่านไฟล์(สามารถเลือกไฟล์ได้) เพื่อนับจำนวนสระ และแสดงผลจำนวนสระแต่ละตัวในไฟล์ใหม่ (สามารถเลือกชื่อไฟล์ได้) โดยในไฟล์ที่เก็บคำตอบให้เก็บในรูปแบบดังนี้

Number of Vowel

A :	10
E :	0
I :	2
O:	3
U :	12

2) จะเขียนโปรแกรมเพื่อนับจำนวนครั้งการรันโปรแกรมหนึ่ง ๆ ว่ามีการรันไปทั้งหมดกี่ครั้งแล้ว โดยให้เก็บจำนวนครั้งการรันไว้ในไฟล์

3) จะเขียนโปรแกรมเพื่อกีบข้อมูลของคนจำนวน 10 คน โดยเก็บชื่อ อายุ ไว้ในไฟล์ (สามารถเลือกชื่อไฟล์ได้) และค้นหาคนที่มีอายุมากกว่า 20 ปีจากข้อมูลในไฟล์ที่เก็บไว้ และแสดงผลที่ output

## การทดลองที่ 11 การเขียนโปรแกรมติดต่อกับ Text File

### วัตถุประสงค์

1. เพื่อให้นักศึกษาทดลองเขียนโปรแกรมติดต่อกับไฟล์
2. เพื่อให้นักศึกษาสามารถเปิดไฟล์ข้อมูลเพื่อนำมาใช้งานได้
3. เพื่อให้นักศึกษาสามารถสร้างไฟล์เพื่อเขียนข้อมูลลงไฟล์ได้

### ทฤษฎี

การสร้างไฟล์ และคำสั่งทำงานกับไฟล์

ไฟล์ข้อมูลถูกสร้างขึ้นเพื่อใช้ข้อมูลสำหรับการใช้งานครั้งต่อไป การจะใช้งานไฟล์ข้อมูล ต้องเขียนโปรแกรมสร้างไฟล์ข้อมูลขึ้นมา หรือเขียนโปรแกรมเพื่ออ่านไฟล์ข้อมูลที่มีอยู่ โดยการเรียกใช้คำสั่งต่างๆ เช่น เปิดไฟล์ เก็บหรือเขียนข้อมูลลงไฟล์ ดึงหรืออ่านข้อมูลจากไฟล์ แล้วจึงนำข้อมูลไปประมวลผลต่อไป และปิดไฟล์ข้อมูลเมื่อทำงานเสร็จสำหรับภาษา C สามารถสร้างไฟล์สำหรับเก็บข้อมูลได้ 2 แบบคือ ไฟล์ข้อมูลตัวอักษร (Text file) กับไฟล์ข้อมูลใบหน้า(Non-text file) สำหรับการทดลองครั้งนี้เป็นการทดลองกับ Text file ซึ่งไฟล์ประเภทนี้เก็บข้อมูลเป็นตัวอักษร เช่น A-Z,a-z, 0-9, หรือ ตัวอักษรพิเศษ

ขั้นตอนแรกของการเขียนโปรแกรมติดต่อกับ Text file คือการประกาศตัวแปรไฟล์ ซึ่งจะถูกประกาศในรูปของตัวแปร pointer เราเรียกว่า file pointer ซึ่งจะใช้ตัวแปรดังกล่าวภายในโปรแกรมเพื่อทำการติดต่อกับไฟล์ข้อมูลจริงที่ถูกเก็บไว้ในอุปกรณ์สำรองข้อมูล เช่น แผ่นดิสก์ หรือ ฮาร์ดดิสก์ เป็นต้น การประกาศตัวแปรไฟล์มีรูปแบบดังต่อไปนี้

```
FILE *fp;
```

ความหมาย

FILE เป็นการสร้างตัวแปรชนิด file

\* เป็นเครื่องหมายที่แสดงว่าเป็น pointer

fp เป็น file pointer ที่สร้างขึ้นเพื่อเก็บตำแหน่งของไฟล์ข้อมูล

หลังจากที่ซึ่งเป็นตัวแปรนั้นเพื่อจัดการกับไฟล์โดยใช้ fp โดยมีคำสั่งต่างๆ ที่ใช้ในการเขียนโปรแกรมติดต่อกับไฟล์ดังนี้

### 1.1 คำสั่งเปิดไฟล์

```
fp = fopen("name", "mode");
```

ความหมาย

name คือ ชื่อของไฟล์ที่ทำการเปิด

mode คือ โหมดของการจัดการกับไฟล์

r เปิดไฟล์เพื่ออ่านอย่างเดียว ต้องมีไฟล์นั้นอยู่จริง

w เปิดไฟล์เพื่อเขียนข้อมูลทับลงไฟล์เก่า ถ้าไม่มีไฟล์เครื่องจะสร้างขึ้น

a เปิดไฟล์เพื่อเขียนข้อมูลต่อท้ายไฟล์เก่า ถ้าไม่มีไฟล์เครื่องจะสร้างขึ้น

ตัวอย่าง การเปิดไฟล์

```
fp = fopen ("c:\temp\mydata.txt", "r");
```

เป็นคำสั่งเปิดไฟล์เพื่ออ่านข้อมูลลงไฟล์ชื่อ mydata.txt ถ้าในดิสก์ไม่พบไฟล์ในชื่อดังกล่าว โปรแกรมจะหยุดทำงานและแสดงข้อผิดพลาด

```
fp = fopen ("c:\temp\mydata.txt", "w");
```

เป็นคำสั่งเปิดไฟล์เพื่อเขียนข้อมูลลงไฟล์ชื่อ mydata.txt ถ้าในดิสก์ไม่พบไฟล์ในชื่อดังกล่าว คำสั่งนี้จะสร้างไฟล์ว่างเปล่าให้โดยไม่มีข้อมูล แต่ถ้ามีไฟล์ชื่อดังกล่าวอยู่แล้ว ข้อมูลเดิมในไฟล์ทั้งหมดจะหายไป กลายเป็นไฟล์ว่างเปล่า

```
fp = fopen ("c:\temp\mydata.txt", "a");
```

เป็นคำสั่งเปิดไฟล์เพื่อเขียนข้อมูลลงไฟล์ชื่อ mydata.txt โดยในดิสก์จะต้องมีไฟล์ชื่อดังกล่าวอยู่ก่อนหน้านี้แล้ว คำสั่งนี้จะเขียนข้อมูลใหม่ต่อท้ายข้อมูลเดิมที่มีอยู่ในไฟล์ ถ้าในดิสก์ไม่พบไฟล์ในชื่อดังกล่าว โปรแกรมจะหยุดทำงานและแสดงข้อผิดพลาด

### 1.2 คำสั่งปิดไฟล์

```
fclose (fp)
```

เป็นคำสั่งปิดไฟล์เมื่อทำงานเสร็จ เมื่อโปรแกรมทำงานกับไฟล์ข้อมูลเสร็จแล้ว การเรียกใช้คำสั่งปิดไฟล์จะเติมตัวอักษรพิเศษปิดท้ายไฟล์บอกจุดสิ้นสุดของข้อมูลในไฟล์ หากสามารถปิดไฟล์ได้สำเร็จจะ return ค่า 0 กลับ

### 1.3 คำสั่งอ่านข้อมูลจากไฟล์

```
fscanf (fp,"format string",address variable);
```

ความหมาย

fp (file pointer) คือ ตัวแปรที่ใช้เก็บตำแหน่งไฟล์

format string คือ การใช้รหัสรูปแบบ

address list คือ ตัวแปรที่เก็บค่าข้อมูลจากไฟล์

fscanf เป็นฟังก์ชันในการอ่านข้อมูลจากไฟล์ ซึ่งประกอบด้วยพารามิเตอร์ สามตัวคือ file pointer, format string และ address list รูปแบบในการอ่านข้อมูลจากไฟล์จะถูกกำหนดโดย format string ซึ่งสอดคล้องกับตัวแปรต่างๆที่อยู่ใน address list

```
var = getc(fp);
```

```
var = fgetc(fp);
```

getc และ fgetc เป็นฟังก์ชันที่ใช้ในการอ่านข้อมูลประเภทตัวอักษรจากไฟล์ข้อมูลที่อยู่ในอุปกรณ์สำรองข้อมูลผ่านตัวแปรไฟล์ fp

### 1.4 คำสั่งเขียนข้อมูลลงไฟล์

```
fprintf (fp,"format string",data list);
```

fprintf เป็นฟังก์ชันในการเขียนข้อมูลลงไฟล์ ซึ่งประกอบด้วยพารามิเตอร์ สามตัวคือ filepointer, format string และ data list รูปแบบในการเขียนข้อมูลลงไฟล์จะถูกกำหนดโดย format string ซึ่งสอดคล้องกับตัวแปรต่างๆ ที่อยู่ใน data list

```
fputc (ch,fp);
```

```
fputc (ch,fp);
```

`putc` และ `fputc` เป็นฟังก์ชันที่ใช้ในการเขียนข้อมูลประเภทตัวอักษร (ที่ถูกเก็บในตัวแปร ch) ลงไฟล์ข้อมูลผ่านตัวแปรไฟล์ fp

### การทดลองที่ 11.1

ให้นักศึกษาพิมพ์โปรแกรมตามตัวอย่างต่อไปนี้ และสั่งให้โปรแกรมทำงาน

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *fp;
    fp = fopen("test.txt", "r");
    if (fp == NULL)
    {
        printf("Error \nCannot open file\n");
        getch();
        return (0);
    }
    else
        printf("Can open file\n");
    getch();
    return 0;
}
```

ผลลัพธ์ที่ได้เป็นอย่างไร เหตุใดจึงเป็นเช่นนั้น

---



---

### การทดลองที่ 11.2

ให้นักศึกษาเขียนโปรแกรมสร้างไฟล์ชื่อ demo.txt ที่มีรายละเอียดคือ ชื่อ นามสกุล รหัสนักศึกษา โดยอยู่คุณละบรรทัด

### การทดลองที่ 11.3

ให้นักศึกษาเขียนโปรแกรมสำหรับ copy ไฟล์ชื่อ demo.txt ไปยังไฟล์ชื่อ backup.txt

### การทดลองที่ 11.4

จะเขียนโปรแกรม copy ไฟล์ โดยอ่านข้อมูลจากไฟล์ (สามารถเลือกชื่อไฟล์ได้) และสร้างไฟล์ใหม่ที่เหมือนกับไฟล์ต้นฉบับ (สามารถตั้งชื่อไฟล์ได้)

### การทดลองที่ 11.5

ให้นักศึกษาพิมพ์โปรแกรมตามตัวอย่างต่อไปนี้ และสั่งให้โปรแกรมทำงาน

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *fp;
    struct people
    {
        char name[20];
        char surname[20];
        int day;
```

```

        char month[15];
        int year;
    } pe;

    fp = fopen("data.txt", "w");
    if (fp == NULL)
    {
        printf("Error \nCannot open file\n");
        getch();
        return (0);
    }
    else
        printf("Can open file\nPlease Enter to Continue..");
    getch();
    printf ("\nEnter your name : ");
    scanf ("%s",pe.name);
    printf ("Enter your surname : ");
    scanf ("%s",pe.surname);
    printf ("Enter your birthday \n");
    printf ("\t Day :");
    scanf ("%d",&pe.day);
    printf ("\t Month :");
    scanf ("%s",pe.month);
    printf ("\t Year :");
    scanf ("%d",&pe.year);
    fprintf (fp,"%s %s ",pe.name,pe.surname);
    fprintf (fp,"%d %s %d\n",pe.day,pe.month,pe.year);
    if (!fclose(fp))
    {
        printf("\nClose file\n");
    }
    getch();
    return 0;
}

```

ผลลัพธ์ที่ได้จากการรันโปรแกรมเป็นอย่างไร

---



---



---

หากต้องการเก็บข้อมูลคน 5 คนต้องแก้ไขโปรแกรมอย่างไร

---



---



---

### การทดลองที่ 11.6

1. ให้นักศึกษา Save as ไฟล์ HTML ในอินเตอร์เน็ตแล้วเขียนโปรแกรมเพื่อนับลิ้งค์ใน HTML ไฟล์นั้น
2. ให้นักศึกษา Save as ไฟล์ HTML ในอินเตอร์เน็ตแล้วเขียนโปรแกรมเพื่อแสดงรายการลิงค์ Image ทั้งหมดใน HTML ไฟล์นั้น
3. ให้นักศึกษาหาไฟล์รูปภาพตัวอย่างที่มีนามสกุล .bmp และเขียนโปรแกรมเพื่อหาว่าในไฟล์รูปภาพ .bmp นั้นมีจุดสีแดง ทั้งหมดกี่จุด

## หนังสืออ้างอิง

- [1] The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie. Prentice Hall
- [2] Computer Science: A Structured Programming Approach Using C, Second Edition , Behrouz Forouzan, Richard Gilberg
- [3] Turbo C Programming Principles and Practices, M.Tim Grady
- [4] คอมพิวเตอร์และการเขียนโปรแกรม (Computers and Programming) C และ Java, ชนัญชัย ตรีภาก, สนพ. ซีเอ็ด ยูเคชั่น จำกัด