

# บทที่ 1

## ความรู้เกี่ยวกับคอมพิวเตอร์และการประมวลผลข้อมูล

### จุดประสงค์

1. เพื่อให้ทราบหลักการทำงานของตัวแปลภาษาประเภทต่าง ๆ
2. เข้าใจขั้นตอนของหลักการเขียนโปรแกรม
3. เพื่อให้ทราบความรู้เบื้องต้นเกี่ยวกับภาษาซี

### 1.1 ความหมายของคอมพิวเตอร์

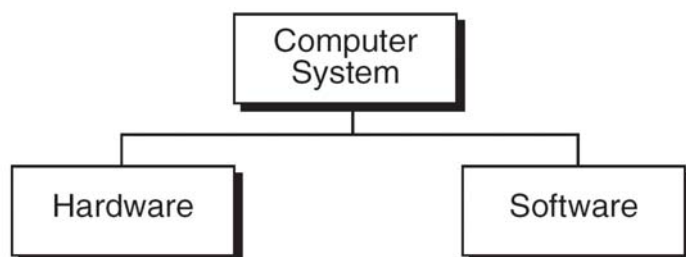
คอมพิวเตอร์เป็นอุปกรณ์ทางไฟฟ้าชนิดหนึ่งที่สามารถประมวลผลและจำข้อมูลต่าง ๆ ได้ สามารถคิดคำนวณตัวเลข สามารถตอบสนองต่อการกระทำของผู้ใช้ได้และมีความสามารถในการเชื่อมต่อกับอุปกรณ์ไฟฟ้าบางชนิด เพื่อสั่งให้อุปกรณ์นั้นทำงานตามคำสั่งได้ ใช้สำหรับแก้ปัญหาต่าง ๆ ทั้งที่ง่ายและซับซ้อนโดยวิธีทางคณิตศาสตร์

เครื่องคอมพิวเตอร์เป็นเครื่องอิเล็กทรอนิกส์ที่ช่วยให้การประมวลผลทำได้อย่างมีประสิทธิภาพ และสามารถทำงานที่มีความซับซ้อนได้อย่างแม่นยำโดยสามารถรับข้อมูลที่อยู่ในรูปแบบที่เครื่องสามารถรับได้ แล้วนำมาทำการประมวลผลคือ ทำการคำนวณ เปรียบเทียบ จนได้ผลลัพธ์ตามที่ต้องการ

### 1.2 องค์ประกอบของระบบคอมพิวเตอร์



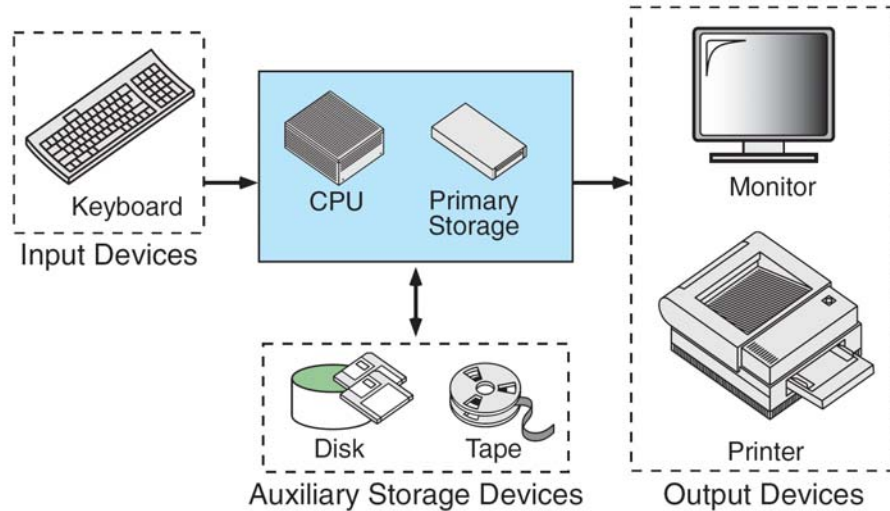
คอมพิวเตอร์มีองค์ประกอบที่สำคัญแบ่งได้เป็นสองส่วนหลัก ๆ คือ ฮาร์ดแวร์ และซอฟต์แวร์ สามารถแสดงองค์ประกอบได้ดังรูปที่ 1.1



รูปที่ 1.1 องค์ประกอบของระบบคอมพิวเตอร์ [1]

### 1.2.1 คอมพิวเตอร์ฮาร์ดแวร์ (Hardware Computer)

เป็นส่วนประกอบของคอมพิวเตอร์ที่จับต้องสัมผัสได้ ประกอบด้วยวงจรทางไฟฟ้าอิเล็กทรอนิกส์ที่เรียกว่าฮาร์ดแวร์(Hardware) โดยมีส่วนประกอบที่สำคัญได้แก่ หน่วยรับข้อมูล หน่วยแสดงผลข้อมูล ระบบประมวลผลและหน่วยเก็บข้อมูล



รูปที่ 2 คอมพิวเตอร์ฮาร์ดแวร์ (Hardware Computer) [1]

#### 1.หน่วยรับข้อมูล(Input Unit)

หน่วยรับข้อมูลหรืออินพุต(Input) จะมีอุปกรณ์อินพุตเป็นส่วนที่ใช้รับข้อมูลและคำสั่งจากภายนอกเข้าสู่เครื่องคอมพิวเตอร์เพื่อนำไปประมวลผล อุปกรณ์ประเภทนี้ได้แก่ แป้นพิมพ์ (Keyboard) เมาส์(Mouse) สแกนเนอร์(Scanner) ไมโครโฟน(Microphone) เครื่องอ่านบาร์โค้ด(Barcode Reader) อุปกรณ์อินพุตจะเปลี่ยนข้อมูลที่มนุษย์เข้าใจเปลี่ยนเป็นรหัสข้อมูลที่เครื่องคอมพิวเตอร์เข้าใจ อุปกรณ์เหล่านี้จะทำงานได้ ข้อมูลคำสั่งจะต้องถูกเก็บไว้ในสื่อ(Input media) ที่อุปกรณ์นั้น ๆ รู้จักเรียกว่า Input Device เช่น หากเก็บข้อมูลไว้ใน Disk จะต้องอ่านข้อมูลที่เก็บภายใน Disk นั้นด้วย Disk Drive เป็นต้น

#### 2. หน่วยแสดงผลหรือเอาต์พุต(Output Unit)

หน่วยแสดงผลหรือเอาต์พุต(Output) เป็นส่วนที่ใช้แสดงผลลัพธ์จากการประมวลผลออกมาในรูปแบบต่าง ๆ ที่มนุษย์เข้าใจตัวอย่างของอุปกรณ์เอาต์พุตได้แก่ จอภาพ (Monitor) ลำโพง (Speaker) และเครื่องพิมพ์ (Printer) เป็นต้น

#### 3. หน่วยประมวลผลกลาง(Central Processing Unit)

หน่วยประมวลผลกลาง(Central Processing Unit) หรือเรียกว่า CPU มีหน้าที่เก็บข้อมูลคำสั่งทำการประมวลผลทางคณิตศาสตร์ เปรียบเทียบข้อมูล เมื่อข้อมูลเข้าสู่ระบบแล้วหน่วยประมวลผลจะทำหน้าที่ประมวลผลตามคำสั่ง หรือโปรแกรมที่กำหนดไว้ โดยโปรแกรมและข้อมูลต่าง ๆ จะถูกเก็บเอาไว้ในหน่วยความจำ เมื่อหน่วยประมวลผลทำงานสำเร็จแล้วจะเก็บข้อมูลลงหน่วยเก็บข้อมูลหรือส่งผลลัพธ์ที่ได้ออกจากหน่วยแสดงผลต่อไป หน่วยประมวลผลกลาง มีหน้าที่ 2 อย่างคือ

1. ทำหน้าที่ประสานการทำงานในระบบคอมพิวเตอร์
2. ทำหน้าที่ประมวลผลทางคณิตศาสตร์และตรรกะของข้อมูล

หน่วยประมวลผลกลางแบ่งหน่วยการทำงานออกเป็น 3 หน่วยหลัก คือ

1. หน่วยควบคุม(Control Unit) ทำหน้าที่แจกแจงงาน ติดตามงาน ควบคุมให้ส่วนอื่น ๆ ทำงานตามคำสั่งของผู้ใช้สั่งให้ทำ
2. หน่วยคำนวณและตรรกะ(Arithmetic and Logic Unit) ทำหน้าที่ทำการคำนวณ บวก ลบ คูณ หาร เปรียบเทียบค่าให้ตามคำสั่งของผู้ใช้สั่งให้ทำ นิยมเรียกย่อ ๆ ว่า ALU
3. หน่วยความจำหลัก(Main Memory หรือ Primary Storage) เป็นหน่วยความจำหลักที่อยู่ภายในตัวเครื่อง มีราคาแพงและเข้าถึงข้อมูลได้อย่างรวดเร็ว โดยภายในหน่วยความจำหลักจะประกอบไปด้วยตำแหน่งที่อยู่(Address) และข้อมูล(Content)

การวัดขนาดของหน่วยความจำหลัก จะวัดจากจำนวนข้อมูลที่เก็บโดยมีหน่วยของการวัดดังนี้

1 KB(Kilo Byte)=1024 Bytes

1 MB(Mega Byte) =1024 K Bytes

1 GB(Giga Byte) = 1024 M Bytes

1 TB(Tera Byte) = 1024 G Bytes

หน่วยความจำหลักแบ่งได้ 2 ชนิดคือ

1. ROM (Read Only Memory) เป็นหน่วยความจำหลักที่เก็บข้อมูลคำสั่งใด ๆ ไว้แล้วอ่านออกมาทำงานได้อย่างเดียวเท่านั้น ทำงานได้เร็ว มีราคาแพง ถ้าไฟดับหรือปิดเครื่องข้อมูลจะยังอยู่จึงนิยมใช้เก็บข้อมูลคำสั่งที่สำคัญ ๆ เช่น ตัวระบบดำเนินงาน (Operating System) เป็นต้น
2. RAM (Random Access Memory) เป็นหน่วยความจำหลักที่เก็บข้อมูลคำสั่ง ขณะที่โปรแกรมกำลังทำงานอยู่ ส่วนของหน่วยความจำนี้ถ้าไฟดับหรือปิดเครื่องข้อมูลที่เก็บอยู่จะหายไป
4. หน่วยความจำสำรอง (Auxiliary Memory หรือ Secondary storage) เป็นหน่วยความจำที่อยู่นอกเครื่องคอมพิวเตอร์ มีหน้าที่ช่วยให้หน่วยความจำหลักทำงานได้มากขึ้น โดยจะเก็บข้อมูลที่หรือการประมวลผล และข้อมูลที่ประมวลผลเสร็จแล้ว อุปกรณ์ที่นำมาใช้เป็นหน่วยความจำรอง เช่น Hard Disk Drive เป็นต้น

### 1.2.2 คอมพิวเตอร์ซอฟต์แวร์ (Software Computer)

เครื่องคอมพิวเตอร์จะทำงานได้จะต้องมีซอฟต์แวร์หรือโปรแกรมสำหรับการควบคุมการทำงานของเครื่องเพื่อประมวลผลตามที่ต้องการ

Software คือ ชุดคำสั่งที่มีไว้เพื่อทำงานอย่างใดอย่างหนึ่ง ซอฟต์แวร์สามารถแบ่งตามการทำงานได้ 2 ประเภท คือ ซอฟต์แวร์ระบบ (System Software) และซอฟต์แวร์ประยุกต์ (Application Software)

#### ซอฟต์แวร์ระบบ (System Software)

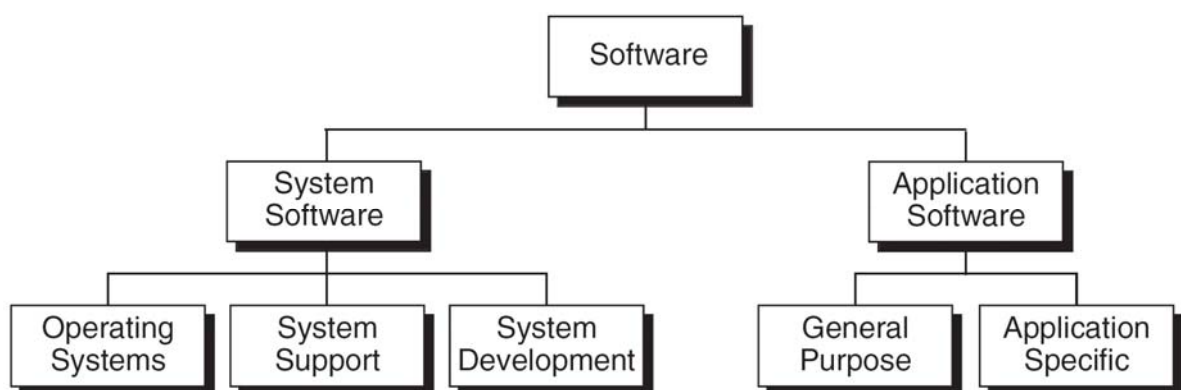
เป็นซอฟต์แวร์ที่ทำหน้าที่ควบคุมการทำงานของเครื่องเพื่อให้สามารถทำงานต่าง ๆ ได้สะดวก แบ่งออกเป็น

- โปรแกรมระบบปฏิบัติการ(OS: Operating System) เป็นโปรแกรมที่ออกแบบมาเพื่อทำงานร่วมกับตัวเครื่องคอมพิวเตอร์ เพื่อควบคุมการทำงานส่วนต่าง ๆ
- โปรแกรมอรรถประโยชน์ (Utilities Program) เป็นโปรแกรมที่ช่วยอำนวยความสะดวกในการใช้เครื่อง ใช้ปรับปรุงคอมพิวเตอร์ให้ทำงานได้ดีขึ้น เช่น ช่วยในการแบ็คอัพข้อมูล ช่วยในการตรวจสอบไวรัส เป็นต้น
- โปรแกรมไดรเวอร์ (Device Driver) เป็นโปรแกรมที่ออกแบบมาเพื่อฮาร์ดแวร์ต่าง ๆ ที่ใช้ในเครื่องคอมพิวเตอร์ เนื่องจากฮาร์ดแวร์บางประเภทออกแบบมาหลายรุ่น บางรุ่นมีความสามารถพิเศษซึ่งต้องมี Device Driver ช่วยในการจัดการต่าง ๆ

### ซอฟต์แวร์ประยุกต์ (Application Software)

เป็นโปรแกรมที่พัฒนาขึ้นสำหรับงานเฉพาะต่าง ๆ อาจเป็นโปรแกรมที่เขียนขึ้นเองหรือโปรแกรมที่มีอยู่ทั่วไป Application Software ผลิตขึ้นมาเพื่อให้ผู้ใช้ใช้งานเฉพาะทาง เช่น ซอฟต์แวร์ประมวลผลคำ ซอฟต์แวร์ตารางจัดการ ประเภทของโปรแกรมประยุกต์ที่มองเห็นทั่ว ๆ ไปมีดังนี้

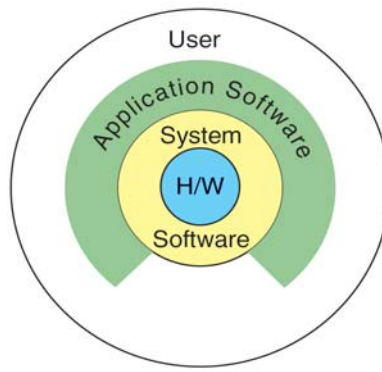
- ซอฟต์แวร์สำเร็จรูป (Package Software) เช่น ซอฟต์แวร์ประมวลผลคำ(Word Processing)
- ซอฟต์แวร์เฉพาะ(Custom Software) เป็นซอฟต์แวร์ที่เขียนขึ้นมาเฉพาะในงานด้านธุรกิจ
- ซอฟต์แวร์แบบเปิด (Open Source Software) ผู้ใช้สามารถแก้ไขเพิ่มเติมได้
- แร่แวร์ (Shareware) ผู้ใช้สามารถนำมาทดลองใช้ฟรีก่อน
- ซอฟต์แวร์ฟรี(Freeware) สามารถดาวน์โหลดจากอินเทอร์เน็ตหรือหามาใช้งานได้ฟรี



รูปที่ 1.3 ประเภทของซอฟต์แวร์ [1]

## 1.3 ความสัมพันธ์ระหว่างคอมพิวเตอร์ฮาร์ดแวร์และคอมพิวเตอร์ซอฟต์แวร์

คอมพิวเตอร์ฮาร์ดแวร์จะทำงานติดต่อกับผู้ใช้ได้จะต้องผ่านโปรแกรมระบบ เป็นผู้ประสานงานให้ โดยผู้ใช้จะต้องเขียนโปรแกรมประยุกต์ขึ้นมาเพื่อสั่งให้คอมพิวเตอร์ทำงานอย่างหนึ่งอย่างใด สามารถอธิบายได้โดยแผนภาพดังต่อไปนี้



รูปที่ 1.4 ความสัมพันธ์ระหว่างฮาร์ดแวร์และซอฟต์แวร์ [1]

## 1.4 ภาษาคอมพิวเตอร์(Computer Language)

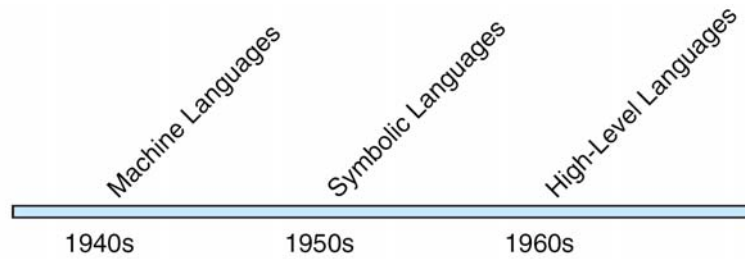
ภาษาคอมพิวเตอร์ เป็นสัญลักษณ์ที่ถูกสร้างขึ้นสำหรับควบคุมการทำงานของอุปกรณ์ต่าง ๆ และสั่งการให้คอมพิวเตอร์ สามารถทำงาน ได้ตามที่มนุษย์ต้องการ เริ่มแรกนั้นการสั่งให้คอมพิวเตอร์ทำงาน ต้องเขียนคำสั่งอยู่ในรูปของเลขฐานสอง ซึ่งประกอบด้วยเลข 0 และ 1 จึงทำให้ การเขียนโปรแกรมควบคุมการทำงานของคอมพิวเตอร์ยุ่งยากมาก เพราะถ้าเข้ารหัสผิดพลาด การทำงานของคอมพิวเตอร์ ก็จะผิดพลาด หรือได้ผลลัพธ์ไม่ตรงตามจุดประสงค์ที่ต้องการ ต่อมามนุษย์จึงพัฒนารูปแบบของภาษาขึ้นมาใหม่ โดยใช้รหัสข้อความในภาษาอังกฤษที่เข้าใจได้ง่าย โดยมีกฎเกณฑ์ต่าง ๆ ของภาษาแตกต่างกันไป ภาษาคอมพิวเตอร์ปัจจุบันมีหลายภาษามากมาย แต่ละภาษาก็มีความยากง่าย และมีวัตถุประสงค์ ของภาษาแตกต่างกันไป ดังนั้นผู้พัฒนาซอฟต์แวร์ จึงจำเป็นต้องเลือกใช้ภาษาคอมพิวเตอร์ให้เหมาะสมกับงาน ที่ต้องการพัฒนา และความสามารถ ในการใช้ภาษาคอมพิวเตอร์ของบุคคลนั้น ๆ

### 1.4.1 ยุคของภาษาคอมพิวเตอร์

ภาษาคอมพิวเตอร์มีการพัฒนาหรือมีวิวัฒนาการมาโดยลำดับเช่นเดียวกับคอมพิวเตอร์ โดยจะสามารถแบ่งออกเป็นยุค หรือเป็นรุ่นของภาษา (Generation) ซึ่งในยุคหลัง ๆ จะมีการพัฒนาภาษาให้มีความสะดวก ในการอ่าน และเขียนง่ายขึ้นกว่าภาษาในยุคแรก ๆ เนื่องจากจะมีโครงสร้างภาษาใกล้เคียงกับภาษาอังกฤษ สามารถแบ่งภาษาคอมพิวเตอร์ออกได้เป็น 5 ยุค ดังนี้

1. ภาษาเครื่อง (machine language)
2. ภาษาแอสเซมบลี (Assembly Language)
3. ภาษาชั้นสูง (High level Language)
4. ภาษาชั้นสูงมาก (Very High level Language)
5. ภาษาธรรมชาติ (National Language)

หรือจะสรุปภาษาคอมพิวเตอร์ออกเป็น 3 กลุ่มหลัก ๆ ดังรูปต่อไปนี้



รูปที่ 1.5 พัฒนาการของภาษาคอมพิวเตอร์ [1]

## 1.ภาษาเครื่อง (Machine Language)

เป็นภาษาที่เครื่องสามารถเข้าใจคำสั่งได้เลย มีลักษณะคำสั่งเป็นตัวเลขล้วน เป็นภาษาคอมพิวเตอร์ระดับต่ำที่สุด เพราะเป็นตัวเลขฐานสอง แทนข้อมูลและคำสั่งต่างๆ ทั้งหมด แสดงดังรูปที่ 1.6

1	00000000	00000100	0000000000000000
2	01011110	00001100	11000010 0000000000000010
3		11101111	00010110 00000000000000101
4		11101111	10011110 00000000000001011
5	11111000	10101101	11011111 0000000000010010
6		01100010	11011111 0000000000010101
7	11101111	00000010	11111011 0000000000010111
8	11110100	10101101	11011111 0000000000011110
9	00000011	10100010	11011111 000000000100001
10	11101111	00000010	11111011 000000000100100
11	01111110	11110100	10101101
12	11111000	10101110	11000101 000000000101011
13	00000110	10100010	11111011 0000000000110001
14	11101111	00000010	11111011 000000000110100
15		01010000	11010100 000000000111011
16			00000100 000000000111101

รูปที่ 1.6 ภาษาเครื่อง [1]

## 2 ภาษาระดับต่ำ(Low level Language)

เป็นภาษาที่มีลักษณะใกล้เคียงกับภาษาเครื่อง เพียงแต่มีการใช้สัญลักษณ์หรือตัวอักษรมาแทนคำสั่งในส่วนต่าง ๆ ตัวอย่างของภาษานี้ได้แก่ ภาษาแอสเซมบลี(Assembly) เป็นภาษาคอมพิวเตอร์ที่พัฒนาขึ้นมาเพื่อให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมติดต่อกับคอมพิวเตอร์ได้ง่ายกว่าภาษาเครื่อง โดยใช้คำย่อภาษาอังกฤษในการเขียนคำสั่ง แสดงดังรูปที่ 1.7

1	entry	main, ^m<r2>
2	subl2	#12, sp
3	jsb	C\$MAIN_ARGS
4	movab	\$CHAR_STRING_CON
5		
6	pushal	-8(fp)
7	pushal	(r2)
8	calls	#2, SCANF
9	pushal	-12(fp)
10	pushal	3(r2)
11	calls	#2, SCANF
12	mull3	-8(fp), -12(fp), -
13	pusha	6(r2)
14	calls	#2, PRINTF
15	clrl	r0
16	ret	

### รูปที่ 1.7 ภาษาแอสเซมบลี [1]

#### 3 ภาษาระดับสูง(High level Language)

เป็นภาษาที่มีลักษณะใกล้เคียงกับภาษาอังกฤษที่มนุษย์ใช้กันอยู่ เรียนรู้่าย เข้าใจได้ง่าย สะดวกในการใช้งาน และใช้ได้กับทุกเครื่อง ตัวอย่างของภาษาระดับสูงได้แก่ ภาษาโคบอล(COBOL) ภาษาปาสคาล (PASCAL) ภาษาซี(C) ภาษาจาวา (JAVA) เป็นต้น แสดงดังรูปที่ 1.8

```
1  /* This program reads two integers from the keyboard
2     and prints their product.
3     Written by:
4     Date:
5  */
6  #include <stdio.h>
7
8  int main (void)
9  {
10 // Local Definitions
11     int number1;
12     int number2;
13     int result;
14
15 // Statements
16     scanf ("%d", &number1);
17     scanf ("%d", &number2);
18     result = number1 * number2;
19     printf ("%d", result);
20     return 0;
21 } // main
```

### รูปที่ 1.8 ภาษาซี [1]

## 1.5 ขั้นตอนการพัฒนาโปรแกรม

ขั้นตอนการพัฒนาโปรแกรมแบ่งเป็นขั้นตอนต่าง ๆ ดังนี้

1. การวิเคราะห์ปัญหา (Problem Analysis)
2. เขียนผังงาน (Pseudo Coding)
3. เขียนโปรแกรม (Programming)
4. ทดสอบและแก้ไขโปรแกรม (Program testing and Debugging)
5. ทำเอกสารและบำรุงรักษาโปรแกรม (Program document and Maintenance)

1.5.1 การวิเคราะห์ปัญหา เป็นขั้นตอนแรกสุดที่นักเขียนโปรแกรมต้องทำ มีขั้นตอนย่อย ๆ ดังนี้

1 กำหนดขอบเขตของปัญหา โดยการกำหนดให้ชัดเจนว่าจะทำ งานอะไร ตัวแปรค่าคงที่ที่ต้องใช้มีลักษณะใด

2. กำหนดลักษณะของข้อมูลเข้าและออกจากระบบ (Input/output Specification) กำหนดว่าข้อมูลที่จะส่งเข้าไปเป็นอย่างไร และต้องการแสดงผลอะไรบ้าง โดยต้องคำนึงถึงผู้ใช้งานโปรแกรมนั้นเป็นหลัก

3. กำหนดวิธีการประมวลผล (Process Specification)

เราสามารถวิเคราะห์ปัญหาโดยการวิเคราะห์ส่วนต่าง ๆ ดังนี้

1. วิเคราะห์ข้อมูลนำเข้า (Input Analysis)



2. วิเคราะห์ขั้นตอนการทำงาน (Process Analysis)

3. วิเคราะห์ผลลัพธ์ (Output Analysis)

### ตัวอย่างการวิเคราะห์ปัญหา#1

จงเขียนแนวทางการแก้ปัญหาด้วยคอมพิวเตอร์สำหรับให้คอมพิวเตอร์คำนวณค่าจ้างพนักงานเป็นรายชั่วโมง จากนั้นแสดงค่าจ้างที่คำนวณได้

- **ต้องการอะไร?** ต้องการทราบค่าจ้างของพนักงานแต่ละคน
- **ต้องการผลลัพธ์อย่างไร(Output)** ต้องการผลลัพธ์เป็นค่าจ้างสุทธิของพนักงานทางจอภาพ
- **ข้อมูลเข้า(Input)** รหัสพนักงาน ชื่อพนักงาน จำนวนชั่วโมงทำงานเก็บในตัวแปรชื่อ Hours ค่าจ้างรายชั่วโมงเก็บในตัวแปรชื่อ PayRate
- **วิธีการประมวลผล(Process)**
- กำหนดวิธีการคำนวณ
  - $\text{ค่าจ้างสุทธิ} = \text{จำนวนชั่วโมง} \times \text{อัตราต่อชั่วโมง}$
- ขั้นตอนการประมวลผล
  - 1. เริ่มต้น
  - 2. รับรหัสพนักงาน ชื่อพนักงาน จำนวนชั่วโมงทำงานเก็บในตัวแปรชื่อ Hours ค่าจ้างรายชั่วโมงเก็บในตัวแปรชื่อ PayRate
  - 3. คำนวณ ค่าจ้างสุทธิ = Hours x PayRate
  - 4. แสดงผลลัพธ์เป็นรหัสพนักงาน ชื่อ ค่าจ้างสุทธิของพนักงานทางจอภาพ
  - 5. จบการทำงาน

### ตัวอย่างการวิเคราะห์ปัญหา#2

จงเขียนโปรแกรมเพื่อรายงานผลสอบของนักศึกษาวิชาคอมพิวเตอร์ โดยให้แสดงคะแนนรวมและเกรดออกมา

- **ต้องการอะไร?** ต้องการพิมพ์คะแนนผลสอบและเกรดของนักศึกษา
- **ต้องการผลลัพธ์อย่างไร(Output)** ต้องการผลลัพธ์เป็นคะแนนรวมและเกรดของนักศึกษาแต่ละคน
- **ข้อมูลเข้า(Input)** รหัสประจำตัวนักศึกษา(ID) ชื่อนักศึกษา(name) คะแนนสอบกลางภาค(mid) คะแนนสอบย่อย(test) คะแนนสอบปลายภาค(final)
- **วิธีการประมวลผล(Process)**
- กำหนดวิธีการคำนวณ
  - $\text{คะแนนรวม} = \text{คะแนนกลางภาค} + \text{คะแนนสอบย่อย} + \text{คะแนนปลายภาค}$
  - ถ้าคะแนนรวม  $\geq 80$  ได้เกรด “A”
  - ถ้าคะแนนรวม  $\geq 70$  และ  $< 80$  ได้เกรด “B”
  - ถ้าคะแนนรวม  $\geq 60$  และ  $< 70$  ได้เกรด “C”
  - ถ้าคะแนนรวม  $\geq 50$  และ  $< 60$  ได้เกรด “D”



- ถ้าคะแนนรวม < 50 ได้เกรด “F”
- ขั้นตอนการประมวลผล
  - 1. เริ่มต้น
  - 2. รับค่าตัวแปร ID name mid test final
  - 3. คำนวณคะแนนรวมและเกรด
    - Total = mid + test + final
    - ถ้า Total  $\geq 80$ , Grade = “A”
    - ถ้า Total  $\geq 70$  และ  $< 80$ , Grade = “B”
    - ถ้า Total  $\geq 60$  และ  $< 70$ , Grade = “C”
    - ถ้า Total  $\geq 50$  และ  $< 60$ , Grade = “D”
    - ถ้า Total  $< 50$ , Grade = “F”
  - 4. แสดง ID name Total Grade ของนักศึกษา
  - 5. กลับไปข้อ 2 เพื่อรับจนครบทุกคน ถ้าครบแล้วไปข้อ 6
  - 6. หยุดการทำงาน

#### 1.5.2 การเขียนผังงาน

เมื่อวิเคราะห์ปัญหาที่จะทำเรียบร้อยแล้ว ขั้นตอนต่อมาจะเป็นการเขียนผังงาน โดยใช้เครื่องมือ ในการออกแบบ ซึ่งยังไม่ได้เขียนเป็นโปรแกรมจริง ๆ โดยลำดับขั้นตอนของการทำงาน โปรแกรม เราเรียกว่าอัลกอริทึม (Algorithm) โดยจะถูกเขียนอยู่ในรูปของ ซูโดโค้ด (Pseudo Code) หรือ เขียนเป็นผังงาน (Flowchart) โดยแต่ละส่วนจะเป็นแนวทางในการเขียนโปรแกรม ในขั้นตอนต่อไปได้ง่ายขึ้น

#### 1.5.3 การเขียนโปรแกรม

เป็นขั้นตอนของการเขียนโปรแกรม เพื่อให้คอมพิวเตอร์สามารถประมวลผลได้ การเขียนโปรแกรมจะต้องเขียนตามภาษาที่คอมพิวเตอร์เข้าใจ โดยจะใช้ภาษาระดับใดก็ได้ ซึ่งจะต้องเขียนให้ถูกต้องตามหลักไวยากรณ์ (Syntax) ของภาษานั้น ๆ

#### 1.5.4 การทดสอบและแก้ไขโปรแกรม

หลังจากการเขียนโปรแกรมจะต้องทดสอบความถูกต้องของโปรแกรมที่เขียนขึ้นว่ามีข้อผิดพลาดหรือไม่ ซึ่งเรียกว่า ดีบั๊ก (Debug) ซึ่งโดยทั่วไปข้อผิดพลาด (Bug) มี 2 ประเภท คือ

1. Syntax error คือ การเขียนคำสั่งไม่ถูกต้องตามหลักการเขียนโปรแกรมของภาษานั้น ๆ กรณีที่เกิด Syntax error โปรแกรมจะไม่สามารถทำงานได้
2. Logic error เป็นข้อผิดพลาดทางตรรกะ โปรแกรมสามารถทำงานได้แต่ผลลัพธ์จะไม่ถูกต้อง

#### 1.5.5 การทำเอกสารและบำรุงรักษาโปรแกรม

การทำเอกสารประกอบโปรแกรม จะทำให้ผู้ใช้สามารถใช้งานโปรแกรมได้อย่างมีประสิทธิภาพ และสะดวกในการตรวจสอบข้อผิดพลาด โดยทั่วไปจะแบ่งเป็น 2 ประเภทคือ

1. คู่มือการใช้ หรือ User Document หรือ User Guide ซึ่งจะเป็นส่วนที่อธิบายการใช้โปรแกรม
2. คู่มือโปรแกรมเมอร์ หรือ Program Document หรือ Technical Document ซึ่งจะทำให้มีความสะดวกในการแก้ไข และพัฒนาโปรแกรมต่อไปในอนาคต

## 1.6 การเขียนผังงานของโปรแกรม

ก่อนที่จะมีการเขียนโปรแกรมจะต้องมีการวางแผนขั้นตอนการทำงานมาก่อน ผังงานหรือ Flowchart เป็นอีกรูปแบบหนึ่งที่สามารถนำมาช่วยในการพัฒนาโปรแกรมคอมพิวเตอร์ได้ โดยจะเป็นการอธิบายขั้นตอนวิธีการทำงานในลักษณะของรูปภาพ ทำให้สามารถนำผังมาช่วยในลำดับขั้นตอนวิธีการแก้ปัญหาได้อย่างชัดเจน เมื่อได้ผังงานที่ใช้สำหรับแก้ปัญหาแล้ว ขั้นตอนต่อไปก็จะเป็นการเขียนขั้นตอนการทำงานในลักษณะของข้อความ และพัฒนาเป็นโปรแกรมภาษาคอมพิวเตอร์ต่อไปตามต้องการ

### 1.6.1 ความหมายของผังงาน

ผังงานเป็นแผนภาพที่ใช้รูปแบบและอธิบายการทำงานของโปรแกรมโดยอาศัยรูปทรงต่างๆ ควบคุมไปกับลูกศร แต่ละรูปภาพจะแสดงการทำงานหนึ่งขั้นตอน ส่วนลูกศรจะแสดงลำดับการทำงานขั้นตอนต่าง ๆ รวมทั้งทิศทางการไหลของข้อมูลตั้งแต่เริ่มต้นจนได้ผลลัพธ์

ผังงานในการทำงานของคอมพิวเตอร์สามารถแบ่งได้ 2 ประเภทคือ ผังงานระบบ และผังงานโปรแกรม

#### ผังงานระบบ (System Flowchart)

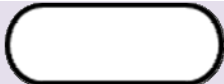






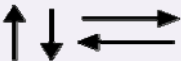
เป็นแผนภาพที่แสดงขอบเขตและลำดับขั้นตอนการทำงานของระบบโดยรวม โดยจะแสดงขั้นตอนการทำงานภายในระบบหนึ่ง ๆ และแสดงภาพกว้างถึงองค์ประกอบที่มีอยู่ในระบบทั้งหมด เช่น เอกสารเบื้องต้น วัสดุที่ใช้ ใช้หน่วยความจำประเภทใด จะต้องส่งผ่านไปหน่วยใด วิธีการแสดงผลและผลลัพธ์เป็นอย่างไร แต่จะไม่มุ่งเน้นรายละเอียดในการปฏิบัติ ไม่สามารถนำมาเขียนเป็นโปรแกรมได้ทันที

#### ผังงานโปรแกรม(Program Flowchart)

เป็นแผนภาพที่แสดงลำดับขั้นตอนในการทำงานของโปรแกรม โดยจะแยกย่อยมาจากผังงานระบบโดยมีการลงรายละเอียด ใส่วิธีการ และจัดลำดับขั้นตอนของโปรแกรม ตั้งแต่เริ่มต้นจากการรับข้อมูล การประมวลผล ไปจนถึงการแสดงผลการทำงาน

การเขียนผังงานที่ดี

1. เขียนตามสัญลักษณ์ที่กำหนด
2. ใช้ลูกศรแสดงทิศทางการทำงานจากบนลงล่าง
3. อธิบายสั้น ๆ ให้เข้าใจง่าย
4. ทุกแผนภาพต้องมีทิศทางเข้าออก
5. ไม่ควรโยงลูกศรไปที่ไกล ๆ มาก ถ้าต้องทำให้ใช้สัญลักษณ์ของการเชื่อมต่อแทน

สัญลักษณ์	ความหมาย
	จุดเริ่มต้น หรือ สิ้นสุด
	รับข้อมูล (Input) แสดงข้อมูล (Output)
	การคำนวณ (Process)
	การตัดสินใจ (Decision) การเปรียบเทียบ (Compare)
	การแสดงออกทางเครื่องพิมพ์ (Printer)
	การทำงานย่อย (SubProgram)
	จุดเชื่อมต่อ (Connection)
	ทิศทาง (Flow)

รูปที่ 1.9 รูปแสดงสัญลักษณ์ที่ใช้ในการเขียนผังงาน

## 1.7 การเขียนอัลกอริทึมของโปรแกรม

อัลกอริทึม(Algorithms) หมายถึงลำดับขั้นตอนเชิงคำนวณที่แปลงข้อมูลด้านอินพุตของปัญหาไปเป็นผลลัพธ์ที่ต้องการ ขั้นตอนต่าง ๆ ในอัลกอริทึมสามารถเปลี่ยนไปเป็นคำสั่งที่ให้คอมพิวเตอร์ทำงานได้ ถ้าหากทำตามอัลกอริทึมแล้ว ปัญหาจะต้องถูกแก้ได้สำเร็จและได้คำตอบที่ถูกต้องสำหรับทุกกรณีตามที่กำหนดในอัลกอริทึม ดังนั้นเราจะไม่ยอมรับอัลกอริทึมที่ทำงานติดอยู่ใน Loop ไม่มีที่สิ้นสุดหรืออัลกอริทึมที่ทำงานแล้วได้คำตอบถูกบ้างผิดบ้าง

จุดประสงค์ของการออกแบบอัลกอริทึมสำหรับแก้ปัญหาหนึ่ง ๆ คือการทำงานที่ถูกต้องและทำงานได้อย่างมีประสิทธิภาพ เราสามารถบรรยายอัลกอริทึมด้วยคำบรรยายสั้นๆ ที่ได้ใจความโดยเขียนรหัสจำลองหรือ Pseudo code ซึ่งคล้ายกับโปรแกรมภาษาคอมพิวเตอร์แต่จะบรรยายด้วยคำที่ง่ายกว่า

Pseudo code เป็นคำอธิบายขั้นตอนการทำงานของโปรแกรมโดยการผสมระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้างที่เข้าใจง่าย มาแสดงลำดับการทำงานของโปรแกรม โดยให้ผู้เขียนโปรแกรมสามารถพัฒนาขั้นตอนโปรแกรมต่าง ๆ ให้เป็นโปรแกรมได้ง่ายขึ้น ส่วนใหญ่แล้วคำที่ใช้มักเป็นคำเฉพาะ

Pseudo code ที่ดีจะต้องมีความชัดเจน สั้น และได้ใจความ ข้อมูลต่าง ๆ ที่ใช้ จะถูกเขียนอยู่ในรูปของตัวแปรบางครั้งเรียกอัลกอริทึม รูปแบบทั่วไปคือ

Algorithms <ชื่ออัลกอริทึม>

1. ...

2. ...

3. ...

...

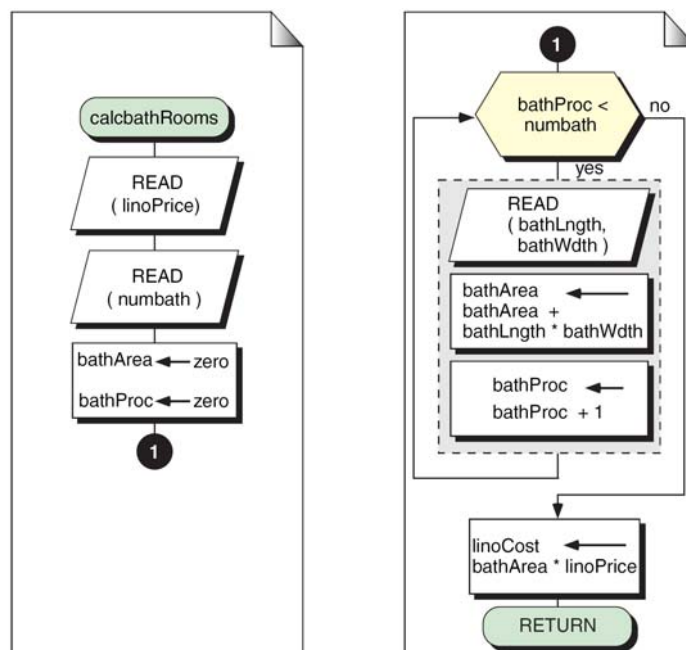
END

ตัวอย่างการเขียนอัลกอริทึม

#### Pseudocode for Calculate Bathrooms

```
Algorithm Calculate BathRooms
1 prompt user and read linoleum price
2 prompt user and read number of bathrooms
3 set total bath area and baths processed to zero
4 while ( baths processed < number of bathrooms )
  1 prompt user and read bath length and width
  2 total bath area =
  3     total bath area + bath length * bath width
  4 add 1 to baths processed
5 bath cost = total bath area * linoleum price
6 return bath cost
end Algorithm Calculate BathRooms
```

รูปที่ 1.10 ตัวอย่างการเขียนอัลกอริทึม [1]



รูปที่ 1.11 ตัวอย่างการเขียน Flowchart [1]

ตัวอย่างการเขียนอัลกอริทึมของโปรแกรม#1

- จงวิเคราะห์ปัญหาและเขียนอัลกอริทึมสำหรับหาค่าเฉลี่ยของอุณหภูมิประจำวันโดยรับค่าอุณหภูมิสูงสุดและอุณหภูมิต่ำสุดเป็นเลขจำนวนเต็มเข้าไปและให้แสดงค่าอุณหภูมิเฉลี่ยออกจากจอภาพ
- ต้องการอะไร? ต้องการทราบค่าเฉลี่ยของอุณหภูมิประจำวัน

- ต้องการผลลัพธ์อย่างไร(Output) ต้องการผลลัพธ์เป็นค่าเฉลี่ยของอุณหภูมิประจำวันโดยใช้ตัวแปรชื่อ avg\_temp
- ข้อมูลเข้า(Input) รับค่าอุณหภูมิสูงสุดอยู่ในตัวแปรชื่อ max\_temp  
อุณหภูมิต่ำสุดอยู่ในตัวแปรชื่อ min\_temp
- วิธีการประมวลผล(Process)
  - 1. รับค่าอุณหภูมิสูงสุดและอุณหภูมิต่ำสุด
  - 2. หาค่าเฉลี่ยโดยใช้  $\text{avg\_temp} = (\text{max\_temp} + \text{min\_temp}) / 2$
  - 3. แสดงค่า avg\_temp ทางจอภาพ
- อัลกอริทึม(Algorithms)

Algorithms Find\_avg\_temperature

1. READ max\_temp, min\_temp
2.  $\text{avg\_temp} = (\text{max\_temp} + \text{min\_temp}) / 2$
3. Output avg\_temp to the screen

End

## 1.8 ตัวอย่างการวิเคราะห์ปัญหาและการเขียนโปรแกรมเพื่อช่วยแก้ปัญหา

ตัวอย่าง จงเขียนผังงานและโปรแกรมเพื่อรับข้อมูลตัวเลขจำนวนจริง ความยาวฐาน (base) และความสูง (height) ของรูปสามเหลี่ยม แล้วให้ทำการคำนวณพื้นที่และแสดงผลในรูปแบบต่อไปนี้

Enter base value:      **10**      (กดแป้น Enter)

Enter height value:      **5**      (กดแป้น Enter)

Area is: 25.000

### วิเคราะห์ปัญหา

ข้อมูลนำเข้า ความยาวฐาน และความสูง

แสดงผล พื้นที่

กำหนดตัวแปร

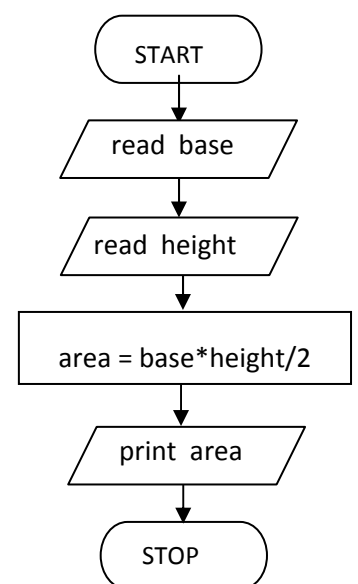
ชื่อตัวแปร      ความหมาย

base      ความยาวฐานของรูปสามเหลี่ยม

height      ความสูงของรูปสามเหลี่ยม

area      พื้นที่ของรูปสามเหลี่ยม

### เขียนผังงาน



### เขียนโปรแกรม

```

/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     float base, height, area;
/* 6 */     printf("Enter base value: ");           /* prompt to input base */
/* 7 */     scanf("%f", &base);                     /* input base */
/* 8 */     printf("Enter height value: ");          /* prompt to input height */
/* 9 */     scanf("%f", &height);                     /* input height */
/* 10 */     area = base*height/2;                   /* compute area */
/* 11 */     printf("Area = %7.2f\n", area);          /* display result */
/* 12 */     system("PAUSE");
/* 13 */     return 0;
/* 14 */ }

```

**ตัวอย่าง** จงเขียนผังงานและโปรแกรมเพื่อหาพื้นที่ (area) ของวงกลมวงหนึ่งเมื่อรับค่ารัศมี (r) และเปรียบเทียบขนาดของพื้นที่เพื่อแสดงผลที่ได้ ถ้าพื้นที่มีค่าตั้งแต่ ศูนย์ถึง 300 ตารางหน่วย ให้พิมพ์ ค่าพื้นที่นั้น และต่อด้วยคำว่า “small” ถ้าพื้นที่มีค่ามากกว่า 300 ตารางหน่วย ให้พิมพ์ ค่าพื้นที่นั้น และต่อด้วยคำว่า “large”

#### วิเคราะห์ปัญหา

ข้อมูลนำเข้า    รัศมีวงกลม

แสดงผล        พื้นที่วงกลม

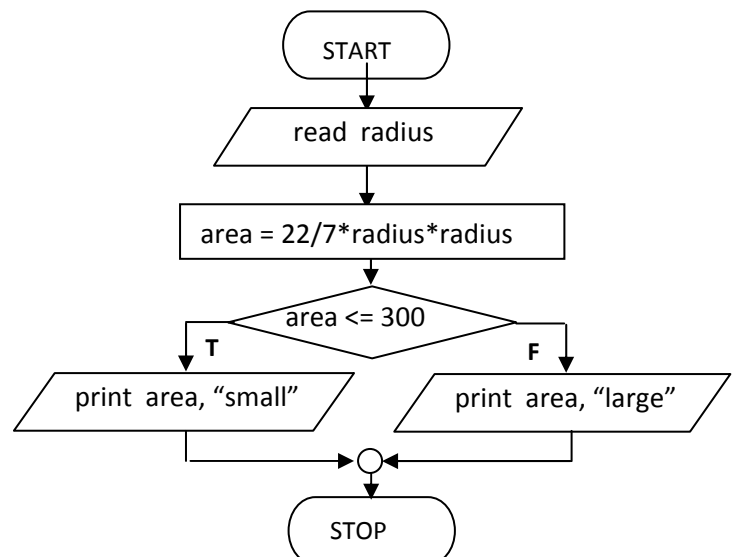
กำหนดตัวแปร

ชื่อตัวแปร    ความหมาย

radius        รัศมีวงกลม

area         พื้นที่วงกลม

#### เขียนผังงาน



#### เขียนโปรแกรม

```

/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     float radius, area;
/* 6 */     printf("Please input radius : ");
/* 7 */     scanf("%f ", &radius);
/* 8 */     area = 22/7*radius*radius;
/* 9 */     if (area <= 300)
/* 10 */         printf("%f small\n", area) ;
/* 11 */     else
/* 12 */         printf("%f large\n", area);
/* 13 */     system("PAUSE");
/* 14 */     return 0 ;
/* 15 */ }

```

## แบบฝึกหัดปฏิบัติการคาบที่ 1: การพัฒนาโปรแกรม

1. จงเขียนผังงานและโปรแกรมแสดงคะแนนสอบวิชาคอมพิวเตอร์เบื้องต้นของนักเรียนสามคน โดย นายไผ่สอบได้ 60 คะแนน นายภูรส์สอบได้น้อยกว่านายไผ่ 10 % ส่วนนางสาวปันปัน สอบได้เป็นค่ากึ่งกลางของคนทั้งสอง

### วิเคราะห์ปัญหา

### เขียนผังงาน

ข้อมูลนำเข้า - ไม่มี -

แสดงผล คะแนนของนางสาวปันปัน

กำหนดตัวแปร

ชื่อตัวแปร                      ความหมาย

pai                      คะแนนของนายไผ่

phu                      คะแนนของนายภูรส์

panpan                      คะแนนของนางสาวปันปัน

### เขียนโปรแกรม

```
/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     float pai, phu, panpan ;    /* data declaration */
/* 6 */     pai = 60;                  /* process */
/* 7 */     phu = pai - pai *10/100;
/* 8 */     panpan = (pai + phu )/2;
/* 9 */     printf("pai = %6.2f, phu = %6.2f, panpan = %6.2f\n", pai, phu, panpan );
/* 10 */     system("PAUSE");
/* 11 */     return 0;
/* 12 */ }
```

1.1 ผลลัพธ์ของโปรแกรมคือ	..... .....
1.2 ถ้าเปลี่ยนบรรทัดที่ 9 เป็น printf("pai = %6.2f, \n phu = %6.2f, \n panpan = %6.2f\n", pai, phu , panpan); จะได้ผลลัพธ์คือ	..... .....



<p>1.3 ถ้ากำหนดปัญหาเป็น “นายภูริสอบได้มากกว่านายไผ่ 10 % ส่วนนางสาวบันปันสอบได้เป็นค่ากึ่งกลางของคนทั้งสอง” จะต้องแก้ไขโปรแกรมบรรทัดใด เป็นอย่างไร</p>	<p>.....</p> <p>.....</p>
---	---------------------------

2. จงเขียนผังงานและโปรแกรมเพื่อรับข้อมูลตัวเลขจำนวนจริงความยาวฐาน (base) และความสูง (height) ของรูปสามเหลี่ยม แล้วให้ทำการคำนวณพื้นที่และแสดงผลในรูปแบบต่อไปนี้

Enter base value:      **10**      (กดแป้น Enter)

Enter height value:      **5**      (กดแป้น Enter)

Area is : 25.000

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า    ความยาวฐาน และความสูง

แสดงผล      พื้นที่

กำหนดตัวแปร

ชื่อตัวแปร      ความหมาย

base    ความยาวฐานของรูปสามเหลี่ยม

height    ความสูงของรูปสามเหลี่ยม

area    พื้นที่ของรูปสามเหลี่ยม

#### เขียนโปรแกรม

```

/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     float base, height, area;
/* 6 */     printf("Enter base value: ");      /* prompt to input base */
/* 7 */     scanf("%f", &base);              /* input base */
/* 8 */     printf("Enter height value: ");   /* prompt to input height */
/* 9 */     scanf("%f", &height);            /* input height */
/* 10 */    area = base*height/2;             /* compute area */
/* 11 */    printf("Area = %.2f\n", area);    /* display result */
/* 12 */    system("PAUSE");
/* 13 */    return 0;
/* 14 */ }

```

2.1 ถ้ารันโดยใช้ข้อมูล  $\text{base} = 15, \text{height} = 10$  ผลลัพธ์ของโปรแกรมคือ	.....
2.2 ถ้าเปลี่ยนบรรทัดที่ 10 เป็น $\text{area} = 1/2 * \text{base} * \text{height};$ และรันโดยใช้ข้อมูล $\text{base} = 15, \text{height} = 10$ ผลลัพธ์ของโปรแกรมคือ	.....
2.3 ถ้ากำหนด <b>base</b> และ <b>height</b> เป็นความยาว และความสูงของรูปสี่เหลี่ยมผืนผ้า และต้องการ คำนวณพื้นที่สี่เหลี่ยมผืนผ้านี้ จะต้องแก้ไขโปรแกรม บรรทัดใด เป็นอย่างไร	.....

3. จงเขียนผังงานและโปรแกรมเพื่อรับข้อมูล ชื่อ (สายอักขระ) และ ส่วนสูง (จำนวนจริง) ของนักเรียนสองคน จากนั้นให้แสดงผลว่า นักเรียนคนแรกสูงกว่าคนที่สองเท่าไร ตามตัวอย่างต่อไปนี้

Please enter name and height of the first student: **Panya 160** (กดแป้น Enter)

Please enter name and height of the second student: **Triphop 170** (กดแป้น Enter)

Panya is taller than Triphop = -10.00

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า ชื่อ (สายอักขระ) และ ส่วนสูง (จำนวนจริง) ของนักเรียนสองคน

แสดงผล นักเรียนคนแรกสูงกว่าคนที่สองเท่าไร

กำหนดตัวแปร

ชื่อตัวแปร                      ความหมาย

name1    ชื่อของนักเรียนคนแรก

ht1        ส่วนสูงของนักเรียนคนแรก

name2    ชื่อของนักเรียนคนที่สอง

ht2        ส่วนสูงของนักเรียนคนที่สอง

## เขียนโปรแกรม

```
/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     char name1[10], name2[10];          /* data declaration */
/* 6 */     float ht1, ht2;
/* 7 */     printf("Please enter name and height of the first student: ");
/* 8 */     /* prompt to input name and height */
/* 9 */     scanf("%s %f", name1, &ht1);          /* input name and height */
/* 10 */     printf("Please enter name and height of the second student: ");
/* 11 */     /* prompt to input name and
/* 12 */     height */
/* 13 */     scanf("%s %f", name2, &ht2);          /* input name and height */
/* 14 */
/* 15 */     printf("%s is taller than %s = %7.2f\n", name1, name2, ht1-ht2);
/* 16 */     system("PAUSE");
/* 17 */     return 0;
/* 18 */ }
```

3.1 ถ้ารันโปรแกรมโดยใช้ข้อมูลต่อไปนี้ ผลลัพธ์ของโปรแกรมคือ

Panya 160 ↵ และ

Triphop 170 ↵ .....

3.2 ถ้ารันโปรแกรมโดยใช้ข้อมูลต่อไปนี้ ผลลัพธ์ของโปรแกรมคือ

Por 172 ↵ และ

Film 165.5 ↵ .....

4. จงเขียนผังงานและโปรแกรมเพื่อหาพื้นที่ (area) ของวงกลมวงหนึ่งเมื่อรับค่ารัศมี (r) และเปรียบเทียบขนาดของพื้นที่เพื่อแสดงผลที่ได้ ถ้าพื้นที่มีค่าตั้งแต่ ศูนย์ถึง 300 ตารางหน่วย ให้พิมพ์ ค่าพื้นที่นั้น และต่อด้วย คำว่า "small" ถ้าพื้นที่มีค่ามากกว่า 300 ตารางหน่วย ให้พิมพ์ ค่าพื้นที่นั้น และต่อด้วยคำว่า "large"

## วิเคราะห์ปัญหา

## เขียนผังงาน

ข้อมูลนำเข้า    รัศมีวงกลม

แสดงผล        พื้นที่วงกลม

กำหนดตัวแปร

ชื่อตัวแปร    ความหมาย

radius        รัศมีวงกลม

area        พื้นที่วงกลม

## เขียนโปรแกรม

```

/* 1 */ #include <stdio.h>
/* 2 */ #include <stdlib.h>
/* 3 */ int main()
/* 4 */ {
/* 5 */     float radius, area;
/* 6 */     printf("Please input radius : ");
/* 7 */     scanf("%f ", &radius);
/* 8 */     area = 22/7*radius*radius;
/* 9 */     if (area <= 300) printf("%f small\n", area) ;
/* 10 */     else printf("%f large\n", area);
/* 11 */     system("PAUSE");
/* 12 */     return 0 ;
/* 13 */ }

```

4.1 รันโปรแกรมโดยใช้ข้อมูล 25.5 ↵ ผลลัพธ์ของโปรแกรมคือ	.....
4.2 ถ้าเปลี่ยนบรรทัดที่ 8 เป็น <b>area = 22.0/7*pow(radius,2) ;</b> รันโปรแกรมโดยใช้ข้อมูล 25.5 ↵ ผลลัพธ์ของโปรแกรมคืออะไร	.....
4.3 ผลลัพธ์ของโปรแกรมในข้อ 4.2 ต่าง กับผลลัพธ์ในข้อ 4.1 หรือไม่ เพราะเหตุ ใด	.....

## บทที่ 2

### การเขียนโปรแกรมภาษาซี(Introduction to C Language)

---

#### จุดประสงค์

1. เพื่อให้ทราบความหมายของข้อมูล
  2. เพื่อให้ทราบประเภทของข้อมูลในภาษาซี
  3. เพื่อให้ทราบความหมายของตัวแปรและ ประเภทของตัวแปรในภาษาซี
  4. เพื่อให้ทราบลักษณะการทำงานของฟังก์ชัน รับ - แสดง ข้อมูลประเภทต่าง ๆ
  5. สามารถเลือกใช้งานฟังก์ชัน รับ - แสดง ข้อมูลได้อย่างเหมาะสม
- 

#### 2.1 ประวัติความเป็นมาของภาษาซี

ภาษา C เป็นภาษาโปรแกรมในลักษณะภาษาเชิงโครงสร้าง เป็นภาษาในระดับสูง ไม่มีข้อจำกัดเรื่องรุ่นของเครื่องคอมพิวเตอร์ที่ใช้ ภาษา C มีพัฒนาการมาจากภาษา ALGO โดยได้ถูกพัฒนาขึ้นโดยเดนนิช ริชชี (Dennis Ritchies) ภาษา C ถูกนำมาใช้เขียนโปรแกรมระบบปฏิบัติการยูนิกซ์เนื่องจากมีความสามารถหลายอย่างที่เหมาะสมกับภาษาระดับสูงทั่วไป เช่น Basic Pascal เป็นต้น สามารถนำมาใช้ในการเขียนโปรแกรมสำเร็จรูปต่าง ๆ



**Dennis MacAlistair Ritchie (born September 9, 1941; found dead October 12, 2011) was an American computer scientist who "helped shape the digital era." He created the C programming language and, with long-time colleague Ken Thompson, the Unix operating system. Ritchie and Thompson received the Turing Award from the ACM in 1983, the Hamming Medal from the IEEE in 1990 and the National Medal of Technology from President Clinton in 1999. Ritchie was the head of Lucent Technologies System Software Research Department when he retired in 2007. He was the 'R' in K&R C and commonly known by his username dmr.**

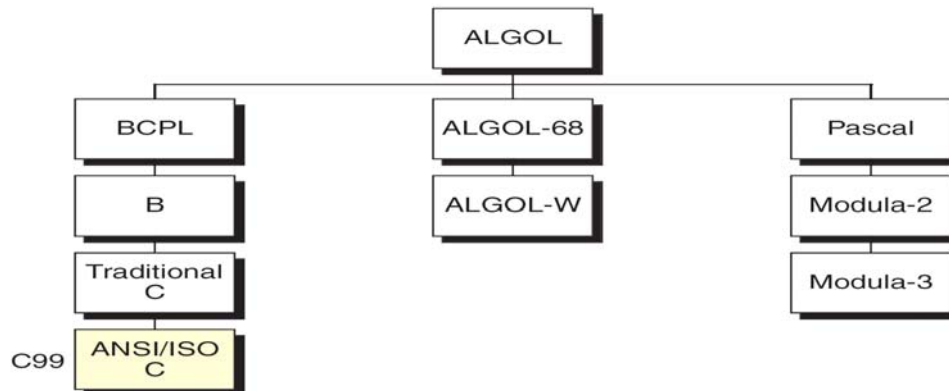
ที่มา [http://en.wikipedia.org/wiki/Dennis\\_Ritchie](http://en.wikipedia.org/wiki/Dennis_Ritchie)

สำหรับพัฒนาการของภาษา C มีดังนี้

- ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำ งานบนเครื่อง DEC PDP-7 ซึ่งทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอดมาจากภาษา BCPL ซึ่งเขียนโดย Marth Richards)
- ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก
- ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยม

ใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค. ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมี บทบาทสำคัญในการนำมาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลาย ๆ ค่าย ดังนั้นเพื่อกำหนดทิศทางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียกว่า ANSI C

- ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้นรายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ แนวความคิดของการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวความคิดการเขียนโปรแกรมที่เหมาะสมกับการพัฒนาโปรแกรมขนาดใหญ่ที่มีความสลับซับซ้อนมาก มีข้อมูลที่ใช้ในโปรแกรมจำนวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้



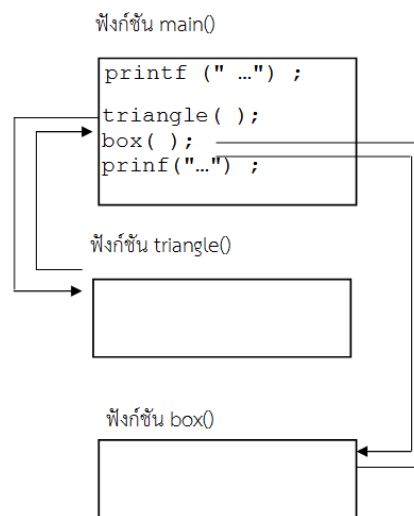
รูปที่ 2.1 พัฒนาการของการภาษา C [1]

## 2.2 ขั้นตอนการพัฒนาโปรแกรมด้วยภาษาซี

### 2.2.1 รูปแบบการทำงานของภาษาซี

ในภาษาซีจะเขียนโปรแกรมโดยการเรียกใช้แต่ละชุดของโปรแกรมที่เรียกว่าฟังก์ชัน (Function) หรือในโปรแกรมภาษาอื่นอาจจะเรียกว่า โปรแกรมย่อย หรือชุดคำสั่งย่อย (Procedure) นั่นเอง ฟังก์ชันเหล่านี้จะมีชื่ออะไรก็ได้ ก็ฟังก์ชันก็ได้ แต่อย่างน้อยต้องมี 1 ฟังก์ชันที่ชื่อ main เพื่อให้โปรแกรมเริ่มทำงานที่ฟังก์ชันนี้ ดังรูป

```
1  #include <stdio.h>
2  main ( )
3  {
4      printf ("This is a program for painting house") ;
5      triangle( );
6      box( );
7      printf("finish") ;
8  }
9  triangle( )
10 {
11     /* draw triangle */
12 }
13 box ( )
14 {
15     /* draw box */
16 }
```



รูปที่ 2.2 แสดงรูปแบบการทำงานของภาษาซี

จากรูปที่ 2.2 แสดงรูปแบบการทำงานของภาษาซีซึ่งแสดงให้เห็นว่า ภาษาซีสามารถมีได้หลายฟังก์ชัน แต่จะมีฟังก์ชันหลัก คือ ฟังก์ชัน main ในการควบคุมการทำงานของโปรแกรมให้ทำฟังก์ชันใดบ้าง แต่ถ้าโปรแกรมขนาดเล็กไม่มีการทำงานที่ซับซ้อน อาจจะไม่จำเป็นต้องมีฟังก์ชันอื่นๆ มีฟังก์ชัน main เพียงฟังก์ชันเดียวก็ได้



### 2.2.2 ฟังก์ชันในภาษาซี

ฟังก์ชันในภาษาซีแบ่งได้เป็น 2 ส่วน ได้แก่ ฟังก์ชันที่เป็นโปรแกรมหลักหรือฟังก์ชันหลักที่ถูกกำหนดให้มีเพียงฟังก์ชันเดียวเท่านั้น นั่นก็คือ ฟังก์ชัน `main()` สำหรับฟังก์ชันในส่วนที่ 2 เป็นฟังก์ชันที่ไม่ใช่ฟังก์ชันหลัก ซึ่งมีกี่ฟังก์ชันก็ได้

#### โครงสร้างทั่วไปในฟังก์ชัน

1 ส่วนหัวของฟังก์ชัน (Heading) เป็นส่วนที่นิยามชื่อฟังก์ชัน กำหนดชนิดและจำนวนตัวแปรที่ใช้ส่งผ่านค่าเข้าออก มีรูปแบบดังนี้

ชนิดพารามิเตอร์ที่ใช้ส่งค่าออก	ชื่อฟังก์ชัน( พารามิเตอร์ที่ใช้ส่งค่าเข้า )
--------------------------------	---

เช่น `int main(char)`

หมายถึง ฟังก์ชัน `main` มีการรับพารามิเตอร์ชนิดเป็น `char` และมีการส่งค่ากลับออกมาเป็นชนิด `int`

2 ส่วนกลุ่มคำสั่ง (Compound Statements) ส่วนนี้จะประกอบด้วย 3 ส่วนย่อย

- ส่วนกำหนดตัวแปร (Variable Declaration) ใช้สำหรับกำหนดตัวแปรเพื่อใช้งานในฟังก์ชัน
- ส่วนคำสั่ง (Statement) ประกอบด้วยคำสั่งต่างๆ เรียงกันไป แต่ละคำสั่งต้องปิดท้ายด้วยเครื่องหมาย ; เสมอ
- เครื่องหมาย { } เครื่องหมายนี้ทำหน้าที่กำหนดขอบเขตของกลุ่มคำสั่งในฟังก์ชัน

#### ส่วนหัวของฟังก์ชัน (Heading)

{

ส่วนกำหนดตัวแปร (Variable Declarations)

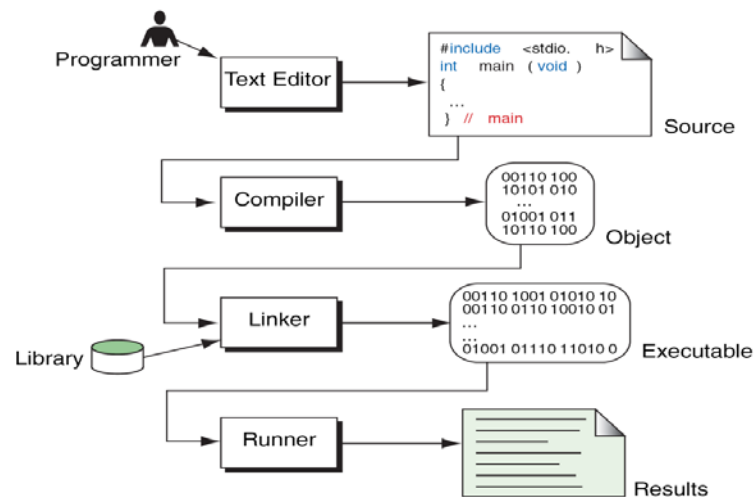
ส่วนคำสั่ง (Statements)

}

### 2.2.3 การคอมไพล์และลิงค์โปรแกรมในภาษาซี

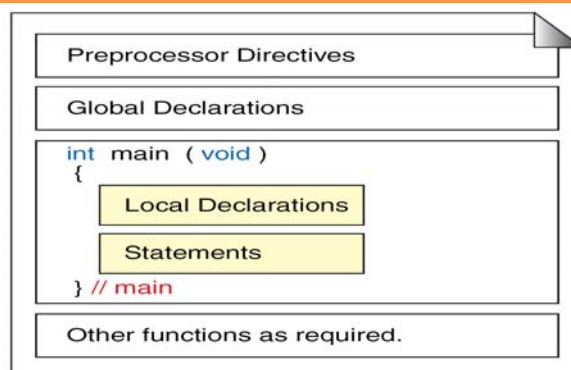
การสร้างโปรแกรมที่สามารถใช้งานได้ขึ้นมาโปรแกรมหนึ่ง ในภาษาซีมีขั้นตอนดังนี้

1. สร้างตัวโปรแกรมที่เป็นตัวอักษร หรือเรียกว่า **ซอร์สไฟล์ (Source file)** โดยมีนามสกุลเป็น `.c` หรือ `.cpp` ขึ้นมาก่อน โดยใช้โปรแกรมที่สามารถเขียนไฟล์ที่เก็บอักขระ (Editor) ใดๆ ก็ได้ อักษรหรืออักขระใดๆ นั้น จะต้องอยู่ในรูปแบบของการโปรแกรมภาษา (ขั้นตอนนี้คือการสร้างโปรแกรมที่เป็นภาษามนุษย์นั่นเอง)
  2. คอมไพเลอร์ของภาษาซี (C Compiler) จะทำการแปลงซอร์สไฟล์ จากอักขระใดๆ ให้เป็นรหัสที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้เก็บไว้ในอีกไฟล์หนึ่งเรียกว่า **Object file** ที่มีนามสกุล `.obj` ขั้นตอนนี้เรียกว่า **การคอมไพล์** เป็นการแปลงภาษามนุษย์เป็นภาษาเครื่อง
  3. ตัวเชื่อม (Linker) จะทำการตรวจสอบว่าในโปรแกรมที่เขียนขึ้นนั้น มีการเรียกใช้งานฟังก์ชันมาตรฐานใด จาก C Library บ้างหรือไม่ ถ้ามี ตัวเชื่อมจะทำการรวมเอาฟังก์ชันเหล่านั้นเข้ากับ **Object file** แล้วจะได้ไฟล์ที่สามารถทำงานได้ โดยมีนามสกุลเป็น `.exe` ขั้นตอนนี้เรียกว่า **การลิงค์** เป็นการรวมฟังก์ชันสำเร็จรูปเข้าไป แล้วสร้างไฟล์ที่ทำงานได้
- สำหรับขั้นตอนการคอมไพล์และลิงค์โปรแกรมภาษาซีสามารถแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 แสดงขั้นตอนการคอมไพล์และลิงค์โปรแกรมภาษาซี

## 2.3 โครงสร้างโปรแกรมภาษาซี



รูปที่ 2.4 โครงสร้างโปรแกรมภาษา C [1]

ภาษาซีมีโครงสร้างและลำดับการเขียนดังนี้

- คำสั่งตัวประมวลผลก่อน(Preprocessor statement/Preprocessor Directive)
- รหัสต้นฉบับ (Source code) มีลำดับการเขียนดังนี้
  - คำสั่งประกาศครอบคลุม (Global declaration statements)
  - ต้นแบบฟังก์ชัน(function prototypes)
  - ฟังก์ชันหลัก(main function) มีฟังก์ชันเดียว
  - ฟังก์ชัน(functions) มีได้หลายฟังก์ชัน
  - คำสั่งประกาศตัวแปรเฉพาะที่(Local declaration statements)
- หมายเหตุ(Comment) สามารถแทรกไว้ที่ใดก็ได้ภายในโปรแกรม

## ตัวอย่างที่ 2.1 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อแสดงข้อความทางหน้าจอ

1	/* This program demonstrates some of the components
2	of a simple C program.
3	Written by: your name here
4	Date: date program written
5	*/
6	#include <stdio.h>
7	int main ()
8	{
9	// Local Declarations
10	
11	// Statements
12	printf("Hello World!\n");
13	return 0;
14	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
Hello World!	

### คำอธิบาย

- `#include <stdio.h>` คือการบอกคอมไพเลอร์ให้นำไฟล์ `stdio.h` มารวมด้วย
- `main` คือชื่อของฟังก์ชัน โปรแกรมจะเริ่มทำงานที่นี้ และเมื่อจบฟังก์ชัน `main` หมายถึงจบโปรแกรมด้วย
- `printf("Hello World!\n")` คือการใช้ฟังก์ชัน `printf` พิมพ์ข้อความที่อยู่ในเครื่องหมาย " " ออกทางอุปกรณ์เอาต์พุตมาตรฐาน ส่วน `\n` คือการสั่งให้ขึ้นบรรทัดใหม่หลังจากพิมพ์ข้อความเสร็จ
- `return 0` คือ การคืนค่าไปยังโปรแกรมที่เรียกใช้ฟังก์ชัน `main()` ซึ่งจากตัวอย่างเป็นฟังก์ชัน `main()` ดังนั้น `return 0` จึงหมายถึงการจบการทำงานของโปรแกรม

หมายเหตุ ทุกประโยคภาษาซี (C Statement) จะต้องมีความหมาย ; ปิดท้าย

## ตัวอย่างที่ 2.2 ตัวอย่างการเขียนโปรแกรมเพื่ออ่านค่าตัวเลขจำนวนเต็มสองจำนวนจากคีย์บอร์ดแล้วพิมพ์ผลคูณของตัวเลขทั้งสองออกทางหน้าจอ

1	/*This program reads two integers from the keyboard and prints their
2	product.
3	Written by:
4	Date:
5	*/
6	#include <stdio.h>
7	int main ()
8	{
9	// Local Definitions
10	int number1;
11	int number2;
12	int result;
13	// Statements
14	scanf ("%d", &number1);
15	scanf ("%d", &number2);
16	result = number1 * number2;
17	printf ("%d", result);
18	return 0;
19	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
5	25
125	

## คำอธิบาย

- **#include <stdio.h>** คือการบอกคอมไพเลอร์ให้นำไฟล์ **stdio.h** มารวมด้วย
- **main** คือชื่อของฟังก์ชัน โปรแกรมจะเริ่มทำงานที่นี่ และเมื่อจบฟังก์ชัน **main** หมายถึงจบโปรแกรมด้วย
- **scanf ("%d", &number1);** คือการใช้ฟังก์ชัน **scanf** รับค่าจากผู้ใช้เก็บไว้ในตัวแปร **number1**
- **result = number1 \* number2;** คือ ผลคูณของตัวเลขที่อยู่ในตัวแปร **number1** และ **number2** และเก็บลงในตัวแปร **result**
- **printf ("%d", result);** คือการใช้ฟังก์ชัน **printf** พิมพ์ข้อความที่อยู่ในเครื่องหมาย " " ออกทางอุปกรณ์เอาต์พุตมาตรฐาน
- **return 0** คือ การคืนค่าไปยังโปรแกรมที่เรียกใช้ฟังก์ชัน **main()** ซึ่งจากตัวอย่างนี้เป็นฟังก์ชัน **main()** ดังนั้น **return 0** จึงหมายถึงการจบการทำงานของโปรแกรม

### 2.3.1 คำสั่งตัวประมวลผลก่อน (Preprocessor statement)

**ตัวประมวลผลก่อน (Preprocessor)** คือ ส่วนที่คอมไพเลอร์จะต้องทำก่อนทำการแปลโปรแกรม คำสั่งของตัวประมวลผลก่อนจะนำหน้าด้วยเครื่องหมาย **#** มีคำสั่งต่างๆ ต่อไปนี้

#include	#define	#if	#program	#elif	#ifdef
#endif	#error	#ifndef	#undef	#else	

คำสั่งที่น่าสนใจมี 2 คำสั่งคือ

**#include** ทำหน้าที่แจ้งให้คอมไพเลอร์อ่านไฟล์อื่นเข้ามาแปลรวมด้วย มีรูปแบบดังนี้

**#include <filename> หรือ #include "filename"**

เช่น

<b>#include &lt;dos.h&gt;</b>	อ่านไฟล์ dos.h จากไดเรกทอรีที่กำหนด
<b>#include "sample.h"</b>	อ่านไฟล์ sample.h จากไดเรกทอรีปัจจุบันหรือที่กำหนด
<b>#include "stdio.h"</b>	อ่านไฟล์ stdio.h จากไดเรกทอรีปัจจุบันหรือที่กำหนด

การใช้เครื่องหมาย **<...>** คร่อมชื่อ เพื่อบอกให้คอมไพเลอร์ค้นหาไฟล์จากไดเรกทอรีที่กำหนดใน Option directory แต่ถ้าใช้เครื่องหมาย **"..."** จะเป็นการกำหนดให้ค้นหาจากไดเรกทอรีปัจจุบัน และไดเรกทอรีที่กำหนดพาร์ทไว้

ชื่อฟังก์ชันการทำงานพื้นฐานของภาษาซี ตัวแปร ค่าคงที่ และมาโครพื้นฐาน โดยปกติจะรวมกันเป็นกลุ่มในไฟล์ที่เรียกว่าไฟล์หัว (Header Files) คือไฟล์ ที่มีนามสกุล **.h** ไฟล์หัวแต่ละไฟล์จะเก็บโมดูลแยกกันเป็นส่วนการทำงานแต่ละส่วนเพื่อให้ผู้เขียนโปรแกรมสามารถเรียกใช้งานได้ เช่น การเรียกใช้ไฟล์ **stdio.h** ซึ่งในไฟล์ดังกล่าวมีการกำหนดคำสั่ง **printf** ซึ่งถูกเรียกใช้ในฟังก์ชัน **main()** และฟังก์ชัน **process()** แต่ถ้าเราต้องการเรียกใช้ฟังก์ชัน **clrscr()** ซึ่งไม่มีในไฟล์นี้ แต่มีในไฟล์ **conio.h** เราจะต้อง include ไฟล์ **conio.h** เข้ามาด้วย

**#define** ทำหน้าที่ใช้กำหนดค่าคงที่ ที่เป็นชื่อแทน คำ นิพจน์ คำสั่ง หรือคำสั่งหลายคำสั่ง มีรูปแบบการใช้งานดังนี้

<b>#define</b> ชื่อตัวแปร (ชื่อที่ใช้แทน)    ค่าที่ต้องการกำหนด
---

เช่น                    #define                    TWO 2                    กำหนดตัวแปร TWO แทนค่า 2

                         #define PI    3.141592654                    กำหนดตัวแปร PI แทนค่า 3.141592654

### 2.3.2 รหัสต้นฉบับ (Source code)

รหัสต้นฉบับหมายถึง    ตัวโปรแกรมที่ประกอบด้วยคำสั่ง(statements)และตัวฟังก์ชันต่าง ๆ  
ประกอบด้วย

- คำสั่งประกาศครอบคลุม (Global declaration statements) ใช้ประกาศตัวแปรส่วนกลาง โดยที่ตัวส่วนกลางนี้จะสามารถถูกเรียกใช้จากทุกส่วนของโปรแกรม
- ต้นแบบฟังก์ชัน(function prototypes) เพื่อบอกให้ตัวแปลโปรแกรมทราบถึง ชนิดของค่าที่ส่งกลับ และชนิดของค่าต่าง ๆ ที่ส่งไปกระทำการในฟังก์ชัน
- ฟังก์ชันหลัก(main function) มีฟังก์ชันเดียว เมื่อสั่งให้โปรแกรมทำงาน ฟังก์ชันหลักจะเป็นจุดเริ่มต้นของการทำงาน ภายในฟังก์ชันหลักจะประกอบด้วยคำสั่งและคำสั่งที่เรียกใช้ฟังก์ชัน เมื่อมีการทำงานตามคำสั่งและฟังก์ชันต่าง ๆ แล้วจะมีการส่งค่าและกลับมาทำงานที่ฟังก์ชันหลักจนจบ
- ฟังก์ชัน(functions) มีได้หลายฟังก์ชัน ฟังก์ชันหมายถึงกลุ่มของคำสั่งที่ทำงานใดงานหนึ่งโดยการทำงานที่เป็นอิสระจากฟังก์ชันหลัก แต่อาจมีการรับส่งค่าระหว่างฟังก์ชันและฟังก์ชันหลัก การเขียนฟังก์ชันขึ้นต้นด้วย ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน วงเล็บ และตามด้วยเครื่องหมายปีกกา ภายในเครื่องหมายปีกกาประกอบด้วยชุดคำสั่งภาษาซี
- คำสั่งประกาศตัวแปรเฉพาะที่(Local declaration statements) ใช้ประกาศตัวแปรเฉพาะที่ โดยที่ตัวแปรเฉพาะที่จะสามารถถูกเรียกใช้เฉพาะภายในฟังก์ชันนั้น
- ตัวอย่าง source code ที่ประกอบไปด้วยส่วนต่าง ๆ ของโปรแกรมที่อธิบายไว้ด้านบนสามารถแสดงได้ดังตัวอย่างต่อไปนี้

**ตัวอย่างที่ 2.3** แสดงตัวอย่างโครงสร้างของโปรแกรมภาษาซีโดยเป็นตัวอย่างโปรแกรมที่ทำหน้าที่ในการเรียกใช้ฟังก์ชันสำหรับคูณตัวเลขจำนวนเต็มสองจำนวน

```
1  /* This program demonstrates function calls to multiply two numbers.
2      Written by:
3      Date:
4  */
5  #include <stdio.h>
6  int multiply (int num1, int num2);
7  int main ()
8  {
9      // Local Declarations
10     int multiplier;
11     int multiplicand;
12     int product;
13     // Statements
14     printf("Enter two integers: ");
15     scanf ("%d%d", &multiplier, &multiplicand);
```

```

16     product = multiply (multiplier, multiplicand);
17     printf("Product of %d & %d is %d\n",
18           multiplier, multiplicand, product);
19     return 0;
20 } // main
21 /* ===== multiply =====
22    Multiples two numbers and returns product.
23 */
24 int multiply (int num1, int num2)
25 {
26     return (num1 * num2);
27 } // multiply

```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```

Enter two integers: 17 21
Product of 17 & 21 is 357

```

### 2.3.3 หมายเหตุ (Comment)

สามารถแทรกไว้ที่ใดก็ได้ภายในโปรแกรม ภาษาซีนิยมการเขียนข้อความอธิบายการทำงานในส่วนต่างๆ ของโปรแกรมเพื่อให้เข้าใจและอ่านโปรแกรมง่ายขึ้น การเขียนอธิบายจะใช้เครื่องหมาย /\* และ \*/ คร่อมข้อความที่ต้องการอธิบาย ดังนี้

/\* .....ข้อความที่ต้องการอธิบาย.....\*/

แต่ถ้าต้องการเขียนอธิบายหลายๆบรรทัดจะเขียนได้ดังนี้

/\* .....  
..... ข้อความที่ต้องการอธิบาย.....  
.....\*/

การเขียนคำอธิบายจะเขียนไว้ที่ไหนในโปรแกรมก็ได้ โดยมากจะนิยมเขียนอธิบายขั้นตอนการทำงานก่อนเขียนคำสั่ง หรือ อธิบายความหมายของคำสั่งไว้ข้างคำสั่งนั้นก็ได้

**ตัวอย่างที่ 2.4** แสดงตัวอย่างการเขียนหมายเหตุแบบต่าง ๆ

```

1  /* This program demonstrates three ways to use constants.
2     Written by:
3     Date:
4  */
5  #include <stdio.h>
6  #define PI 3.1415926536
7  int main ()
8  {
9      // Local Declarations
10     const double cPi = PI;
11     // Statements
12     printf("Defined constant PI: %f\n", PI);
13     printf("Memory constant cPi: %f\n", PI);
14     printf("Literal constant:    %f\n", 3.1415926536);
15     return 0;
16 } // main

```

## 2.4 รูปแบบคำสั่งในภาษาซี

รูปแบบคำสั่งในภาษาซี มีกฎเกณฑ์ในการเขียนคำสั่ง ดังนี้

1. คำสั่งทุกคำสั่งต้องเขียนด้วยอักษรตัวเล็กเสมอ เช่นคำสั่ง `printf`, `scanf`, `for` เป็นต้น
2. ทุกคำสั่งจะใช้เครื่องหมาย ; แสดงการจบของคำสั่ง เช่น `printf("Hello");` ;
3. การเขียนคำสั่ง จะเขียนได้แบบอิสระ (Free Format) คือ สามารถเขียนหลายๆคำสั่งต่อกันได้ เช่น

```
printf("Hello World"); printf("C Programming"); f = 3.414;
```

แต่เพื่อความเป็นระเบียบและอ่านง่าย ควรจะเขียน 1 คำสั่งต่อ 1 บรรทัด

## 2.5 ชนิดของข้อมูล (Data types)

### 2.5.1 กลุ่มคำในภาษาซี

กลุ่มคำที่มีใช้งานในภาษาซีมี 2 ประเภท คือ

1. **คำสงวน (Keywords)** คือคำที่ภาษาซีกำหนดไว้ก่อนแล้ว เพื่อใช้งาน ได้แก่

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

2. **คำที่ผู้ใช้ตั้งขึ้นใหม่ (User Defines words)** คือ กลุ่มอักษรที่นิยามขึ้นใช้ในโปรแกรม โดยผู้เขียนโปรแกรมกำหนดขึ้นเอง มีข้อกำหนดดังนี้

- ตัวอักษรภาษาอังกฤษตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ภาษาซีถือว่าเป็นคนละตัวกัน เช่น `Area` และ `area` เป็นตัวแปรคนละตัวกัน
- ตัวอักษรตัวแรกต้องเป็นตัวอักษรหรือ `_` จะเป็นตัวเลขไม่ได้
- ตัวอักษรที่ไม่ใช่ตัวแรกจะเป็นตัวอักษรหรือ `_` หรือตัวเลขก็ได้
- ก่อนการใช้ชื่อใดๆ ต้องนิยามก่อนเสมอ เช่นต้องการใช้ชื่อข้อมูลที่ชื่อ `area` ที่มีชนิดข้อมูลข้อมูลเป็น `float` ต้องมีการนิยามก่อนนำไปใช้ คือ `float area;`
- ห้ามตั้งชื่อซ้ำกับคำสงวน
- ภายในกลุ่มคำสั่ง สามารถกำหนดชื่อขึ้นใหม่ได้ ชื่อนั้นจะถูกใช้งานภายในกลุ่มคำสั่ง และกลุ่มคำสั่งที่ย่อยลงไปเท่านั้น หากชื่อในกลุ่มคำสั่งไปซ้ำกับที่นิยามไว้ภายนอก จะถือเอาชื่อที่นิยามใหม่เป็นหลัก
- ความยาวชื่อจะขึ้นอยู่กับตัวแปรในภาษาซี สำหรับโปรแกรม Borland C กำหนดไว้ 32 ตัวอักษร

### 2.5.2 ตัวแปร (Variable) ในภาษาซี



ตัวแปร หมายถึงชื่อเรียกแทนพื้นที่เก็บข้อมูลในหน่วยความจำ มีชนิดของข้อมูล หรือแบบของตัวแปรคือ char, int, long, float, double, unsigned int, unsigned long int,

- การประกาศตัวแปร ทำได้ 2 แบบ คือ
  1. ประกาศไว้นอกกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า **Global Variable** กำหนดไว้นอกฟังก์ชัน ใช้งานได้ทั้งโปรแกรม มีค่าเริ่มต้นเป็น 0 (กรณีไม่ได้กำหนดค่าเริ่มต้น)
  2. ประกาศไว้ในกลุ่มคำสั่ง หรือฟังก์ชัน เรียกตัวแปรนี้ว่า **Local Variable** กำหนดไว้ภายในฟังก์ชัน ใช้งานได้ภายในฟังก์ชันนั้น และไม่ถูกกำหนดค่าเริ่มต้นโดยอัตโนมัติ
- การประกาศตัวแปร มีลักษณะดังนี้

ชนิดตัวแปร ชื่อตัวแปร , ชื่อตัวแปร, ชื่อตัวแปร,.....;

กฎการตั้งชื่อตัวแปร การตั้งชื่อตัวแปรมีข้อกำหนดดังนี้

1. ประกอบด้วย a ถึง z, 0 ถึง 9 และ \_ เท่านั้น
2. อักขระตัวแรกต้องเป็น a ถึง z และ \_
3. ห้ามใช้ชื่อเฉพาะ
4. ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก มีความหมายที่แตกต่างกัน (Case sensitive)
5. ยาวสูงสุดไม่เกิน 31 ตัวอักษร

ตัวอย่าง การตั้งชื่อตัวแปรที่ถูกต้องและไม่ถูกต้องแสดงดังรูป

Valid Names		Invalid Name	
a	// Valid but poor style	\$sum	// \$ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

รูปที่ 2.5 ตัวอย่างของการตั้งชื่อตัวแปรที่ถูกต้องและไม่ถูกต้อง [1]

### 2.5.3 ชนิดของตัวแปรในภาษาซี

1. ตัวแปรแบบ char เป็นตัวแปรที่ใช้สำหรับเก็บข้อมูลที่เป็นตัวอักษรขนาด 1 ตัว โดยใช้เนื้อที่ในการเก็บ 1 ไบต์(8 บิต) ตัวอย่างตัวแปรชนิดนี้ เช่น 'A' , 'b' , '1' , '?'
2. ตัวแปรแบบ integer เป็นตัวแปรที่ใช้สำหรับการเก็บค่าตัวเลขที่เป็นจำนวนเต็มที่มีค่าระหว่าง -32768 ถึง 32767 ใช้เนื้อที่ในการเก็บ 2 ไบต์(16 บิต) ตัวอย่างตัวแปรชนิดนี้ เช่น 5 -10
3. ตัวแปรแบบ long เป็นตัวแปรที่เก็บค่าเป็นจำนวนเต็มที่มีจำนวนไบต์เป็น 2 เท่าของจำนวนเต็ม(32 บิต) (มักจะใช้เป็นคำนำหน้าตัวแปร เช่น long int )
4. ตัวแปรแบบ float เป็นตัวแปรที่ใช้เก็บข้อมูลที่เป็นเลขทศนิยม โดยจะเก็บอยู่ในรูป  $a.b \times 10^e$  ใช้พื้นที่ในการเก็บ 4 ไบต์(32 บิต) มีค่าระหว่าง  $3.4E-38$  ถึง  $3.4E+38$  หรือ แสดงเป็น เลขทศนิยมได้ไม่เกิน 6 ตำแหน่ง ตัวอย่างตัวแปรชนิดนี้ เช่น 10.625 -6.67

5. ตัวแปรแบบ double เป็นตัวแปรที่เก็บข้อมูลที่เป็นเลขทศนิยมเหมือนกับ float แต่จะใช้พื้นที่ในการเก็บมากกว่าเดิม 2 เท่า คือมีขนาด 8 ไบต์(64 บิต) มีค่าระหว่าง 1.7E-308 ถึง 1.7E+308

6. ตัวแปรแบบ unsigned แสดงว่าเป็นตัวแปรที่เก็บค่าเป็นจำนวนเต็มแบบไม่คิดเครื่องหมาย (เป็นบวกเท่านั้น) มักจะใช้เป็นค่านำหน้าตัวแปร ตัวอย่างการใช้งาน เช่น unsigned int

ชนิด	ขนาดความกว้าง	ช่วงของค่า
char	8 บิต	ASCII character (-128 ถึง 127)
unsigned char	8 บิต	0-255
int	16 บิต	-32768 ถึง 32767
long int	32 บิต	-2147483648 ถึง 2147483649
float	32 บิต	3.4E-38 ถึง 3.4E+38 หรือ ทศนิยม 6 ตำแหน่ง
double	64 บิต	1.7E-308 ถึง 1.7E+308 หรือ ทศนิยม 12 ตำแหน่ง
unsigned int	16 บิต	0 ถึง 65535
unsigned long int	32 บิต	0 ถึง 4294967296

รูปที่ 2.6 แสดงชนิดและขนาดของข้อมูลที่ใช้ในภาษาซี

หมายเหตุ E คือสัญลักษณ์ในเลขจำนวนทศนิยมในรูปของเลขชี้กำลัง อาจใช้ e แทนก็ได้ หมายถึงเขียนเลขจำนวนทศนิยมฐานสิบ

รูปแบบ 9.9999e+n หมายถึง  $9.9999 \times 10^{+n}$

หรือ 9.9999e-n หมายถึง  $9.9999 \times 10^{-n}$

ดังนั้นตัวเลข  $1.5 \times 10^{-3}$  จะเขียนแทนด้วย 1.5E-3 1.5e-3 0.15e-2 หรือ 15e-4

การหาขนาดของชนิดตัวแปรต่าง ๆ จะใช้คำสั่ง sizeof(ประเภทข้อมูล) โดยลำดับขนาดของประเภทข้อมูลเรียงลำดับจากน้อยไปหามากมีดังนี้

$$\text{sizeof (short)} \leq \text{sizeof (int)} \leq \text{sizeof (long)} \leq \text{sizeof (long long)}$$

สามารถสรุปรูปแบบของการประกาศข้อมูลประเภทต่าง ๆ ได้ดังรูปที่ 2.7

กลุ่มของข้อมูล	ประเภทข้อมูล	C implementation
Void	Void	void
Integral	Boolean	bool
	Character	char, wchar_t
	Integer	short int, int, long int, long long int
Floating-Point	Real	float, double, long double
	Imaginary	float imaginary, double imaginary, long double imaginary
	Complex	float complex, double complex, long double complex

รูปที่ 2.7 สรุปประเภทข้อมูล

## ตัวอย่างที่ 2.5 ตัวอย่างการเขียนโปรแกรมสำหรับการคำนวณตัวเลขประเภทต่าง ๆ

1	/* This program demonstrates binary expressions.
2	Written by:
3	Date:
4	*/
5	#include <stdio.h>
6	int main (void)
7	{
8	// Local Declarations
9	int a = 17;
10	int b = 5;
11	float x = 17.67;
12	float y = 5.1;
13	// Statements
14	printf("Integral calculations\n");
15	printf("%d + %d = %d\n", a, b, a + b);
16	printf("%d - %d = %d\n", a, b, a - b);
17	printf("%d * %d = %d\n", a, b, a * b);
18	printf("%d / %d = %d\n", a, b, a / b);
19	printf("%d %% %d = %d\n", a, b, a % b);
20	printf("\n");
21	printf("Floating-point calculations\n");
22	printf("%f + %f = %f\n", x, y, x + y);
23	printf("%f - %f = %f\n", x, y, x - y);
24	printf("%f * %f = %f\n", x, y, x * y);
25	printf("%f / %f = %f\n", x, y, x / y);
26	return 0;
27	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
Integral calculations	
17 + 5 = 22	
17 - 5 = 12	
17 * 5 = 85	
17 / 5 = 3	
17 % 5 = 2	
Floating-point calculations	
17.670000 + 5.100000 = 22.770000	
17.670000 - 5.100000 = 12.570000	
17.670000 * 5.100000 = 90.116997	
17.670000 / 5.100000 = 3.464706	

### ตัวอย่างที่ 2.6

int a ;

หมายความว่า ประกาศตัวแปร a เป็นตัวแปรที่ใช้สำหรับเก็บค่าที่เป็นเลขจำนวนเต็มที่มีค่าอยู่ระหว่าง -35768 ถึง 32767

### ตัวอย่างที่ 2.7

int num1=8;

หมายความว่า ประกาศตัวแปร num1 เป็นตัวแปรที่เก็บค่าตัวเลขจำนวนเต็มให้ค่าเริ่มต้นเท่ากับ 8

### ตัวอย่างที่ 2.8

float money,price ;

หมายความว่า money และ price เป็นตัวแปรที่ใช้สำหรับเก็บค่าที่เป็นเลขทศนิยม โดยจะให้ตำแหน่งทศนิยมได้ไม่เกิน 6 หลัก

### ตัวอย่างที่ 2.9

char ch='A'

หมายความว่า ประกาศตัวแปร ch เป็นตัวแปรที่เก็บค่าตัวอักษรเพียง 1 ตัว คือ ตัวอักษร 'A'

**ตัวอย่างที่ 2.10** unsigned long int test;

หมายความว่า ประกาศตัวแปร test เป็นตัวแปรที่ใช้สำหรับเก็บค่าที่เป็นเลขจำนวนเต็ม แบบยาวที่ไม่คิดเครื่องหมาย

**ตัวอย่างที่ 2.11**

```
char a,b,c,d;      /* ตัวแปร a,b,c,d เป็นตัวแปรชนิด character */
unsigned e;        /* ตัวแปร e เป็นตัวแปรชนิด unsigned int */
char key = 'A';    /* ตัวแปร key เป็นตัวแปรชนิด character มีค่า 'A' */
char name = "SAM" /* ตัวแปร name เป็นตัวแปรชนิด character มีค่า "SAM" */
```

#### 2.5.4 ค่าคงที่ constant

คือ ค่าของข้อมูลที่มีจำนวนแน่นอน เขียนได้ 3 ลักษณะ คือ

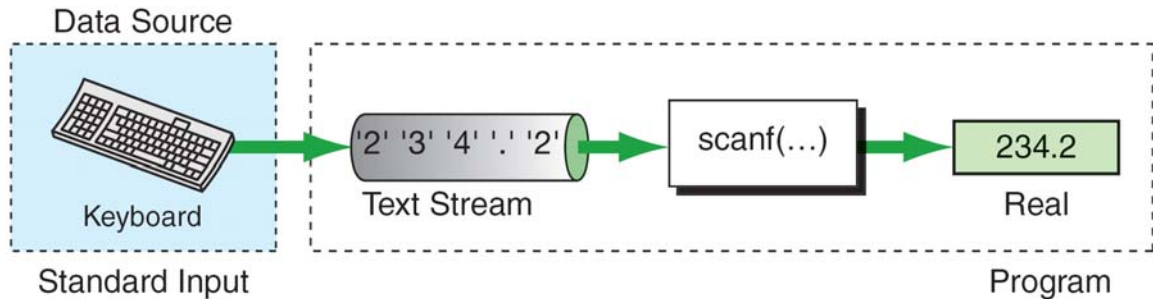
- 1 ค่าคงที่จำนวนเต็ม เขียนอยู่ในรูปตัวอักษร อาจมีเครื่องหมายลบนำหน้า
- 2 ค่าคงที่จำนวนจริง เขียนในรูปตัวเลขมีทศนิยม
- 3 ค่าคงที่ที่หมายถึงรหัสตัวอักษร (ตัวอักษรถูกจำในรูปแบบตัวเลข ตามรหัส ASCII)

ตัวอย่างการประกาศค่าคงที่แสดงได้ดังรูปที่ 2.8

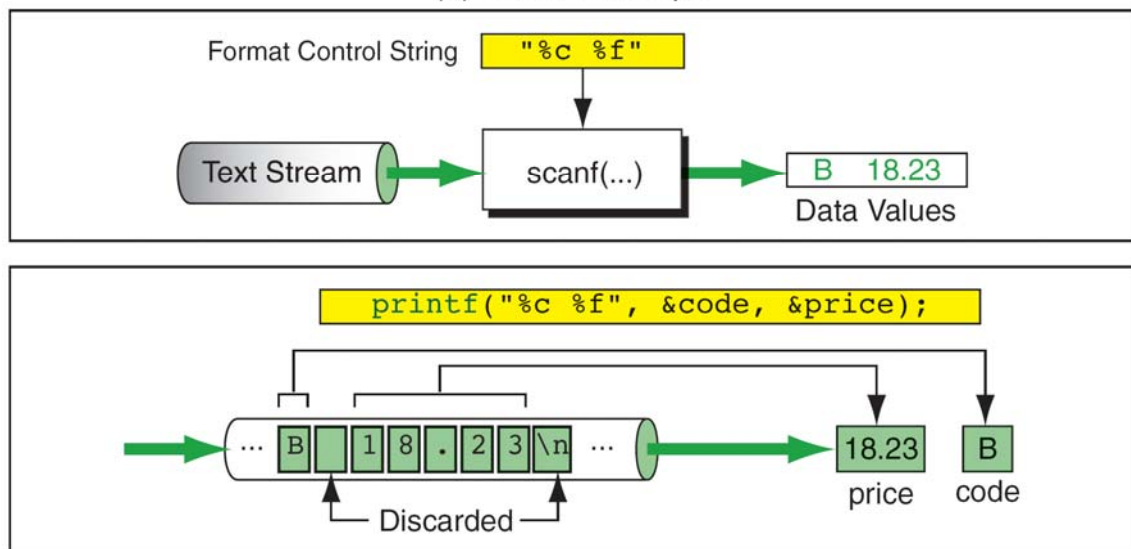
ตัวอย่างค่าคงที่	ชนิด	ความหมาย
255	decimal int	จำนวนเต็มฐานสิบ
0xFF	hexadecimal int	จำนวนเต็มฐานสิบหก
0377	octal int	จำนวนเต็มฐานแปด
255L หรือ 255l	long int	จำนวนเต็มฐานสิบแบบยาว
255U หรือ 255u	unsigned int	จำนวนเต็มฐานสิบไม่คิดเครื่องหมาย
0xFFUL	unsigned long int	เลขฐานสิบหกแบบยาวไม่คิดเครื่องหมาย
15.75E2	floating point	เลขทศนิยมแบบยกกำลัง
-1.23	floating point	เลขทศนิยมแบบค่าติดลบ
.123	floating point	เลขทศนิยม
123F	floating point	เลขทศนิยม
'a'	Character	ตัวอักษร
""	String	ประโยค, ข้อความ

รูปที่ 2.8 แสดงตัวอย่างค่าคงที่

## 2.6 การรับและแสดงผลข้อมูล



(a) Basic Concept



(b) Implementation

รูปที่ 2.9 แสดงตัวอย่างการรับและแสดงผลข้อมูล

### 2.6.1 การแสดงผลข้อมูล

#### ฟังก์ชัน printf()

เราสามารถแสดงค่าข้อมูลที่อยู่ในตัวแปรได้โดยใช้ฟังก์ชัน printf() ซึ่งเป็นฟังก์ชันจากคลังที่มาพร้อมกับตัวแปลโปรแกรมภาษาซีใช้สำหรับการแสดงผล มีรูปแบบดังนี้

printf("สายอักขระควบคุม", ตัวแปร); โดย

1. สายอักขระควบคุมประกอบด้วย 3 ส่วนคือ

-ตัวอักขระที่จะแสดง

-รูปแบบการแสดงผล ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%)

-ลำดับหลัก (escape sequence)

2. ตัวแปร คือชื่อของตัวแปรที่จะแสดงผล (format specifies)

รูปแบบการแสดงผล(Format specifies) มีการกำหนดรูปแบบการแสดงผลดังนี้ คือ

ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์(%) ตามด้วยอักขระ 1 ตัว หรือหลายตัวโดยที่อักขระมีความหมายดังนี้

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

### ลำดับหลัก(Escape sequence)

ในการแสดงผลบางสิ่งบางอย่างที่จะแสดงอาจไม่ใช่ตัวอักษร จึงไม่สามารถที่จะเขียนสิ่งที่จะแสดงไว้ในโปรแกรมได้ เช่น ต้องการเขียนโปรแกรมให้ส่งเสียง หรือต้องการให้เลื่อนขึ้นบรรทัดใหม่ก่อนแสดงข้อความ ดังนั้นในการเขียนโปรแกรมเพื่อแสดงผลสิ่งที่ไม่ใช่ตัวอักษรปกติ จะต้องใช้ลำดับหลักเพื่อช่วยในการกำหนดอักขระพิเศษหรือสิ่งที่ไม่ใช่อักขระ ที่ต้องการให้โปรแกรมแสดง

ลำดับหลัก จะเขียนขึ้นต้นด้วยเครื่องหมาย \ แล้วตามด้วยอักขระ ในการทำงานเครื่องหมาย \ จะบอกให้เครื่องคอมพิวเตอร์ทราบว่าให้หลีกเลี่ยงการตีความอักขระที่ตามหลังมานี้ในลักษณะปกติ เพราะอักขระเหล่านี้จะมีความหมายพิเศษแตกต่างออกไป ลำดับหลักที่ใช้ในการแสดงผลสิ่งที่ไม่ใช่อักขระปกติ ได้แก่

ลำดับหลัก	ผลการกระทำ
\n	ขึ้นบรรทัดใหม่(new line)
\t	เลื่อนไปยังจุดตั้งระยะ tab ต่อไป
\a	เสียงกระดิ่ง (bell)
\b	ถอยไปหนึ่งทีว่าง(backspace)
\f	ขึ้นหน้าใหม่(form feed)
\\	แสดงเครื่องหมายทับกลับหลัง (backslash)
\'	แสดงเครื่องหมายฝนทอง(single quote)
\"	แสดงเครื่องหมายพันหนุ(double quote)

### การแสดงค่าของตัวแปร

ในการแสดงค่าของตัวแปรภายในฟังก์ชัน printf() จะมีรูปแบบและความหมายของการใช้ตัวอักษรแต่ละตัวอักษรดังนี้

ชนิดข้อมูล	ชนิดตัวแปร	รูปแบบสำหรับ printf
จำนวนเต็ม	short integer long unsigned short unsigned int unsigned long	%hd หรือ %hi %d หรือ %i %ld หรือ %li %hu %u %lu
จำนวนจริง	float double	%f %lf
อักขระ สายอักขระ	char char s[]	%c %s
เลขฐาน	จำนวนเต็มฐาน 10 จำนวนเต็มฐาน 8 จำนวนเต็มฐาน 16	%d %o %x
	แสดงเครื่องหมาย %	%%

### การจัดรูปแบบผลลัพธ์ printf()

เมื่อต้องการจัดรูปแบบการแสดงผลออกมาทางหน้าจอสามารถใช้ฟังก์ชัน printf() โดยภายในฟังก์ชันดังกล่าวจะมีรูปแบบของการจัดหน้าจอดังนี้

"%"	Flag	Maximum width	.Pecision	Size	Conversion code	"
-----	------	---------------	-----------	------	-----------------	---

Flag คือ - หมายถึงการจัดให้ชิดซ้าย  
+ ให้แสดงเครื่องหมาย + หน้าตัวเลข

Maximum width คือ จำนวนสดมภ์ที่จองสำหรับแสดงผลข้อมูล

Precision คือ จำนวนตัวเลขหลังจุดทศนิยม

Size คือ ชนิดตัวแปร เช่น long จะเป็น l

### ตัวอย่างที่ 2.12 ตัวอย่างการจัดรูปแบบผลลัพธ์

"%"	+	10	.2		f	"
-----	---	----	----	--	---	---

```

1  #include <stdio.h>
2  main ( )
3  {
4      printf ("%+10.2f\n", 12.34);
5  }
```



ผลลัพธ์จากการประมวลผลโปรแกรม

+12.34

ตัวอย่างการจัดรูปแบบผลลัพธ์

+ 1 2 . 3 4

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Conversion code คือ การกำหนดชนิดข้อมูลในการแสดงผล

%d : พิมพ์ int ด้วยตัวเลขฐานสิบ

%o : พิมพ์ int ด้วยตัวเลขฐานแปด

%x : พิมพ์ int ด้วยตัวเลขฐานสิบหก

%f : พิมพ์ float, double แบบจุดทศนิยม (หกตำแหน่ง)

%e : พิมพ์ float, double แบบวิทยาศาสตร์ เช่น 1.23e+23

%c : พิมพ์ char ตัวอักษร 1 ตัว

%s : พิมพ์ข้อความ เช่น "Hello"

ตัวอย่างที่ 2.13 โปรแกรมแสดงตัวเลขในรูปแบบต่าง ๆ

```

1 #include <stdio.h>
2 main ( )
3 {
4     int a;
5     double b;
6     a=1;
7     b=1040.041;
8     printf ("%4i=%10.2f\n", a, b);
9     a=2;
10    b=5.05;
11    printf ("%4i=%10.2f\n", a, b);
12    a=3;
13    b=1234567.351;
14    printf ("%4i=%10.2f\n", a, b);
15    printf("100%");
16 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

1= 1040.04  
2= 5.05  
3=1234567.35  
100%

ตัวอย่างการจัดรูปแบบผลลัพธ์

			1	=				1	0	4	0	.	0	4
			2	=							5	.	0	5
			3	=	1	2	3	4	5	6	7	.	3	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ตัวอย่างที่ 2.14 การแสดงตัวเลขที่ระบุเฉพาะจำนวนตัวเลขหลังจุดทศนิยม

1	#include <stdio.h>
2	main ( )
3	{
4	printf ("%e\n", 9876543.21);
5	printf ("%e\n", +9876543.21);
6	printf ("%e\n", -9876543.21);
7	printf ("%E\n", -9876543.21);
8	printf ("%f\n", 9876543.21);
9	printf ("%g\n", 9876543.21);
10	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
9.876543e+006	
9.876543e+006	
-9.876543e+006	
-9.876543E+006	
9876543.210000	
9.87654e+006	

ตัวอย่างการจัดรูปแบบผลลัพธ์

9	.	8	7	6	5	4	3	e	+	0	0	6		
9	.	8	7	6	5	4	3	e	+	0	0	6		
-	9	.	8	7	6	5	4	3	e	+	0	0	6	
9	.	8	7	6	5	4	3	E	+	0	0	6		
9	8	7	6	5	4	3	.	2	1					
9	.	8	7	6	5	4	e	+	0	0	6			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ตัวอย่างที่ 2.15 การจัดรูปแบบจำนวนเต็ม

1	#include <stdio.h>
2	main ( )
3	{
4	printf ("%5d\n", 9);
5	printf ("%5d\n", 98);
6	printf ("%5d\n", 987);
7	printf ("%5d\n", -9);
8	printf ("%5d\n", -98);
9	printf ("%5d\n", -987);
10	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
9	
98	
987	
-9	
-98	
-987	

### ตัวอย่างการจัดรูปแบบผลลัพธ์

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

ตัวอย่าง 2.16

1	/* This program calculates the area and circumference
2	of a circle using PI as a defined constant.
3	Written by:
4	Date:
5	*/
6	#include <stdio.h>
7	#define PI 3.1416
8	int main ()
9	{
10	// Local Declarations
11	float circ;
12	float area;
13	float radius;
14	// Statements
15	printf("\nPlease enter the value of the radius: ");
16	scanf("%f", &radius);
17	circ = 2 * PI * radius;
18	area = PI * radius * radius;
19	printf("\nRadius is : %10.2f", radius);
20	printf("\nCircumference is : %10.2f", circ);
21	printf("\nArea is : %10.2f", area);
22	return 0;
23	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
Please enter the value of the radius: 23	
Radius is : 23.00	
Circumference is : 144.51	
Area is : 1661.91	

ตัวอย่าง 2.17

1	#include <stdio.h>
2	int main ()
3	{
4	// Statements
5	printf("\tPart Number\tQty On Hand");
6	printf("\tQty On Order\tPrice\n");

```

7 // Print data
8 printf("\t %06d\t\t%7d\t\t%7d\t\t $%7.2f\n",
9      31235, 22, 86, 45.62);
10 printf("\t %06d\t\t%7d\t\t%7d\t\t $%7.2f\n",
11      321, 55, 21, 122.);
12 printf("\t %06d\t\t%7d\t\t%7d\t\t $%7.2f\n",
13      28764, 0, 24, .75);
14 printf("\t %06d\t\t%7d\t\t%7d\t\t $%7.2f\n",
15      3232, 12, 0, 10.91);
16 // Print end message
17 printf("\n\tEnd of Report\n");
18 return 0;
19 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม			
Part Number	Qty On Hand	Qty On Order	Price
031235	22	86	\$ 45.62
000321	55	21	\$ 122.00
028764	0	24	\$ 0.75
003232	12	0	\$ 10.91
End of Report			

## 2.6.2 การรับค่าข้อมูล

### ฟังก์ชัน scanf()

ฟังก์ชัน scanf() เป็นฟังก์ชันจากคลังก่อนเรียกใช้ต้อง #include<stdio.h> จึงจะสามารถใช้คำสั่งได้  
ใช้ในการรับข้อมูลจากแป้นพิมพ์ โดยจะบอกเลขที่อยู่ของตัวแปรในหน่วยความจำ แล้วจึงนำค่าที่รับมาไปเก็บ  
ไว้ตามที่อยู่นั้น ฟังก์ชัน scanf() มีรูปแบบดังนี้ scanf("%รูปแบบ", &ตัวแปร);

โดยที่ &ตัวแปร หมายถึงเลขที่อยู่ (address) ของตัวแปรที่จะรับค่ามาเก็บในหน่วยความจำ

ตัวอย่าง การรับข้อมูลชนิด int แล้วไปเก็บไว้ในตัวแปร age

รูปแบบ scanf("%d",&age);

ตัวอย่าง การรับข้อมูลที่ละหลายตัวแปรได้

รูปแบบ scanf("%s %f", name, &GPAX);

สิ่งที่ควรระวัง การรับข้อมูล int float char ต้องมีเครื่องหมาย & แต่ string ไม่ต้องมี &

## 2.6.3 ฟังก์ชันอื่นๆ ที่ใช้ในการรับและแสดงข้อมูล

1. ฟังก์ชัน getchar ( ) เป็นฟังก์ชันที่ทำงานกับตัวอักษรโดยเฉพาะ

getchar ( ) เป็นฟังก์ชันที่ใช้รับข้อมูลเข้ามาทางแป้นพิมพ์ทีละ 1 ตัวอักษร โดยต้องกด enter ทุก  
ครั้งเมื่อสิ้นสุดข้อมูล และข้อมูลที่ป้อนจะปรากฏให้เห็นบนหน้าจอภาพด้วย

รูปแบบ

ชื่อตัวแปร =getchar ( ) ;

### ตัวอย่าง 2.18

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

3	{ <code>char</code> ch;
4	ch=getchar ( );
5	}

**ผลลัพธ์จากการประมวลผลโปรแกรม**

เครื่องจะรอรับข้อมูลจากแป้นพิมพ์ที่ผู้ใช้ป้อน จำนวน 1 ตัวอักษรเก็บไว้ในตัวแปร ch หลังจากที่ใช้ต้องกดปุ่ม enter เพื่อให้ฟังก์ชันรับค่าข้อมูล

## 2. ฟังก์ชัน getch ( )

getch ( ) เป็นฟังก์ชันที่รับข้อมูลที่เป็นตัวอักษร 1 ตัว เข้ามาทางแป้นพิมพ์ โดยเมื่อป้อน ข้อมูลเสร็จไม่ต้องกดปุ่ม enter และอักขระที่ป้อนเข้ามาจะไม่ปรากฏบนจอภาพ ต้อง #include <conio.h> จึงจะสามารถใช้ฟังก์ชัน getch ( ) ได้

รูปแบบ

<b>ชื่อตัวแปร =getch ( ) ;</b>
--------------------------------

## ตัวอย่าง 2.19

1	#include<stdio.h>
2	main ( )
3	{ <code>char</code> ch;
4	x=getch ( );
5	}

**ผลลัพธ์จากการประมวลผลโปรแกรม**

เครื่องจะรอรับข้อมูลจากแป้นพิมพ์เข้ามา 1 ตัวเพื่อนำมาเก็บไว้ในตัวแปร x โดยผู้ใช้ไม่ต้องกด enter หลังจากที่ย้อนข้อมูลเสร็จแล้ว

## 3 ฟังก์ชัน gets ( ) เป็นฟังก์ชันที่ทำงานกับข้อความ(String) โดยเฉพาะ

gets ( ) เป็นฟังก์ชันที่รับข้อมูลที่เป็นข้อความ (ตัวอักษรจำนวนหนึ่ง) จากแป้นพิมพ์เข้ามาเก็บไว้ในตัวแปร gets มาจากคำว่า get string เช่น gets(n) โดยที่ n เป็นชื่อตัวแปรชนิดที่เก็บค่าข้อความ โดยรับค่าข้อความจากแป้นพิมพ์ ฟังก์ชันจะทำการใส่ ‘ \0 ‘ เอาไว้ที่ตัวสุดท้ายของข้อความ เพื่อแสดงการสิ้นสุดของข้อความที่รับเข้ามาเมื่อผู้ใช้กดปุ่ม enter

รูปแบบ

<b>gets ( ชื่อตัวแปร ) ;</b>
------------------------------

## ตัวอย่าง 2.20

1	#include <stdio.h>
2	main( )
3	{ <code>char</code> name[10];
4	gets(name) ;
5	}

**ผลลัพธ์จากการประมวลผลโปรแกรม**

เครื่องจะจองที่ตัวแปรชุดที่ชื่อ name ซึ่งเป็นอักขระ ไขว้ 10 ตัว และรอรับค่าที่เป็นข้อความเข้ามาเก็บไว้ในตัวแปรชุดที่ชื่อ nameได้ยาวไม่เกิน 9 ตัวอักษรเพื่อให้ name ตัวที่ 10 (ตัวสุดท้าย) เก็บ \0 เอาไว้

## 4. ฟังก์ชัน putchar( )

putchar ( ) เป็นฟังก์ชันที่ใช้ให้คอมพิวเตอร์แสดงผลบนจอภาพทีละ 1 ตัวอักษร

รูปแบบ

**putchar (ชื่อตัวแปร) ;**

#### ตัวอย่าง 2.21

```
1  #include<stdio.h>
2  main ( )
3  {   char x;
4      x=getch ( ) ;
5      printf ("The result is \n") ;
6      putchar ( x );
7  }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

ลักษณะข้อมูลที่ป้อน

A

ผลลัพธ์

The result is

A

5. ฟังก์ชัน puts ( ) เป็นฟังก์ชันที่ทำงานกับข้อความโดยเฉพาะ

puts ( ) เป็นฟังก์ชันที่ใช้แสดงผลข้อมูลที่เป็นข้อความที่เก็บไว้ในตัวแปรชุดออกมาบนจอภาพ puts()

มาจากคำว่า put string ใช้สำหรับพิมพ์สตริงออกทางจอภาพ

รูปแบบ

**puts ( ชื่อตัวแปร ) ;**

เช่น puts(n) เมื่อ n เป็นตัวแปรชุดที่ต้องการนำค่ามาแสดงผล

#### ตัวอย่าง 2.22

```
1  #include <stdio.h>
2  main ( )
3  {   char name [10];
4      gets (name) ;
5      puts (name);
6  }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

เครื่องจะนำค่าที่เก็บในตัวแปรชุด name มาแสดงผลบนจอภาพ

6. ฟังก์ชัน getche ( )

getche ( ) เป็นฟังก์ชันในการรับข้อมูล 1 ตัวอักษรโดยจะปรากฏตัวอักษรให้เห็นในการป้อนข้อมูล

โดยไม่ต้องกด Enter การใช้ getche() จะต้องทำการ #include <conio.h> จึงจะสามารถใช้ getche() ได้

รูปแบบ

**ชื่อตัวแปร = getche ( ) ;**

## 2.6.4 ตารางรหัสแอสกี สามารถแสดงได้ดังตารางต่อไปนี้

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: [www.LookupTables.com](http://www.LookupTables.com)

## แบบฝึกหัดปฏิบัติการคาบที่ 2: Basic C Programming

1. ให้เขียนคำสั่งเพื่อประกาศตัวแปรเพื่อใช้เก็บค่าต่อไปนี้

1.1 อุณหภูมิ

---

1.2 ความยาวของเส้นรอบวงกลม

---

1.3 จำนวนรถที่ผ่านสี่แยกแต่ละช่วงเวลา

---

1.4 ระดับคะแนนของนักเรียน คือ A B C D F

---

1.5 คะแนนเฉลี่ยของนักเรียน

---

2. กำหนดให้  $a=5$   $b=3$   $c=2$   $d=0.5$  ให้แสดงค่าของ  $y$  ถ้ากำหนด  $y$  มีชนิดข้อมูล float

2.1  $y=a*b+c;$

---

2.2  $y=b+c*b;$

---

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

---

2.4  $y=c\%5;$

---

2.5  $y=a/c;$

---

2.6  $y=a/d$

---



3. ให้เขียนโปรแกรมสำหรับคำนวณค่าความยาว (length) โดยกำหนดพื้นที่ (area) ความกว้าง (width) และความยาว เป็นข้อมูลชนิดจำนวนจริง และรับข้อมูลพื้นที่และความกว้าง จากผู้ใช้ แสดงผลลัพธ์บนจอภาพ

```

/* 1 */ #include <stdio.h>
/* 2 */
/* 3 */ int main ()
/* 4 */ {
/* 5 */     float    width , length ;           /* Declaration of Variables */
/* 6 */     float    area ;

/* 7 */     printf("Please enter area: ") ;           /* Read data */
/* 8 */     scanf("%f", &area) ;
/* 9 */     printf("Please enter width: ") ;
/* 10 */    scanf("%f", &width) ;
/* 11 */    length = area / width ;                 /* Expression Statements */
/* 12 */    printf ("Area = %f , width = %f and length = %f \n", area, width, length) ;
/* 13 */
/* 14 */    return 0 ;
/* 15 */ }

```

<p>3.1 รันโปรแกรมโดยใส่ข้อมูล</p> <p>257.5 ↵ และ</p> <p>10 ↵</p> <p>จะได้ผลลัพธ์คือ</p>	<p>.....</p> <p>.....</p> <p>.....</p>
<p>3.2 ถ้าแก้ไขบรรทัดที่ 12 เป็น</p> <p>printf ("Area = %7.4f , width = %7.3f and length = %7.2f \n", area, width, length) ;</p> <p>และรันโปรแกรมโดยใส่ข้อมูล 45.0789 ↵ และ</p> <p>12.50 ↵</p> <p>จะได้ผลลัพธ์คือ</p>	<p>.....</p> <p>.....</p> <p>.....</p>
<p>3.3 ถ้าแก้ไขบรรทัดที่ 12 เป็น</p> <p>printf ("Area = %7.3f , width = %7.5f and length = %7.7f \n", area, width, length) ;</p> <p>และรันโปรแกรมโดยใส่ข้อมูล 45.0789 ↵ และ</p> <p>12.50 ↵</p> <p>จะได้ผลลัพธ์คือ</p>	<p>.....</p> <p>.....</p> <p>.....</p>

3.4 ถ้าสลับบรรทัดที่ 9 และ 10 จะได้ผลลัพธ์คือ	<p>.....</p> <p>.....</p>
---	---------------------------

4. กำหนดค่าของตัวแปรดังนี้

```
#define commission 5000.00
```

```
#define percent 0.05
```

```
#define no_of_day 7
```

จงเขียนโปรแกรมเพื่อแสดงข้อความและตัวแปรให้มีผลการกระทำการดังรูป

r	a	t	e	:				0	.	0	5	%		
c	o	m	m			:	5	0	0	0	.	0	0	
n	o	.	o	f		d	a	y	:	7	d	a	y	s
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

5. จงเขียนผังงานและโปรแกรมสำหรับให้ผู้ใช้ป้อนอุณหภูมิ 3 ค่าเป็นฟาเรนไฮต์แล้วแปลงอุณหภูมิทั้งสามค่าเป็นเซลเซียสตามสูตร  $C = 5 * (F - 32) / 9$  แล้วพิมพ์ผลลัพธ์ออกมาดังนี้

Please input temperature1 (F): 15

Please input temperature2 (F): 32

Please input temperature3 (F): 0

Result:

temperature1:15 F is -9.44 C

temperature2:32 F is 0.00 C

temperature3:0 F is -17.77 C

วิเคราะห์ปัญหา

เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร      ความหมาย

### เขียนโปรแกรม

6. จงเขียนผังงานและโปรแกรมเพื่อคำนวณค่ากลางหรือค่ามัธยฐาน(Median) ของข้อมูล(input) 4 ค่าที่ได้รับจากคีย์บอร์ด(แบบเรียงค่าจากน้อยไปมาก) เพื่อเก็บในตัวแปร (x1,x2,x3,x4) และแสดงผลลัพธ์จากการคำนวณเมื่อ median คือค่ากลางของข้อมูลที่เรียงแล้วดังนี้

Please input data (x1-x4): 1 2 3 4

Result:

Median is 2.5

### วิเคราะห์ปัญหา

### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร      ความหมาย

### เขียนโปรแกรม

7. จงเขียนผังงานและโปรแกรมสำหรับรับค่าความสูงและรัศมีของทรงกระบอกแล้วคำนวณหาปริมาตรจากสูตรต่อไปนี้

ปริมาตรทรงกระบอก =  $3.1416 \times \text{ความสูง} \times \text{รัศมี} \times \text{รัศมี}$

โดยให้โปรแกรมทำงานดังตัวอย่างต่อไปนี้

Enter height and radius of the cylinder in cm: 3.0 4.0

Volume of the cylinder is 150.8

วิเคราะห์ปัญหา

เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร

ความหมาย

เขียนโปรแกรม

8. จงเขียนโปรแกรมเพื่อพิมพ์ใบส่งของ (Invoice) ตามรายละเอียดต่อไปนี้

ให้ลูกค้าทำการรายการโดยถามหมายเลขใบส่งของ (Invoice number) วันที่ส่ง (date) วันครบกำหนด (due date) และชื่อลูกค้า โดยให้ข้อมูลทั้งหมดเป็นแบบข้อความ (String)

Please enter the invoice number: A230/02

Please enter date: 19/11/2012

Please enter due date: 28/11/2012

Please enter the customer name: Apple Store

จากนั้นสมมติว่าลูกค้าต้องการซื้อ 3 รายการ แล้วถามชื่อสินค้า(ItemName) จำนวน (quantity) และราคา  
สินค้าต่อหน่วย (UnitPrice)

Please enter the name of item1: Iphone5

Please enter the quantity of item 1: 3

Please enter the unit price of item 1: 20000

Please enter the name of item2: Earphones

Please enter the quantity of item 2: 10

Please enter the unit price of item 2: 1000

Please enter the name of item3: USB cable

Please enter the quantity of item 3: 9

Please enter the unit price of item 3: 500

คำนวณราคารวมของสินค้าแต่ละรายการ (TotalPrice) และราคารวมของสินค้าทุกรายการ (TotalAmount)

คำนวณ Vat 7%ของราคารวม และคำนวณยอดรวมทั้งหมด (AmountDue)

แสดงผลลัพท์ใบส่งของซึ่งมีลักษณะดังนี้

Invoice No.: A230/02

Date: 19/11/2012

Customer: Apple Store

Due Date: 28/11/2012

#	Item Name	Unit Price	Quantity	Total Price
1	Iphone5	20000.00	3	60000.00
2	Earphones	1000.00	10	10000.00
3	USB cable	500.00	9	4500.00

Total Amount :	74500.00
VAT:	3725.00
Amount Due:	78225.00

---

### วิเคราะห์ปัญหา

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร

ความหมาย

เขียนผังงาน

### เขียนโปรแกรม

## บทที่ 3

### ตัวดำเนินการและนิพจน์ (Operator & Expression)

#### จุดประสงค์

1. เพื่อให้ทราบความหมายของตัวดำเนินการ
2. เพื่อให้ทราบหน้าที่ของตัวดำเนินการประเภทต่าง ๆ
3. เพื่อให้ทราบความหมายของนิพจน์

#### 3.1 คำสั่งและนิพจน์ (Statement and Expression)

**คำสั่ง, ข้อคำสั่ง (Statement)** คือขั้นตอนในการทำงานหนึ่งขั้นตอน ทุกคำสั่งต้องจบด้วยเครื่องหมาย;

**กลุ่มคำสั่ง** คือคำสั่งที่อยู่ในวงเล็บปีกกา {}

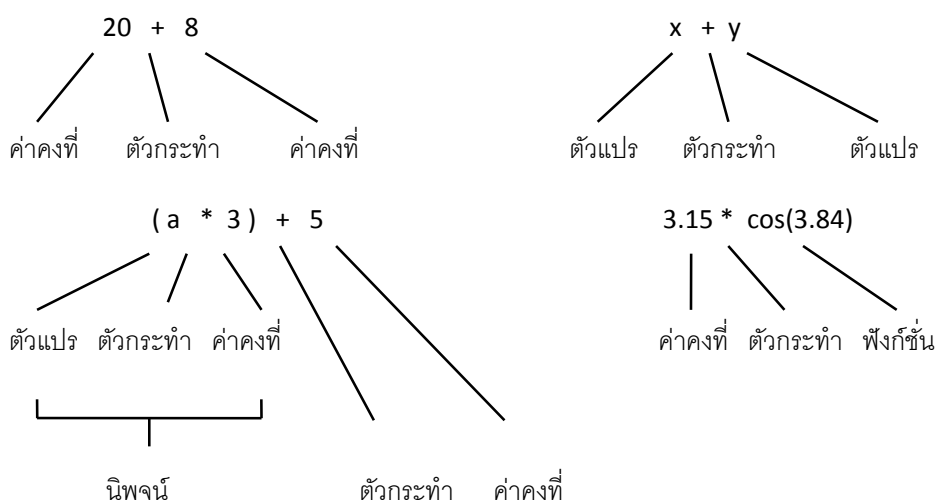
**นิพจน์ (Expression)** คือการกระทำเพื่อให้ได้ผลลัพธ์ค่าหนึ่งค่า ประกอบไปด้วยตัวถูกกระทำ (Operands) และตัวกระทำ (Operators) เขียนเรียงกันไป เช่น  $3 * 2 - 1 + 7$  หรือ  $a * 5$  เป็นต้น

ในภาษา C นิพจน์หมายถึง สิ่งที่จะประมวลผลแล้วสามารถให้เป็นค่าตัวเลขได้ ซึ่งแต่ละนิพจน์จะมีระดับความยากง่ายในการประมวลผลที่ต่างกัน นิพจน์ที่มีระดับการประมวลผลแบบง่ายที่สุด จะประกอบด้วยตัวแปรเพียงตัวเดียว หรือค่าคงที่นิพจน์ที่มีลักษณะเป็นค่าคงที่ เช่น

100

'a'

ตัวอย่างการเขียนนิพจน์ เช่น



### ตัวอย่างการเขียนนิพจน์คณิตศาสตร์และนิพจน์ภาษา C

นิพจน์คณิตศาสตร์	นิพจน์ภาษา C
$a+b/cd$	$(a+b)/(c*d)$
$10x+3xy+10y^2$	$10*x+3*x*y+10*y*y$

นิพจน์แบ่งออกเป็น

#### 1. นิพจน์คณิตศาสตร์ (Arithmetic Expression)

หมายถึง การนำตัวแปร ค่าคงที่มาสัมพันธ์กันโดยใช้เครื่องหมายคณิตศาสตร์เป็นตัวเชื่อมผลที่ได้จากนิพจน์แบบนี้จะเป็นตัวเลข เช่น  $(a+b)/(c*d)$

#### 2. นิพจน์ตรรก (Logical Expression)

หมายถึงการนำตัวแปรค่าคงที่หรือนิพจน์มาสัมพันธ์กันโดยใช้เครื่องหมายเปรียบเทียบและเครื่องหมายตรรกะเป็นตัวเชื่อม ผลที่ได้จะออกมาเป็นจริงหรือเท็จ เช่น  $a\&b$ ,  $a>b$

กฎเกณฑ์การเขียนนิพจน์

- ห้ามเขียนตัวแปร 2 ตัวติดกันโดยไม่มีเครื่องหมายเช่น  $ab$
- ถ้าเขียนนิพจน์โดยมีค่าของตัวแปร หรือค่าคงที่ต่างชนิดกันในนิพจน์เดียวกันภาษาซีจะเปลี่ยนชนิดของข้อมูลที่มีขนาดเล็กให้เป็นชนิดข้อมูลที่มีขนาดใหญ่ขึ้น

## 3.2 เครื่องหมายดำเนินการ (Operators)

เครื่องหมายดำเนินการ (Operators) หรือ ตัวดำเนินการ นั้นกำหนดการกระทำที่เกิดขึ้นกับตัวแปร และค่าคงที่ โดยที่นิพจน์ประกอบด้วยตัวแปร และค่าคงที่ และใช้ตัวดำเนินการคำนวณเพื่อให้ได้ค่า ภาษาซีได้แบ่งตัวดำเนินการออกเป็น 3 ประเภท ได้แก่ ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการสัมพันธ์และตัวดำเนินการตรรกะ และ ตัวดำเนินการประกอบ นอกจากนี้ยังมีโอเปอเรเตอร์พิเศษอีกหลายตัวซึ่งได้แก่ ตัวดำเนินการบอกชนิดตัวแปร (type) และตัวดำเนินการบอกขนาดหน่วยความจำของตัวแปร (size of) เป็นต้น

### 3.2.1 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

ตัวดำเนินการทางคณิตศาสตร์ เป็นตัวดำเนินการที่คำนวณทางด้านคณิตศาสตร์ให้มีค่าออกมามีชนิดสอดคล้องกับชนิดของตัวถูกกระทำ ประกอบด้วยตัวดำเนินการต่างๆ ดังนี้

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	การบวก	$A+B$
-	การลบ	$A-B$
*	การคูณ	$A*B$



/	การหาร	A/B
%	การหารที่เก็บเศษเหลือจากการหาร (Modulo)	5%3=1 เศษ 2 จะเก็บเศษ 2 เอาไว้
--	การลดค่าลงครั้งละ 1	A -- จะเหมือนกับ A=A-1
++	การเพิ่มค่าขึ้นครั้งละ 1	A++ จะเหมือนกับ A=A+1

ตารางที่ 3.1 แสดงประเภทของตัวดำเนินการทางคณิตศาสตร์

การโอเปอเรชัน (Operation) หรือการทำงานของตัวดำเนินการต้องเป็นไปตามลำดับของการวางค่าคงที่หรือตัวแปรที่จะกระทำต่อกัน ซึ่งเรียกว่า **ลำดับของการโอเปอเรชัน** ซึ่งลำดับการทำงานของโอเปอเรเตอร์โดยคอมพิวเตอร์ภาษา C ต้องเป็นไปตาม **กฎการกำหนดลำดับการทำงานของตัวดำเนินการ (Precedence)** ดังแสดงในตารางที่ 3.2

โอเปอเรเตอร์	ลำดับของการโอเปอเรชัน
( )	ซ้ายไปขวา
- แสดงความเป็นลบ ของตัวเลข	ซ้ายไปขวา
* /	ซ้ายไปขวา
+ -	ซ้ายไปขวา

ตารางที่ 3.2 แสดงลำดับการทำงานของตัวดำเนินการทางคณิตศาสตร์ ตามกฎ Precedence

ในการคำนวณ  $a + b * c$  ภาษา C จะกระทำเครื่องหมาย \* ก่อนเครื่องหมาย + แต่ถ้าเครื่องหมายมีความเท่าเทียมกันจะกระทำจากซ้ายไปขวา เช่น  $a+b+c$  จะเริ่มทำจาก a บวก b แล้วจึงนำค่าที่ได้มาบวก c ถ้าต้องการกำหนดลำดับให้ทำโดยการใส่เครื่องหมายวงเล็บในส่วนที่ต้องการให้ทำงานก่อน

ตัวดำเนินการ % เป็นตัวดำเนินการทางคณิตศาสตร์เพียงตัวเดียวที่กำหนดให้ใช้กับค่าจำนวนเต็มเท่านั้น นอกนั้นสามารถใช้กับค่าอื่นๆ ได้

ตัวดำเนินการทางคณิตศาสตร์ที่น่าสนใจอีกตัวหนึ่งคือ เครื่องหมาย / เช่น  $a / b$  ถ้าทั้งตัวแปร a และ b เป็น integer ค่าที่ได้จากการดำเนินการของหารจะมีชนิดเป็น integer และถ้า a หรือ b ตัวแปรใดตัวแปรหนึ่งเป็นตัวแปรชนิด float จะได้คำตอบเป็นชนิด float เช่น

$39 / 5$	จะมีค่าเท่ากับ 7
$39.0 / 5$	จะมีค่าเท่ากับ 7.8

จะเห็นได้ว่ามีความแตกต่างที่สำคัญของเครื่องหมายหาร คือ เครื่องหมายหารสำหรับตัวแปร integer และตัวแปรแบบ floating ขึ้นอยู่กับชนิดของ operand ที่เราใช้กับเครื่องหมายหารนั้นๆ เช่น  $1/3$

(ซึ่งมีค่าเท่ากับ 0) แทนที่จะเป็น 1.0/3.0 ซึ่งมีค่าเท่ากับ 0.333... โดยการที่การเขียน 1.0/3.0 เราสามารถเขียนได้เป็นว่า 1.0/3 หรือ 1/3.0 และในแต่ละแบบของนิพจน์จะไม่เป็น 0

### ตัวดำเนินการเพิ่ม และลดค่า (Increment & Decrement)

ตัวดำเนินการเพิ่ม และลดค่า เป็นตัวดำเนินการในการเพิ่มหรือลดค่าของตัวแปร โดย

- ตัวดำเนินการเพิ่มค่า ++ จะบวกหนึ่งเข้ากับตัวถูกดำเนินการ
- ตัวดำเนินการลดค่า -- จะลบหนึ่งออกจากตัวถูกดำเนินการ

ตัวดำเนินการเพิ่ม และลดค่ามีวิธีใช้งาน 2 แบบคือ

- ใช้เป็นตัวดำเนินการแบบมาก่อน (prefix) เช่น ++n
- ใช้ตัวดำเนินการแบบตามหลัง (postfix) ก็ได้เช่น n++

ในทั้งสองกรณี ผลลัพธ์คือการเพิ่มค่า 1 ให้กับ n แต่นิพจน์ ++n จะเพิ่มค่าก่อนที่จะนำค่าไปใช้ ขณะที่ n++ จะเพิ่มค่าหลังจากนำค่าไปใช้ ซึ่งหมายความว่าถ้ามีการนำค่าไปใช้ (ไม่เพียงหวังเฉพาะผลลัพธ์) ++n และ n++ จะแตกต่างกัน ตัวอย่างดังนี้

**ตัวอย่างที่ 3.1** ถ้า n มีค่า 5

x = n++;      จะเป็นการกำหนดค่า x ให้กับ 5

x = ++n;      จะเป็นการกำหนดค่า x ให้กับ 6

ตัวดำเนินการนี้จะใช้ได้กับเฉพาะตัวแปรเท่านั้น จะใช้กับนิพจน์อื่นๆ ไม่ได้เช่น (i+j)++

### **3.2.2 ตัวดำเนินการสัมพันธ์และตัวดำเนินการตรรกะ (Relational, Equality, and Logical Operators)**

Operator	เครื่องหมาย	ความหมาย
ตัวดำเนินการสัมพันธ์ (Relational Operator)	<	น้อยกว่า
	>	มากกว่า
	<=	น้อยกว่า หรือ เท่ากับ
	>=	มากกว่า หรือ เท่ากับ
ตัวดำเนินการเท่ากับ (Equality Operator)	==	เท่ากับ
	!=	ไม่เท่ากับ
ตัวดำเนินการตรรกะ (Logical Operator)	!	นิเสธ
	&&	และ
		หรือ

ตารางที่ 3.3 แสดงตัวดำเนินการสัมพันธ์และตัวดำเนินการตรรกะ

#### 1 ตัวดำเนินการสัมพันธ์ (Relational Operator)

ตัวดำเนินการสัมพันธ์ เป็นเครื่องหมายที่ใช้ในการเปรียบเทียบและตัดสินใจ ซึ่งผลของการเปรียบเทียบจะเป็นได้ 2 กรณีเท่านั้นคือ จริงและเท็จ ภาษาซีถือว่าค่าทางตรรกจริงและเท็จมีชนิดเป็น int ดังนั้นผลการกระทำทางตรรกจึงมีค่าเป็นจำนวนเต็ม และมีค่าได้เพียงสองค่าคือ 1 หรือตัวเลขใดๆ แทนค่าความจริงเป็นจริง และ 0 แทนค่าความจริงเป็นเท็จ (0 เป็นเท็จ ส่วนตัวเลขอื่นๆ ทั้งหมดมีค่าเป็นจริง) เครื่องหมายที่เป็นตัวดำเนินการสัมพันธ์มี 4 ตัวคือ

< (น้อยกว่า) > (มากกว่า) <= (น้อยกว่าเท่ากับ) และ >= (มากกว่าเท่ากับ)

ตัวอย่างการใช้งาน เช่น

```
a<3 /* ถ้า a มีค่าเป็น 5 ประโยคนี้จะได้ผลลัพธ์เป็น 0 (เท็จ) */
```

```
a>b /* ถ้า a=10 b=8 ประโยคนี้จะได้ผลลัพธ์เป็น 1 (จริง) */
```

```
a<b<c /* รูปแบบถูกต้อง แต่ลำดับการวางตัวแปรอาจทำให้สับสนได้ โดยจะทำการเปรียบเทียบค่า b และ c ก่อน จึงนำผลลัพธ์มาเปรียบเทียบกับ a */
```

ตัวอย่างการใช้งานผิดแบบ

```
a =< b /* ผิดลำดับ เครื่องหมาย = มาก่อน < */
```

```
a <= b /* ผิดรูปแบบ ห้ามเว้นวรรคระหว่าง < และ = */
```

## 2 ตัวดำเนินการเท่ากับ (Equality Operator)

ตัวดำเนินการเท่ากับ ใช้ในการเปรียบเทียบค่า 2 ค่า ว่ามีค่าเท่ากันหรือไม่ ผลลัพธ์ที่ได้จะมีเพียง 2 ค่า เช่นเดียวกับผลลัพธ์ของตัวดำเนินการสัมพันธ์คือ จริง และ เท็จ เครื่องหมายที่เป็นตัวดำเนินการเท่ากับมี 2 ตัวคือ

== (เท่ากับ) และ != (ไม่เท่ากับ)

ตัวอย่างการใช้งาน เช่น

```
c == 'A' /* ถ้าตัวแปร c เก็บค่าอักขระ A ผลลัพธ์จะได้ค่าจริง */
```

```
k != -2 /* ถ้าตัวแปร k เก็บค่าตัวเลข -2 ผลลัพธ์จะได้ค่าเท็จ */
```

```
x+y == 2 * z - 5
```

ตัวอย่างการใช้งานผิดแบบ

```
a=b /* การเปรียบเทียบค่าตัวแปร a กับ b ว่ามีค่าเท่ากันหรือไม่
```

```
กลายเป็นการให้ค่าตัวแปร a ด้วยค่าตัวแปร b */
```

```
a= =b -1 /* ใช้งานผิดรูปแบบ โดยมีช่องว่างระหว่างเครื่องหมาย = ทั้ง 2 ตัว */
```

## 3 ตัวดำเนินการตรรกะ (Logical Operator)

ตัวดำเนินการตรรกะ ใช้ในการเปรียบเทียบ และกระทำทางตรรกะกับค่าตัวเลข หรือค่าที่อยู่ในตัวแปร ผลลัพธ์ที่ได้ จะมีเพียง 2 ค่าเช่นเดียวกับผลลัพธ์ของตัวดำเนินการสัมพันธ์คือ จริง และ เท็จ เครื่องหมายที่เป็นตัวดำเนินการทางตรรกะ มี 3 ตัวคือ

! (นิเสธ) && (และ) และ || (หรือ)

นิเสธ คือการกลับค่าความจริงของค่าตัวเลข หรือตัวแปรที่ตามหลังเครื่องหมาย

### ตัวดำเนินการนิเสธ

ต้องการตัวถูกกระทำเพียง 1 ตัวเท่านั้น ตัวอย่างการใช้งาน

!a /\* ถ้าค่าความจริงของตัวแปร a มีค่าจริง ผลลัพธ์จากการนิเสธจะได้ค่าเป็นเท็จ \*/

!(x+7.7)

!(a<b || c<d) /\* a มากระทำกับ b แล้วเอาผลลัพธ์มา || กับ ผลลัพธ์ระหว่างการกระทำระหว่าง c กับ d จากนั้นนำผลลัพธ์ที่ได้จากการ || มา ทำการนิเสธอีก \*/

ตัวอย่างการใช้งานผิดแบบ

a! /\* ผิดลำดับ เครื่องหมาย ! มาก่อนตัวแปร \*/

a! = b /\* ผิดความต้องการ โดยความต้องการคือ กลายเป็นการใช้เครื่องหมาย != ไป \*/

### ตัวดำเนินการ && และ ||

ต้องการตัวถูกกระทำ 2 ตัว ตัวอย่างการใช้งาน

a&& b /\* a และ b ถ้าค่าความจริงของ a เป็นจริง และ ค่าความจริง b เป็นจริง ผลลัพธ์ที่ได้จะมีค่าความจริงเป็นจริง \*/

a || b /\* a หรือ b ถ้าค่าความจริงของ a เป็นเท็จ และ ค่าความจริง b เป็นเท็จ ผลลัพธ์ที่ได้จะมีค่าความจริงเป็นเท็จ \*/

!(a<b) && C /\* นำ a มาเปรียบเทียบกับ b แล้วนำผลลัพธ์มานิเสธ จากนั้นมา && กับ C \*/

3 && (-2 \* a + 7) /\* หาค่าภายในวงเล็บก่อน แล้วนำค่าผลลัพธ์ที่ได้มา && กับ 3 \*/

ตัวอย่างการใช้งานผิดแบบ

a && /\* ตัวกระทำไม่ครบ \*/

a| |b /\* เว้นช่องว่างระหว่างเครื่องหมาย || \*/

a & b /\* เครื่องหมาย & ไม่ครบ \*/

&b /\* ตำแหน่งในหน่วยความจำของตัวแปร b \*/

### การหาค่าของตัวดำเนินการตรรกะ

- && (และ)

นำค่าความจริง 2 ค่ามาเปรียบเทียบกัน ได้ผลของการเปรียบเทียบตามตารางที่ 3.4

P	Q	P && Q
0	0	0
0	1	0
1	0	0
1	1	1

ตารางที่ 3.4 แสดงผลของตัวดำเนินการ && ระหว่างตัวแปร P และ Q

- || (หรือ)

นำค่าความจริง 2 ค่ามาเปรียบเทียบกัน ได้ผลของการเปรียบเทียบตามตารางที่ 3.5

P	Q	P    Q
0	0	0
0	1	1
1	0	1
1	1	1

ตารางที่ 3.5 แสดงผลของตัวดำเนินการ || ระหว่างตัวแปร P และ Q

- ! (นิเสธ)

นำค่าความจริงมาเปรียบเทียบ ได้ผลของการเปรียบเทียบตามตารางที่ 3.6

P	!P
0	1
1	0

ตารางที่ 3.6 แสดงผลของตัวดำเนินการ ! กับตัวแปร P

### ตัวอย่างที่ 3.2

```

1  /* Demonstrates the results of relational operators.
2      Written by:
3      Date:
4  */
5  #include <stdio.h>
6  int main (void)
7  {
8      // Local Declarations
9      int a = 5;
10     int b = -3;
11     // Statements *
12     printf(" %2d < %2d is %2d\n", a, b, a < b);
13     printf(" %2d == %2d is %2d\n", a, b, a == b);
14     printf(" %2d != %2d is %2d\n", a, b, a != b);
15     printf(" %2d > %2d is %2d\n", a, b, a > b);
16     printf(" %2d <= %2d is %2d\n", a, b, a <= b);
17     printf(" %2d >= %2d is %2d\n", a, b, a >= b);
18     return 0;
19 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

5 < -3 is 0
5 == -3 is 0
5 != -3 is 1
5 > -3 is 1
5 <= -3 is 0
5 >= -3 is 1

```

### 3.2.3 ตัวดำเนินการประกอบ (Compound Operator)

ตัวดำเนินการประกอบ คือ ตัวดำเนินการที่เป็นรูปแบบย่อ ของตัวดำเนินการและตัวแปรที่ถูกดำเนินการ โดยตัวดำเนินการประกอบในภาษาซี มีทั้งหมด 10 ตัวดังนี้

`+=      -=      /=      %=  
<<=    >>=    |=      ^=`

การกำหนดรูปแบบของตัวดำเนินการประกอบให้กับตัวแปรมีรูปแบบดังนี้

**Variable operation= expression;**

จะมีความหมายเท่ากับ

**Variable = variable operation expression;**

<code>i = i + 1;</code>	ตัวดำเนินการประกอบคือ <code>i += 1;</code>
<code>i = i - a;</code>	ตัวดำเนินการประกอบคือ <code>i -= a;</code>
<code>i = 1 * (a + 1);</code>	ตัวดำเนินการประกอบคือ <code>i *= a+1;</code>
<code>i = i / (a-b);</code>	ตัวดำเนินการประกอบคือ <code>i /= a-b;</code>
<code>i = i &lt;&lt; 1;</code>	ตัวดำเนินการประกอบคือ <code>i &lt;&lt;= 1;</code>
<code>i = i &gt;&gt; i;</code>	ตัวดำเนินการประกอบคือ <code>i &gt;&gt;= i;</code>
<code>i = i   01;</code>	ตัวดำเนินการประกอบคือ <code>i  = 0xf;</code>
<code>i = i &amp; 01;</code>	ตัวดำเนินการประกอบคือ <code>i &amp;= 0xf;</code>
<code>i = i ^ (07   0xb);</code>	ตัวดำเนินการประกอบคือ <code>i ^= 07   0xb;</code>

### 3.2.4. ตัวกระทำบอกขนาด (Sizeof Operator)

เป็นตัวกระทำใช้รายงานขนาดของหน่วยความจำที่ใช้ในการเก็บค่า โดยมีรูปแบบดังนี้

**sizeof ค่าคงที่ หรือ sizeof (แบบของตัวแปร)**

ตัวอย่าง 3.3

1	<code>#include&lt;stdio.h&gt;</code>
2	<code>int main()</code>
3	<code>{</code>
4	<code>    int ABC;</code>
5	<code>    printf("%d",sizeof(ABC));</code>
6	<code>    return 0;</code>
7	<code>}</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
4	

จะเห็นว่าการพิมพ์ค่าขนาดของตัวแปรชื่อ ABC ที่มีชนิดเป็น integer ออกทางหน้าจอ ค่า 4 คือขนาดของ integer ทั่วไบนั่นเอง

### 3.2.5. ตัวดำเนินการแบบมีเงื่อนไข (Conditional Operator)

ตัวกระทำสำหรับเลือกค่า เป็นตัวกระทำใช้ในการเลือกการให้ค่าของนิพจน์ ตัวกระทำชนิดนี้ประกอบไปด้วยเครื่องหมาย ? และ นิพจน์เลือกค่า (Condition Expressions) จะเขียนอยู่ในรูป

นิพจน์1 ? นิพจน์2 : นิพจน์3

นิพจน์เลือกค่ามีการให้ค่าดังนี้คือ หากค่าในนิพจน์1 มีค่าไม่เท่ากับ 0 (เป็นจริง) จะให้ค่านิพจน์เป็นไปตามนิพจน์ที่ 2 แต่ถ้าค่าในนิพจน์1 เป็น 0 จะให้ค่านิพจน์เป็นไปตามนิพจน์ที่ 3 เช่น

$c = ((a+5)? (a+1) : 0);$

สมมติให้ค่า a เป็น 1 ซึ่งทำให้นิพจน์ a+5 เป็นจริง ก็จะทำให้ค่า c เป็น 2 ซึ่งได้จาก a+1 และถ้าสมมติให้ a เป็น -5 ก็จะทำให้นิพจน์ a+5 เป็นเท็จ c ก็จะได้รับค่า 0 ดังตัวอย่างต่อไปนี้

#### ตัวอย่างที่ 3.4

1	#include <stdio.h>
2	int main()
3	{
4	int a,b;
5	a = 15;
6	b = ((a+5)? (a+1) : 0);
7	printf("b= %d\n",b);
8	a=0;
9	b=((a+5)?(a+1) : 0);
10	printf("b= %d\n",b);
11	a=5;
12	b=((a+5)?(a+1) : 0);
13	printf("b= %d\n",b);
14	return 0;
15	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
b= 16	
b= 1	
b= 6	

### 3.2.6 ลำดับการทำงานก่อน-หลังของตัวดำเนินการ

ลำดับการทำงานก่อน-หลังของตัวดำเนินการเป็นไปตามตารางที่ 3.7

ตัวดำเนินการ	ทิศทางการดำเนินการ
(), [], ->,	ซ้ายไปขวา
!, ~, ++, --, +(ค่าบวก), -(ค่าลบ), *, &(type), sizeof	ขวาไปซ้าย
*, /, %	ซ้ายไปขวา
+, - (ตัวกระทำทางคณิตศาสตร์)	ซ้ายไปขวา
<<, >>	ซ้ายไปขวา

<, <=, >, >=	ซ้ายไปขวา
==, !=	ซ้ายไปขวา
&	ซ้ายไปขวา
^	ซ้ายไปขวา
	ซ้ายไปขวา
&&	ซ้ายไปขวา
	ซ้ายไปขวา
? :	ขวาไปซ้าย
=, +=, -=, /=, %=, &=, ^=,  =, <<=, >>=	ขวาไปซ้าย
,	ซ้ายไปขวา

ตารางที่ 3.7 แสดงลำดับความสำคัญของตัวดำเนินการทั้งหมด เรียงลำดับจากมากไปน้อย

### 3.2.7 การเปลี่ยนแปลงค่าผลลัพธ์เป็นตัวแปรชนิดใหม่ (Casting)

ผลของการกระทำของนิพจน์จะให้ค่าออกมาค่าหนึ่งเสมอ ค่าที่ได้จะมีชนิดสอดคล้อง กับตัวกระทำ และตัวถูกกระทำภายในนิพจน์นั้น ๆ เช่น ตัวถูกกระทำเป็น int ค่าที่ได้จะเป็นชนิด int ด้วย เราอาจเปลี่ยนชนิดของค่า นั้น ๆ ให้มีชนิดตามที่เรต้องการได้โดย การเขียนชนิดของข้อมูลแบบใหม่ ภายในวงเล็บ นำหน้า นิพจน์นั้น ๆ ดังรูปแบบต่อไปนี้

(แบบข้อมูลแบบใหม่) นิพจน์

#### ตัวอย่างที่ 3.5

(int)(a\*5.2)                      เป็นการเปลี่ยนค่า output เป็นข้อมูลชนิด int

(float)(b+5)                      เป็นการเปลี่ยนค่า output เป็นข้อมูลชนิด float



## แบบฝึกหัดปฏิบัติการคาบที่ 3: Operator & Expression

1. จงเขียนนิพจน์ที่กำหนดในรูปของนิพจน์ทางคอมพิวเตอร์ แล้วตอบคำถามข้อ 1.1 - 1.5

นิพจน์ทางคณิตศาสตร์	นิพจน์ทางคอมพิวเตอร์
1. $\left(3\frac{a}{5} + \frac{1}{b}\right)$	
2. $\left(\frac{3a+5b}{2+c}\right)$	
3. $\frac{2}{7}\left((4^{3+c}) - 5d\right)$	
4. $\sqrt{\frac{2+8b}{a}}$	
5. $\sqrt[3]{b^2 - 4d}$	

1.1 จากนิพจน์ข้างต้น ตัวแปร a มีค่าเป็น 0 ได้หรือไม่ ตอบ..... เพราะ.....

1.2 จากนิพจน์ข้างต้น ตัวแปร b มีค่าเป็น 0 ได้หรือไม่ ตอบ..... เพราะ.....

1.3 ถ้าตัวแปร c มีค่าเป็น -2 และ d = -2 นิพจน์ในข้อ 3 จะให้ผลลัพธ์เป็นเท่าใด ตอบ.....

1.4 ตัวแปร c เป็นเลขจำนวนเต็มหรือเลขจำนวนทศนิยมก็ได้ยกเว้นค่าใด ตอบ.....

1.5 ถ้าตัวแปร b มีค่าเป็น 2 และตัวแปร d มีค่าเป็น 1 นิพจน์ในข้อ 5 จะให้ผลลัพธ์เป็นเท่าใด ตอบ.....

2. เมื่อกำหนดให้ค่าของตัวแปรต่าง ๆ ในหน่วยความจำเป็นดังนี้

	หน่วยความจำ
i	10
j	3
x	1.525
y	-0.008
z	12.26
c	'A'
d	'F'

### 3.จงหาค่าของนิพจน์ต่อไปนี้

นิพจน์ทางคณิตศาสตร์	ค่าของนิพจน์
1. $(y-2)*(y+z)/j$	
2. $j\%(i-j)/(z-x)$	
3. $((i/3-1)+((j-1)*6)\%(i-9))*3$	
4. $-x+(y*y+4*x*z)/x$	
5. $(c/d)*(-d)$	
6. $!(c<99)$	
7. $!(i*j<c)$	
8. $(c==97)\&\&!(z>15)$	
9. $(z/2-j<x)   (i-j!=0)   (c>d)$	
10. $(j-i/j)>(d-c/d)$	

4. จงเขียนผังงานและโปรแกรมเพื่อรับค่าราคาต่อหน่วยของสินค้า จำนวนหน่วยที่ซื้อ เพื่อคำนวณหาค่าจำนวนเงินที่ลูกค้าต้องจ่าย ซึ่งมีการคำนวณภาษี 7% ด้วยพร้อมทั้งแสดงผลในรูปแบบต่อไปนี้

Please enter unit price: **90** (กดแป้น Enter)

Please enter number: **3** (กดแป้น Enter)

Total amount = 288.90 baht

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร    ชนิดตัวแปร    ความหมาย

#### เขียนโปรแกรม

5. จงเขียนผังงานและโปรแกรมเพื่อรับค่ามุมเป็นองศา (x) แล้วให้คำนวณหาค่า  $\sin(x)$  และ  $\cos(x)$  และแสดงผลในรูปแบบต่อไปนี้

Please enter angle in degree: **90** (กดแป้น Enter)

sine of **90.0** degree is 1.0000

cos of **90.0** degree is 0.0000

กำหนด ฟังก์ชันคำนวณ  $\sin(a)$ ,  $\cos(a)$  เมื่อต้องการคำนวณ sine ของมุม a และ cosine ของมุม a ตามลำดับ

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร      ชนิดตัวแปร      ความหมาย

#### เขียนโปรแกรม

5.1 ถ้ารันโดยใส่ข้อมูล 0 ผลลัพธ์ของโปรแกรมคือ	.....
5.2 ถ้ารันโดยใส่ข้อมูล 3.1415 ผลลัพธ์ของโปรแกรมคือ	.....

## การเขียนโปรแกรมส่งผ่าน Grader

6. [Seven] ที่ร้านสะดวกซื้อแห่งหนึ่งเมื่อทำการรับเงินจากลูกค้าจะทำการแยกเงินแต่ละราคาใส่ไว้ในช่องเก็บเงินที่ประกอบด้วยชนิดของเงินแต่ละราคา คือ 1000, 500, 100, 50, 20, 10, และ 1 บาท จงเขียนผังงานและโปรแกรมเพื่อที่จะรับจำนวนเงินจากลูกค้าเพื่อคำนวณหาจำนวนเงินแต่ละชนิดราคา

**ข้อมูลอินพุต** มี 1 บรรทัด ประกอบด้วยจำนวนตัวเลข 1 จำนวนเป็นจำนวนเงินจากลูกค้า ( $0 \leq a \leq 1000000$ )

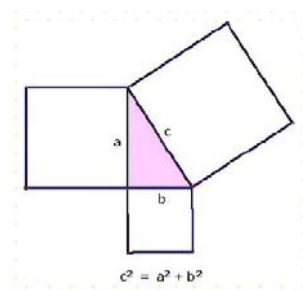
**ข้อมูลเอาต์พุต** มี 1 บรรทัด แสดงผลลัพธ์ที่ประกอบด้วยช่องเก็บเงินแต่ละชนิดราคา คือ 1000, 500, 100, 50, 20, 10, และ 1 บาท ตามลำดับ

**ตัวอย่าง**

อินพุต	เอาต์พุต
1751	1 1 2 1 0 0 1

7. [Pythagorus] รูปสามเหลี่ยมมุมฉาก มีมุมภายในมุมหนึ่งมีขนาด  $90^\circ$  (มุมฉาก) ด้านที่อยู่ตรงข้ามกับมุมฉากเรียกว่า ด้านตรงข้ามมุมฉาก ซึ่งเป็นด้านที่ยาวที่สุดในรูปสามเหลี่ยม อีกสองด้านเรียกว่า ด้านประกอบมุมฉาก

มีทฤษฎีที่เกี่ยวข้องกับสามเหลี่ยมมุมฉาก ทฤษฎีนั้นคือ ทฤษฎีบทพีทาโกรัส กล่าวไว้ว่า "ผลรวมของพื้นที่ของรูปสี่เหลี่ยมจัตุรัสบนด้านประชิดมุมฉากทั้งสอง จะเท่ากับ พื้นที่ของรูปสี่เหลี่ยมจัตุรัสบนด้านตรงข้ามมุมฉาก"



จงคำนวณความยาวของด้านตรงข้ามมุมฉาก เมื่อระบุความยาวของด้านประกอบมุมฉากทั้งสองด้านมาให้

**ข้อมูลอินพุต** บรรทัดแรก ประกอบไปด้วยจำนวนจริงบวก 2 จำนวน คั่นด้วยช่องว่าง 1 ช่อง แต่ละจำนวนจะบ่งบอกถึงความยาวของด้านประกอบมุมฉากของรูปสามเหลี่ยมรูปหนึ่ง

**ข้อมูลเอาต์พุต** บรรทัดแรกเพียงบรรทัดเดียว แสดงความยาวของด้านตรงข้ามมุมฉากของรูปสามเหลี่ยมมุมฉากที่มีด้านประกอบมุมฉากที่มีความยาวเท่ากับที่ระบุไว้ในข้อมูลนำเข้า ตอบเป็นทศนิยม 6 ตำแหน่ง

**ตัวอย่าง**

อินพุต	เอาต์พุต
3.000000 4.000000	5.000000

8. [GCD] จงเขียนโปรแกรมสำหรับหาค่า หรม. (หารร่วมมาก) หรือ GCD (Great Common Divisor) ของค่า 2 ค่าแล้วพิมพ์ผลลัพธ์ คือค่า GCD เช่น GCD ของ 150 และ 35 คือ 5

**ข้อมูลอินพุต** มี 1 บรรทัด ประกอบด้วยจำนวนตัวเลข 2 จำนวนที่เว้นด้วยช่องว่าง

**ข้อมูลเอาต์พุต** มี 1 บรรทัด แสดงค่าหารร่วมมากของตัวเลข 2 จำนวนจากข้อมูลอินพุต

**ตัวอย่าง**

อินพุต	เอาต์พุต
150 35	5

9. [กบ (frog)] มี เจ้ากบน้อยอยู่ตัวหนึ่ง สามารถกระโดดได้ในทุกทิศทางบนระนาบ และจะกระโดดเป็นระยะทางครั้งละ  $X$  หน่วยพอดี อยู่มาวันหนึ่ง เจ้ากบน้อยต้องการกระโดดจากจุด A ไปยังจุด B ซึ่งเป็นจุดบนระนาบ ที่ตั้งอยู่ห่างกัน  $Y$  หน่วย เจ้ากบน้อยอยากให้คุณช่วยหาว่า มันจะต้องกระโดดอย่างน้อยกี่ครั้ง จึงจะไปหยุดที่จุด B พอดี

จงเขียนโปรแกรมเพื่อรับจำนวนเต็ม  $X$  และ  $Y$  แล้วคำนวณหาจำนวนครั้งที่น้อยที่สุดที่เจ้ากบน้อยต้องใช้ในการกระโดดจากจุด A ไปยังจุด B

**ข้อมูลอินพุต** มีบรรทัดเดียว ระบุจำนวนเต็ม  $X$  และ  $Y$  ( $1 \leq X, Y \leq 1,000$ ) แทนระยะทางในการกระโดดแต่ละครั้งของเจ้ากบน้อย และระยะห่างระหว่างจุด A และจุด B

**ข้อมูลเอาต์พุต** มีบรรทัดเดียว แสดงจำนวนครั้งที่น้อยที่สุดที่เจ้ากบน้อยต้องใช้ในการกระโดดจากจุด A ไปยังจุด B

**ตัวอย่าง**

อินพุต	เอาต์พุต
3 12	4
5 23	5

## บทที่ 4

### คำสั่งควบคุม (Control Statement)

#### จุดประสงค์

เพื่อให้ทราบความหมายของคำสั่งควบคุมการทำงานของโปรแกรมแบบเลือกทำ เช่น if, if-else, switch

โดยปกติการทำงานของคอมพิวเตอร์จะทำงานเรียงลำดับคำสั่งลงมา ตั้งแต่ต้นโปรแกรมจนจบโปรแกรม แต่ถ้าต้องการเปลี่ยนแปลงขั้นตอนการทำงานของคำสั่ง เช่น กระโดดข้ามไปที่ คำสั่งใดคำสั่งหนึ่ง หรือให้วนกลับมาทำคำสั่งที่เคยทำไปแล้ว ลักษณะการสั่งงานแบบนี้จะต้องใช้คำสั่งควบคุม ดังนั้นคำสั่งควบคุมจึงเป็นคำสั่งที่ใช้เปลี่ยนแปลงลำดับขั้นตอนการทำงานของโปรแกรม ซึ่งแบ่งออกเป็น 3 ชนิด คือ

1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข (Unconditional branch Statement) ได้แก่คำสั่ง goto
2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข (Condition Statement) ได้แก่คำสั่ง if, switch
3. คำสั่งให้ไปทำงานแบบเป็นวงจร (Loop Control Statement) ได้แก่คำสั่ง while, do while, for

#### 4.1 รูปแบบของคำสั่งให้ไปทำงานโดยมีเงื่อนไข

คำสั่ง if ใช้สำหรับการตัดสินใจเลือกทำงานอย่างใดอย่างหนึ่งโดยใช้เงื่อนไขเป็นส่วนช่วยในการตัดสินใจเลือก

ไวยากรณ์

```
if (expression)
    Statement;
หรือ
if (expression)
    Statement;
else
    Statement;
```

โดยที่

Expression เป็นนิพจน์ที่จะต้องให้ผลลัพธ์เป็น true หรือ false เท่านั้น

Statement อาจเป็นคำสั่งเพียงคำสั่งเดียว เช่น

```
printf("Hello world");
```

Statement อาจเป็นกลุ่มของคำสั่งหลายคำสั่ง ซึ่งต้องปิดด้วยเครื่องหมาย {...} โดยใช้เครื่องหมายเซมิโคลอน ";" คั่นระหว่างแต่ละคำสั่ง เช่น

#### ตัวอย่างที่ 4.1

```
1  #include <stdio.h>
2  int main()
3  {
4      int a,b;
5      a = 15;
6      b = ((a+5)? (a+1) : 0);
7      printf("b= %d\n",b);
8      a=0;
```

```

9      b=((a+5)?(a+1) : 0);
10     printf("b= %d\n",b);
11     a=5;
12     b=((a+5)?(a+1) : 0);
13     printf("b= %d\n",b);
14     return 0;
15 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

b= 16
b= 1
b= 6

```

## 4.2 คำสั่ง if

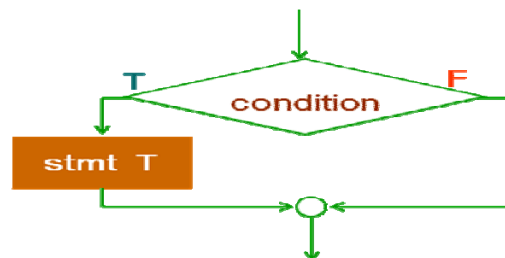
การใช้คำสั่ง if โดยไม่มี else (*If statements without ELSE*) คือ การตัดสินใจว่าจะทำคำสั่งนั้นถ้าเงื่อนไขเป็นจริง ถ้าไม่เป็นจริงก็ไม่ทำ

รูปแบบที่ง่ายที่สุดของประโยค if แสดงได้ดังนี้

```

if ( condition )
    statements;

```



โดยที่ condition คือเงื่อนไขสำหรับตัดสินใจ

statements คือชุดคำสั่งที่จะถูกทำเมื่อเงื่อนไขใน condition เป็นจริง

ตัวอย่างที่ 4.2

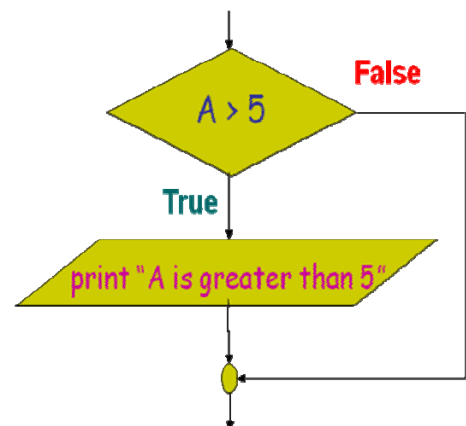
```

1  #include <stdio.h>
2  int main()
3  {
4      int A=6;
5      if (A > 5 )
6          printf("A is greater than 5");
7      return 0;
8  }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
A is greater than 5
```



ตัวอย่างที่ 4.3

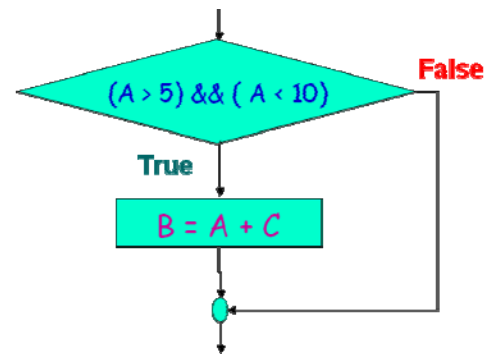
```

1  #include <stdio.h>
2  int main()
3  {
4      int A=6;
5      int B=5;
6      int C=0;
7      if (A > 5 && A < 10){
8          B = A + C;
9          printf("B=%d",B);
10     }
11     return 0;
12 }

```

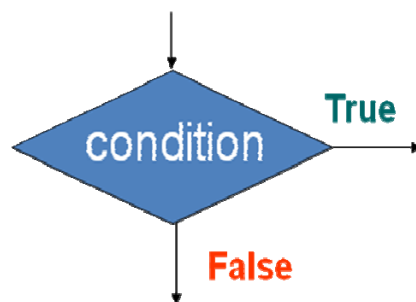
ผลลัพธ์จากการประมวลผลโปรแกรม

B=6



### 4.3 คำสั่ง if-else

เมื่อมีทางเลือก 2 ทางเลือก ใช้คำสั่ง if-elseตัดสินใจเพื่อเลือกการทำงานอย่างใดอย่างหนึ่ง



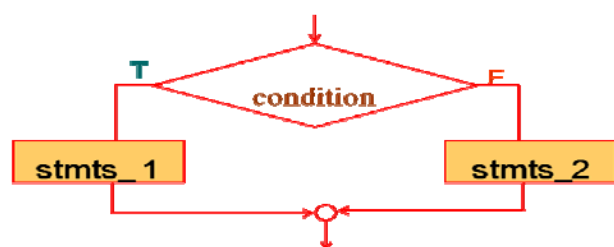
โดยคำสั่ง if-else จะทำการทดสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงให้ทำงานอย่างหนึ่ง แต่ถ้าเงื่อนไขเป็นเท็จจะให้เลือกทำงานอีกอย่างหนึ่ง

รูปแบบคำสั่ง if-else

```

if (condition )
    statements_1;
else
    statements_2;

```



โดยที่

condition คือเงื่อนไขหรือ operand ที่ให้ผลลัพธ์เป็นเลขจำนวนจริง

- ค่าผลลัพธ์ที่เป็น ศูนย์ หรือ NULL หมายถึงเท็จ
- ค่าผลลัพธ์ที่ไม่ใช่ ศูนย์ หรือไม่ใช่ NULL หมายถึงจริง

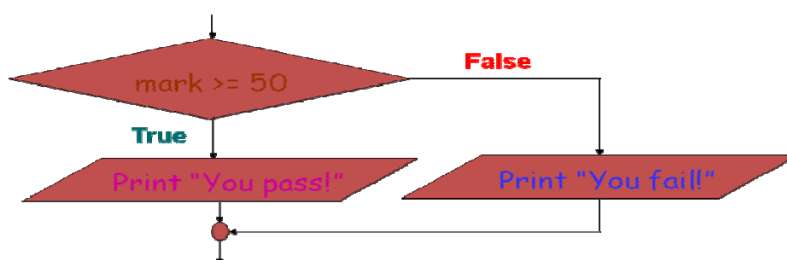
ส่วน statements\_1 คือชุดคำสั่งที่จะทำกรณี เงื่อนไขเป็นจริง ส่วน statements\_2 คือชุดคำสั่งที่จะทำกรณีเงื่อนไขเป็นเท็จ



การเปรียบเทียบทางตรรกะมักใช้ในการเขียนนิพจน์เพื่อเปรียบเทียบเงื่อนไขในการทดสอบ (if, loop)

สัญลักษณ์	ความหมาย	สัญลักษณ์	ความหมาย
= =	เท่ากับ	!=	ไม่เท่ากับ
<	น้อยกว่า	<=	น้อยกว่าหรือเท่ากับ
>	มากกว่า	>=	มากกว่าหรือเท่ากับ

ตัวอย่างที่ 4.4



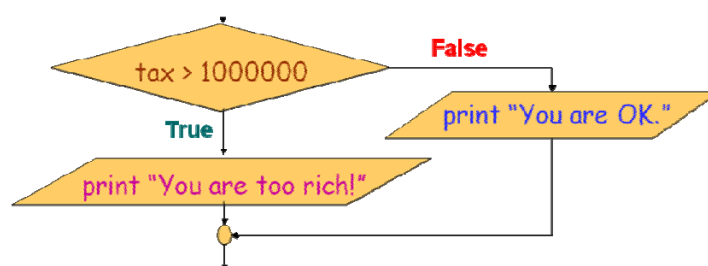
```

1  #include <stdio.h>
2  int main (void)
3  {
4      int mark=70;
5      if (mark >= 50)
6          printf("You pass!");
7      else
8          printf("You fail!");
9      return 0;
10 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

You pass!

ตัวอย่างที่ 4.5



```

1  #include <stdio.h>
2  int main (void)
3  {
4      int tax= 500;
5      if (tax > 1000000)
6          printf("You are too rich!\n");
7      else
8          printf("You are OK.\n");
9      return 0;
10 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

You are OK.

## 4.4 การทดสอบเงื่อนไขที่มี && (and), || (or) และ !(not)

การใช้ if-else นั้นเราสามารถเพิ่มเติมเงื่อนไขที่มีความซับซ้อนโดยการใช้ตัวเชื่อมทางตรรกะคือ && (and), || (or) หรือ !(not) โดยตัวเชื่อม ! มีค่าความสำคัญในการคำนวณมากกว่าตัวเชื่อม && และ || ส่วนตัวเชื่อม && มีค่าความสำคัญในการคำนวณมากกว่าตัวเชื่อม ||

ตารางค่าความจริง

A	B	A&&B	A  B	!A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

ตัวอย่างที่ 4.6

```

1  #include <stdio.h>
2  int main ()
3  {   int month =12;
4      if (month == 12||month == 1||month == 2)
5          printf("Winter");
6      else if (month == 3||month == 4||month == 5)
7          printf("Spring");
8      else if (month == 6||month == 7||month == 8)
9          printf("Summer");
10     else if (month == 9||month == 10||month == 11)
11         printf("Autumn");
12     else printf("No season");
13     return 0;
14 }
```

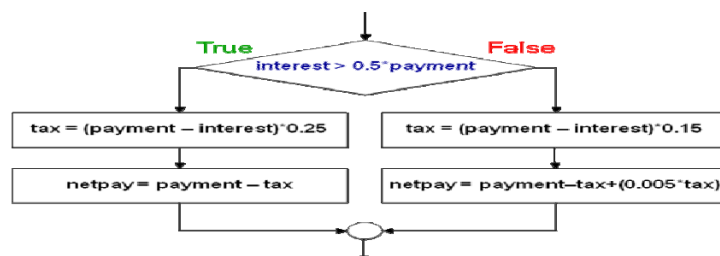
ผลลัพธ์จากการประมวลผลโปรแกรม

Winter

## 4.5 Block

ส่วน statements ที่มีคำสั่งที่ต้องการทำมากกว่าหนึ่งคำสั่ง ต้องใช้เครื่องหมาย { ... } เพื่อรวมให้เป็นชุดของคำสั่ง ซึ่งเรียกว่า block เช่น

ตัวอย่างที่ 4.7



```

1  #include <stdio.h>
2  int main ()
3  {
4      float payment =10000;
```

```

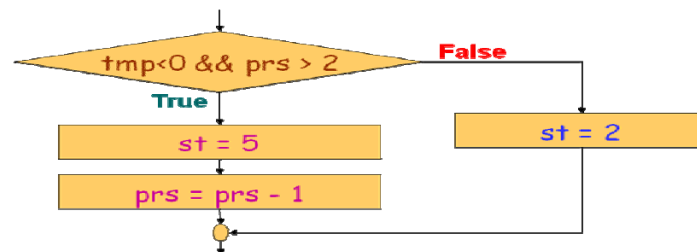
5   float interest=500;
6   float tax;
7   float netpay;
8   if(interest > 0.5*payment){
9       tax = 0.25*(payment-interest);
10      netpay = payment - tax;
11      printf("tax=%.2f netpay=%.2f",tax,netpay);
12  }
13  else{
14      tax = (payment -interest)*0.15;
15      netpay = payment-tax+(0.005*tax);
16      printf("tax=%.2f netpay=%.2f",tax,netpay);
17  }
18  return 0;
19  }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

tax=1425.00 netpay=8582.13

ตัวอย่างที่ 4.8



```

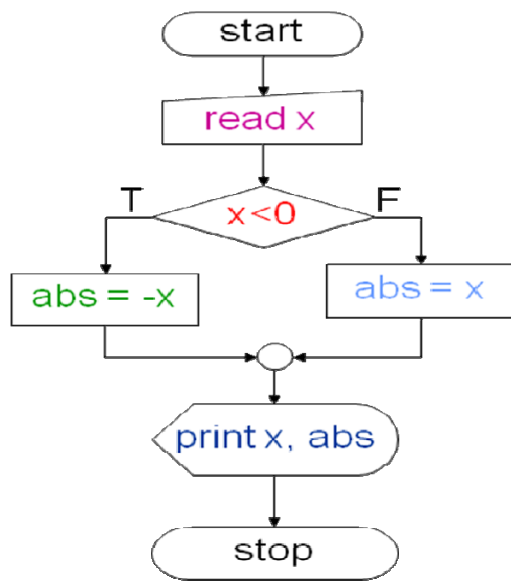
1   #include <stdio.h>
2   int main ()
3   {
4       int tmp=5;
5       int prs=5;
6       int st=0;
7       if (tmp<0 && prs>2)
8       {
9           st = 5;
10          prs = prs - 1;
11      }
12      else
13          st = 2;
14      printf("st=%d prs=%d",st,prs);
15
16      return 0;
17  }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

st=2 prs=5

ตัวอย่างที่ 4.9 โปรแกรมคำนวณค่า absolute



```

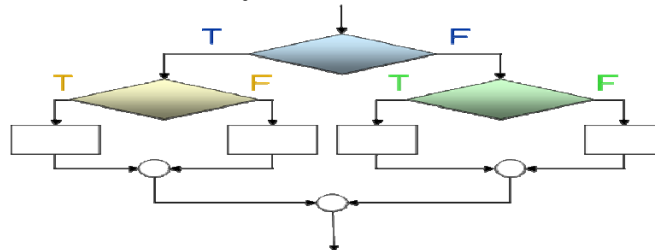
1  #include <stdio.h>
2  int main()
3  {
4      int abs, x;
5      printf ("Input an integer :");
6      scanf ("%d", &x);
7      if (x<0)
8          abs = -x;
9      else
10         abs = x;
11     printf ("abs(%d) = %d\n", x, abs);
12     return 0;
13 }
  
```

ผลลัพธ์จากการประมวลผลโปรแกรม

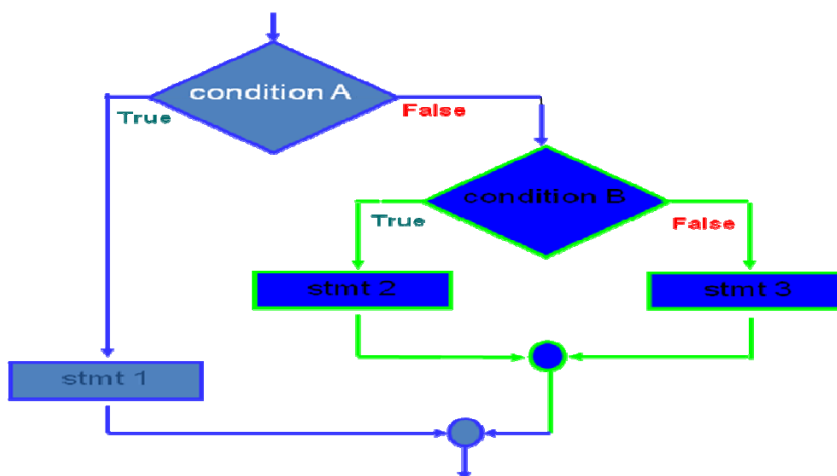
Input an integer :-5  
abs(-5) = 5

## 4.6 Nested if statements

Nested if คือการที่มีคำสั่ง if ซ้อนอยู่ใน block ของ if หรือ block ของ else เช่น



if-else-if Ladder หลังจากผ่านการเปรียบเทียบมาแล้วขั้นหนึ่ง อาจมีการเปรียบเทียบอีกก็ได้ จึงต้องมีการใช้คำสั่ง if-else หลายครั้งซ้อนกันเรียกว่า if-else- if ladder



if (condition A)

stmt1;

else if (condition B)

stmt2;

else

stmt3;

## รูปแบบ if-else-if Ladder

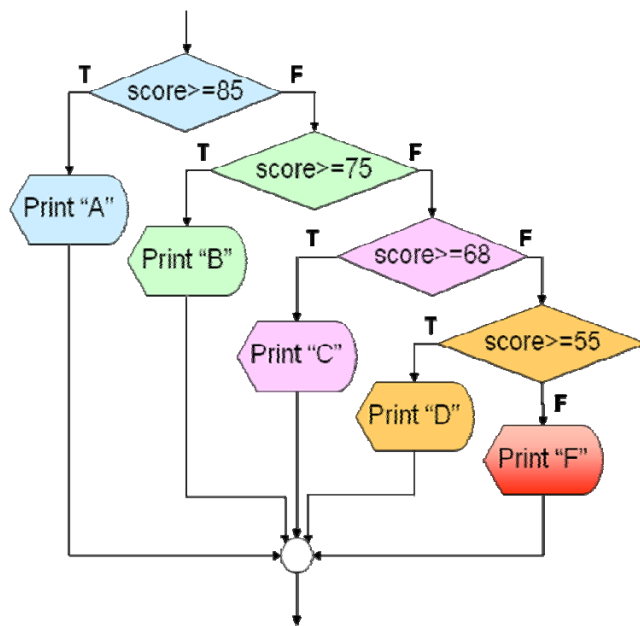
if (condition\_1) statements\_1;

else if (condition\_2) statements\_2;

...

else statements\_n;

ตัวอย่างที่ 4.10 โปรแกรมแสดงการคำนวณเกรดของนักศึกษา



```

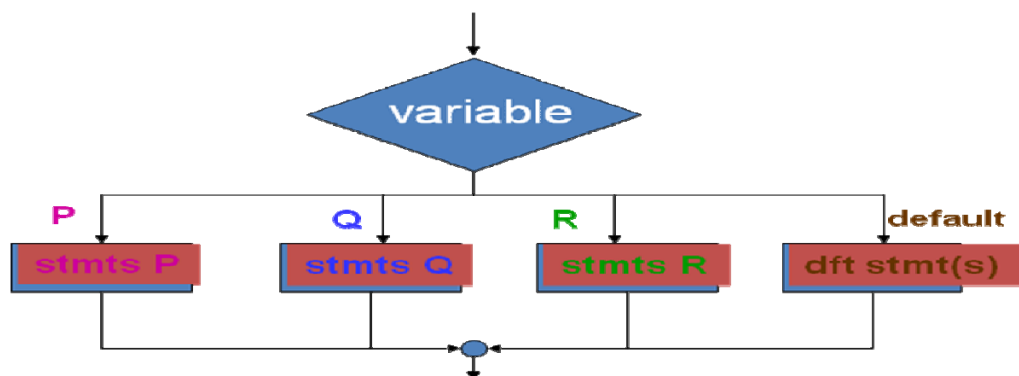
1  #include <stdio.h>
2  int main()
3  {
4      int score=50;
5      if (score >= 85)
6          printf("A");
7      else if (score >= 75)
8          printf("B");
9      else if (score >= 68)
10         printf("C");
11     else if (score >= 55)
12         printf("D");
13     else printf("F");
14     return 0;
15 }
  
```

ผลลัพธ์จากการประมวลผลโปรแกรม

A

## 4.7 การใช้คำสั่ง switch

คำสั่ง switch เป็นคำสั่งสำหรับการเลือกใช้คำสั่งหรือกลุ่มของคำสั่งจากหลาย ๆ กลุ่ม โพลีชาร์ตแสดงการทำงานของคำสั่ง switch คือ



โดยที่ variable คือ ตัวแปรที่จะใช้ในการทดสอบเงื่อนไข

P Q R คือ case หรือตัวอย่างของค่าที่จะถูกนำมาทดสอบกับค่าที่อยู่ภายในตัวแปร (variable) ซึ่งกรณี  
ที่ค่าในตัวแปรมีค่าตรงกับ case ใดก็จะทำงานตามคำสั่งที่อยู่ภายใต้ case นั้นๆ แต่หากตรวจสอบแล้วไม่ตรง  
กับ case ใด ๆ เลยก็จะเข้าสู่การทำงานในส่วนของ default ทั้งนี้คำสั่ง switch-case จะมีหรือไม่มีส่วนของ  
default ก็ได้

ถ้ามี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่ตรงกับ case ใดๆ ก็จะเข้ามาทำในส่วน  
ของ default

ถ้าไม่มี default เมื่อตรวจสอบค่าของตัวแปรหรือนิพจน์แล้วไม่ตรงกับ case ใดๆ ก็จะไม่ทำงานตาม  
คำสั่งภายใน switch-case นั้นเลย

รูปแบบ

```
switch (expression) {  
    case value_1: statements_1; break;  
    ...  
    case value_n: statements_n; break;  
    default: statements_default;  
}
```

โดยที่

expression ต้องให้ผลเป็น int หรือ char เท่านั้น

ค่าของ value\_i ต้องเป็นค่าคงที่ชนิดเดียวกับ expression

value\_i เป็นตัวแปรไม่ได้

ตัวอย่างที่ 4.11

```
1  #include <stdio.h>  
2  int main()  
3  {  
4      char grade='A';  
5      switch (grade)  
6      {  
7          case 'A': printf("Excellent"); break;  
8          case 'B': printf("Good"); break;  
9          case 'C': printf("So So"); break;  
10         case 'D': printf("Fails"); break;  
11         case 'F': printf("Get lost"); break;  
12         default : printf("Invalid");  
13     }  
14     return 0;  
15 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Excellent

## คำสั่ง break

คำสั่ง break ถูกใช้เพื่อให้โปรแกรมกระโดดข้ามการทำงานในคำสั่ง switch แล้วไปทำคำสั่งแรกที่ตามหลังคำสั่ง switch จากตัวอย่างก่อนหน้านี้ ถ้าไม่ใช่ คำสั่ง break จะเกิดอะไรขึ้น?

ตัวอย่างที่ 4.12

```
1  #include <stdio.h>
2  int main()
3  {
4      char grade='D';
5      switch (grade)
6      {
7          case 'A': printf("Excellent ");
8          case 'B': printf("Good ");
9          case 'C': printf("So So ");
10         case 'D': printf("Fails ");
11         case 'F': printf("Get lost ");
12         default : printf("Invalid ");
13     }
14     return 0;
15 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Fails Get lost Invalid

ถ้า grade มีค่าเท่ากับ 'D' ผลลัพธ์คือ Fails Get lost Invalid

ตัวอย่างที่ 4.13 กรณีที่ต้องการให้หลาย cases ทำงานอย่างเดียวกัน

```
1  #include <stdio.h>
2  int main()
3  {
4      int month=5;
5      switch(month)
6      {
7          case 12: case 1: case 2: printf("Winter"); break;
8          case 3: case 4: case 5: printf("Spring"); break;
9          case 6: case 7: case 8: printf("Summer"); break;
10         case 9: case 10: case 11: printf("Autumn"); break;
11         default : printf("No season");
12     }
13     return 0;
14 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Spring

## แบบฝึกหัดปฏิบัติการคาบที่ 4: Control Statement

1. จงเขียนผังงานและโปรแกรมเครื่องคิดเลข โดยโปรแกรมรับข้อมูลนำเข้า 3 ตัว ได้แก่ ตัวเลขตัวที่ 1 ตัวเลขตัวที่ 2 เครื่องหมาย (+,-,\*,/) และ จากนั้นแสดงค่าผลลัพธ์

Please enter number1: **1** (กดแป้น Enter)

Please enter number2: **2** (กดแป้น Enter)

Please enter operator: **+**

Result is = 3

### วิเคราะห์ปัญหา

### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร    ชนิดตัวแปร    ความหมาย

### เขียนโปรแกรม

|



2. จงเขียนผังงานและโปรแกรมเพื่อรับค่า A B C และ m จากแป้นพิมพ์ เพื่อนำมาคำนวณหาค่า Y โดยมีเงื่อนไขต่อไปนี้

กำหนดให้ A B C m เป็นเลขจำนวนเต็ม

$$Y = Am^2 + Bm + C \quad \text{เมื่อ } m > 7$$

$$Y = Am^2 - Bm - C \quad \text{เมื่อ } m = 7$$

$$Y = Am^2 + Bm \quad \text{เมื่อ } m < 7$$

โดยแสดงผลในรูปแบบต่อไปนี้

Please enter A : (กดแป้น Enter)

Please enter B : (กดแป้น Enter)

Please enter C : (กดแป้น Enter)

Please enter m : (กดแป้น Enter)

The result of Y =

วิเคราะห์ปัญหา

เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร      ชนิดตัวแปร      ความหมาย

เขียนโปรแกรม

3. จงเขียนผังงานและโปรแกรมเพื่อให้โปรแกรมทำงานโดยการถามอายุ เพศ น้ำหนัก ส่วนสูง จากนั้นโปรแกรมจะพิมพ์ผลลัพธ์ว่าน้ำหนักนั้น Overweight หรือ Underweight ไปกี่กิโลกรัม น้ำหนักมาตรฐานของผู้ชายคือส่วนสูง-105 และน้ำหนักมาตรฐานของผู้หญิงคือ ส่วนสูง-110

#### ตัวอย่าง

Please enter your name: **Yaya**

Are you male or female, **Yaya** (M, F): **F**

**Yaya**, what is your height in c.m. and weight in k.g: **170 55**

**Yaya**, your ideal weight is **60** kg, you are **5** kg **underweight**.

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร

ชนิดตัวแปร

ความหมาย

#### เขียนโปรแกรม

4. จงเขียนโปรแกรม เพื่อคำนวณคะแนนรวมของผลสอบวิชา C Programming จากคะแนน Mid-term (100 คะแนน), คะแนน Final (100 คะแนน), และคะแนน Homework (10 คะแนน) เป็นข้อมูลเข้า (Input) จาก คีย์บอร์ด และแสดงผลลัพธ์จากการคำนวณ เมื่อคะแนนรวม (x) = Mid-term (40%) + Final (50%) + HW (10%) และตัดเกรดด้วยคำสั่ง nested-if ตามเงื่อนไขต่อไปนี้

คะแนน  $90 \leq x \leq 100$  จะได้ grade = 'A'

คะแนน  $85 \leq x < 90$  จะได้ grade = 'B+'

คะแนน  $80 \leq x < 85$  จะได้ grade = 'B'

คะแนน  $70 \leq x < 80$  จะได้ grade = 'C+'

คะแนน  $60 \leq x < 70$  จะได้ grade = 'C'

คะแนน  $55 \leq x < 60$  จะได้ grade = 'D+'

คะแนน  $50 \leq x < 55$  จะได้ grade = 'D'

คะแนน  $x < 50$  จะได้ grade = 'F'

#### วิเคราะห์ปัญหา

#### เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร

ชนิดตัวแปร

ความหมาย

#### เขียนโปรแกรม

## บทที่ 5

# คำสั่งควบคุมการวนซ้ำ (Loop Statement)

### จุดประสงค์

เพื่อให้ทราบความหมายของคำสั่งควบคุมการทำงานของโปรแกรมแบบวนรอบ เช่น for, while, do-while

การวนซ้ำ หมายถึง การควบคุมให้การกระทำบางคำสั่งซ้ำหลายรอบ ซึ่งจะช่วยให้การเขียนโปรแกรมทำได้ง่ายสะดวก ไม่ต้องเขียนคำสั่งเดิมหลายครั้ง ทำให้โปรแกรมมีความกระชับ สามารถตรวจสอบความผิดพลาดได้ง่าย คำสั่งการวนรอบมี 3 ชนิดหลัก ๆ คือ

คำสั่ง for

คำสั่ง do-while

คำสั่ง while

โดยที่แต่ละคำสั่งมีรูปแบบและวิธีการใช้งานที่แตกต่างกัน สามารถเลือกใช้ตามความเหมาะสมของการใช้งานในโปรแกรม

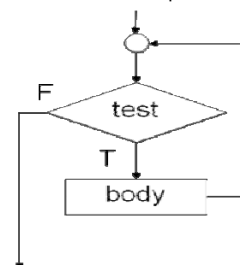
**ส่วนประกอบของคำสั่งแบบวนซ้ำ** ประกอบด้วย

- ส่วนของการตรวจสอบ (loop test)
  - เป็นเงื่อนไขเพื่อทดสอบว่าจะทำวนซ้ำอีกหรือไม่
- ส่วนของการทำวนซ้ำ (loop body)
  - เป็นชุดคำสั่งที่จะถูกดำเนินการ

**การสร้าง loop**

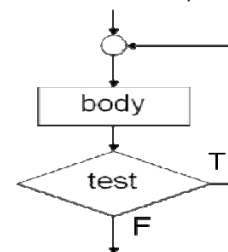
- ระบุส่วนของการทำงานที่ต้องทำซ้ำ (loop body)
  - ระบุเงื่อนไข (loop test) ที่จะทำซ้ำ หรือเลิกทำซ้ำ
- ระบุชนิดของ loop ที่จะใช้

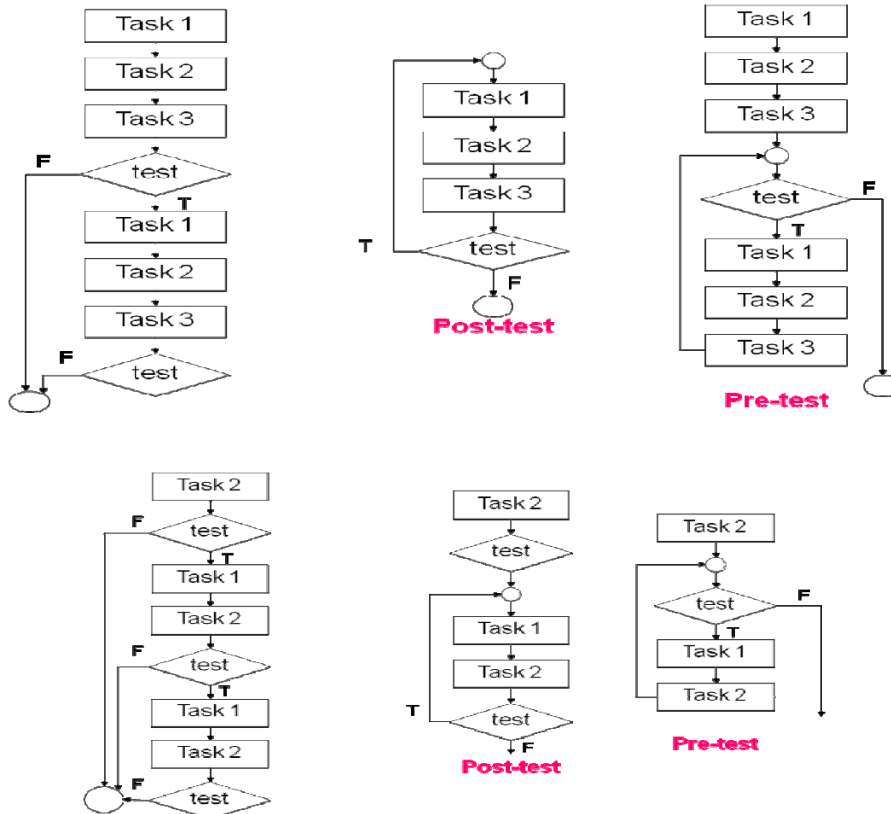
Pre-test Loop



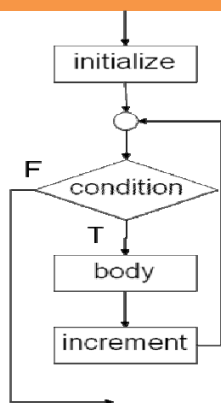
ตัวอย่างการสร้าง loop

Post-test Loop





## 5.1 คำสั่ง for



คำสั่ง for เป็นคำสั่งที่สั่งให้ทำคำสั่ง หรือกลุ่มของคำสั่งวนซ้ำหลายรอบ โดยมีจำนวนรอบในการวนซ้ำที่แน่นอน

รูปแบบ

for (ค่าเริ่มต้น; เงื่อนไข; ค่าเพิ่มหรือค่าลด)

คำสั่ง

โดยที่ ค่าเริ่มต้น, เงื่อนไข, ค่าเพิ่มหรือค่าลด เป็นนิพจน์

คำสั่ง หมายถึง คำสั่งที่จะถูกกระทำซ้ำ ซึ่งอาจจะมีเพียงคำสั่งเดียว หรือหลายคำสั่งก็ได้

คำสั่ง for มีขั้นตอนการทำงานดังนี้

1. คำนวณค่าเริ่มต้นของตัวแปรที่ใช้ควบคุมการวนซ้ำซึ่งอยู่ในรูปของคำสั่งกำหนดค่า
2. คำนวณผลลัพธ์จากเงื่อนไข ที่อยู่ในรูปของนิพจน์ความสัมพันธ์ ซึ่งจะให้ผลเป็นจริง(0) หรือเท็จ (ค่าที่ไม่เป็น 0)
3. ถ้าผลลัพธ์จากข้อ 2 มีค่าเป็นเท็จหรือ 0 ไปที่ 7
4. ถ้าผลลัพธ์จากข้อ 2 มีค่าเป็นจริง หรือค่าที่ไม่ใช่ 0 ข้อความสั่งที่อยู่ภายในคำสั่ง for จะถูกกระทำ
5. คำนวณค่าใหม่ของตัวแปรที่ใช้ควบคุมการวนซ้ำ
6. กลับไปที่ข้อ 2
7. จบการกระทำคำสั่ง for และข้อความแรกที่อยู่ถัดจากคำสั่ง for จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 5.1 ตัวอย่างการใช้คำสั่ง for รูปแบบต่าง ๆ

```
for (i=1, sum=0; i<=10; i++)
    sum += i;
for (i=1; i<=25; i++)
{
    scanf("%f", &score);
    printf("%f\n", score);
}
for (scanf("%f", &sc), sum=0;
     sc>=0; scanf("%f", &sc))
    sum += sc;
```

ตัวอย่างที่ 5.2

```
1  #include <stdio.h>
2  int main ()
3  {
4      // Local Declarations
5      int limit;
6      int i;
7      // Statements
8      printf ("\nPlease enter the limit: ");
9      scanf ("%d", &limit);
10     for (i = 1; i <= limit; i++)
11     {
12         printf("\t%d\n", i);
13     }
14     return 0;
15 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter the limit: 5
    1
    2
    3
    4
    5
```

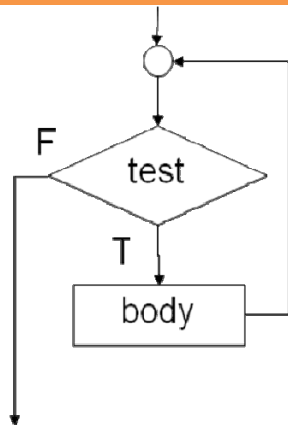
ตัวอย่างที่ 5.3

```
1  #include <stdio.h>
2  int main ()
3  {
4      // Statements
5      int i = 1, j=1;
6      for ( i = 1; i <= 3; i++)
7      {
8          printf("Row %d: ", i);
9          for ( j = 1; j<= 5; j++)
10             printf("%3d", j);
11             printf("\n");
12         } // for i
13     return 0;
14 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Row 1:   1  2  3  4  5
Row 2:   1  2  3  4  5
Row 3:   1  2  3  4  5
```

## 5.2 คำสั่ง while



คำสั่ง while เป็นคำสั่งวนซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายในคำสั่ง while หลายรอบจนกระทั่งเงื่อนไขเป็นเท็จ หรือ 0 จึงจะจบการวนซ้ำ

รูปแบบ

while (เงื่อนไข)

คำสั่ง

โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และคำสั่งอยู่ภายในคำสั่ง while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง

คำสั่ง while มีขั้นตอนการทำงานดังนี้

1. คำนวณหาค่าของเงื่อนไข
2. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จหรือศูนย์ ไปที่ข้อ 5
3. ถ้าค่าของเงื่อนไข มีค่าเป็นจริงหรือค่าที่ไม่ใช่ศูนย์ คำสั่งที่อยู่ภายในคำสั่ง while จะถูกกระทำ
4. กลับไปที่ข้อ 1
5. จบการกระทำการคำสั่ง while และข้อความแรกที่อยู่ถัดจากคำสั่ง while จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 5.4 ตัวอย่างการใช้คำสั่ง while รูปแบบต่าง ๆ

```
while (x<0)
    x++;
while (sum<20)
{
    scanf("%d", &prc);
    if (prc<20)
    {
        vat = .07*prc;
        sum += prc + vat;
    }
}
```

ตัวอย่างที่ 5.5

```
1  #include <stdio.h>
2  int main (void)
3  {
4      // Local Declarations
5      int num;
6      int lineCount;
7      // Statements
8      printf ("Enter an integer between 1 and 100: ");
9      scanf ("%d", &num);           // Initialization
10     // Test number
11     if (num > 100)
12         num = 100;
13     lineCount = 0;
14     while (num > 0)
15     {
16         if (lineCount < 10)
17             lineCount++;
18         else
19             {
```

20	printf("\n");
21	lineCount = 1;
22	} // else
23	printf("%4d", num--); // num-- updates loop
24	} // while
25	return 0;
26	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
Enter an integer between 1 and 100: 20	
20 19 18 17 16 15 14 13 12 11	
10 9 8 7 6 5 4 3 2 1	

ตัวอย่างที่ 5.6

1	#include <stdio.h>
2	int main ()
3	{
4	// Local Declarations
5	int number;
6	int count = 0;
7	int sum = 0;
8	// Statements
9	printf("Enter an integer: ");
10	scanf ("%d", &number);
11	printf("Your number is: %d\n\n", number);
12	while (number != 0)
13	{
14	count++;
15	sum += number % 10;
16	number /= 10;
17	} // while
18	printf("The number of digits is: %3d\n", count);
19	printf("The sum of the digits is: %3d\n", sum);
20	return 0;
21	} // main
ผลลัพธ์จากการประมวลผลโปรแกรม	
Enter an integer: 5894	
Your number is: 5894	
The number of digits is: 4	
The sum of the digits is: 26	

ตัวอย่างที่ 5.7

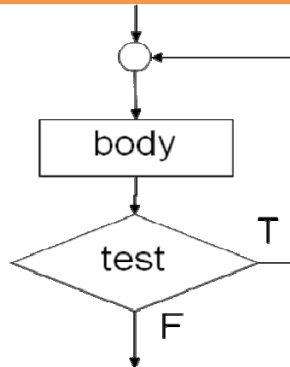
1	#include <stdio.h>
2	int main (void)
3	{
4	// Local Declarations
5	long num;
6	int digit;
7	// Statements
8	printf("Enter a number and I'll print it backward: ");
9	scanf ("%d", &num);
10	while (num > 0)
11	{
12	digit = num % 10;
13	printf("%d", digit);
14	num = num / 10;
15	} // while
16	printf("\nHave a good day.\n");
17	return 0;
18	} // main



### ผลลัพธ์จากการประมวลผลโปรแกรม

```
Enter a number and I'll print it backward: 12345678
87654321
Have a good day.
```

## 5.3 คำสั่ง do-while



คำสั่ง do-while เป็นคำสั่งวนซ้ำ ที่สั่งให้ทำคำสั่งที่อยู่ภายในคำสั่ง do-while หนึ่งรอบ แล้วจึงจะตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นเท็จ จะจบการทำงานทันที

รูปแบบ

```
do
    คำสั่ง;
while (เงื่อนไข);
```

โดยที่ เงื่อนไข เป็นนิพจน์ที่ให้ผลลัพธ์เป็นจริงหรือเท็จ และคำสั่งอยู่ภายในคำสั่ง do-while อาจมีเพียงคำสั่งเดียว หรือหลายคำสั่ง

คำสั่ง do-while มีขั้นตอนการทำงานดังนี้

1. กระทำข้อความสั่งที่อยู่ภายในข้อความสั่ง do-while
2. คำนวณหาค่าของเงื่อนไข
3. ถ้าค่าของเงื่อนไข มีค่าเป็นเท็จหรือศูนย์ ไปที่ข้อ 5
4. ถ้าค่าของเงื่อนไข มีค่าเป็นจริงหรือค่าที่ไม่ใช่ศูนย์ กลับไปที่ข้อ 1
5. จบการกระทำคำสั่ง do-while และข้อความแรกที่อยู่ถัดจากคำสั่ง do-while จะถูกทำในลำดับต่อไป

ตัวอย่างที่ 5.8

```
do x++;
while (x<0);
do
{   scanf("%d", &prc);
    if (prc<20)
    {   vat = .07*prc;
        sum += prc + vat;
    }
} while (sum<20);
```

### ตัวอย่างที่ 5.9

```
1  #include <stdio.h>
2  int main (void)
3  {
4  // Local Declarations
5  int loopCount;
6  // Statements
7  loopCount = 5;
8  printf("while loop      : ");
9  while (loopCount > 0)
10     printf ("%3d", loopCount--);
11     printf("\n\n");
12
13     loopCount = 5;
14     printf("do...while loop: ");
15     do
16         printf ("%3d", loopCount--);
17     while (loopCount > 0);
18     printf("\n");
19     return 0;
20 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

while loop : 5 4 3 2 1

do...while loop: 5 4 3 2 1

## 5.4 การวนซ้ำซ้อน

การวนซ้ำซ้อน หมายถึง การควบคุมให้กระทำบางคำสั่งหลายรอบ และในการทำงานแต่ละรอบก็จะควบคุมให้ทำคำสั่งที่อยู่ภายในนั้นอีกหลายรอบ

การวนซ้ำซ้อน อาจจะเขียนได้โดยการใช้คำสั่งวนซ้ำใด ๆ ซ้อนกัน 2 คำสั่ง เช่น

คำสั่ง for ซ้อนกับคำสั่ง for

คำสั่ง while ซ้อนกับคำสั่ง while

คำสั่ง while-do ซ้อนกับคำสั่ง while-do

คำสั่ง while ซ้อนกับคำสั่ง for

หลักการทำงานของ การวนซ้ำซ้อน จะเริ่มที่การวนซ้ำรอบนอกก่อน 1 รอบ ในระหว่างนั้นจะไปทำซ้ำการวนรอบในอีกหลาย ๆ รอบ จากนั้นจึงจะไปกระทำการวนซ้ำรอบนอก และวนซ้ำรอบในอีกหลาย ๆ รอบ เช่นนี้เรื่อยไปจนกระทั่งจบการวนซ้ำรอบนอก

### ตัวอย่างที่ 5.10

```

1  #include <stdio.h>
2  int main (void)
3  {
4      // Local Declarations
5      double presVal;
6      double futureVal;
7      double rate;
8      int    years;
9      int    loopier;
10     // Statements
11     printf("Enter value of investment:  ");
12     scanf ("%lf", &presVal);
13     printf("Enter rate of return (nn.n): ");
14     scanf ("%lf", &rate);
15     printf("Enter number of years:      ");
16     scanf ("%d", &years);
17     printf("\nYear      Value\n");
18     printf("====  =====\n");
19     for (futureVal = presVal, loopier = 1;
20         loopier <= years;
21         loopier++)
22     {
23         futureVal = futureVal * (1 + rate/100.0);
24         printf("%3d%11.2lf\n", loopier, futureVal);
25     } // for
26     return 0;
27 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

Enter value of investment:  10000
Enter rate of return (nn.n): 7.2
Enter number of years:      5

```

```

Year      Value
====  =====
  1    10720.00
  2    11491.84
  3    12319.25
  4    13206.24
  5    14157.09

```

### ตัวอย่างที่ 5.11

```

1  #include <stdio.h>
2  int main ()
3  {
4      // Local Declarations
5      int limit;
6      int lineCtrl = 0;
7      int numCtrl=0;
8      // Statements
9      printf("\nPlease enter a number between 1 and 9: ");
10     scanf ("%d", &limit);
11     for ( lineCtrl = 1; lineCtrl <= limit; lineCtrl++)
12     {
13         for ( numCtrl = 1; numCtrl <= lineCtrl; numCtrl++)
14             printf("%1d", numCtrl);
15         printf("\n");
16     } // for lineCtrl
17     return 0;
18 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter a number between 1 and 9: 9
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

ตัวอย่างที่ 5.12

```
1  #include <stdio.h>
2  int main ()
3  {
4      int limit;
5      int row=0;
6      int col=0;
7      printf("Please enter a number between 1 and 9: ");
8      scanf("%d", &limit);
9      for ( row = 1; row <= limit; row++)
10     {
11         for ( col = 1; col <= limit; col++)
12             if (row >= col)
13                 printf("%d", col);
14             else
15                 printf("*");
16         printf("\n");
17     }
18     return 0;
19 } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter a number between 1 and 9: 9
1*****
12*****
123*****
1234*****
12345****
123456***
1234567**
12345678*
123456789
```

## แบบฝึกหัดปฏิบัติการคาบที่ 5: Loop Statement

1. จงเขียนผังงานและโปรแกรมสำหรับหาค่าแฟคทอเรียลดังตัวอย่างต่อไปนี้

Please enter number of factorial: 3

Result is: 6

**หมายเหตุ** การหาค่า Factorial มีดังนี้

$$1! = 1$$

$$2! = 2 * 1 = 2$$

$$3! = 3 * 2 * 1 = 6$$

$$4! = 4 * 3 * 2 * 1 = 24$$

$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

$$6! = 6 * 5 * 4 * 3 * 2 * 1 = 720$$

วิเคราะห์ปัญหา

เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร

ชนิดตัวแปร

ความหมาย

เขียนโปรแกรม

2. จงเขียนผังงานและโปรแกรมเครื่องขายอาหารอัตโนมัติซึ่งขายอาหาร 3 ประเภท โดยเครื่องดังกล่าวจะถามว่าเราต้องการอาหารประเภทใดระหว่าง sandwich หรือ cake หรือ beverage

- หากเลือก sandwich เครื่องจะให้เลือกชนิดเป็น Tuna ราคา 30 บาท Hamburger ราคา 40 บาท หรือ Ham ราคา 35 บาท
- หากเลือก cake เครื่องจะให้เลือกชนิดเป็น Donut ราคา 17 บาท JamRoll ราคา 15 บาท หรือ Pastry ราคา 25 บาท หรือ Cookie ราคา 10 บาท
- หากเลือก beverage เครื่องจะให้เลือกชนิดเป็น Coke ราคา 15 บาท Est ราคา 15 บาท หรือ GreenTea ราคา 60 บาท

ให้ผู้ใช้สามารถเลือกซื้ออาหารได้จนกว่าจะพิมพ์ n หลังจากนั้นให้รวมราคาที่ต้องจ่าย ดังตัวอย่างต่อไปนี้

```
+++++
                                VENDING MACHINE
+++++
Welcome to vending machine. Enter 1-Sandwich, 2-cake, 3-Beverage: 1
Enter 1-Tuna (30), 2- Hamburger (40) , 3- Ham (35): 2
Do you want to continue: Y
Welcome to vending machine. Enter 1-Sandwich, 2-cake, 3-Beverage: 2
Enter 1- Donut (17), 2- JamRoll (15), 3- Pastry (25), 4-Cookie-(10): 2
Do you want to continue: N
+++++
THANK YOU VERY MUCH. THE PRICE IS: 55 BAHT
+++++
```

วิเคราะห์ปัญหา

เขียนผังงาน

ข้อมูลนำเข้า

ข้อมูลส่งออก

กำหนดตัวแปร

ชื่อตัวแปร      ชนิดตัวแปร      ความหมาย

เขียนโปรแกรม

## การเขียนโปรแกรมส่งผ่าน Grader

1. จงเขียนโปรแกรมเพื่อพิมพ์ \* ออกทางจอภาพ

**ข้อมูลอินพุต** มี 1 บรรทัด ประกอบด้วยจำนวนตัวเลข 1 เป็นจำนวน \* ที่มากที่สุด

**ข้อมูลเอาต์พุต** แสดงรูป \* ตามจำนวนอินพุต

**ตัวอย่าง**

อินพุต	เอาต์พุต
9	<pre> * *** ***** ***** ***** ***** ***** ***** *** *</pre>
6	<pre> ** **** ***** **** **</pre>

2. จงเขียนโปรแกรมสำหรับการแสดงผลเป็นข้อมูล 2 มิติ แบบ NxN ซึ่งแต่ละแถวและแต่ละคอลัมน์จะปรากฏเลข 1 - N ที่ไม่ซ้ำกันเมื่อ N เป็นค่าที่รับจากคีย์บอร์ด เช่น N= 5

**ข้อมูลอินพุต** มี 1 บรรทัด ประกอบด้วยจำนวนตัวเลข 1 จำนวน แสดงจำนวนแถวและคอลัมน์

**ข้อมูลเอาต์พุต** แสดงผลลัพธ์เป็นข้อมูล 2 มิติ แบบ NxN ซึ่งแต่ละแถวและแต่ละคอลัมน์จะปรากฏเลข 1 - N ที่ไม่ซ้ำกัน

**ตัวอย่าง**

อินพุต	เอาต์พุต
5	<pre> 1 2 3 4 5 2 3 4 5 1 3 4 5 1 2 4 5 1 2 3 5 1 2 3 4</pre>

3. N จะเป็นเลขเฉพาะถ้า N เป็นเลขที่หารด้วยเลขใด ๆ ไม่ลงตัวยกเว้น N และ 1 ดังนั้น N จะมีค่าเป็น Prime Number ถ้าหารด้วยค่าต่างๆ ตั้งแต่ค่า 2 ถึงค่า N-1 ไม่ลงตัว แต่ถ้า N หารด้วยค่าใดค่าหนึ่งลงตัว จะแสดงว่า N ไม่เป็น Prime Number จงเขียนโปรแกรมสำหรับตรวจสอบว่าค่าเลขจำนวนเต็ม N เป็นค่าเลขเฉพาะหรือไม่

**ข้อมูลอินพุต** มี n+1 บรรทัด ประกอบด้วย

บรรทัดแรกแสดงจำนวนตัวเลขทั้งหมดที่จะทดสอบ

บรรทัดที่ 2 ถึง n+1 แสดงตัวเลข n จำนวน

**ข้อมูลเอาต์พุต** มี n บรรทัด แต่ละบรรทัดแสดงผลลัพธ์

y ถ้า N มีค่าเป็น Prime Number

n ถ้า N มีค่าไม่เป็น Prime Number

**ตัวอย่าง**

อินพุต	เอาต์พุต
3	y
7	n
8	y
13	
2	n
1	y
2	

4. จงเขียนโปรแกรมทดสอบว่า International Standard Book Number (ISBN) ถูกต้อง (Valid) หรือไม่ โดยเลข ISBN เป็นเลขที่เอาไว้ทดสอบความเป็นเอกลักษณ์ของหนังสือแต่ละเล่ม ประกอบด้วย 10 digits เลข ISBN จะ Valid ก็ต่อเมื่อหากผลรวม digit ทั้ง 10 digits คูณกับค่า weight ของแต่ละตำแหน่ง หารด้วย 11 ลงตัว เช่น จะตรวจสอบว่า ISBN = 0078818095 valid หรือไม่ทำได้โดย

Code	Weight	Weight value (Weight*code)
0	10	0
0	9	0
7	8	56
8	7	56
8	6	48
1	5	5
8	4	32
0	3	0
9	2	18
5	1	5



ผลรวม Weight = 220

ดังนั้นเมื่อนำ 220 ไปหารด้วย 11 จะได้ 10 เพราะฉะนั้นเลข ISBN ชุดนี้ valid

ข้อมูลอินพุต มี 1 บรรทัด ประกอบด้วยเลข ISBN ที่จะทดสอบ

ข้อมูลเอาต์พุต มี 1 บรรทัด แสดงผลลัพธ์การทดสอบ

ถ้าเลข ISBN ที่จะทดสอบ ตรงตามเงื่อนไข จะแสดง valid

ถ้าเลข ISBN ที่จะทดสอบ ไม่ตรงตามเงื่อนไขจะแสดง invalid

ตัวอย่าง

อินพุต	เอาต์พุต
0078818095	valid

## บทที่ 6

# โครงสร้างข้อมูลแบบอาร์เรย์ (Arrays)

### จุดประสงค์

1. เพื่อให้ทราบความหมายของตัวแปรอาร์เรย์
2. เพื่อให้ทราบรูปแบบการใช้งานตัวแปรอาร์เรย์
3. เพื่อให้ทราบการอ้างอิงข้อมูลในตัวแปรอาร์เรย์

## 6.1 อาร์เรย์

**ตัวแปรชุด (array) array** เป็นชนิดข้อมูลประเภทหนึ่งที่น่าเอาชนิดข้อมูลข้อมูลพื้นฐาน เช่น ตัวอักษร(char) ชนิดข้อมูลแบบเลขจำนวนเต็ม(int) และชนิดข้อมูลแบบเลขจำนวนจริง(float) มาประยุกต์เป็นชนิดข้อมูลประเภทนี้ เมื่อประกาศโครงสร้างข้อมูลแบบอาร์เรย์(Array) จะเก็บข้อมูลต่างจากชนิดข้อมูลพื้นฐานทั่วไป คือ สามารถเก็บค่าภายในตัวแปรชนิดนี้ได้มากกว่า 1 ค่าซึ่งจำนวนค่าที่เก็บนั้นขึ้นอยู่กับขนาดของอาร์เรย์ที่ได้กำหนดไว้ ประเภทของตัวแปรชุด อาจแบ่งตามลักษณะของจำนวนตัวเลขของดัชนี คือ

1. ตัวแปรชุด 1 มิติ (one dimension arrays หรือ single dimension arrays) เป็นตัวแปรชุดที่มีตัวเลขแสดงขนาดเป็นเลขตัวเดียว เช่น word[20] , num[25] , x[15]
2. ตัวแปรชุดหลายมิติ (multi-dimension arrays) เป็นตัวแปรชุดที่มีชื่อมีตัวเลขแสดงขนาดเป็นตัวเลขหลายตัว ที่นิยมใช้กันมี 2 มิติ กับ 3 มิติ
  - 2.1 ตัวแปรชุด 2 มิติ มีเลขแสดงขนาด 2 ตัว เช่น a[3][5] , name[5][6]
  - 2.2 ตัวแปรชุด 3 มิติ มีเลขแสดงขนาด 3 ตัว เช่น a[3][5][6] , name[5][6][8]

## 6.2 การประกาศและกำหนดค่าตัวแปรชุด 1 มิติ

การประกาศตัวแปรชุด 1 มิติ เพื่อใช้งาน ใช้คำสั่ง ดังนี้

```
type arrayname[size];
```

โดย type คือ ชนิดของตัวแปร เช่น int char float

arrayname คือชื่อของตัวแปรarray

size คือ ขนาดของตัวแปร

### 6.2.1 การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท integer

int a[10]; เป็นการประกาศตัวแปร array ชื่อ a เป็น array ของข้อมูลประเภท integer มีสมาชิกได้จำนวน 10 ตัว คือ a[0] a[1] a[2] a[3] ... a[9] โดยมีการจองเนื้อที่ในหน่วยความจำเปรียบเทียบได้ดังรูป

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

2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์	2 ไบต์

โดยสมาชิกแต่ละตัวจะใช้เนื้อที่เท่ากับตัวแปรประเภท integer ที่ไม่ได้อยู่ใน array คือ 2 ไบต์ ต่อ ตัวแปร 1 ตัว ดังนั้นเนื้อที่หน่วยความจำที่ใช้ทั้งหมดจึงเท่ากับจำนวนสมาชิก คูณ ด้วย 2 ไบต์

การกำหนดค่าให้แก่ตัวแปร array อาจกำหนดพร้อมกับการประกาศ เช่น

`int num1[3] = {56,25,89};` เป็นการประกาศว่าตัวแปร `num1` เป็น array ประเภท integer มีสมาชิก 3 ตัว โดย `num1[0] = 56`; ส่วน `num1[1]=25`; และ `num1[2]=89`;

`int a[ ]={200,230};` ประกาศว่า `a` เป็นตัวแปร array ประเภท integer ที่มีสมาชิก 2 ตัว โดย `a[0]` มีค่า เป็น 200 และ `a[1]` มีค่าเป็น 230

แต่ไม่สามารถประกาศว่า `int value[ ]` ; โดยถ้าจะไม่ระบุจำนวนสมาชิก ต้องระบุค่าของแต่ละสมาชิกที่ถูกล้อมรอบด้วย { } โดยระหว่างสมาชิกคั่นด้วยเครื่องหมาย , (คอมม่า) ดังตัวอย่าง `int a[ ]={200,230};`

หรือ ประกาศตัวแปร โดยยังไม่กำหนดค่า เช่น `int money[5];` แล้วไปกำหนดค่าให้สมาชิกแต่ละตัวในภายหลัง เช่น `money[0] = 250`; `money[4] = 500`;

#### 6.2.2 การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท float

`float b[10];` เป็นการประกาศตัวแปร array ของ ตัวแปรจำนวนที่มีทศนิยมได้ คือ float ในชื่อ `b` ซึ่งมีสมาชิกได้ 10 ตัว คือ `b[0]` `b[1]`... `b[9]` มีการจองเนื้อที่ในหน่วยความจำเปรียบเทียบได้ ดังรูป

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



โดยสมาชิกแต่ละตัวใช้หน่วยความจำ 4 ไบต์ ดังนั้นทั้งหมดจะใช้หน่วยความจำ 4 คูณ 10 คือ 40 ไบต์ การกำหนดค่าของตัวแปร array ประเภท float เป็นไปในลักษณะเดียวกับ array ประเภท integer ประกาศพร้อมกับกำหนดค่าให้เลยโดยล้อมรอบด้วย { } และค่าของสมาชิกแต่ละตัวคั่นด้วย , (คอมม่า) เช่น

`float num[5] = {2.00,1.25,5.36,6.32,246.10};` โดยค่าของ `num[0] = 2.00` ส่วนของ `num[1] = 1.25` ค่าของ `num[2] = 5.36` ค่าของ `num[3] = 6.32` และ `num[4] = 246.10` ตามลำดับ หรือประกาศตัวแปรก่อนแล้วไปกำหนด ค่าภายหลัง เช่น

`float salary[10];`

`salary[0] = 25000.00;`

#### 6.2.3 การประกาศตัวแปรชุด 1 มิติของข้อมูลประเภท char

`char c[10];` เป็นการประกาศตัวแปร array ชื่อ `c` เป็น array ของ ตัวแปรอักขระ char มีสมาชิกได้ 10 ตัว คือ `a[0]` `a[1]` ... `a[9]` โดยการใช้เนื้อที่ในหน่วยความจำเปรียบเทียบได้ ดังรูป

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



โดยที่สมาชิกแต่ละตัวใช้เนื้อที่ 1 ไบต์ ดังนั้น array ประเภท char ชุดนี้ ใช้เนื้อที่เป็น 10 bytes

## 6.3 ตัวอย่างการใช้งานอาร์เรย์ 1 มิติ

ตัวอย่างที่ 6.1 ตัวอย่างการเก็บตัวเลขจำนวนเต็มในอาร์เรย์และแสดงผลข้อมูลในอาร์เรย์ตัวแรก

1	#include <stdio.h>
2	int main()
3	{
4	int a[5]={11,12,13,14,15};
5	printf("%d\n",a[0]);
6	return 0;
7	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
11	

### คำอธิบายชุดคำสั่ง

บรรทัดที่ 4 int a[5]={11,12,13,14,15}; เป็นการกำหนดค่าตัวแปรในลำดับต่างๆ

บรรทัดที่ 5 printf("%d\n",a[0]); เป็นการแสดงค่าตัวแปรในตำแหน่ง 0 คือ 11

โดยถ้าเปลี่ยนเป็น

printf("%d\n",a[1]); จะแสดงค่าตัวแปรในตำแหน่ง 1 คือ 12

printf("%d\n",a[2]); จะแสดงค่าตัวแปรในตำแหน่ง 2 คือ 13

โดยใน int a[ ] บรรทัดที่ 4 จะต้องใส่ขนาดข้อมูลให้ตรงกัน ถ้าใส่ต่ำกว่า จะเกิดข้อผิดพลาด ถ้าใส่มากกว่า คำตอบจะไม่ถูกต้อง

ตัวอย่างที่ 6.2 ตัวอย่างการนำข้อมูลในอาร์เรย์มาหาผลรวมและแสดงข้อมูลในอาร์เรย์แต่ละช่อง

1	#include <stdio.h>
2	#include <conio.h>
3	int a[] = {20,30,40,50,60};
4	void main()
5	{
6	int t = 0,n=0;
7	do {
8	t += a[++n];
9	printf ("Room %d has %3d pupils, \n", n+1, a[n]);
10	}while (n<5);
11	printf ("====\n");
12	printf ("Total no is %3d pupils \n", t);
13	getch();
14	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
Room 2 has 30 pupils,	
Room 3 has 40 pupils,	
Room 4 has 50 pupils,	
Room 5 has 60 pupils,	
Room 6 has 0 pupils,	
====	
Total no is 180 pupils	

### ตัวอย่างที่ 6.3 ตัวอย่างการนำข้อมูลในอาร์เรย์มาแสดงผล

```
1  #include <stdio.h>
2  void main()
3  {
4      int a;
5      char c[3];
6      c[0] = 'A', c[1] = 'B', c[2]='C';
7      for (a=0;a<=2;a++)
8      {
9          printf("data in array %d = %c \n",a,c[a]);
10     }
11     getch();
12 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
data in array 0 = A
data in array 1 = B
data in array 2 = C
```

ตัวอย่างที่ 6.4 ตัวอย่างการเก็บคะแนนรายบุคคล โดยจะให้อาจารย์ใส่คะแนนของนักศึกษาทั้งสาม แล้วหากอาจารย์ต้องการดูคะแนนของนักศึกษาคนไหน(คนแรกเป็นคน ที่ 0 คนสุดท้ายเป็นคน ที่ 2) ให้โปรแกรมแสดงคะแนนของนักศึกษาคนนั้น

```
1  #include <stdio.h>
2  int main()
3  {
4      int score[3];
5      int number;
6      printf("Student No.#0 : ");
7      scanf("%d",&score[0]);
8      printf("Student No.#1 : ");
9      scanf("%d",&score[1]);
10     printf("Student No.#2 : ");
11     scanf("%d",&score[2]);
12     printf("Enter number : ");
13     scanf("%d",&number);
14     printf("Student no.%d = %d point",number,score[number]);
15     return 0;
16 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Student No.#0 : 80
Student No.#1 : 72
Student No.#2 : 68
Enter number : 1
Student no.1 = 72 point
```

**ตัวอย่างที่ 6.5** ตัวอย่างการเก็บข้อมูลในอาร์เรย์โดยการระบุจำนวนที่ต้องการเก็บและวนลูปเก็บคะแนนของนักเรียนทั้งหมด N คนลงในอาร์เรย์ โดยให้อาจารย์ใส่คะแนนของนักเรียน N คน แล้วหากอาจารย์ต้องการดูคะแนนของนักศึกษาคนไหน(คนแรกเป็นคนที่ 0 คนสุดท้ายเป็นคนที่ N-1) ให้โปรแกรมแสดงคะแนนของนักศึกษาคนนั้น

```

1  #include <stdio.h>
2  int main()
3  {
4      int score[100];
5      int number;
6      int n,i;
7      printf("How many students in class : ");
8      scanf("%d",&n);
9      for( i=0 ; i<n;i++){
10         printf("Student No.##%d : ",i);
11         scanf("%d",&score[i]);
12     }
13     printf("Enter number : ");
14     scanf("%d",&number);
15     printf("Student no.%d = %d point",number,score[number]);
16     return 0;
17 }
```

**ผลลัพธ์จากการประมวลผลโปรแกรม**

```

How many students in class : 10
Student No.#0 : 32
Student No.#1 : 90
Student No.#2 : 65
Student No.#3 : 78
Student No.#4 : 95
Student No.#5 : 37
Student No.#6 : 66
Student No.#7 : 85
Student No.#8 : 49
Student No.#9 : 12
Enter number : 5
Student no.5 = 37 point
```

## 6.4 การประกาศและกำหนดค่าตัวแปรชุด 2 มิติ(two dimension arrays)

array 2 มิติ มีการจัดการจัดเก็บเปรียบเทียบกับ ตาราง 2 มิติ มิติที่ 1 เปรียบเหมือนแถว(row) ของตาราง มิติที่ 2 เปรียบคล้ายกับสดมภ์(column)ของตาราง ดังรูป

	column 0	column 1	column2	
row 0	x[0][0]	x[0][1]	x[0][2]	first dimension (row)
row 1	x[1][0]	x[1][1]	x[1][2]	
	second dimension (column)			

รูปนี้เป็น array ของ x[6] ซึ่งมีสมาชิกทั้งหมด 6 ตัว เปรียบเหมือนเก็บไว้ในตารางช่องละ 1 ตัว ดังนั้นจำนวนสมาชิกจะมีจำนวน เท่ากับ จำนวนแถว คูณ จำนวนสดมภ์ โดยหน่วยความจำที่ใช้เท่ากับหน่วยความจำที่ใช้โดยตัวแปรแต่ละตัว คูณ ด้วยจำนวนตัวแปรทั้งหมดใน array เช่น ในรูป ถ้าเป็น array ของ จำนวนเต็ม สมาชิกแต่ละตัวใช้หน่วยความจำ 2 ไบต์ หน่วยความจำที่ใช้ทั้ง คือ 2 x 6 คือ 12 ไบต์ เราอาจพิจารณา array 2 มิติ ว่าเป็น array ของ array 1 มิติ ดังรูป

10	25	12	
table[0][0]	table[0][1]	table[0][2]	table[0][3]
table[0]			
65			
table[1][0]	table[1][1]	table[1][2]	table[1][3]
table[1]			
table[2][0]	table[2][1]	table[2][2]	table[2][3]
table[2]			
table			

ในรูปมี array 1 มิติ 3 ชุด คือ table[0] table[1] และ table[2] ต่างก็เป็นสมาชิกใน array 2 มิติ ที่ชื่อ table ดังนั้น array ชื่อ table จะมีสมาชิกทั้งหมด คือ สมาชิกของ array ทั้งสาม คือ table[0][0] , table[0][1] ... ถึง table[2][3] ทั้งหมด 12 ตัว โดยในสมาชิกเหล่านี้ บางตำแหน่งมีข้อมูลแล้ว เช่น table[0][0] มีข้อมูลเป็น 10

การประกาศตัวแปร array 2 มิติ มีรูปแบบ คือ type arrayname [r] [c];

เมื่อ type คือ ชนิดของข้อมูล เช่น int , float ,char

arrayname คือ ชื่อของ array เช่น num , word ,x

r , c คือ จำนวนแถวและจำนวนสดมภ์ ตามลำดับ

โดยตัวเลขกำกับตำแหน่ง(ดัชนี) ที่อยู่แถว r เป็น 0,1,2 ... , r-1 ในคอลัมน์ c เป็น 0,1,2 ... , c-1

เช่น int num[3][4]; หมายความว่า มี array ของ integer ที่มีจำนวน 3 แถว 4 คอลัมน์ ดังนั้นมีสมาชิกทั้งหมด 12 ตัว เริ่มด้วย num[0][0] , num[0][1] ,num[0][2],num[0][3],num[1][0],...,num[2][3]

การกำหนดค่าอาจกำหนดในขั้นตอนประกาศตัวแปร เช่น

```
char word[ ][ ] = {"KMUTNB","CS","Computer Programming"};
```

หมายความว่า word เป็นตัวแปร array 2 มิติ ขนาด 3 x 20 ของสมาชิกที่เป็นอักขระ โดยแต่ละสมาชิก มีขนาด 1 ไบต์ โดย word[0] = "KMUTNB" word[1] = "CS" word[2] = " Computer Programming "

หรือ ประกาศ โดยยังไม่กำหนดค่า แล้วกำหนดค่าภายหลังเช่น int num[2][2];

```
num[0][1]=3;
```

## 6.5 ตัวอย่างการใช้งานอาร์เรย์ 2 มิติ

ตัวอย่างที่ 6.6 ตัวอย่างการเก็บตัวเลขจำนวนเต็มในอาร์เรย์ 2 มิติ และแสดงผลข้อมูลในอาร์เรย์

```
1  #include<stdio.h>
2  void main()
3  {
4      int tw[3][4] = {{2,4,6,8},{1,3,5,7},{1,2,3,4}};
5      int r,c;
6      for (r = 0;r<=2;r++)
7      {
8          printf ("\n");
9          for (c=0;c<=3;c++)
10             printf ("%d\t",tw[r][c]);
11     }
12     getch();
13 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
2      4      6      8
1      3      5      7
1      2      3      4
```

ตัวอย่างที่ 6.7 ตัวอย่างการเก็บชื่อคณะและรายนามนักเรียนในอาร์เรย์สองมิติ

```
1  #include <stdio.h>
2  void main()
3  {
4      char student[4];
5      int num[4][3],i,j;
6      char f[20];
7      for(i=0;i<=3;i++)
8      {
9          printf ("faculty :");
10         gets(f);
11         for (j=0;j<=2;++j)
12         {
13             printf ("student[%d][%d] = ",i,j);
14             gets(student);
15             num[i][j] = atoi(student);
16         }
17     }
18     getch();
19 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
faculty :Science
student[0][0] = Yaya
student[0][1] = Nadech
student[0][2] = Wier
faculty :Engineering
student[1][0] = Mark
student[1][1] = Tu
student[1][2] = Pu
faculty :Medicine
student[2][0] = Mario
student[2][1] = Kimberry
student[2][2] = Mai
faculty :Art
student[3][0] = Ying
student[3][1] = Ohm
student[3][2] = Weng
```



ตัวอย่างที่ 6.8 ตัวอย่างเรียงลำดับด้วยวิธี Bubble Sort จากน้อยไปมาก

```

1  #include <stdio.h>
2  void main()
3  {
4      int i,j;
5      float temp;
6      int N=10;
7      int list[10]={10,9,8,7,6,5,4,3,2,1};
8      printf("Befor sort:");
9      for (i=0;i<10;i++){
10         printf("%d ", list[i]);
11     }
12     for(i = N-1; i>=0; i--)
13     {
14         for(j = 1; j <=i; j++)
15         {
16             if(list[j-1] > list[j])
17             {
18                 temp = list[j-1];
19                 list[j-1] = list[j];
20                 list[j] = temp;
21             }
22         }
23     }
24     printf("\nAfter sort:");
25     for (i=0;i<10;i++){
26         printf("%d ", list[i]);
27     }
28 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

Befor sort:10 9 8 7 6 5 4 3 2 1  
After sort:1 2 3 4 5 6 7 8 9 10

ตัวอย่างที่ 6.9 ตัวอย่างเรียงลำดับด้วยวิธี Selection Sort จากน้อยไปมาก

```

1  #include <stdio.h>
2  void main()
3  {
4      int N=10;
5      int list[10]={10,9,8,7,6,5,4,3,2,1};
6      int i,j, MinIndex;
7      float temp;
8      printf("Befor sort:");
9      for (i=0;i<N;i++){
10         printf("%d ", list[i]);
11     }
12     for ( i = 0 ; i < ( N - 1 ) ; i++ )
13     {
14         MinIndex = i;
15         for ( j = i + 1 ; j < N ; j++ )
16         {
17             if ( list[MinIndex] > list[j] )
18                 MinIndex = j;
19         }
20         if ( MinIndex != i )
21         {
22             temp = list[i];
23             list[i] = list[MinIndex];
24             list[MinIndex] = temp;
25         }
26     }

```

27	printf("\nAfter sort:");
28	for (i=0;i<10;i++){
29	printf("%d ", list[i]);
30	}
31	}

ผลลัพธ์จากการประมวลผลโปรแกรม

Before sort:10 9 8 7 6 5 4 3 2 1

After sort:1 2 3 4 5 6 7 8 9 10

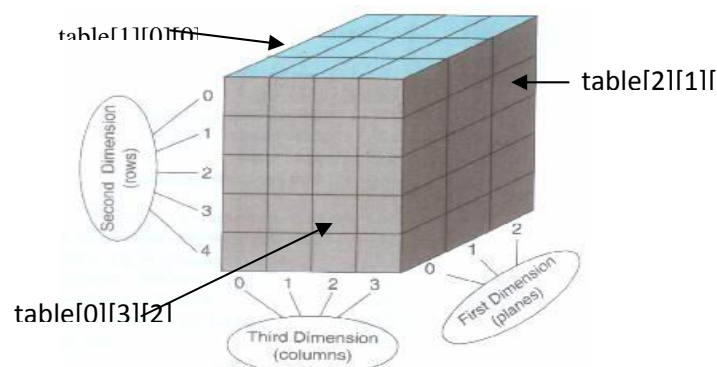
## 6.6 การประกาศและกำหนดค่าตัวแปรชุด 3 มิติ(Three dimension arrays)

ตัวแปร array 3 มิติ มีการประกาศ ดังนี้ type arrayname[p] [r][c];

type คือ ชนิดของตัวแปร เช่น int ,float,char, double

arrayname คือชื่อของตัวแปร

p,r,c คือตัวเลขแสดงจำนวนในมิติที่ 1 มิติที่ 2 และมิติที่ 3 ของ array ตามลำดับ โดยตัวเลขกำกับตำแหน่ง(ดัชนี)เป็นดังนี้ p เป็น 0,1,2 .. , p-1 r เป็น 0,1,2, ... ,r-1 c เป็น 0,1,2 ... ,c-1  
ลักษณะของ array 3 มิติ เป็น arrays of arrays ดังรูป



จากตัวอย่างข้างต้นเป็นการประกาศอาร์เรย์ที่มีจำนวน 3 มิติ โดยมิติที่ 1 จะมีจำนวน 3 ระนาบ มิติที่ 2 จะมีจำนวนแถวทั้งหมด 5 แถว มิติที่ 3 จะมีจำนวนคอลัมน์ทั้งหมด 4 คอลัมน์หรืออาจจะเขียนในรูปแบบของการจองพื้นที่ของภาษาซีได้เป็น table[3][5][4] เราสามารถพิจารณารูปแบบในการเก็บข้อมูลของการประกาศอาร์เรย์ดังกล่าวได้ดังนี้ เช่น

table[0][0][0] ถึง table[0][0][2] เป็นสมาชิกใน table[0][0]

table[0][0] ถึง table[0][4] เป็นสมาชิกใน table[0]

table[0] ถึง table[2] เป็นสมาชิกใน table

โดยจำนวนสมาชิกใน array เท่ากับ p \* r \* c เช่น array ในรูป มีสมาชิกทั้งหมด 3 \* 5 \* 4 คือ 60 สมาชิก ส่วนการกำหนดค่าของสมาชิกของ array แต่สมาชิกก็เป็นทำนองเดียวกับ array 1 มิติ และ array 2 มิติ เช่นเดียวกับจำนวนหน่วยความจำที่ใช้ก็คือหน่วยความจำที่สมาชิกแต่ละตัวใช้ คุณด้วยจำนวนสมาชิกทั้งหมด ในขณะเดียวกันเรายังสามารถสร้างอาร์เรย์ 4 มิติได้โดยเขียนให้อยู่ในรูปสัญลักษณ์ type arrayname[ ][...][ ]

## 6.7 ตัวอย่างการใช้งานอาร์เรย์ 2 มิติ

ตัวอย่างที่ 6.10 ตัวอย่างการเก็บตัวเลขจำนวนเต็มในอาร์เรย์และแสดงผลข้อมูลในอาร์เรย์ตัวแรก

```
1  #include <stdio.h>
2  int main()
3  {
4      int arr[3][4][5];
5      int i, j, k, sum = 0;
6      for(i = 0; i < 3; i++)
7          for(j = 0; j < 4; j++)
8              for(k = 0; k < 5; k++)
9                  {
10                     scanf("%d", &arr[i][j][k]);
11                     sum = sum + arr[i][j][k];
12                 }
13     printf("sum is %d", sum);
14 }
```

### ผลลัพธ์จากการประมวลผลโปรแกรม

โปรแกรมจะวนรับค่าใส่ในอาร์เรย์ตามมิติทั้งหมด 60 รอบและหาผลรวม

## แบบฝึกหัดปฏิบัติการคาบที่ 6: Arrays

1. จงเขียนผังงานและโปรแกรมเพื่อรับตัวเลขเข้ามา n จำนวน จากนั้นหาค่าเฉลี่ย หาค่ามากที่สุด ค่าน้อยสุด ค่า SD
2. ในการแข่งขันวิ่งระยะ 100 เมตร รายการหนึ่ง มีนักวิ่งแข่งขันทั้งสิ้น 10 คน จงเขียนโปรแกรมเพื่อรับอินพุตจากผู้เข้าซึ่งได้แก่ ใอดีของนักวิ่ง (ID) และเวลาที่ใช้ในการวิ่ง (time) มีหน่วยเป็นวินาที จากนั้นโปรแกรมจะต้องแสดงสามอันดับแรกของนักวิ่ง (ID) ที่ใช้เวลาในการวิ่งน้อยที่สุดตามลำดับ กำหนดให้เวลามากที่สุดที่ใช้ในการวิ่ง 100 เมตร ต้องไม่เกิน 20.00 วินาที

### 3. [ตรวจสอบรหัสบัตรประชาชนอย่างไร]

แบบฟอร์มในการสมัครสมาชิกของหลายๆ เว็บไซต์บังคับให้กรอกเลขที่บัตรประชาชน เช่น เว็บไซต์ขายของ, เว็บไซต์ประเภทเกมออนไลน์ ฯลฯ ในการตรวจสอบเลขที่บัตรประชาชนนั้นทำได้โดยการใช้ Check Digit หรือการตรวจสอบความถูกต้องโดยใช้ตัวเลขหลักสุดท้ายในการตรวจสอบ วิธีการ Check Digit มีดังนี้

1. ตัวเลขบนบัตรประชาชนจะมีทั้งหมด 13 หลัก นำเลขใน 12 หลักแรก มาคูณกับเลขประจำตำแหน่ง (เลขประจำหลักได้แก่ 13 บวก 1 ลบด้วยตำแหน่งที่) จะได้ตัวเลขประจำตำแหน่งดังนี้

ตำแหน่งที่	เลขประจำตำแหน่ง
1	13
2	12
3	11
4	10
5	9
6	8
7	7
8	6
9	5
10	4
11	3
12	2

2. หลังจากนั้นเอาผลคูณของทั้ง 12 หลักมารวมกัน แล้ว modulation (การหารเอาเศษ) ด้วย 11
3. เอาเศษที่ได้จากการหารในข้อ 2 มาลบด้วย 11 จะได้ Check Digit (ถ้าผลจากข้อ 2 ได้ 10 ให้เอาเลขหลักหน่วยเป็น Check Digit ก็คือ 0 นั่นเอง)

### ตัวอย่าง

ต้องการเช็คว่ารหัสบัตรประชาชน 1234567890129 ถูกต้องหรือไม่ ทำได้โดย

1. นำตัวเลขคูณเลขประจำตำแหน่ง

$$(1*13)+(2*12)+(3*11)+(4*10)+(5*9)+(6*8)+(7*7)+(8*6)+(9*5)+(0*4)+(1*3)+(2*2) = 352$$

2.เอาผลคูณของทั้ง 12 หลักมารวมกัน แล้ว modulation (การหารเอาเศษ) ด้วย 11 จะได้

$$352 \% 11 = 0$$

3.นำ 11 ตั้งแล้วลบด้วย 0 จะได้

$$11 - 0 = 11 \text{ เอาเลขหลักหน่วย ดังนั้น Check Digit คือ } 1$$

4. นำ Check digit ที่ได้ในข้อ 3 ไปเปรียบเทียบกับรหัสบัตรประชาชนตำแหน่งที่ 13

เพราะฉะนั้นเลขที่บัตรประชาชน 1234567890129 ไม่ถูกต้อง ที่ถูกต้องคือ 1234567890121

จากขั้นตอนดังกล่าว จงเขียนโปรแกรมเพื่อรับค่าเลขรหัสประจำตัวประชาชนจากผู้ใช้ หลังจากนั้นให้เช็คว่ารหัสดังกล่าวถูกต้องหรือไม่ โดยกรณีถูกต้องแสดงผลคำว่า Valid ส่วนถ้าไม่ถูกต้อง ให้แสดงผลคำว่า Invalid เช่น

Please Enter ID: 3340100019856

Your ID is Valid

### การเขียนโปรแกรมส่งผ่าน Grader

- [SD] จงเขียนโปรแกรมคำนวณส่วนเบี่ยงเบนมาตรฐาน ส่วนสูงของคน n คน รับข้อมูลส่วนสูงจากผู้ใช้นั้นคำนวณหาค่าส่วนเบี่ยงเบนมาตรฐาน

$$SD = \left( \frac{1}{5} \left( \sum_{i=1}^5 (x_i - \bar{x}) \right) \right)^{1/2}$$

$x_i$  = ส่วนสูงคนที่  $i$

$\bar{x}$  = ส่วนสูงเฉลี่ยของข้อมูล

ข้อมูลอินพุต มี 2 บรรทัด บรรทัดแรกแสดงจำนวนคน n คน

บรรทัดที่ 2 รับข้อมูลส่วนสูงจากผู้ใช้นั้น n คน

ข้อมูลเอาต์พุต มี 1 บรรทัด แสดงค่าส่วนเบี่ยงเบนมาตรฐาน

ตัวอย่าง

อินพุต	เอาต์พุต
5 160 178 169 177 168	6.59
5 185 166 172 167 177	7.00

- [Matrix Transpose] จงเขียนโปรแกรมเพื่อคำนวณหาค่าทรานโพสของเมตริกซ์(Matrix Transpose)

A ขนาด  $n \times n$  คือ  $A^T_{n \times n}$

$$A = \begin{matrix} 1 & 5 & 3 & 7 \\ 2 & 6 & 9 & 2 \\ 4 & 10 & 1 & 0 \\ 9 & 0 & 8 & 7 \end{matrix}$$

$$A^T = \begin{matrix} 1 & 2 & 4 & 9 \\ 5 & 6 & 10 & 0 \\ 3 & 9 & 1 & 8 \\ 7 & 2 & 0 & 7 \end{matrix}$$

ข้อมูลอินพุต มี n+1 บรรทัด บรรทัดแรกเป็นขนาดของเมตริกซ์

บรรทัดที่ 2-+1 ประกอบสมาชิกของเมทริกซ์ A ขนาด nxn

ข้อมูลเอาต์พุต มี n บรรทัด ประกอบสมาชิกของเมทริกซ์  $A^T$  ขนาด nxn

ตัวอย่าง

อินพุต	เอาต์พุต
4 4	1 2 4 9
1 5 3 7	5 6 10 0
2 6 9 2	3 9 1 8
4 10 1 0	7 2 0 7
9 0 8 7	

3. [Matrix Addition] จงเขียนโปรแกรมเพื่อคำนวณหาผลบวกของเมทริกซ์(Matrix Addition) ขนาด nxn คือ

$$C_{n \times n} = A_{n \times n} + B_{n \times n}$$

เช่น n=4

A =    1 5 3 7  
          2 6 9 2  
          4 10 1 0  
          9 0 8 7

B =    11 3 2 1  
          5 -5 0 4  
          12 1 1 3  
          7 5 -4 6

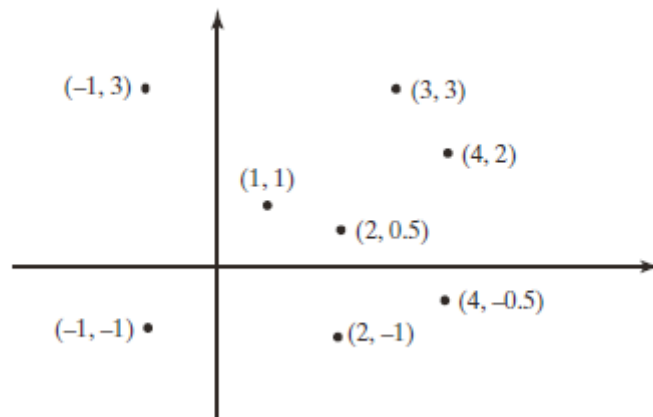
ข้อมูลอินพุต มี n+1 บรรทัด บรรทัดแรกแสดงจำนวนของแถวและคอลัมน์ของเมทริกซ์ทั้งสอง  
 บรรทัดที่ 2 ถึง n\*2 รับข้อมูลของเมทริกซ์ A และ B

ข้อมูลเอาต์พุต มี n บรรทัด แสดงผลบวกของเมทริกซ์(Matrix Addition) ขนาด nxn

ตัวอย่าง

อินพุต	เอาต์พุต
4 4	12 8 5 8
1 5 3 7	7 1 9 6
2 6 9 2	16 11 2 3
4 10 1 0	16 5 4 13
9 0 8 7	
11 3 2 1	
5 -5 0 4	
12 1 1 3	
7 5 -4 6	

4. [Nearest point] จงเขียนโปรแกรมเพื่อคำนวณหาจุดที่ใกล้เคียงกันมากที่สุด



ข้อมูลอินพุต มี  $n+1$  บรรทัด บรรทัดแรกแสดงจำนวนของจุด

บรรทัดที่  $2 - n+1$  รับข้อมูลของจุดแต่ละจุด

ข้อมูลเอาต์พุต มี  $n$  บรรทัด แสดงคู่จุดที่ใกล้เคียงกันมากที่สุด และระยะห่างระหว่างจุดดังกล่าว

ตัวอย่าง

อินพุต	เอาต์พุต
8	3 5 1.12
-1 3	
3 3	
1 1	
4 2	
2 0.5	
4 -0.5	
2 -1	
-1 -1	

5. [MH320] ในวันปีใหม่ สนามกีฬาแห่งหนึ่งได้ประดับไฟที่สนามฟุตบอลเพื่อความสวยงาม ในการประดับไฟนั้นทำโดยแบ่งสนามสี่เหลี่ยมเป็นช่องย่อยๆ จำนวน  $N$  แถว แถวละ  $M$  คอลัมน์ รวม  $N \times M$  ช่อง เจ้าของสนามได้เปิดสนามให้ประชาชนทั่วไปเข้าชมเพื่อความสวยงาม

อย่างไรก็ตาม โลกนี้ไม่มีอะไรฟรี เจ้าของสนามจะต้องจ่ายค่าไฟให้กับไฟประดับเหล่านี้ เนื่องจากมีการประดับไฟเป็นสวดลายต่างๆ ค่าไฟของไฟแต่ละช่องไม่จำเป็นต้องเท่ากัน

เพื่อไม่ให้เป็นการขาดทุน เจ้าของสนามจึงได้จัดเครื่องโพนเจ็ตส่วนบุคคลให้กับประชาชนเช่าเพื่อบินดู ไฟประดับ เครื่องโพนเจ็ตแต่ละเครื่องเมื่อเช่าไปแล้วจะผู้ใช้จะสามารถบินได้ทั้งสิ้น  $K$  ครั้ง ในการบินแต่ละครั้งจะใช้เชื้อเพลิงมูลค่าเท่ากับ  $L$  บาท ดังนั้น ค่าใช้จ่ายรวมทั้งหมดของเจ้าของสนามคือค่าไฟรวมของไฟประดับ และค่าเชื้อเพลิงรวมของการบินเครื่องโพนเจ็ตในการบินทั้งหมด

เจ้าของสนามทราบว่าจะมีคนมาชมและเช่าเครื่องโพนเจ็ตจำนวน  $C$  คน เขาต้องการคำนวณค่าเช่าเครื่องโพนเจ็ตต่อคนที่น้อยที่สุด ที่จะทำให้เขาไม่ขาดทุน เพื่อให้การเช่าเป็นไปได้สะดวก ค่าเช่าจะต้องเป็น

จำนวนเต็มเสมอด้วย

เขียนโปรแกรมรับราคาค่าไฟ ของสนามแต่ละช่อง รวมทั้งข้อมูลของการใช้เครื่องไอพ่นเจ็ต จากนั้นคำนวณหาค่าเช่าเครื่องไอพ่นเจ็ตต่อคนที่เป็จำนวนเต็มทีน้อยที่สุด ที่จะทำให้เจ้าของสนามไม่ขาดทุน

### ข้อมูลอินพุท

บรรทัดแรกระบุจำนวนเต็มบวก N และ M คั่นด้วยช่องว่าง แทนขนาดความกว้างและความยาวของสนาม ( $1 \leq N \leq 100$ ;  $1 \leq M \leq 100$ )

บรรทัดที่สองระบุจำนวนเต็มบวก L และ K คั่นด้วยช่องว่าง โดยที่ L แทนราคาเชื้อเพลิงต่อการบินหนึ่งครั้ง และ K แทนจำนวนครั้งที่เครื่องไอพ่นใช้บินได้ต่อคนเช่าหนึ่งคน ( $1 \leq L \leq 100$ ;  $1 \leq K \leq 100$ )

บรรทัดที่สามระบุจำนวนเต็มบวก C แทนจำนวนผู้เล่นทั้งหมดที่เข้ามาเล่น ( $1 \leq C < 1,000$ )

บรรทัดถัดไปอีก N บรรทัด แต่ละบรรทัดรับจำนวนเต็มบวก M ตัว แต่ละตัวถูกคั่นด้วยช่องว่าง แทนค่าไฟในแต่ละช่องที่ประดับไฟ ซึ่งจะเป็นจำนวนเต็มบวกที่มีค่าไม่เกิน 3,000

### ข้อมูลเอาต์พุท

มีบรรทัดเดียวเป็นจำนวนเต็มบวกหนึ่งจำนวน แทนค่าเช่าเครื่องไอพ่นเจ็ตต่อคนที่เป็จำนวนเต็มทีน้อยที่สุด ที่จะทำให้เจ้าของสนามไม่ขาดทุน

### ตัวอย่าง

อินพุท	เอาต์พุท
3 3 2 1 1 1 1 1 1 1 1 1 1 1	11
3 4 3 2 7 1 2 3 4 4 3 2 1 1 1 1 1	10

## 6. [Racing] ขับรถหลบสิ่งกีดขวาง (Car)[คะแนนพิเศษ+10 คะแนน]

ในการแสดงขับรถผาดโผนบนถนนที่มีเลนทั้งหมด m เลน โดยให้หมายเลขประจำเลนจากซ้ายไปขวามีค่าตั้งแต่ 1 จนถึง m ตามลำดับ นักแสดงขับรถผาดโผนต้องบังคับรถให้แล่นไปบนถนนดังกล่าวให้ปลอดภัยตลอดระยะเวลา t หน่วย การแสดงเริ่มต้น ณ เวลา 0 ถึง t นักแสดงขับรถผาดโผนอยู่ในเลนที่ n ในแต่ละ 1 หน่วยเวลา อาจมีสิ่งกีดขวางตกลงมายังถนนบางเลน ทำให้เขาต้องบังคับรถเพื่อหลีกเลี่ยงสิ่งกีดขวาง ซึ่งมีทางเลือกในการบังคับรถอยู่ 3 แบบ ได้แก่ 1 หมายถึง การเปลี่ยนเลนไปทางซ้าย 1 เลนในเวลาถัดไปยังเลนที่มี



หมายเลขประจำเลนน้อยกว่า 2 หมายถึงการเปลี่ยนเลนไปทางขวา 1 เลนในเวลาถัดไป (ไปยังเลนที่มีหมายเลขประจำเลนมากกว่า) และ 3 หมายถึง การขับอยู่ในเลนเดิม กำหนดให้ถนนเป็นเส้นตรงตลอดทาง จงเขียนโปรแกรมเพื่อบังคับให้รถแล่นไปตามเส้นทางนี้โดยปลอดภัย โดยชุดข้อมูลทดสอบจะมีคำตอบที่ถูกต้องเพียง 1 คำตอบเสมอ

### ข้อมูลนำเข้า

- บรรทัดแรกระบุจำนวนเลน  $m$  โดยที่  $2 \leq m \leq 40$
- บรรทัดที่สองระบุหมายเลขเลนเริ่มต้น  $1 \leq n \leq m$
- บรรทัดที่สามระบุระยะเวลา  $t$  โดยที่  $1 \leq t \leq 100$
- บรรทัดที่สี่ถึงบรรทัดที่  $t+3$  แสดงสถานะของถนน ณ เวลา  $t=1, 2, \dots, K$  ตามลำดับ แต่ละบรรทัดระบุตัวเลข  $m$  ตัวเลขแต่ละตัวแสดงสถานะของถนนตั้งแต่เลนที่ 1 ถึงเลนที่  $m$  โดยเลข 0 หมายถึงเลนนั้นไม่มีสิ่งกีดขวาง และเลข 1 หมายถึงมีสิ่งกีดขวางอยู่

### ข้อมูลส่งออก

มีอยู่  $t$  บรรทัด แต่ละบรรทัดมีตัวเลข 1 ตัวเพื่อแสดงถึงทางเลือกในการบังคับรถของนักแสดงขับรถผาดโผน ในแต่ละช่วงเวลา บรรทัดที่  $i$  หมายถึงการเปลี่ยนเลนจากเวลาที่  $i-1$  ไปยังเวลาที่  $i$  เมื่อ  $i=1, 2, \dots, t$  โดยที่เลข 1 จะหมายถึงขับไปทางซ้าย 1 เลน, เลข 2 หมายถึงขับไปทางขวา 1 เลน, และเลข 3 หมายถึงขับอยู่ในเลนเดิม

### ตัวอย่าง

อินพุต	เอาต์พุต
7	1
5	1
5	1
0 0 0 0 0 0 0	1
0 0 0 0 0 0 0	2
0 0 0 0 0 0 0	
0 1 1 0 0 0 0	
1 0 1 1 1 1 1	

## บทที่ 7

### ฟังก์ชัน (Function)

#### จุดประสงค์

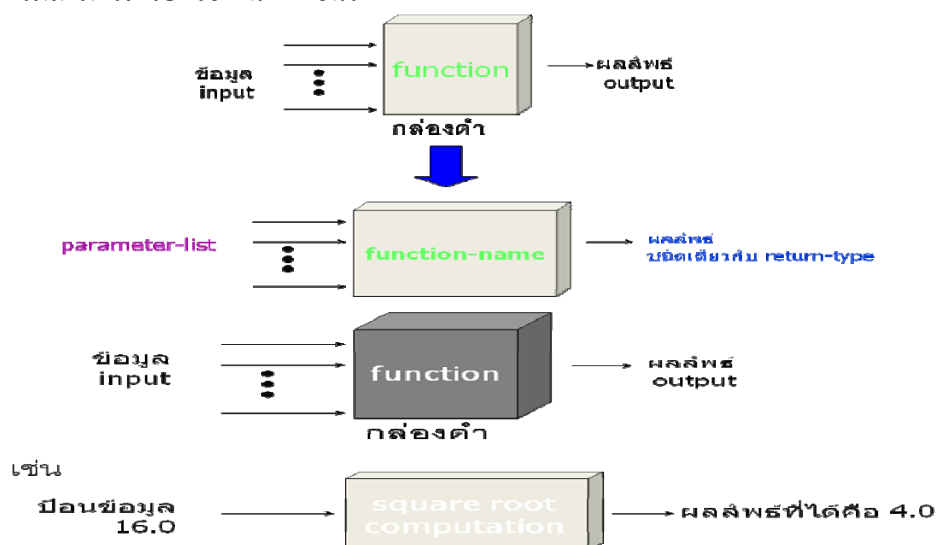
1. เพื่อให้ทราบรูปแบบการใช้งานของฟังก์ชัน
2. เพื่อให้ทราบวิธีการเรียกใช้ฟังก์ชันที่สร้างขึ้นเอง
3. สามารถสร้างฟังก์ชันขึ้นเพื่อใช้งานในโปรแกรมได้อย่างมีประสิทธิภาพ
4. เพื่อให้ทราบความหมายและลักษณะของโครงสร้างโปรแกรม

#### 7.1 ฟังก์ชันคืออะไร

โปรแกรมคอมพิวเตอร์ที่ใช้แก้ปัญหาต่าง ๆ ส่วนมากจะมีขนาดใหญ่ มีจำนวนชุดคำสั่ง จำนวนมาก วิธีการที่จะช่วยในการพัฒนาโปรแกรมขนาดใหญ่เหล่านี้เพื่อช่วยให้โปรแกรมเขียนได้ง่ายและสามารถแก้ไขต่อไปในอนาคต คือการเขียนให้อยู่ในรูปของฟังก์ชันหรือการแบ่งการทำงานของโปรแกรมออกเป็นส่วน ๆ สำหรับการแก้ปัญหาที่มีขนาดเล็ก ซึ่งภายหลังจากที่มีการแก้ปัญหาเล็ก ๆ เหล่านั้นเสร็จ สามารถที่จะเอาคำตอบจากปัญหาเล็ก ๆ เข้ามารวมให้เป็นผลลัพธ์ที่มีขนาดใหญ่ขึ้น ซึ่งเรียกเทคนิคการแบ่งปัญหาแบบนี้ว่า divide and conquer

สรุปแล้วความหมายของฟังก์ชัน(function) คือ ส่วนของโปรแกรมที่ทำงานเสร็จสิ้นภายในตัวเอง มีลักษณะเหมือนกับโปรแกรมย่อย ที่รวมอยู่ในโปรแกรมหลักอีกทีหนึ่งใช้หลักการแบ่งแยกและรวม( *Divide and Conquer*) แบ่งการทำงานออกเป็นส่วนเล็กๆ ที่ทำงานเสร็จสมบูรณ์ในตัว มีการทดสอบและแก้ไขส่วนเล็กๆนี้ ประกอบส่วนเล็กๆนี้ ขึ้นมาเป็นโปรแกรมขนาดใหญ่ที่สมบูรณ์ในขั้นตอนสุดท้าย

ฟังก์ชัน หรือ Procedure คือ ชุดของ statements ที่ทำงานอย่างใดอย่างหนึ่ง และมีชื่อเรียก ส่วนอื่นของโปรแกรมสามารถเรียกใช้งานฟังก์ชันได้



รูปที่ 7.1 แสดงลักษณะการทำงานของ function เมื่อเปรียบเทียบกับกล่องดำ

โปรแกรมภาษา C ประกอบไปด้วยหนึ่งฟังก์ชัน (main) หรือมากกว่าแต่ละฟังก์ชันประกอบไปด้วยหนึ่ง statement หรือมากกว่า

- ภาษา C แบ่งฟังก์ชันเป็น 2 แบบ
  - ฟังก์ชันมาตรฐานใน C (Standard Library function)
  - ฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม (User-Defined function)

โดยฟังก์ชันมาตรฐานของภาษาซี เป็นฟังก์ชันที่ภาษาซีได้จัดเตรียมไว้ให้ผู้เขียนโปรแกรมเรียบร้อยแล้ว ผู้ใช้สามารถเรียกใช้ได้ทันที เพียงแต่ต้องทราบว่าฟังก์ชันดังกล่าวมีการประกาศไว้ในเฮดเดอร์ไฟล์ตัวใดแล้วเพิ่มบรรทัดคำสั่ง `#include <ชื่อ header file>` เพื่อรวม header file นั้นเข้าไปในการแปลโปรแกรมด้วย ส่วนฟังก์ชันสร้างเองจะเป็นฟังก์ชันที่ผู้เขียนโปรแกรมสร้างขึ้น เพื่อใช้ในการจัดการกับปัญหาต่าง ๆ ตามที่ต้องการ

## 7.2 ฟังก์ชันมาตรฐานใน C (C Standard Library)

ฟังก์ชันมาตรฐานเป็นฟังก์ชันที่ผู้ผลิต C compiler เป็นผู้เขียนขึ้น โดยเก็บไว้ใน C library ประกอบด้วยฟังก์ชันเกี่ยวกับ

- disk I/O (input/output), standard I/O เช่น `printf()`, `scanf()`, ...
- string manipulation เช่น `strlen()`, `strcpy()`, ...
- mathematics เช่น `sqrt()`, `sin()`, `cos()`, ...
- และฟังก์ชันอื่น ๆ เป็นต้น

ฟังก์ชันมาตรฐานสามารถเรียกใช้งานได้เลย แต่ต้องมีการ include header file ที่นิยามฟังก์ชันนั้นๆไว้ เช่น

1. จะใช้ `printf()`, `scanf()` ต้องเขียน `#include <stdio.h>`
2. จะใช้ `sqrt()`, `sin()`, `cos()` ต้องเขียน `#include <math.h>`

### 7.2.1 ฟังก์ชันมาตรฐานทางคณิตศาสตร์

ถ้าต้องการใช้ฟังก์ชันมาตรฐานทางคณิตศาสตร์จะต้องนำเข้า Standard Library Files 2 ไฟล์ คือ `math.h` และ `stdlib.h` ฟังก์ชันทางคณิตศาสตร์ เป็นฟังก์ชันที่ใช้ทางการคำนวณทางคณิตศาสตร์ ปกติอยู่ใน `math.h` ผลลัพธ์ ที่ได้จากฟังก์ชันกลุ่มนี้เป็นข้อมูลประเภท `double` ดังนั้นตัวแปรที่ใช้จึงเป็นพวกที่มีชนิดเป็น `double`

1. ฟังก์ชัน **`sin(x)`** เป็นฟังก์ชันใช้คำนวณค่าของ sine โดย `x` มีค่าของมุมในหน่วย เรเดียน
2. ฟังก์ชัน **`cos(x)`** ใช้หาค่า cosine โดย `x` มีหน่วยเป็นเรเดียน (radian)
3. ฟังก์ชัน **`tan(x)`** ใช้หาค่า tangent โดย `x` มีหน่วยเป็นเรเดียน (radian)

## ตัวอย่างที่ 7.1

```

1  #include <stdio.h>
2  #include <math.h>
3  main()
4  {
5      float deg , angle, pi = 3.141592654;
6      printf("Please enter value of angle in degree that you want to find tan
7  cos sin :");
8      scanf("%f",&deg);
9      angle = deg * pi / 180;
10     printf("\nvalue of tangent %4.0f degree is %4.2f ",deg,tan(angle));
11     printf("\nvalue of sine %4.0f degree is %4.2f ",deg,sin(angle));
12     printf("\nvalue of cosine %4.0f degree is %4.2f ",deg,cos(angle));
13 }

```

### ผลลัพธ์จากการประมวลผลโปรแกรม

Please enter value of angle in degree that you want to find tan cos sin :45

value of tangent 45 degree is 1.00

value of sine 45 degree is 0.71

value of cosine 45 degree is 0.71

4. ฟังก์ชัน `sqrt(x)` ใช้หาค่ารากที่สองของ  $x$  โดย  $x$  เป็นตัวเลขหรือตัวแปรที่ไม่ติดลบ
5. ฟังก์ชัน `exp(x)` ใช้หาค่า  $e^x$  โดย  $e$  มีค่าประมาณ 2.718282
6. ฟังก์ชัน `pow(x,y)` ใช้หาค่า  $x^y$
7. ฟังก์ชัน `log(x)` ใช้หาค่า  $\log$  ฐาน  $e$  เรียกว่า natural logarithm โดย  $x$  เป็นตัวเลขหรือตัวแปรที่ไม่ติดลบ
8. ฟังก์ชัน `log10(x)` ใช้หาค่า  $\log$  ฐาน 10 โดย  $x$  เป็นตัวเลขหรือตัวแปรที่ไม่ติดลบ
9. ฟังก์ชัน `ceil(x)` ใช้ในการปัดเศษทศนิยมของ  $x$  เมื่อ  $x$  เป็นเลขทศนิยม
10. ฟังก์ชัน `floor(x)` ใช้ในการตัดเศษทศนิยมของ  $x$  ทั้งเมื่อ  $x$  เป็นเลขทศนิยม
11. ฟังก์ชัน `fabs(x)` ใช้ในการหาค่าสัมบูรณ์ของค่าคงที่หรือตัวแปรที่มีทศนิยม โดยเป็นบวกหรือลบก็ได้

## ตัวอย่างที่ 7.2

```

1  #include <stdio.h>
2  #include <math.h>
3  main()
4  {
5      double x = 10.0, y = 2.0 ,z = 16.0,a = 2.718282, b = -2.718282 , m=1.0;
6      printf("\npow\(\x,y\) = %4.2f when x=10.0 y=2.0", pow(x,y));
7      printf("\nsqrt\(\z\) = %4.2f when z=16.0", sqrt(z));
8      printf("\nexp\(\m\) = %4.6f when m=1.0", exp(m));
9      printf("\nlog\(\a\) = %4.2f when a=2.718282", log(a));
10     printf("\nlog10\(\x\) = %4.2f when x=10.0", log10(x));
11     printf("\nceil\(\a\) = %4.2f when a=2.718282", ceil(a));
12     printf("\nceil\(\b\) = %4.2f when b=-2.718282", ceil(b));
13     printf("\nfloor\(\a\) = %4.2f when a=2.718282", floor(a));
14     printf("\nfloor\(\b\) = %4.2f when b=-2.718282", floor(b));
15     printf("\nfabs\(\a\) = %4.6f when a=2.718282", fabs(a));
16     printf("\nfabs\(\b\) = %4.6f when b=-2.718282", fabs(b));
17 }

```

### ผลลัพธ์จากการประมวลผลโปรแกรม

pow(x,y) = 100.00 when x=10.0 y=2.0

sqrt(z) = 4.00 when z=16.0

exp(m) = 2.718282 when m=1.0

```
log(a) = 1.00 when a=2.718282
log10(x) = 1.00 when x=10.0
ceil(a) = 3.00 when a=2.718282
ceil(b) = -2.00 when b=-2.718282
floor(a) = 2.00 when a=2.718282
floor(b) = -3.00 when b=-2.718282
fabs(a) = 2.718282 when a=2.718282
fabs(b) = 2.718282 when b=-2.718282
```

## 7.2.1 ฟังก์ชันที่จัดการเกี่ยวกับตัวอักษร(character functions)

เป็นฟังก์ชันที่จัดการกับตัวอักษร single char เท่านั้น ตัวอักษรนี้ใช้หน่วยความจำเพียง 1 ไบต์ ฟังก์ชันเหล่านี้อยู่ใน header file ชื่อ ctype.h ก่อนจะทำการเขียนโปรแกรมจึงต้อง #include <ctype.h>

**1.ฟังก์ชัน isalnum(cha)** เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลในตัวแปร(ซึ่งคือตัวแปรประเภท char ) เป็นตัวอักษรหรือตัวเลขหรือไม่ ถ้าเป็นตัวอักษรหรือตัวเลข ฟังก์ชันจะส่งค่าที่ไม่ใช่ 0 มาให้ ถ้าข้อมูลในตัวแปร เป็นอักขระพิเศษอื่นที่ไม่ใช่ตัวอักษรหรือตัวเลขจะส่งค่าออกมาเป็น 0

**2.ฟังก์ชัน isalpha(cha)** เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลในตัวแปร(ซึ่งคือตัวแปรประเภท char ) เป็นตัวอักษรหรือไม่ ถ้าเป็นตัวอักษรฟังก์ชันจะให้ค่าที่ไม่ใช่ 0 ออกมาถ้าเป็นตัวเลขหรืออักขระพิเศษอื่น ฟังก์ชันจะส่งค่า 0 ออกมา

**3.ฟังก์ชัน isdigit(cha)** เป็นฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลในตัวแปร(ซึ่งคือตัวแปรประเภท char) ฟังก์ชัน เป็นตัวเลขหรือไม่ ถ้าเป็นตัวเลขฟังก์ชันจะให้ค่าที่ไม่ใช่ 0 ออกมา ถ้าเป็นตัวอักษรหรืออักขระพิเศษอื่น ฟังก์ชันจะส่ง 0 ออกมา

ตัวอย่างที่ 7.3

```
1  #include <stdio.h>
2  #include <ctype.h>
3  main()
4  {
5      char cha1 = 'B', cha2 = '3', cha3 = '&';
6      printf("\n %d is return value of isdigit\\(cha1\\) of %c", isdigit(cha1), cha1);
7      printf("\n %d is return value of isdigit\\(cha2\\) of %c ", isdigit(cha2), cha2);
8      printf("\n %d is return value of isalpha\\(cha3\\) of %c ", isalpha(cha3), cha3);
9      printf("\n %d is return value of isalpha\\(A\\) of %c ", isalpha('A'), 'A');
10     printf("\n %d is return value of isalpha\\('0'\\) of %c ", isalpha('0'), '0');
11     printf("\n %d is return value of isalpha\\('$'\\) of %c ", isalpha('$'), '$');
12     printf("\n %d is return value of isalnum\\(cha1\\) of %c", isalnum(cha1), cha1);
13     printf("\n %d is return value of isalnum\\(cha2\\) of %c ", isalnum(cha2), cha2);
14     printf("\n %d is return value of isalnum\\(cha3\\) of %c ", isalnum(cha3), cha3);
15     printf("\n %d is return value of isalnum\\(A\\) of %c ", isalnum('A'), 'A');
16     printf("\n %d is return value of isalnum\\('0'\\) of %c ", isalnum('0'), '0');
17     printf("\n %d is return value of isalnum\\('$'\\) of %c ", isalnum('$'), '$');
18 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
0 is return value of isdigit(cha1) of B
4 is return value of isdigit(cha2) of 3
0 is return value of isalpha(cha3) of &
1 is return value of isalpha(A) of A
0 is return value of isalpha('0') of 0
0 is return value of isalpha('$') of $
1 is return value of isalnum(cha1) of B
4 is return value of isalnum(cha2) of 3
0 is return value of isalnum(cha3) of &
1 is return value of isalnum(A) of A
4 is return value of isalnum('0') of 0
0 is return value of isalnum('$') of $
```

4. ฟังก์ชัน `islower(cha)` ฟังก์ชันที่ใช้ตรวจสอบว่าตัวอักษรในตัวแปร `cha` เป็นตัวพิมพ์เล็กหรือไม่ ถ้าเป็นตัวพิมพ์เล็กฟังก์ชันจะส่งค่ากลับเป็นจำนวนเต็มที่ไม่ใช่ 0 แต่ถ้าไม่ใช่ตัวพิมพ์เล็กจะส่งค่ากลับเป็น 0

5. ฟังก์ชัน `isupper(cha)` ฟังก์ชันที่ใช้ตรวจสอบว่าตัวอักษรในตัวแปร `cha` เป็นตัวพิมพ์ใหญ่หรือไม่ ถ้าเป็นตัวพิมพ์ใหญ่ ฟังก์ชันจะส่งค่ากลับเป็นจำนวนเต็มที่ไม่ใช่ 0 แต่ถ้าไม่ใช่ตัวพิมพ์ใหญ่จะส่งค่ากลับเป็น 0

6. ฟังก์ชัน `tolower(cha)` ฟังก์ชันที่ใช้เปลี่ยนตัวพิมพ์ใหญ่ที่เก็บอยู่ในตัวแปรให้เป็นตัวพิมพ์เล็ก

7. ฟังก์ชัน `toupper(cha)` ฟังก์ชันที่ใช้เปลี่ยนตัวพิมพ์เล็กที่เก็บอยู่ในตัวแปรให้เป็นตัวพิมพ์ใหญ่

ตัวอย่างที่ 7.4

1	<code>#include &lt;stdio.h&gt;</code>
2	<code>#include &lt;ctype.h&gt;</code>
3	<code>main()</code>
4	<code>{</code>
5	<code>    char cha1 = 'D' , cha2 = 'a' , cha3 = 'f' , cha4 = 'N' ;</code>
6	<code>    printf("\ncheck cha1 = 'D' is uppercase yes or no : %d ", isupper(cha1));</code>
7	<code>    printf("\ncheck cha2 = 'a' is lower yes or no : %d ", islower(cha2));</code>
8	<code>    printf("\ncheck cha2 = 'a' is upper yes or no : %d ", isupper(cha2));</code>
9	<code>    printf("\ncheck 'i' is lower yes or no : %d ", islower('i'));</code>
10	<code>    printf("\ncheck 'L' is uppercase yes or no : %d ", isupper('L'));</code>
11	<code>    printf("\nchange cha3 = 'f' to uppercase %c : ", toupper(cha3));</code>
12	<code>    printf("\nchange cha4 = 'N' to lowercase %c : ", tolower(cha4));</code>
13	<code>}</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
<pre>check cha1 = 'D' is uppercase yes or no : 1 check cha2 = 'a' is lower yes or no : 2 check cha2 = 'a' is upper yes or no : 0 check 'i' is lower yes or no : 2 check 'L' is uppercase yes or no : 1 change cha3 = 'f' to uppercase F : change cha4 = 'N' to lowercase n :</pre>	

8. ฟังก์ชัน `isspace(cha)` ฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลในตัวแปร `cha` เป็น whitespace หรือไม่ whitespace ได้แก่ space , tab , vertical tab , formfeed , carriage return , newline ถ้ามี whitespace จริง ฟังก์ชันจะส่งค่าไม่เท่ากับ 0 ถ้าไม่จริงจะส่งค่า 0

9. ฟังก์ชัน `isxdigit(cha)` ฟังก์ชันที่ใช้ตรวจสอบว่าข้อมูลในตัวแปร `cha` เป็น เลขฐานสิบหก ( คือ 0-9 , A-F , a - f) หรือไม่ ถ้าจริงส่งค่าตัวเลขที่ไม่ใช่ 0 ถ้าไม่จริงส่งตัวเลข 0

ตัวอย่างที่ 7.5

1	<code>#include &lt;stdio.h&gt;</code>
2	<code>#include &lt;ctype.h&gt;</code>
3	<code>main()</code>
4	<code>{</code>
5	<code>    char cha1 = '\r', cha2 = '\n', cha3 = '\v' , cha4 = 'A';</code>
6	<code>    printf("\n%d is value return from isspace %c ", isspace(cha1), cha1);</code>
7	<code>    printf("\n%d is value return from isspace %c ", isspace(cha2), cha2);</code>
8	<code>    printf("\n%d is value return from isspace %c ", isspace(cha3), cha3);</code>
9	<code>    printf("\n%d is value return from isxdigit %c ", isxdigit(cha4), cha4);</code>
10	<code>    printf("\n%d is value return from isxdigit %c ", isxdigit('0'), '0');</code>
11	<code>    printf("\n%d is value return from isxdigit %c ", isxdigit('g'), 'g');</code>
12	<code>}</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
<pre>is value return from isspace 8 is value return from isspace 8 is value return from isspace 128 is value return from isxdigit A 128 is value return from isxdigit 0 0 is value return from isxdigit g</pre>	

10. ฟังก์ชัน `gotoxy(x,y)`; เป็นฟังก์ชันอยู่ใน `conio.h` ใช้สั่งให้เคอร์เซอร์เคลื่อนที่ไปตามตำแหน่งที่ระบุ โดย `x` คือ ตำแหน่งของสดมภ์บนจอภาพ (คล้ายค่า `x` ของกราฟ)ค่าเพิ่มจากซ้ายไปขวามีค่าตั้งแต่ 1 ถึง 79 ตำแหน่งที่ 80 สงวนไว้ไม่ให้ใช้ ส่วน `y` คือตำแหน่งแถวบนจอภาพนับจากบนลงล่าง มีค่าได้ตั้งแต่ 1 ถึง 24 ตำแหน่งที่25 สงวนไว้

11.ฟังก์ชัน `clreol()`; เป็นฟังก์ชันอยู่ใน `conio.h` ใช้ลบข้อความตั้งแต่ตำแหน่งที่เคอร์เซอร์อยู่ไปจนจบบรรทัด

12.ฟังก์ชัน `delline()`; เป็นฟังก์ชันอยู่ใน `conio.h` ใช้ลบข้อความทั้งบรรทัดที่เคอร์เซอร์อยู่ไปจนจบบรรทัดและเลื่อนข้อความในบรรทัดล่างขึ้นมาแทน

13.ฟังก์ชัน `insline()`; เป็นฟังก์ชันอยู่ใน `conio.h` ใช้แทรกบรรทัดว่าง 1 บรรทัดใต้บรรทัดที่เคอร์เซอร์อยู่

14.ฟังก์ชัน `system("dos command")`; เป็นฟังก์ชันอยู่ใน `stdlib.h` ใช้เรียกคำสั่งของ dos ขึ้นมาทำงาน เช่นคำสั่ง `cls dir date time`

15.ฟังก์ชัน `abort()`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้ ยกเลิกการทำงานของโปรแกรมทันทีไม่ว่าจะทำงานสำเร็จหรือไม่ และมีข้อความ Abnomal program termination แสดงทางจอภาพ

16.ฟังก์ชัน `abs(x)`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้หาค่าสัมบูรณ์ของ `x` โดย `x` ต้องเป็นจำนวนเต็ม

17.ฟังก์ชัน `labs(x)`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้หาค่าสัมบูรณ์ของ `x` โดย `x` ต้องเป็น `long int`

18.ฟังก์ชัน `atoi(s)`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้เปลี่ยนข้อความให้เป็นเลขจำนวนเต็ม

19.ฟังก์ชัน `atol(s)`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้เปลี่ยนข้อความให้เป็น `long integer`

20.ฟังก์ชัน `atof(s)`; ฟังก์ชันที่อยู่ใน `<stdlib.h>` ใช้เปลี่ยนข้อความให้เป็น `floating point`

#### ตัวอย่างที่ 7.6

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      char numstring1[10],numstring2[10],numstring3[10];
6      int in1;
7      float flo1;
8      long lon1;
9      printf("\nEnter number as string1 : ");
10     scanf("%s",numstring1);
11     printf("\nEnter number as string2 : ");
12     scanf("%s",numstring2);
13     printf("\nEnter number as string3 : ");
14     scanf("%s",numstring3);
15     in1 = atoi(numstring1); flo1 = atof(numstring2); lon1=atol(numstring3);
16     printf("\nnumstring1 =%s change to integer %d ",numstring1,in1);
17     printf("\nnumstring2 =%s change to floating point %4.4f ",numstring2,flo1);
18     printf("\nnumstring3 =%s change to long integer %d ",numstring3,lon1);
19     printf("\nsummation of in1,flo1,lon1 is %6.4f ",in1+flo1+lon1);
20     printf("\nsummation of atoi(numstring1),atof(numstring2),atol(numstring2) is
21     %6.4lf:",atoi(numstring1)+atof(numstring2)+atol(numstring3));
22 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

Enter number as string1 : 2559

Enter number as string2 : 2016

Enter number as string3 : 2555

```
numstring1 =2559 change to integer 2559
numstring2 =2016 change to floating point 2016.0000
numstring3 =2555 change to long integer 2555
summation of in1,flo1,lon1 is 7130.0000
summation of atoi(numstring1),atof(numstring2),atol(numstring3) is 7130.0000:
```

## 7.3 ฟังก์ชันสร้างเอง (User-Defined function)

การสร้างฟังก์ชันด้วยตนเองมีขั้นตอน 2 ขั้นตอนคือ

1. ต้นแบบฟังก์ชัน (function prototype)
2. นิยามฟังก์ชัน (function definition)

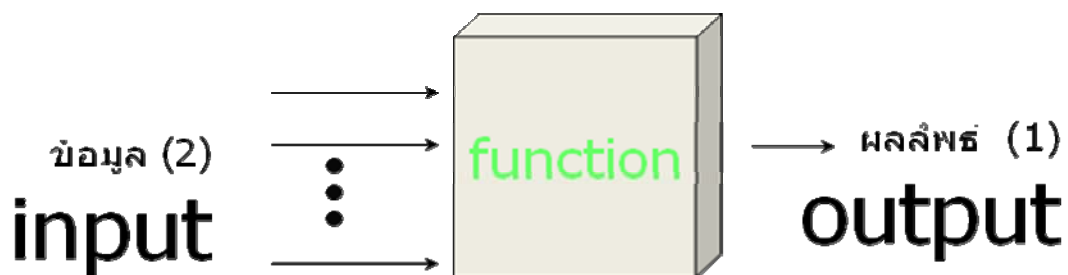
ต้นแบบฟังก์ชันจะมีหรือไม่ ขึ้นกับตำแหน่งที่เรานิยามฟังก์ชันในโปรแกรม

- ถ้าฟังก์ชันนิยามก่อน main (in-line function) ไม่จำเป็นต้องมีต้นแบบฟังก์ชัน
- ถ้าฟังก์ชันนิยามหลัง main จำเป็นต้องมีการประกาศต้นแบบฟังก์ชันก่อนฟังก์ชัน main

### 7.3.1 Function prototype

การประกาศโปรโตไทป์เป็นสิ่งจำเป็นในภาษาซีเนื่องจากภาษาซีเป็นภาษาในลักษณะที่ต้องมีการประกาศฟังก์ชันก่อนจะเรียกใช้ฟังก์ชันนั้น (Pre-defined Function) ในโปรโตไทป์จะไม่มีการประกาศชื่อตัวแปร มีแต่การเขียนประเภทของตัวแปรไว้ภายในเป็นการช่วยให้คอมไพเลอร์ สามารถตรวจสอบจำนวนของตัวแปร ประเภทของตัวแปร ประเภทของการคืนค่า ภายในโปรแกรมว่ามีการเรียกใช้งานสิ่งต่างๆเกี่ยวกับฟังก์ชัน นั้นถูกต้องหรือไม่ นอกจากนี้เราอาจจะแยกส่วน โปรโตไทป์ไปเขียนไว้ใน include file ก็ได้เช่นเดียวกัน ต้นแบบของฟังก์ชันเป็นตัวบอกให้คอมไพเลอร์รู้ถึง:

1. ชนิดข้อมูลที่จะส่งค่ากลับ (1)
2. จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (2)
3. ชนิดของพารามิเตอร์แต่ละตัว รวมทั้งลำดับของพารามิเตอร์เหล่านั้น



รูปที่ 7.2 แสดงลักษณะการทำงานของ function

รูปแบบของการประกาศโปรโตไทป์ของฟังก์ชัน

```
return-type function-name (parameter-list) ;
```

โดยที่



- return-type หรือชนิดข้อมูลที่จะส่งค่ากลับได้แก่ void, int, double, char, ...
  - **function-name** ชื่อของฟังก์ชัน
  - parameter-list จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ แต่ละพารามิเตอร์ประกอบด้วย ชนิดตัวแปรและชื่อตัวแปรแต่ละพารามิเตอร์แยกด้วยเครื่องหมาย “ , ”
- ตัวอย่างการประกาศโปรโตไทป์ของฟังก์ชัน

```
int add (int a, int b) ;
```

### 7.3.2 นิยามฟังก์ชัน (function definition)

นิยามฟังก์ชันเป็นการเขียนรายละเอียดการทำงานของฟังก์ชันนั้นๆ ประกอบด้วยส่วนของ header และ algorithm โดย header จะมีการเขียนเหมือนต้นแบบฟังก์ชัน แต่ไม่มี ;

algorithm เขียนอยู่ภายใน { }

#### รูปแบบของนิยามฟังก์ชัน

ชนิดข้อมูลที่คืนค่า ชื่อฟังก์ชัน ( การประกาศตัวแปร )

```
{
    การประกาศตัวแปรภายในฟังก์ชัน;
    คำสั่ง;
    return (ค่าข้อมูลที่ต้องการส่งค่ากลับ);
}
```

### 7.3.3 การเรียกใช้ฟังก์ชัน การเรียกใช้ฟังก์ชันที่มีการคืนค่า จะใช้รูปแบบดังต่อไปนี้

ค่าที่รับ = ฟังก์ชัน (อาร์กิวเมนต์)

ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับโดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

#### ตัวอย่างฟังก์ชันที่มี return

```
int add (int a, int b)
{
    int result;
    result = a+b;
    return result;
}
```

#### ตัวอย่างฟังก์ชันที่ไม่มี return

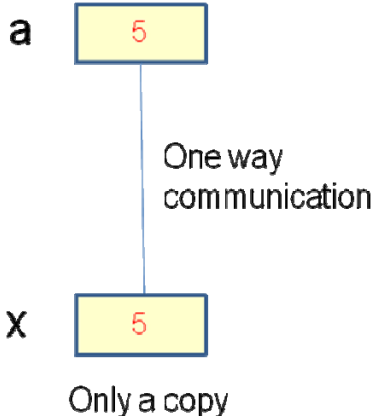
```
void checkresult (int result)
{
    if(result < 0)
        printf("result is negative");
    if(result == 0)
        printf("result is zero");
    if(result > 0)
        printf("result is positive")
}
```

### 7.3.4 การส่งค่าพารามิเตอร์ (Parameter Passing)

## 1. การส่งโดยค่า Pass by Value

เป็นการส่งค่าพารามิเตอร์จากโปรแกรมที่เรียกไปยังโปรแกรมย่อยเพียงด้านเดียว ไม่มีการส่งค่ากลับ ค่าที่ส่งไปยังจะถูกคัดลอกไปไว้ที่หน่วยความจำใหม่ โปรแกรมย่อยใช้ทำงาน จึงไม่มีผลต่อตัวพารามิเตอร์ที่เรียกใช้

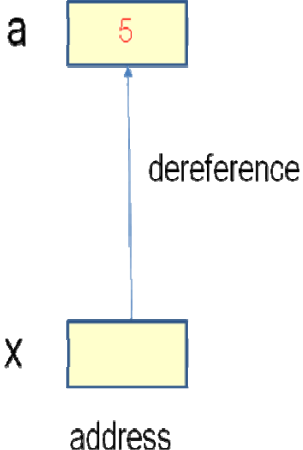
ตัวอย่าง

<pre>1  #include &lt;stdio.h&gt; 2  void func(int x); 3  main() 4  { 5      int a=5; 6      func(a); 7      printf("%d\n",a); 8      return 0; 9  } 10 void func(int x) 11 { 12     x=x+3; 13     return; 14 }</pre>	
ผลลัพธ์จากการประมวลผลโปรแกรม 5	

## 2. การส่งโดยที่อยู่ของตัวแปร Pass by Reference

เป็นการผ่านค่าที่อยู่ของตัวแปรในโปรแกรมหลักไปยังโปรแกรมย่อยเมื่อโปรแกรมย่อยทำงานจึงทำให้ค่าในโปรแกรมหลักเปลี่ยนแปลง

ตัวอย่าง

<pre>1  #include &lt;stdio.h&gt; 2  void func(int *x); 3  main() 4  { 5      int a=5; 6      func(&amp;a); 7      printf("%d\n",a); 8      return 0; 9  } 10 void func(int *x) 11 { 12     *x=*x+3; 13     return; 14 }</pre>	
ผลลัพธ์จากการประมวลผลโปรแกรม 8	

### 7.3.5 ฟังก์ชันแบ่งตามการรับส่งข้อมูล

เราสามารถแบ่งประเภทการทำงานของฟังก์ชันออกเป็นประเภทต่าง ๆ ดังนี้

- |  |                                 |
|--|---------------------------------|
| แบบที่ 1 - ฟังก์ชันที่ไม่รับผ่านค่า และไม่ส่งผ่านค่ากลับ   | <code>void func1();</code>      |
| แบบที่ 2 - ฟังก์ชันที่มีการรับผ่านค่า แต่ไม่ส่งผ่านค่ากลับ | <code>void func2(int a);</code> |
| แบบที่ 3 - ฟังก์ชันที่มีการรับผ่านค่า และส่งผ่านค่ากลับ    | <code>int func3(int a);</code>  |
| แบบที่ 4 - ฟังก์ชันที่ไม่รับผ่านค่า แต่มีการส่งผ่านค่ากลับ | <code>int func4();</code>       |

ตัวอย่างที่ 7.7 ตัวอย่างโปรแกรมที่พิมพ์ตัวเลขตัวแรกทีอ่านจากคีย์บอร์ดโดยใช้ฟังก์ชัน

1	<code>#include &lt;stdio.h&gt;</code>
2	<code>/* This program prints the first digits of an integer</code>
3	<code>read from the keyboard</code>
4	<code>*/</code>
5	<code>#include &lt;stdio.h&gt;</code>
6	<code>// Function Declarations</code>
7	<code>int firstDigit (int num);</code>
8	<code>int main (void)</code>
9	<code>{</code>
10	<code>// Local Declarations</code>
11	<code>int number;</code>
12	<code>int digit;</code>
13	<code>// Statements</code>
14	<code>printf("Enter an integer: ");</code>
15	<code>scanf ("%d", &amp;number);</code>
16	<code>digit = firstDigit (number);</code>
17	<code>printf("\nLeast significant digit is: %d\n", digit);</code>
18	<code>return 0;</code>
19	<code>} // main</code>
20	<code>int firstDigit (int num)</code>
21	<code>{</code>
22	<code>// Statements</code>
23	<code>return (num % 10);</code>
24	<code>} // firstDigit</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
Enter an integer: 89	
Least significant digit is: 9	

## 7.4 ฟังก์ชันแบบเรียกตัวเอง (recursive function)

ฟังก์ชันแบบเรียกตัวเองคือ ฟังก์ชันที่มีการเรียกตัวเองโดยให้พารามิเตอร์ที่แตกต่างกันออกไปเช่น การหา Factorial หรือการหา Fibonacci

ตัวอย่างที่ 7.8 ตัวอย่างโปรแกรมแบบเรียกตัวเอง

1	<code>#include&lt;stdio.h&gt;</code>
2	<code>int factorial(int x);</code>
3	<code>int main()</code>
4	<code>{</code>
5	<code>int y = factorial(3);</code>
6	<code>printf("3! = %d", y);</code>
7	<code>return 0;</code>
8	<code>}</code>
9	<code>int factorial(int x)</code>
10	<code>{</code>
11	<code>if(x &lt;= 1)</code>
12	<code>return 1;</code>
13	<code>else</code>
14	<code>return x* factorial(x-1);</code>
15	<code>}</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
3! = 6	

## 7.5 ขอบเขตของโปรแกรม

```
#define MAX 950
void one(int anarg,
        double second)
{
    int onelocal;
    ...
}
#define LIMIT 200
int fun_two(int one, char anarg)
{
    int localvar;
    ...
}

int main(void)
{
    int localvar;
    ...
}
```

ตัวแปร	รู้จักใน one	รู้จักใน fun_two	รู้จักใน main
MAX	✓	✓	✓
anarg(int)	✓	✗	✗
second	✓	✗	✗
onelocal	✓	✗	✗
LIMIT	✗	✓	✓
one (parameter)	✗	✓	✓
anarg(char)	✗	✓	✗
localvar(fun_two)	✗	✓	✗
localvar(main)	✗	✗	✓

ฟังก์ชัน	รู้จักใน one	รู้จักใน fun_two	รู้จักใน main
one (function)	✓	✗	✓
fun_two	✗	✓	✓

## 7.6 ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่างที่ 7.9 ตัวอย่างโปรแกรมพิมพ์ปฏิทินโดยใช้ฟังก์ชัน

```
1  #include <stdio.h>
2  void printMonth (int startDay, int days);
3  int main (void)
4  {
5      printMonth (2, 29);
6      return 0;
7  }
8  void printMonth (int startDay, int days)
9  {
10     int weekDay;
11     int dayCount;
12     printf("Sun Mon Tue Wed Thu Fri Sat\n");
13     printf("---- - - - - - - - - - -\n");
14     for (weekDay = 0; weekDay < startDay; weekDay++)
15         printf("    ");
16     for (dayCount = 1; dayCount <= days; dayCount++)
17     {
18         if (weekDay > 6)
19         {
20             printf("\n");
21             weekDay = 1;
22         }
23         else
24             weekDay++;
25         printf("%3d ", dayCount);
26     }
27     printf("\n---- - - - - - - - - - -\n");
28     return;
29 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Sun Mon Tue Wed Thu Fri Sat
---- - - - - - - - - - -
      1  2  3  4  5
6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29
---- - - - - - - - - - -
```

## แบบฝึกหัดปฏิบัติการคาบที่ 7: Function

### [การเรียกใช้ฟังก์ชันมาตรฐาน]

1. ให้แสดงค่าของ x หลังจากการใช้คำสั่งต่อไปนี้

a) x = fabs( 7.5 );

\_\_\_\_\_

b) x = floor( 7.5 );

\_\_\_\_\_

c) x = fabs( 0.0 );

\_\_\_\_\_

d) x = ceil( 0.0 );

\_\_\_\_\_

e) x = fabs( -6.4 );

\_\_\_\_\_

f) x = ceil( -6.4 );

\_\_\_\_\_

g) x = ceil( -fabs( -8 + floor( -5.5 ) ) );

\_\_\_\_\_

2. จงเขียนโปรแกรมเพื่อสุ่มเลขที่อยู่ระหว่าง 0-1000 ด้วยฟังก์ชัน rand() เก็บใส่ในอาร์เรย์จำนวน N ค่าเรียงข้อมูลที่ได้และนับจำนวนความถี่ของเลขแต่ละค่า โดยใช้ฟังก์ชัน พร้อมแสดงผลลัพธ์

3. จากโปรแกรมต่อไปนี้ให้แสดงผลลัพธ์ของโปรแกรมจากการใช้ฟังก์ชันมาตรฐานต่าง ๆ

```
1 #include <stdio.h>
2 #include <math.h>
3 int main( void )
4 {
5     printf( "sqrt(%.1f) = %.1f\n", 900.0, sqrt( 900.0 ) );
6     printf( "sqrt(%.1f) = %.1f\n", 9.0, sqrt( 9.0 ) );
7     printf( "exp(%.1f) = %f\n", 1.0, exp( 1.0 ) );
8     printf( "exp(%.1f) = %f\n", 2.0, exp( 2.0 ) );
9     printf( "log(%.1f) = %.1f\n", 2.718282, log( 2.718282 ) );
10    printf( "log(%.1f) = %.1f\n", 7.389056, log( 7.389056 ) );
11    printf( "log10(%.1f) = %.1f\n", 1.0, log10( 1.0 ) );
12    printf( "log10(%.1f) = %.1f\n", 10.0, log10( 10.0 ) );
13    printf( "log10(%.1f) = %.1f\n", 100.0, log10( 100.0 ) );
14    printf( "fabs(%.1f) = %.1f\n", 13.5, fabs( 13.5 ) );
15    printf( "fabs(%.1f) = %.1f\n", 0.0, fabs( 0.0 ) );
16    printf( "fabs(%.1f) = %.1f\n", -13.5, fabs( -13.5 ) );
17    printf( "ceil(%.1f) = %.1f\n", 9.2, ceil( 9.2 ) );
18    printf( "ceil(%.1f) = %.1f\n", -9.8, ceil( -9.8 ) );
19    printf( "floor(%.1f) = %.1f\n", 9.2, floor( 9.2 ) );
20    printf( "floor(%.1f) = %.1f\n", -9.8, floor( -9.8 ) );
21    printf( "pow(%.1f, %.1f) = %.1f\n", 2.0, 7.0, pow( 2.0, 7.0 ) );
22    printf( "pow(%.1f, %.1f) = %.1f\n", 9.0, 0.5, pow( 9.0, 0.5 ) );
23    printf( "fmod(%.3f/%.3f) = %.3f\n", 13.675, 2.333,
24    fmod( 13.675, 2.333 ) );
25    printf( "sin(%.1f) = %.1f\n", 0.0, sin( 0.0 ) );
26    printf( "cos(%.1f) = %.1f\n", 0.0, cos( 0.0 ) );
27    printf( "tan(%.1f) = %.1f\n", 0.0, tan( 0.0 ) );
28    return 0; /* indicates successful termination */
29 }
```

### [ฟังก์ชันกำหนดเอง]

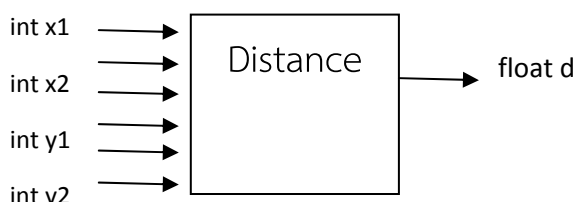
1. จงเขียนโปรแกรมหาค่า  $f(x)$  โดยสมการ  $f(x)$  เป็นดังนี้

$$\begin{aligned} f(x) &= x^2 + 2x + 3 & \text{if } x < 0 \\ &= 0 & \text{if } x = 0 \\ &= x - 2 & \text{if } x > 0 \end{aligned}$$

กำหนดให้ส่วนที่ใช้ในการคำนวณค่า  $f(x)$  อยู่ในฟังก์ชัน `get_Fx` กำหนดให้ส่วนที่รับค่าตัวแปร  $X$  จากคีย์บอร์ด และส่วนที่แสดงผลลัพธ์ของค่า  $f(x)$  อยู่ในฟังก์ชัน `main` ห้ามใช้ตัวแปร `Global` ในโปรแกรมเด็ดขาด

2. จงเขียนโปรแกรมเพื่อคำนวณระยะห่างระหว่างจุดสองจุด คือ  $(x_1, y_1, z_1)$  และ  $(x_2, y_2, z_2)$  โดยการใช้ฟังก์ชัน `Distance` โดยฟังก์ชันจะรับ input เป็นเลขจำนวนเต็ม 6 ตัว สำหรับค่า  $x_1, y_1, z_1, x_2, y_2, z_2$  จากนั้นจะคำนวณส่งกลับค่าตัวเลขทศนิยมสองหลักเป็นค่าระยะห่าง (distance)

$$d = \sqrt{|z_2 - z_1|^2 + |y_2 - y_1|^2 + |x_2 - x_1|^2}$$



3. จงเขียนโปรแกรมเพื่อคำนวณพื้นที่สี่เหลี่ยมและวงกลม ดังนี้

ในส่วนของโปรแกรมหลัก ให้แสดงเมนูให้ผู้ใช้เลือกกว่าต้องการคำนวณพื้นที่ของสี่เหลี่ยมหรือวงกลม

โดยถ้าผู้ใช้เลือกเมนู 1 ให้รับค่าความกว้างและความยาว (กำหนดให้ความกว้างและความยาวเป็นจำนวนเต็ม) แล้วเรียกใช้ฟังก์ชันในการคำนวณสี่เหลี่ยม

แต่ถ้าผู้ใช้เลือกเมนูข้อ 2 ให้รับค่ารัศมี (เป็นทศนิยม) แล้วเรียกใช้ฟังก์ชันในการคำนวณพื้นที่วงกลม

เมื่อเรียกใช้งานฟังก์ชันดังกล่าวเพื่อคำนวณค่าพื้นที่แล้ว ให้ฟังก์ชันนั้นๆ ส่งค่าผลลัพธ์กลับมายัง

โปรแกรมหลัก แล้วแสดงผลออกทางหน้าจอ

ตัวอย่างการทำงานของโปรแกรม

<pre> ===== MENU ===== 1. Calculate area of rectangle 2. Calculate area of circle Please enter 1 or 2: 1 Please enter width: 5 Please enter height: 10 Area = 50.00           </pre>	<pre> ===== MENU ===== 1. Calculate area of rectangle 2. Calculate area of circle Please enter 1 or 2: 2 Please enter radius: 10.5 Area = 346.36           </pre>
--	---

4. จงเติมค่าตัวแปรลงในช่องว่างที่กำหนดให้ (อธิบายการทำงานของโปรแกรมในชั่วโมง)

1	<code>#include &lt;stdio.h&gt;</code>	
2	<code>/* function prototype */</code>	
3	<code>int f1( int );</code>	
4	<code>float f2( int *, int );</code>	
5	<code>/* global variable */</code>	
6	<code>int a = 10;</code>	
7	<code>int b = 5;</code>	
8	<code>int f1( int a )</code>	
9	<code>{</code>	
10	<code>return a - 1;</code>	
11	<code>}</code>	
12	<code>float f2( int *k, int y )</code>	
13	<code>{</code>	
14	<code>*k = f1(a);</code>	
15	<code>a *= 5;</code>	
16	<code>y = f1(a);</code>	
17	<code>return *k + y + 0.5;</code>	
18	<code>}</code>	
19	<code>int main()</code>	
20	<code>{</code>	
21	<code>int x, y;</code>	
22	<code>x = f1(a + b);</code>	
23	<code>y = f1(b);</code>	
24	<code>printf("%d\n", x);</code>	
25	<code>printf("%.2f\n", f2(&amp;x, y));</code>	
26	<code>printf("%d\n", a);</code>	
27	<code>printf("%d\n", x);</code>	
28	<code>printf("%d\n", y);</code>	
29	<code>getchar();</code>	
30	<code>return 0;</code>	
31	<code>}</code>	
32		

5. จงเขียนฟังก์ชันเพื่อหาค่าต่ำสุด (minArray(int A[])) ค่าสูงสุด(maxArray(int A[]))

ผลรวมของตัวเลข(sumArray(int A[])) ที่อยู่ในอาร์เรย์

6. กำหนด function prototype สำหรับวาดเส้น และวาดจุดดังนี้

`void drawline ( char c, int begin, int end );`

ฟังก์ชัน drawline จะแสดงผลอักขระ c จากตำแหน่ง begin จนถึงตำแหน่ง end

เช่น drawline('+', 5, 10); จะแสดงผล

++++++

`void drawpoints ( char c, int *list, int n );`

ฟังก์ชัน drawpoints จะแสดงผลอักขระ c ในตำแหน่งที่ระบุใน list (เรียงลำดับจากซ้ายไปขวา) โดยที่ n คือจำนวนจุดที่ต้องการแสดงผลในบรรทัดนั้น

เช่น `int points[ ] = {1, 3, 5};`

`drawpoint('$', points, 3);` จะแสดงผล

\$ \$ \$

จงเขียนโปรแกรมเพื่อวาดรูปสามเหลี่ยมและสี่เหลี่ยม (โดยเรียกใช้ Function ดังกล่าว) เพื่อให้แสดงผลดังนี้

### ตัวอย่างผลการรันโปรแกรม

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

#####

#      #
#      #
#####
```

กำหนดโปรแกรมหลักในการทดสอบ function drawline และ drawpoints ดังนี้

```
int main( )
{
    int list[10];
    char c = '*';

    /* draw triangle */
    list[0] = 5;
    drawpoints(c, list, 1);
    list[0] = 3; list[1] = 7;
    drawpoints(c, list, 2);
    list[0] = 1; list[1] = 9;
    drawpoints(c, list, 2);
    drawline(c, 1, 10);

    /* draw rectangle */
    printf("\n");
}
```



## บทที่ 8

### ตัวชี้ (Pointer)

#### จุดประสงค์

1. เพื่อให้ทราบความหมายและประโยชน์ของตัวแปรพอยน์เตอร์
2. สามารถเขียนคำสั่งประกาศและกำหนดค่าให้กับตัวแปรพอยน์เตอร์
3. สามารถใช้งานตัวแปรพอยน์เตอร์ได้อย่างมีประสิทธิภาพ

พอยน์เตอร์ หรือ ตัวชี้เป็นชนิดข้อมูลชนิดหนึ่งของภาษา C (int, float, char) ตัวแปรชนิดตัวชี้มีความเร็วในการทำงานสูง ตัวแปรชนิดตัวชี้ช่วยประหยัดเนื้อที่ในหน่วยความจำหลักขณะประมวลผล เมื่อเทียบกับ Array ใช้ตัวชี้ร่วมกับฟังก์ชันเพื่อเพิ่มประสิทธิภาพการเขียนโปรแกรม

#### 8.1 พอยน์เตอร์กับแอดเดรส (Pointers and Addresses)

ตัวแปร คือชื่อที่ใช้แทนข้อมูล การประกาศตัวแปรเป็นการกำหนดชื่อเพื่อใช้แทนข้อมูล เมื่อประกาศตัวแปร จะมีการจองเนื้อที่ในหน่วยความจำเพื่อเก็บข้อมูล เราสามารถเข้าถึงข้อมูลได้โดยอ้างถึงตัวแปร การประกาศตัวแปร เช่น

`int i;` เป็นการประกาศ (Declaration) ตัวแปรชื่อ `i` เป็นตัวแปรชนิด `int` (integer)

การใช้ตัวชี้หรือพอยน์เตอร์เป็นอีกวิธีที่จะเข้าถึงตัวแปรปกติได้ ตัวแปรชนิดพอยน์เตอร์จะเก็บค่าที่อยู่ของหน่วยความจำหลัก ซึ่งต่างกับตัวแปรปกติที่เก็บค่าที่แท้จริงของข้อมูล การใช้ตัวแปรชนิดพอยน์เตอร์จะเป็นการเข้าถึงข้อมูลหรือเป็นการอ้างถึงตำแหน่งที่เก็บข้อมูล

#### ตัวอย่างที่ 8.1 พอยน์เตอร์กับแอดเดรส (Pointers and Addresses)

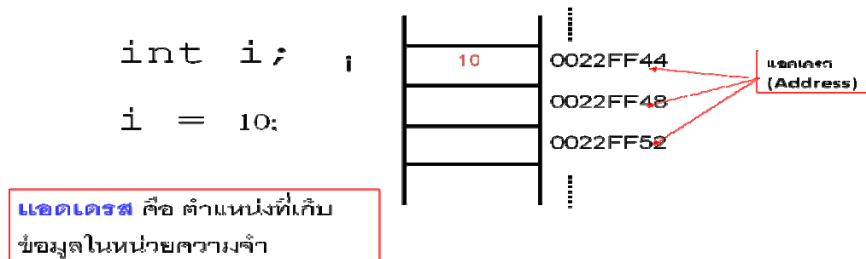
1	<code>#include &lt;stdio.h&gt;</code>
2	<code>#include &lt;stdlib.h&gt;</code>
3	<code>int main()</code>
4	<code>{</code>
5	<code>    float x = 1.0;</code>
6	<code>    printf("Value of x is %.2f\n", x);</code>
7	<code>    printf("Address of x is %p\n", &amp;x);</code>
8	<code>    return 0;</code>
9	<code>}</code>

ผลลัพธ์จากการประมวลผลโปรแกรม

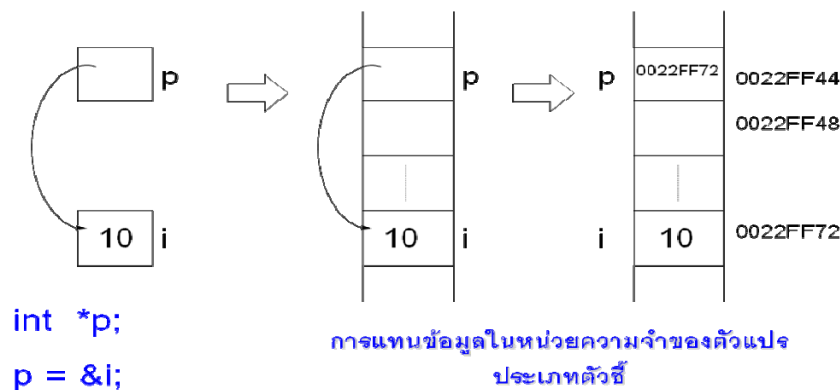
Value of x is 1.00  
Address of x is 0022FF44

ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ

x 1.00  
0022FF44



### 8.1.1 ตัวชี้กับแอดเดรส (Pointers and Address)



## 8.2 การประกาศตัวแปรประเภทตัวชี้

ตัวแปรที่จะทำหน้าที่เป็นตัวแปร pointer จะต้องมีการประกาศไว้ตอนต้นของโปรแกรมโดยใช้รูปแบบ

type \*variable-name

โดย

type หมายถึง ชนิดของตัวแปร

\* เป็นเครื่องหมายที่แสดงว่าตัวแปรที่ตามหลังเครื่องหมายนี้เป็นตัวแปรชนิด pointer

variable-name เป็นชื่อตัวแปรที่ต้องการประกาศว่าเป็นชนิด pointer โดยมีกฎการตั้งชื่อเหมือนกับการตั้งชื่อตัวแปรธรรมดา

การประกาศตัวแปรประเภทพอยน์เตอร์จะใช้ Unary Operator \* ซึ่งมีชื่อเรียกว่า Indirection หรือ Dereferencing Operator โดยจะต้องประกาศประเภทของตัวแปรพอยน์เตอร์ให้สอดคล้องกับประเภทของตัวแปรที่เราต้องการ (ยกเว้นตัวแปรพอยน์เตอร์ประเภท void ที่สามารถชี้ไปยังตัวแปรประเภทใดก็ได้) การประกาศตัวแปรประเภทตัวชี้ `int *ip;` เป็นการประกาศตัวแปร ip ให้เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท int

```
double *dp, atof(char *);
```

เป็นการประกาศตัวแปร dp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท double

ประกาศฟังก์ชัน atof มีพารามิเตอร์เป็นตัวแปรพอยน์เตอร์ประเภท char การกำหนดค่าให้กับตัวแปรพอยน์เตอร์จะเป็นการกำหนดแอดเดรสของตัวแปรที่มีประเภทสอดคล้องกับประเภทของตัวแปรพอยน์เตอร์เท่านั้น

การใช้ Unary Operator & เป็นโอเปอเรเตอร์ที่อ้างถึงแอดเดรสของออบเจกต์ (Object) ใด ๆ การกำหนดค่าและการอ่านค่าตัวแปรตัวชี้

## ตัวอย่างที่ 8.2 การทำงานกับ pointer

1	#include <stdio.h>
2	#include <stdlib.h>
3	int main()
4	{
5	int x = 1, y = 2;
6	int *ip, *iq;
7	ip = &x;
8	printf("ip= %p, *ip=%d, x= %d, &x= %p\n", ip,*ip, x, &x);
9	y = *ip;
10	printf("ip= %p, *ip=%d, y= %d, &y= %p\n", ip,*ip, y, &y);
11	*ip = 0;
12	printf("ip= %p, *ip=%d\n", ip,*ip);
13	y = 5;
14	printf("ip= %p, *ip=%d, y= %d, &y= %p\n", ip,*ip, y, &y);
15	ip = &y;
16	printf("ip= %p, *ip=%d, y= %d, &y= %p\n", ip,*ip, y, &y);
17	*ip = 3;
18	printf("ip= %p, *ip=%d\n", ip,*ip);
19	iq = ip;
20	printf("ip= %p, *ip=%d, iq= %p, *iq=%d\n", ip,*ip, iq,*iq);
21	return 0;
22	}

**ผลลัพธ์จากการประมวลผลโปรแกรม**

```
ip= 0022FF44, *ip=1, x= 1, &x= 0022FF44
ip= 0022FF44, *ip=1, y= 1, &y= 0022FF40
ip= 0022FF44, *ip=0
ip= 0022FF44, *ip=0, y= 5, &y= 0022FF40
ip= 0022FF40, *ip=5, y= 5, &y= 0022FF40
ip= 0022FF40, *ip=3
ip= 0022FF40, *ip=3, iq= 0022FF40, *iq=3
```

ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ

x	0	400
y	3	402
ip	402	500
iq	402	504

## ตัวอย่างที่ 8.3 พอยน์เตอร์กับแอดเดรส (Pointers and Addresses)

1	#include <stdio.h>
2	int main( void )
3	{
4	int a; /* a is an integer */
5	a = 7;
6	int *aPtr; /* aPtr is a pointer to an integer */
7	aPtr = &a; /* aPtr set to address of a */
8	printf( "The address of a is %p\n The value of aPtr is %p",&a, aPtr );
9	printf( "\n\nThe value of a is %d\n The value of *aPtr is %d", a,*aPtr );
10	printf( "\n\nShowing that * and & are complements of each other\n*&aPtr =
11	%p\n*&aPtr = %p\n",&aPtr, *&aPtr );
12	return 0; /* indicates successful termination */
13	} /* end main */

#### ผลลัพธ์จากการประมวลผลโปรแกรม

The address of a is 0022FF44  
The value of aPtr is 0022FF44

The value of a is 7  
The value of \*aPtr is 7

Showing that \* and & are complements of each other  
&\*aPtr = 0022FF44  
\*&aPtr = 0022FF44

### 8.3 ตัวชี้และอาร์กิวเมนต์ของฟังก์ชัน(Pointer and Function Arguments)

เนื่องจากภาษาซีมีการส่งอาร์กิวเมนต์ให้กับฟังก์ชันแบบ By Value และฟังก์ชันสามารถคืนค่า (return) ค่าได้เพียงหนึ่งค่า หากต้องการให้ฟังก์ชันมีการเปลี่ยนแปลงค่าและคืนค่ากลับมายังฟังก์ชันที่เรียกใช้มากกว่าหนึ่งค่าจะต้องนำพอยน์เตอร์เข้ามาช่วย

#### ตัวอย่างที่ 8.4

ต้องการเขียนฟังก์ชันเพื่อสลับค่าของตัวแปร 2 ตัว ผลลัพธ์ที่ต้องการได้จากฟังก์ชันนี้จะมี 2 ค่าของตัวแปรที่ทำการสลับค่า หากอาร์กิวเมนต์เป็นตัวแปรธรรมดาจะไม่สามารถแก้ปัญหานี้ได้ จึงต้องใช้พอยน์เตอร์เข้ามาช่วย โดยการส่งค่าแอดเดรสของตัวแปรทั้ง 2 ให้กับฟังก์ชันที่จะสลับค่าของตัวแปรทั้ง 2 ผ่านทางตัวแปรพอยน์เตอร์ที่เป็นอาร์กิวเมนต์ของฟังก์ชัน โปรแกรมตัวอย่างการสลับค่าตัวแปร 2 ตัวโดยผ่านฟังก์ชัน จะแสดงการส่งอาร์กิวเมนต์ให้เป็นพอยน์เตอร์

```
1  #include<stdio.h>
2  void swap (int *, int *);
3  main ( )
4  {
5      int x = 5, y = 10;
6      printf("Before swap : x = %d, y = %d\n", x, y);
7      swap ( &x, &y);
8      printf("After swap : x = %d, y = %d\n", x, y);
9      getch();
10 }
11 void swap (int *px, int *py)
12 {
13     int temp;
14     temp = *px;
15     *px = *py;
16     *py = temp;
17 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

Before swap : x = 5, y = 10  
After swap : x = 10, y = 5

#### อาร์กิวเมนต์ที่เป็นประเภทพอยน์เตอร์

อาร์กิวเมนต์ที่เป็นประเภทพอยน์เตอร์จะช่วยให้ฟังก์ชันสามารถเปลี่ยนค่าให้กับตัวแปรที่ส่งเข้ามาได้ เนื่องจากอาร์กิวเมนต์นั้นจะเก็บแอดเดรสของตัวแปรที่ส่งเข้ามา เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ผ่าน Dereferencing Operator ( \* ) ค่าของตัวแปรที่ส่งเข้ามาจะถูกเปลี่ยนค่าพร้อมกันในทันที

## 8.4 ตัวชี้กับอาร์เรย์ (Pointer and Arrays)

อาร์เรย์เป็นประเภทข้อมูลที่เก็บชุดของข้อมูลประเภทเดียวกัน หรือ อาร์เรย์เป็นโครงสร้างแบบ homogeneous ที่ประกอบด้วยอีลีเมนต์ (elements) ที่มีชนิด (type) เดียวกัน มักใช้กับการทำงานที่ต้องทำงานกับตัวแปรชนิดเดียวกันหลายตัวที่มีการทำงานเหมือนกัน เช่น คะแนนของนักศึกษาภายในห้อง 20 คน เป็นต้น อาร์เรย์ในภาษาซีจะนำหลักการของพอยน์เตอร์เข้ามาใช้ การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาแทนที่ การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาช่วย ซึ่งจะทำให้มีความเร็วในการทำงานสูงขึ้น



การอ้างถึงสมาชิกในอาร์เรย์สามารถแสดงดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 8.5 อ่านค่าของจำนวนเต็ม 5 จำนวนจากคีย์บอร์ด และแสดงผลในลำดับที่กลับกัน

```
1 #include <stdio.h>
2 #define SIZE 5
3 int main ( )
4 {
5     int k;
6     int table[SIZE];
7     for (k = 0; k < SIZE; k++){
8         printf("Please enter number(%d):",k+1);
9         scanf ("%d", &table[k]);
10    }
11    printf("Data in array are:\n");
12    for (k = SIZE-1; k >= 0; k--){
13        printf ("%d\n", table[k]);
14    }
15    return 0;
16 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter number(1):5
Please enter number(2):6
Please enter number(3):8
Please enter number(4):6
Please enter number(5):4
Data in array are:
4
6
8
6
5
```

การใช้ตัวชี้กับอาร์เรย์

สมมติว่ามีอาร์เรย์ a และพอยน์เตอร์ pa ดังนี้

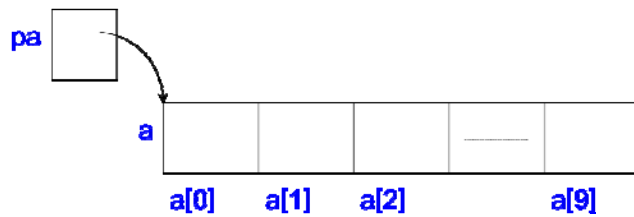
```
int a[10];
```

```
int *pa;
```

กำหนดให้พอยน์เตอร์ pa ชี้ไปยังอาร์เรย์ a ด้วยคำสั่ง

```
pa = &a[0]; /* หรือใช้คำสั่ง pa = a; */
```

pa จะเก็บค่าแอดเดรสเริ่มต้นของอาร์เรย์ a



การนำไปใช้งานจะสามารถอ่านค่าอาร์เรย์ผ่านพอยน์เตอร์ได้ดังนี้

```
x = *pa;
```

จะเป็นการกำหนดค่าให้ x มีค่าเท่ากับ a[0] การเลื่อนไปอ่านค่าสมาชิกตำแหน่งต่าง ๆ ของอาร์เรย์ผ่านทางพอยน์เตอร์สามารถทำได้โดยการเพิ่มค่าพอยน์เตอร์ขึ้น 1 เพื่อเลื่อนไปยังตำแหน่งถัดไป หรือเพิ่มค่าขึ้น N เพื่อเลื่อนไป N ตำแหน่ง หรืออาจจะลดค่าเพื่อเลื่อนตำแหน่งลง

กรณีที่ pa ชี้อยู่ที่ a[0] คำสั่ง pa+1; จะเป็นการอ้างอิงแอดเดรสของ a[1]

หากเป็น pa+i เป็นการอ้างอิงแอดเดรส a[i] หากต้องการอ้างอิงข้อมูลภายในของสมาชิกของอาร์เรย์ตำแหน่งที่ a[i] จะใช้ \*(pa+i) การสั่งให้บวก 1 หรือบวก i หรือ ลบ i เป็นเหมือนการเลื่อนไปยังสมาชิกของอาร์เรย์ตำแหน่งที่ต้องการ

เนื่องจากประเภทของข้อมูลแต่ละประเภทของอาร์เรย์ เช่น int, float, double และอื่น ๆ มีขนาดของข้อมูลที่แตกต่างกัน ทำให้ขนาดของสมาชิกภายในอาร์เรย์แต่ละประเภทมีขนาดแตกต่างกันด้วย การสั่งให้บวกหรือลบด้วยจำนวนที่ต้องการนั้นจะมีกลไกที่ทำหน้าที่คำนวณตำแหน่งที่ต้องการให้สอดคล้องกับข้อมูลแต่ละประเภทโดยอัตโนมัตินอกจากนี้ยังสามารถใช้พอยน์เตอร์แทนอาร์เรย์

การอ้างโดยใช้ a[i] สามารถใช้ \*(a+i) เนื่องจากทุกครั้งที่อ้างถึง a[i] ภาษาซีจะทำหน้าที่แปลงเป็น \*(a+i) เพราะฉะนั้นการเขียนในรูปแบบใดก็ได้ให้ผลลัพธ์ในการทำงานเช่นเดียวกัน

การอ้างอิงแอดเดรส เช่น &a[i] จะมีผลเท่ากับการใช้ a+i ในลักษณะเดียวกันการใช้งานพอยน์เตอร์ก็สามารถใช้คำสั่งในลักษณะอาร์เรย์ก็ได้ เช่น การอ้างอิง \*(pa+i) สามารถเขียนด้วย pa[i] ก็ได้ผลเช่นเดียวกัน

สิ่งที่แตกต่างกันของอาร์เรย์และพอยน์เตอร์ คือ พอยน์เตอร์เป็นตัวแปร แต่อาร์เรย์ไม่ใช่ตัวแปร สมมติให้ a เป็นอาร์เรย์ และ pa เป็นพอยน์เตอร์ การอ้างอิง pa = a หรือ pa++ จะสามารถคอมไพล์ได้ แต่จะไม่สามารถใช้คำสั่ง a = pa หรือ a++ ได้

เมื่อมีการส่งชื่อของอาร์เรย์ให้แก่ฟังก์ชัน จะเป็นการส่งตำแหน่งแอดเดรสของสมาชิกตัวแรกของอาร์เรย์ให้แก่ฟังก์ชัน ดังนั้นพารามิเตอร์ในฟังก์ชันนั้นจะเป็นตัวแปรประเภทพอยน์เตอร์ฟังก์ชันที่รับพารามิเตอร์เป็นพอยน์เตอร์ โดยอาร์กิวเมนต์ที่ส่งมาเป็นอาร์เรย์ นอกจากนี้ยังอาจจะประกาศพารามิเตอร์ภายในฟังก์ชัน strlen ได้ใน 2 ลักษณะ คือ char \*s แบบในตัวอย่าง หรืออาจจะใช้ char s[] ก็ได้ โดยทั่วไปจะใช้ในลักษณะแรก เพราะช่วยในรู้ได้ทันทีว่า s เป็นตัวแปรพอยน์เตอร์ และยังสามารถส่งส่วนใดส่วนหนึ่งของอาร์เรย์ให้แก่ฟังก์ชันก็ได้ โดยไม่จำเป็นต้องส่งสมาชิกตัวแรกก็ได้เช่นกัน

```
f(&a[2]) หรือ f(a+2) เป็นการส่งแอดเดรสของสมาชิก a[2] ให้กับฟังก์ชัน f
```

การประกาศฟังก์ชัน f สามารถทำได้โดยการประกาศ

```
f(int arr[]) { ..... } หรือ
```

```
f(int *arr) { .....}
```

### ตัวอย่างการรับส่งค่าของอาร์เรย์ไปยังฟังก์ชัน

```
#include <stdio.h>
int main { }
{
    char str1[10];
    scanf("%s",str1);
    printf("String length is %d\n",strlen(str1));

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

## 8.5 เลขคณิตของ Pointer กับ Array

ให้ p เป็นพอยน์เตอร์ชี้ไปยังอาร์เรย์ใด ๆ คำสั่ง p++ เป็นการเลื่อน p ไปยังสมาชิกถัดไป และคำสั่ง p += i เป็นการเลื่อนพอยน์เตอร์ไป i ตำแหน่งจากตำแหน่งปัจจุบัน

นอกจากนี้ยังสามารถใช้เครื่องหมายความสัมพันธ์(Relational Operator) เช่น ==, !=, <, >= และอื่น ๆ ทำงานร่วมกับพอยน์เตอร์ได้ สมมติให้ p และ q ชี้ไปยังสมาชิกของอาร์เรย์เดียวกันสมมติให้ p และ q ชี้ไปยังสมาชิกของอาร์เรย์เดียวกัน p < q จะเป็นจริงเมื่อ p ชี้ไปที่สมาชิกที่อยู่ก่อนหน้าสมาชิกที่ q ชี้อยู่ การเปรียบเทียบในลักษณะจะใช้ได้ต่อเมื่อ p และ q ชี้ไปที่อาร์เรย์เดียวกันเท่านั้นนอกจากนี้ยังสามารถใช้การลบหรือการบวกกับพอยน์เตอร์ได้เช่นเดียวกัน แต่สิ่งที่ควรระวังคือ การทำเช่นนั้นจะต้องอยู่ในขอบเขตขนาดของอาร์เรย์เท่านั้น

ฟังก์ชัน strlen ( ) ปรับปรุงให้กระชับขึ้น

- เนื่องจาก s ชี้อยู่ที่ตำแหน่งเริ่มต้น โดยมี p ชี้ไปที่ s เช่นเดียวกัน แต่จะมีการเลื่อน p ไปทีละหนึ่งตำแหน่ง จนกว่าค่าที่ตำแหน่งที่ p ชี้อยู่จะเท่ากับ '\0' เมื่อนำ p ค่าสุดท้ายมาลบกับ s ที่ตำแหน่งเริ่มต้นก็จะได้ความยาวของข้อมูลที่ส่งเข้ามา

```
int strlen (char *s) {
    char *p = s;
    while (*p != '\0')
        p++;
    return p-s;
}
```

## 8.6 ตัวชี้ตัวอักษรและฟังก์ชัน (Character Pointer and Function)

การทำงานกับข้อความหรือที่เรียกว่า สตริง (String) หรืออาร์เรย์ของข้อมูลประเภท char อาจจะใช้พอยน์เตอร์ชี้ไปยังข้อมูลประเภท char

การทำงานกับค่าคงที่สตริง (String Constant) สามารถเขียนภายในเครื่องหมาย “ ” เช่น “I am a string”

เมื่อมีการใช้ค่าคงที่สตริงจะมีการพื้นที่ในหน่วยความจำเท่ากับความยาวของค่าคงที่สตริงบวกด้วย 1 เนื่องจากลักษณะการเก็บข้อมูลประเภทข้อความในหน่วยความจำจะมีการปะตัวอักษร null หรือ ‘\0’ ต่อท้ายเสมอเพื่อให้รู้ว่าเป็นจุดสิ้นสุดของข้อมูล

การจองพื้นที่ดังกล่าวจะเหมือนการจองพื้นที่ของข้อมูลประเภทอาร์เรย์ เป็นอาร์เรย์ของ char ค่าคงที่สตริงที่พบเห็นได้เสมอได้แก่ข้อความที่ใช้ในฟังก์ชัน printf ( ) เช่น

```
printf ( “Hello, world\n” );
```

ฟังก์ชัน printf ( ) จะรับพารามิเตอร์เป็นพอยน์เตอร์ชี้ไปยังแอดเดรสของข้อมูลที่ตำแหน่งเริ่มต้นของอาร์เรย์ และนำข้อความนั้นแสดงออกทางอุปกรณ์แสดงข้อมูลมาตรฐาน

ในการเขียนโปรแกรมจะสามารถใช้พอยน์เตอร์ชี้ไปค่าคงที่สตริงใด ๆ ก็ได้ เช่น

```
char *pmessage = “Hello, world”;
```

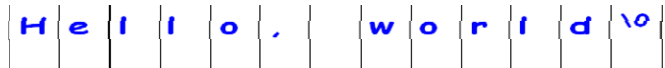
pmessage จะเป็นพอยน์เตอร์ประเภท char ชี้ไปที่อาร์เรย์ของตัวอักษร จะแตกต่างจากการใช้อาร์เรย์ทั่วไปเช่น

```
char amessage[ ] = “Hello, world”;
```

**pmessage**



**amessage**



#### การจองพื้นที่ให้กับอาร์เรย์และตัวชี้ชี้ไปยังค่าคงที่สตริง

ฟังก์ชัน strcpy ( ) ทำหน้าที่สำเนาข้อความจากตัวแปรหนึ่งไปยังอีกตัวแปรหนึ่งเขียนในลักษณะอาร์เรย์

```
void strcpy ( char *s, char *t )
{
    int i=0;
    while( ( s[i] = t[i] ) != '\0' )
        i++;
}
```

ฟังก์ชัน strcpy ( ) เขียนในลักษณะพอยน์เตอร์

```
void strcpy ( char *s, char *t )
{
    while ( ( *s = *t ) != '\0' ) {
        s++;
        t++;
    }
}
```

ฟังก์ชัน strcpy ( ) เขียนในลักษณะพอยน์เตอร์แบบสั้น

```
void strcpy ( char *s, char *t ){
    while ( ( *s++ = *t++ ) != '\0' );
}
```



## 8.7 พอยน์เตอร์ซ้อนพอยน์เตอร์

เป็นวิธีการกำหนดตัวแปร pointer ให้ทำหน้าที่เก็บตำแหน่งที่อยู่ของตัวแปร pointer อีกตัวหนึ่ง บางทีเรียกตัวแปร บางทีเรียกตัวแปร pointer แบบนี้ว่า indirect pointer

การประกาศตัวแปรให้เป็น indirect Pointer

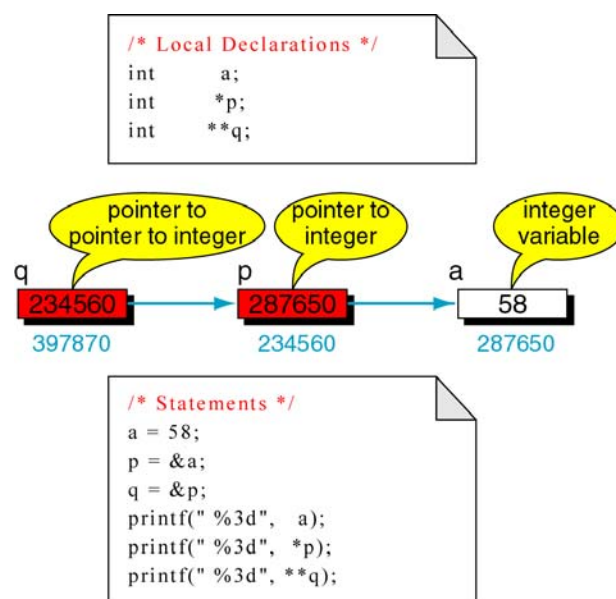
ตัวแปรที่จะทำหน้าที่เป็นตัวแปร indirect pointer จะต้องมีการประกาศหรือบอกให้รู้ไว้ก่อนในตอนต้นของโปรแกรมโดยใช้

รูปแบบ

Type \*\*name-of-pointer

Name-of-pointer คือตัวแปรที่ต้องการใช้เป็นตัวแปร indirect pointer

ตัวอย่างที่ 8.6



## 8.8 ตัวอย่างการใช้ Pointer

ตัวอย่างที่ 8.7 การใช้ pointer สำหรับเรียงข้อมูลในอาร์เรย์โดยใช้การเรียงแบบ Bubble sort

```
1  /*This program puts values into an array, sorts the values into
2  ascending order, and prints the resulting array. */
3  #include <stdio.h>
4  #define SIZE 10
5  void bubbleSort( int * const array, const int size ); /* prototype */
6  void swap( int *element1Ptr, int *element2Ptr ); /* prototype */
7
8  int main( void )
9  {
10     /* initialize array a */
11     int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
12     int i; /* counter */
13     printf( "Data items in original order\n" );
14     /* loop through array a */
15     for ( i = 0; i < SIZE; i++ ) {
16         printf( " %4d", a[ i ] );
17     } /* end for */
```

```

18 bubbleSort( a, SIZE ); /* sort the array */
19 printf( "\nData items in ascending order\n" );
20 /* loop through array a */
21 for ( i = 0; i < SIZE; i++ ) {
22     printf( "%4d", a[ i ] );
23 } /* end for */
24 printf( "\n" );
25 return 0; /* indicates successful termination */
26 } /* end main */
27 /* sort an array of integers using bubble sort algorithm */
28 void bubbleSort( int * const array, const int size )
29 {
30     int pass; /* pass counter */
31     int j; /* comparison counter */
32     /* loop to control passes */
33     for ( pass = 0; pass < size - 1; pass++ ) {
34         /* loop to control comparisons during each pass */
35         for ( j = 0; j < size - 1; j++ ) {
36             /* swap adjacent elements if they are out of order */
37             if ( array[ j ] > array[ j + 1 ] ) {
38                 swap( &array[ j ], &array[ j + 1 ] );
39             } /* end if */
40         } /* end inner for */
41     } /* end outer for */
42 } /* end function bubbleSort */
43 /* swap values at memory locations to which element1Ptr and
44 element2Ptr point */
45 void swap( int *element1Ptr, int *element2Ptr )
46 {
47     int hold = *element1Ptr;
48     *element1Ptr = *element2Ptr;
49     *element2Ptr = hold;
50 } /* end function swap */

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

Data items in original order
 2  6  4  8 10 12 89 68 45 37
Data items in ascending order
 2  4  6  8 10 12 37 45 68 89

```

## แบบฝึกหัดปฏิบัติการคาบที่ 8: Pointer

1. กำหนดตัวแปรดังนี้

```
int    i = 3, j = 5, *p = &i, *q = &j, *r;
double x = 2.50;
```

จงตอบคำถามว่าค่าของตัวแปรต่อไปนี้มีค่าเป็นเท่าใด

(ตอบว่าเป็น illegal ถ้าการกำหนดค่าให้ตัวแปรในข้อนั้นไม่ถูกต้อง)

ตัวแปร	ค่าของตัวแปร
1. *p	
2. *q	
3. *r ( เมื่อกำหนดให้ r = p; )	
4. *r ( เมื่อกำหนดให้ r = &j; )	
5. *r ( เมื่อกำหนดให้ r = &x; )	
6. **&p	
7. *p-1	
8. *p+*q	
9. ++*p	
10. 7**q+7	

2. จากโปรแกรมต่อไปนี้ จงเติมค่าตัวแปรลงในช่องว่างที่กำหนดให้

<pre>/* 1 */ #include &lt;stdio.h&gt; /* 2 */ int main() /* 3 */ { /* 4 */     int    x = 1, y = 2; /* 5 */     int    a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; /* 6 */     int    *ip, *iq; /* 7 */     ip = &amp;x; /* 8 */     y = *ip; /* 9 */     *ip = 0; /* 10 */    ip = &amp;a[0]; /* 11 */    ip = ip + 3; /* 12 */    *ip = 0; /* 13 */    *ip = *ip + 10; /* 14 */    iq = ip; /* 15 */    *iq = 0; /* 16 */    return 0; /* 17 */ }</pre>	<pre>*ip = _____ y   = _____ x   = _____ *ip = _____ *ip = _____ a[3] = _____ a[3] = _____ *iq = _____ *ip = _____</pre>
---	--

3. จงเขียนโปรแกรมทำการรับค่าสายอักขระจากทางแป้นพิมพ์ แล้วทำการแสดงผลสายอักขระนี้จากหลังมาหน้า (Reverse) และแสดงจำนวนของตัวเลขที่อยู่ในสายอักขระดังกล่าว โดยให้ใช้ pointer เท่านั้น

**ตัวอย่างผลลัพธ์โปรแกรม**

**Input**

บรรทัดแรกเป็นสายอักขระ

**Output**

บรรทัดแรกเป็นการแสดงผลสายอักขระนี้จากหลังมาหน้า (Reverse)

บรรทัดถัดไปแสดงจำนวนของตัวเลขที่อยู่ในสายอักขระ

Input	Output
Computer Programming	gnimmargorP retupmoC 0

4. จงเขียนโปรแกรมให้สมบูรณ์ (โดยใช้ Pointer) เพื่อรับและแสดงผล argument พร้อมทั้งสลับลำดับตัวอักษรของ argument ต่างๆ โดยนำตัวอักษรแต่ละลำดับของ argument แต่ละตัวมาเขียนต่อกันเก็บไว้ในตัวแปร str  
ดังนี้ สมมุติว่าโปรแกรมมี Argument ตัวที่ 1, 2 และ 3 คือ 123 abc xyz ผลการจัดเรียงตัวอักษรใหม่ที่ต้องการคือ 1ax2by3cz

**ตัวอย่างผลลัพธ์โปรแกรม**

**Input**

บรรทัดแรกเป็นจำนวน Arguments n ตัว

n บรรทัดถัดไปเป็น Argument

**Output**

บรรทัดแรกเป็นผลลัพธ์

Input	Output
3 123 abc xyz	1ax2by3cz

5. กำหนดให้ Matrix P คือ Matrix ขนาด NxN ที่สร้างจากอาร์เรย์ 1 มิติสองตัว (A และ B) ที่มีความยาว N เท่ากัน  $(1 \leq N \leq 10)$  โดยสมาชิกของ Matrix P ได้จากผลคูณของสมาชิกของอาร์เรย์ A และ B ดังนี้

$$A = [a_1 \quad a_2 \quad a_3 \quad \dots \quad a_N]$$

$$B = [b_1 \quad b_2 \quad b_3 \quad \dots \quad b_N]$$

$$P = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 & \dots & a_1b_N \\ a_2b_1 & a_2b_2 & a_2b_3 & \dots & a_2b_N \\ \dots & \dots & \dots & \dots & \dots \\ a_Nb_1 & a_Nb_2 & a_Nb_3 & \dots & a_Nb_N \end{bmatrix}$$

โปรแกรมสำหรับสร้าง Matrix P จากอาร์เรย์ A และ B ดังนิยามข้างต้น มีตัวอย่างการรันโปรแกรมเป็นดังนี้

Enter N = 2

Input array A

Enter 2 integers: 2 7

Input array B

Enter 2 integers: 9 5

Matrix P

18 10

Enter N = 4

Input array A

Enter 4 integers: 1 2 3 4

Input array B

Enter 4 integers: 5 6 7 8

Matrix P

5 6 7 8

โค้ดของโปรแกรม

```
#include <stdio.h>
#define NMAX 10
void inputArray(int array[ ], int N);
void showArray2D(int matrix[ ][10], int N);
void constructMatrix(int P[ ][10], int N, int A[ ], int B[ ]);
int main()
{
    int a[NMAX], b[NMAX], p[NMAX][NMAX], n;
    printf("Enter N = "); scanf("%d", &n);
    printf("Input array A \n"); inputArray(a, n);
    printf("Input array B \n"); inputArray(b, n);
    constructMatrix(p, n, a, b);
    printf("Matrix P \n");
    showArray2D(p, n);
    return 0;
}
```

โปรแกรมนี้อยู่ภายใต้ลิขสิทธิ์ของทั้งสามฟังก์ชัน จงเขียนรายละเอียดของสามฟังก์ชันนั้นเพื่อให้

โปรแกรมทำงานได้อย่างถูกต้องสมบูรณ์ดังตัวอย่างข้างต้น

## บทที่ 9

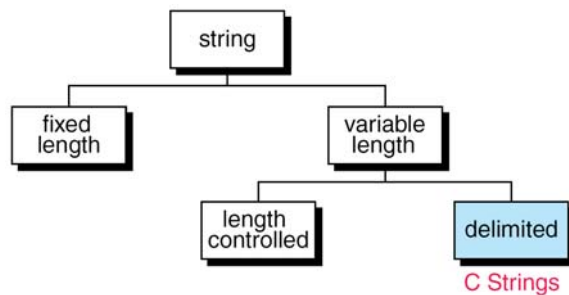
### สายอักขระ

#### จุดประสงค์

เพื่อให้เข้าใจหลักการทำงานของสตริง (string)

#### 9.1 สายอักขระ

สายอักขระเป็นโครงสร้างข้อมูลแบบอาร์เรย์ชนิดหนึ่งที่แต่ละตำแหน่งจะเก็บอักขระใดๆ และจบลงด้วยอักขระพิเศษคือ ‘\0’ ดังนั้นเมื่อโปรแกรมทำงานเพื่อแสดงสายอักขระใดๆ จะแสดงจนกว่าจะพบอักขระพิเศษนี้ อักขระ ‘\0’ เรียกว่า อักขระว่าง (null character)



ในคลาสสตริงจะมีส่วนที่ช่วยในการควบคุมการทำงานเพื่อกำหนดความยาวของสตริงโดยมีตัวที่ควบคุมความยาวของสตริง (Length Control) หรืออาจจะใช้ delimited เป็นตัวที่ควบคุมการจบของข้อความ เช่น



#### 9.2 คำสั่งรับและแสดงผล

คำสั่งรับและแสดงผล จะปรากฏคำสั่งที่ใช้ในการรับค่าและแสดงผลสตริงหรือข้อความ ดังนี้

คำสั่งรับค่า	ตัวอย่าง
<code>scanf("%s",ชื่อตัวแปรสตริง);</code>	<code>scanf("%s",name);</code> //สังเกตว่าไม่ต้องใส่ & หน้าตัวแปร name
<code>gets(ชื่อตัวแปร);</code>	<code>gets(name);</code>

คำสั่งแสดงผล	ตัวอย่าง
<code>printf("%s",ชื่อตัวแปรสตริง);</code>	<code>printf("%s",name);</code>
<code>puts(ชื่อตัวแปร);</code>	<code>puts(name);</code>

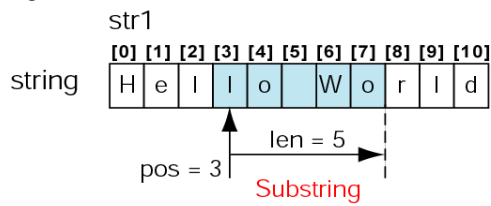
### 9.3 ฟังก์ชันเกี่ยวกับ String

ในภาษาซีจะมีฟังก์ชันที่ใช้สำหรับการจัดการเกี่ยวกับสายอักขระดังแสดงในตารางต่อไปนี้

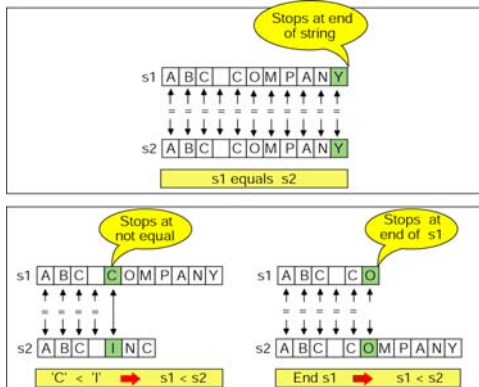
ชื่อฟังก์ชัน	ความหมาย	Library
<code>gets(ชื่อตัวแปร)</code>	รับสตริงหรือข้อความจากคีย์บอร์ด	<code>stdio.h</code>
<code>strlen(ชื่อตัวแปร)</code>	หาความยาวสตริงของตัวแปรที่ระบุ	<code>string.h</code>
<code>strcpy(ชื่อตัวแปร1,ข้อความ/ชื่อตัวแปร2)</code>	คัดลอกข้อความหรือค่าในตัวแปร 2 ไปเก็บไว้ใน ตัวแปร 1	<code>string.h</code>
<code>strcmp(ชื่อตัวแปร1, ชื่อตัวแปร2)</code>	เปรียบเทียบลำดับสตริงระหว่าง ตัวแปร 1 และตัวแปร 2 โดย ถ้าลำดับตัวอักษรใน ตัวแปร1 <u>มาก่อน</u> ตัวแปร2 แสดงว่า ตัวแปร1 < ตัวแปร2 ดังนั้นจะได้เงื่อนไขดังนี้ ตัวแปร1 < ตัวแปร2      ฟังก์ชันจะให้ค่าติดลบ ตัวแปร1 == ตัวแปร2      ฟังก์ชันจะให้ค่าเท่ากับ 0 ตัวแปร1 > ตัวแปร2      ฟังก์ชันจะให้ค่าเป็นบวกที่มากกว่า 0	<code>string.h</code>
<code>strcat(ชื่อตัวแปร1, ชื่อตัวแปร2)</code>	รวมสตริงในตัวแปร 2 ไปต่อท้าย ตัวแปร 1 แล้วเก็บสตริงที่ต่อกันแล้วไว้ในตัวแปร 1	<code>string.h</code>
<code>strcmpi(s1,s2)</code>	เปรียบเทียบอักขระ 2 สองค่าโดยไม่สนใจตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็ก	<code>string.h</code>
<code>strstr(s1,s2)</code>	ค้นหาสายอักขระ s2 ที่พบครั้งแรกในอีกสายอักขระ s1 หรือไม่	<code>string.h</code>
<code>strupr(s1)</code>	แปลงสายอักขระ s1 ให้เป็นตัวพิมพ์ใหญ่	<code>string.h</code>
<code>sprintf(s1,format)</code>	สร้างสายอักขระแล้วแต่อาร์กิวเมนต์ที่ส่งมา	<code>string.h</code>

## ตัวอย่างการใช้งานฟังก์ชันของสตริง

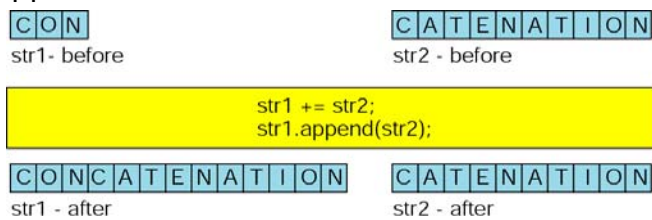
### Substring concept



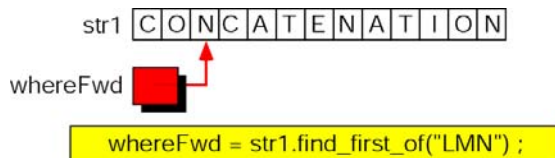
### String compares



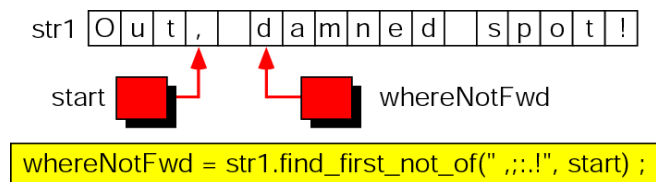
### String append



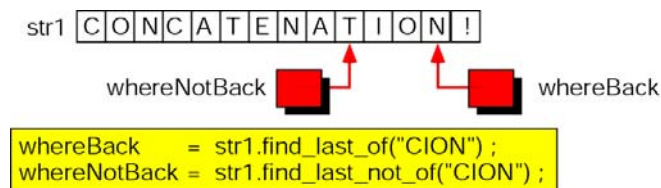
### Find first



### Find first not



### Find last





### ตัวอย่างที่ 9.1 รับชื่อและหาความยาวชื่อ แล้วแสดงผลออกทางจอภาพ

```
1 #include<stdio.h>
2 #include<string.h>
3 void main()
4 { char name[20];
5   int size;
6   printf("Please enter your name: ");
7   gets(name);
8   size = strlen(name);
9   printf("Hello %s\n", name);
10  printf("Your name has %d characters", size);
11 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter your name: Sathit
Hello Sathit
Your name has 6 characters
```

### ตัวอย่างที่ 9.2 โปรแกรมรับชื่อและคัดลอกข้อความ แล้วแสดงผลออกทางจอภาพ

```
1 #include<stdio.h>
2 #include<string.h>
3 void main()
4 { char s1[30], s2[30];
5
6   printf("Please enter string1: ");
7   gets(s1);
8   strcpy(s2, "Welcome to C Programming");
9   printf("s1 : %s\n", s1);
10  printf("s2 : %s\n", s2);
11 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter string1: String1
s1 : String1
s2 : Welcome to C Programming
```

ตัวอย่างที่ 9.3 โปรแกรมรับข้อความ 2 ข้อความแล้วตรวจสอบว่าคำใดมาก่อน-หลัง แล้วแสดงผลทางจอภาพ ตรวจสอบ  $x > y$  หรือ  $x < y$  ถ้า  $x < y$  ให้พิมพ์ s1 มาก่อน s2 ถ้า s1 มาก่อน s2 ให้พิมพ์ s1 ตามด้วย s2 แต่ถ้า s1 มาหลัง s2 ให้พิมพ์ s2 ตามด้วย s1 แต่ถ้านอกจากนี้ให้พิมพ์ว่าทั้งสองคำเป็นคำเดียวกัน

```
1 #include<stdio.h>
2 #include<string.h>
3 void main()
4 { char s1[30], s2[30];
5
6   printf("Please enter string1: ");
7   gets(s1);
8   strcpy(s2, "Welcome to C Programming");
9   printf("s1 : %s\n", s1);
10  printf("s2 : %s\n", s2);
11 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
Please enter string1: Thailand
Please enter string2: Thailand
Two string are similar
```

#### ตัวอย่างที่ 9.4 โปรแกรมต่อข้อความโดยใช้ strcat แล้วแสดงผลออกทางจอภาพ

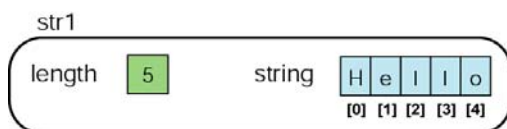
```
1  #include<stdio.h>
2  #include<string.h>
3  void main()
4  {   char s1[20];
5      char s2[20];
6      strcpy(s1," Hello ");
7      printf("s1 : %s\n", s1);
8      strcpy(s2," Yaya ");
9      printf("s2 : %s\n", s2);
10     strcat(s1, s2);
11     printf("After using string concatenate with strcat\n");
12     printf("s1 : %s ", s1);
13 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

```
s1 :  Hello
s2 :  Yaya
After using string concatenate with strcat
s1 :  Hello Yaya
```

### 9.4 คลาสสายอักขระใน C++

แม้ว่าคลังโปรแกรมใน string.h จะมีฟังก์ชันให้ใช้พอสมควร แต่อย่างไรก็ตามส่วนใหญ่มีลักษณะการเขียนโปรแกรมเชิงหน้าที่มากกว่าที่จะเป็นการเขียนโปรแกรมเชิงวัตถุ หัวข้อนี้จะอธิบายการเขียนโปรแกรมเชิงวัตถุในการสร้างคลาส String ซึ่งเป็นคลาสสายอักขระที่ใช้จัดการกับสายอักขระในลักษณะต่างๆ เพื่อเป็นแนวทางในการสร้าง หรือใช้ฟังก์ชันสมาชิกของคลาสนี้ต่อไป คลาส String



คลาสสตริงเป็นหนึ่งใน Standard template library เมื่อนำเอาคลาสสตริงมาจัดการกับสตริงจะง่ายกว่าแบบเดิมมาก นอกจากนี้ยังมีจุดเด่นในเรื่องฟังก์ชันต่าง ๆ มากมาย ที่ไว้จัดการกับสตริงโดยเฉพาะ

ในการสร้างสตริงใหม่ด้วยคลาสสตริง จะมีได้หลายรูปแบบเพราะว่าคลาส string มี constructor หลายรูปแบบ

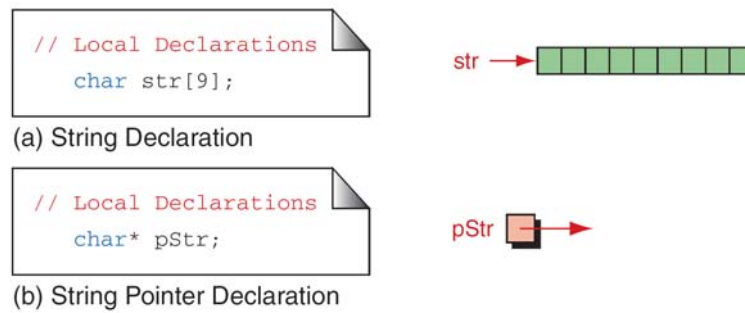
```
เช่น
string s1;

string s2("Hello");

string s3(n, 'c');

string s4(s2);
```

## ตัวอย่างการประกาศสตริง



## ตัวอย่างที่ 9.5

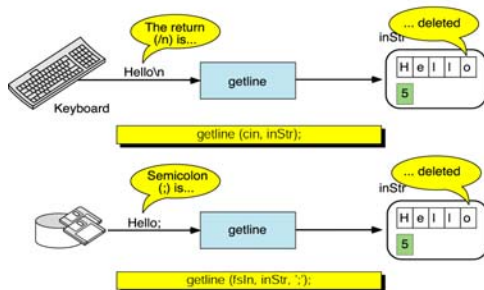
```
1  #include <iostream>
2  #include <iomanip>
3  #include <string>
4  using namespace std;
5  int main ()
6  {
7      string s1;
8      string s2 ("Hello World");
9      string s3 (s2);
10     string s4 (5, 'A');
11     string s5 (s2, 6);
12     string s6 ("Hello", 2);
13     string s7 ("Hello", 3, 25);
14     cout << "Value of s1: " << s1 << endl;
15     cout << "Value of s2: " << s2 << endl;
16     cout << "Value of s3: " << s3 << endl;
17     cout << "Value of s4: " << s4 << endl;
18     cout << "Value of s5: " << s5 << endl;
19     cout << "Value of s6: " << s6 << endl;
20     cout << "Value of s7: " << s7 << endl;
21     return 0;
22 } // main
```

### ผลลัพธ์จากการประมวลผลโปรแกรม

```
Value of s1:
Value of s2: Hello World
Value of s3: Hello World
Value of s4: AAAAAA
Value of s5: World
Value of s6: He
Value of s7: lo
```

### การอ่านค่าสตริงจากคีย์บอร์ด

การอ่านค่าสตริงเพื่อเก็บไว้ในออบเจกต์ string สามารถใช้ออบเจกต์ cin ร่วมกับโอเปอเรเตอร์ >> แต่จะไม่สามารถอ่านคีย์ space หรือ คีย์ tab หรือคีย์ enter เพื่อที่จะอ่าน สตริงที่ต่อเนื่องกันรวมทั้งคีย์ space หรือคีย์แท็บจนกว่าจะกดคีย์ enter เราจะต้องใช้ฟังก์ชัน getline ของคลาส string ซึ่งมีรูปแบบการใช้งานดังนี้



ตัวอย่างที่ 9.6

```

1  #include <iostream>
2  #include <iomanip>
3  #include <string>
4  using namespace std;
5  int main ()
6  {
7      cout << "Enter a name in the form <last,first>: \n";
8      string  lastName;
9      getline (cin, lastName, ',');
10     string  firstName;
11     getline (cin, firstName);
12     cout << "Here is your name:\n\t|"
13          << firstName << " " << lastName << "|\n";
14     return 0;
15 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

Enter a name in the form <last,first>:
Washington,George
Here is your name:
    |George Washington|

```

**เครื่องหมายการกำหนดค่าของสตริง(Assignment Operator)**

เราสามารถที่จะโอเวอร์โหลดเพื่อนำมาใช้กับชนิดข้อมูลที่แตกต่างกันได้ 3 ชนิดคือ

- ค่าจาก string ใน C++
- ค่าจาก string ใน C
- ค่าของคาแรกเตอร์

ในกรณีที่มีการประกาศสตริงในรูปแบบต่อไปนี้จะเกิด Error เกิดขึ้น

```
string str1='A';
```

### ตัวอย่างที่ 9.7

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main ()
5  {
6      string str1 ("String 1");
7      string str2;
8      string str3;
9      string str4;
10     string str5 = "String 5";
11     cout << "String 1: " << str1 << endl;
12     str2 = str1;
13     cout << "String 2: " << str2 << endl;
14     str3 = "Hello";
15     cout << "String 3: " << str3 << endl;
16     str4 = 'A';
17     cout << "String 4: " << str4 << endl;
18     cout << "String 5: " << str5 << endl;
19     return 0;
20 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

String 1: String 1
String 2: String 1
String 3: Hello
String 4: A
String 5: String 5

```

### ตัวอย่างที่ 9.8

```

1  #include <iostream>
2  #include <iomanip>
3  #include <string>
4  using namespace std;
5  int main ()
6  {
7      cout << "Enter some text and I'll parse it into words. \n";
8      string strIn;
9      while (cin >> strIn)
10         cout << strIn << endl;
11     cout << "End of demonstration\n";
12     return 0;
13 } // main

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

Enter some text and I'll parse it into words.
Now is the time
Now
is
the
time
For all good students
For
all
good
students

```

### ตัวอย่างที่ 9.9

```
1  /* Creates a text file from keyboard input.
2      Written by:
3      Date:
4  */
5  #include <iostream>
6  #include <fstream>
7  #include <string>
8  using namespace std;
9  int main ()
10 {
11     ofstream fsOut;
12     cout << "Begin file copy. Enter your text.\n"
13         << "<EOF> to stop.\n";
14     fsOut.open ("progl4-5.txt");
15     if (!fsOut)
16     {
17         cerr << "\aCould not open output file.\n\a";
18         exit (100);
19     } // if
20     string str;
21     while (getline(cin, str))
22         fsOut << str << endl;
23     fsOut << str << endl;
24     fsOut.close();
25     cout << "\nEnd file copy\n";
26     return 0;
27 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```
Begin file copy. Enter your text.
<EOF> to stop.
Now is the time
For all good students
^d
End file copy

Output file...
Now is the time
For all good students
```

### ตัวอย่างที่ 9.10

```
1  /* Typewriter program: adds two spaces to the left
2     margin and writes line to file.
3     Written by:
4     Date:
5  */
6  #include <iostream>
7  #include <fstream>
8  #include <string>
9  using namespace std;
10 int main ()
11 {
12     ifstream fsIn;
13     ofstream fsOut;
14     cout << "Begin file copy and shift.\n";
15     fsIn.open ("progl4-5.txt");
16     if (!fsIn)
17     {
18         cerr << "\aCould not open input file.\n\a";
19         exit (100);
20     } // if
21     fsOut.open ("progl4-6.out");
22     if (!fsOut)
23     {
24         cerr << "\aCould not open output file.\n\a";
25         exit (101);
26     } // if
27     string str;
28     int    lineCount = 0;
29     while (getline (fsIn, str))
30     {
31         fsOut << "  " << str << endl;
32         lineCount++;
33     } // while
34     fsIn.close();
35     fsOut.close();
36     cout << "End of file shift. "
37           << lineCount << " lines written\n";
38     return 0;
39 } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Begin file copy and shift.  
End of file shift. 3 lines written

Output:

Now is the time  
For all good students

### ตัวอย่างที่ 9.11

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5 int main ()
6 {
7     ifstream fsTextIn;
8     fsTextIn.open ("progl4-5.txt");
9     if (!fsTextIn)
10     {
11         cout << "\aCould not open input file.\n\a";
12         exit (100);
13     } // if open
14     string str;
15     while (getline (fsTextIn, str))
16         cout << str << endl << endl;
17     fsTextIn.close();
18     return 0;
19 } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Now is the time

For all good students

**Array of String** การสร้างอาร์เรย์ของสตริงสามารถทำได้ดังตัวอย่างต่อไปนี้

### ตัวอย่างที่ 9.12

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5 int main ()
6 {
7     string daysAry[7];
8     daysAry[0] = "Sunday";
9     daysAry[1] = "Monday";
10    daysAry[2] = "Tuesday";
11    daysAry[3] = "Wednesday";
12    daysAry[4] = "Thursday";
13    daysAry[5] = "Friday";
14    daysAry[6] = "Saturday";
15    cout << "\nThe days of the week\n";
16    for (int daysIndex = 0; daysIndex < 7; daysIndex++)
17        cout << daysAry[daysIndex] << endl;
18    return 0;
19 } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

The days of the week

Sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday



## แบบฝึกหัดปฏิบัติการคาบที่ 9: String

1. จงเขียนโปรแกรมที่รับข้อความมา 2 ข้อความเข้าทางคีย์บอร์ด จากนั้นให้ทำการค้นหาว่าข้อความที่สองพบในข้อความแรกหรือไม่ ถ้าพบมีกี่ครั้งและอยู่ที่ตำแหน่งใดบ้าง
2. จงเขียนโปรแกรมที่รับข้อความมา 2 ข้อความและเลขจำนวนเต็ม 1 ตัวเข้าทางคีย์บอร์ด จากนั้นให้นำข้อความที่สองไปใส่ไว้ในตำแหน่งที่ถูกกำหนดจากเลขที่รับเข้ามา ถ้าตำแหน่งที่กำหนดเป็นไปไม่ได้ให้นำข้อความที่สองนั้นไปใส่ไว้ด้านหลังของข้อความตัวแรก
3. จงเขียนฟังก์ชัน ตัดช่องว่างหัวท้ายของคำ(Trim) ของค่าที่ส่งเข้าไป

ตัวอย่าง

```
input = " Hello world ";
```

```
output = "Hello world";
```

4. จงเขียนฟังก์ชัน หาตำแหน่งแรก(indexOf) ของค่าที่ส่งเข้าไป

ตัวอย่าง

```
string input = "Hello world";
```

```
string pattern = "world";
```

```
int index = indexOf("Hello world", pattern);
```

```
output => index = 6
```

5. จงเขียนฟังก์ชัน Replace คำ ที่ส่งเข้าไป

ตัวอย่าง ให้แทนที่ คำ 'world' ในตัวแปร input ด้วย คำว่า 'XXX'

```
string input = "Hello world 123 world 1234";
```

```
string pattern = "world";
```

```
string replace = "XXX";
```

```
string output = Replace(input, pattern, replace);
```

```
output = "Hello XXX 123 XXX 1234";
```

## บทที่ 10

### ข้อมูลแบบโครงสร้าง (Structure)

#### จุดประสงค์

1. เพื่อให้ทราบความหมายและลักษณะของข้อมูลแบบโครงสร้าง
2. เพื่อให้ทราบวิธีการใช้งานข้อมูลแบบโครงสร้าง
3. สามารถนำข้อมูลแบบโครงสร้างมาใช้เขียนโปรแกรมได้อย่างมีประสิทธิภาพ

สตรัคเจอร์ หรือกลุ่มข้อมูลชนิดโครงสร้าง (Structure / struct ) เป็นการประกาศหน่วยของข้อมูลใหม่ที่เกิดจากการรวมกลุ่มของข้อมูล เป็นโครงสร้าง โดยข้อมูลซึ่งเป็นสมาชิกของโครงสร้างใหม่ อาจมีหลายตัว และเป็นชนิดเดียวกันหรือต่างชนิดกันก็ได้ เช่น มีสมาชิกเป็นจำนวนเต็ม ทศนิยม และอักขระได้ สตรัคเจอร์ อาจมีสมาชิกที่เป็นอาร์เรย์ หรือแม้แต่สตรัคเจอร์ด้วยก็ได้

- ข้อมูลพื้นฐาน (simple data)
  - int, unsigned int, char, float, double, long เป็นต้น
- ข้อมูลซับซ้อน (complex data)
  - array, structure & union
- ประโยชน์สตรัคเจอร์ หรือกลุ่มข้อมูลชนิดโครงสร้าง (Structure)
  - เพื่อกำหนดหน่วยข้อมูลใหม่ ให้เหมาะสมกับข้อมูลที่ต้องการเก็บ เช่น การกำหนดหน่วยข้อมูลนักเรียน ที่ประกอบด้วยสมาชิกเป็น ชื่อและนามสกุลที่เป็น string (char [ ]) กับรหัสนักศึกษาที่เป็นตัวเลข(int) ซึ่งเราสามารถกำหนดให้ตัวแปรนักเรียน A กับ B มีโครงสร้างแบบหน่วยข้อมูลที่สร้างขึ้นนี้ได้ดังรูป

#### 10.1 การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง

การประกาศตัวแปรของกลุ่มข้อมูลชนิดโครงสร้าง structure มีขั้นตอนการทำงานดังนี้

1. Structure definition การนิยามกลุ่มข้อมูลที่สร้างขึ้นใหม่ ว่ามีสมาชิกอะไรบ้าง เป็นชนิดใด ข้อมูลย่อยในสตรัคเจอร์เรียกว่า ฟิวด์ (Field)

2. Structure declaration ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้นมา

##### 9.1.1 การนิยามกลุ่มข้อมูลที่สร้างขึ้นใหม่ว่ามีสมาชิกข้อมูลชนิดใดบ้าง

รูปแบบ

struct ชื่อของสตรัคเจอร์

```
{    ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
    ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
    .....  
    ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
}
```

### ตัวอย่างที่ 10.1 การระบุกลุ่มข้อมูลชนิดโครงสร้างที่สร้างขึ้นใหม่

```
struct address
{   char name[30];
    char detail[50];
    int  age;
    char telephone[10];
};
struct student
{   char name[30];
    char surname[50];
    int  Id;
};
```

### ตัวอย่างที่ 10.2

- จงนิยามสตรักเจอร์ชื่อว่า date เพื่อใช้ในการเก็บข้อมูลของวันที่โดยประกอบด้วยสมาชิก 3 ตัวชื่อว่า day, month และ year ซึ่งสมาชิกทั้งสามเป็นจำนวนเต็ม

```
struct date
{
    int day;
    int month;
    int year;
};
```

หรือ

```
struct date {
    int day,month,year;
};
```

### 9.1.2 ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้น

มีวิธีการประกาศได้ 2 ลักษณะคือ

- 1. การประกาศโดยตรง (ภายหลังจากการนิยามสตรักเจอร์แล้ว) โดยมีรูปแบบ struct ชื่อแบบของสตรักเจอร์ ชื่อตัวแปร[, ชื่อตัวแปร, ...];

เช่น struct student stdA, stdB;

### ตัวอย่างที่ 10.3

```
struct address
{   char name[30];
    char detail[50];
    int  age;
    char telephone[10];
};
struct address input1, input2;
```

- 2. การประกาศตัวแปรพร้อมการสร้างกลุ่มข้อมูลโดยการระบุต่อท้ายก่อนเครื่องหมาย ;

การประกาศโดยตรง โดยมีรูปแบบ

struct ชื่อแบบของสตรักเจอร์ ชื่อตัวแปร[, ชื่อตัวแปร, ...];

การประกาศตัวแปรพร้อมการสร้างกลุ่มข้อมูลโดยการระบุต่อท้ายก่อนเครื่องหมาย ;

### ตัวอย่างที่ 10.4

```
struct address
{   char name[30];
    char detail[50];
    int  age;
    char telephone[10];
} input1, input2 ;
```

หลังจากนี้อาจมีการประกาศตัวแปรเพิ่มได้ เช่น struct address input3, input4 ;

## 10.2 การเข้าถึงสมาชิกในสตรัคเจอร์

สำหรับ Array เราใช้เลขดัชนีหรือ index(subscript) ในการอ้างอิง สมาชิกแต่ละตัวใน array (เช่น a[5], b[1][0] เป็นต้น) แต่รูปแบบการเข้าถึงข้อมูลสมาชิกของ struct ทำได้โดยใช้โอเปอเรเตอร์จุด (.) แล้วตามด้วยชื่อสมาชิก

รูปแบบ:

ชื่อตัวแปรสตรัคเจอร์.ชื่อสมาชิก

structure-variable.field

จากรูปแบบ:                      ชื่อตัวแปรสตรัคเจอร์.ชื่อสมาชิก

**ตัวอย่างที่ 10.5** การประกาศโครงสร้าง และการประกาศตัวแปรของโครงสร้าง

```
struct complex {  
    double real;  
    double image;  
} num0 ;  
struct complex num1;
```

**ตัวอย่างที่ 10.6** การกำหนดค่า เช่น

```
num1.real = 1;  
num1.image = 2.5;  
num0.real = num1.image;  
num0.image = num1.real;
```

**ตัวอย่างที่ 10.7** การเขียนโค้ดภาษาซีเพื่อจัดการกับตัวเลขเชิงซ้อน

1.นิยามกลุ่มข้อมูลชนิดเชิงซ้อน

```
struct complex  
{  
    double real;  
    double image;  
};
```

2.ประกาศตัวแปร

```
struct complex a,b,c;
```

3.ตัวอย่างการใช้งานตัวแปร

```
c.real=a.real+b.real;  
c.imag=a.imag+b.image;
```

## 10.3 การกำหนดค่าด้วยสตรัคเจอร์

กำหนดให้ตัวแปร a และ b เป็นโครงสร้าง เมื่อใช้คำสั่ง a = b; จะมีความหมายเช่นเดียวกับ

a.real=b.real;

a.imag=b.image;

คือการคัดลอกข้อมูลทุกข้อมูลจากตัวแปรโครงสร้าง b ให้กับตัวแปรโครงสร้าง a แต่ไม่มีการเปรียบเทียบโดยตรงระหว่างสตรัคเจอร์ เช่น a == b หรือ a>b หรือ a<b คำสั่งดังกล่าวไม่สามารถใช้ได้

### ตัวอย่างที่ 10.8 การสร้าง struct ของเลขเชิงซ้อน

```
1  #include <stdio.h>
2  int main()
3  {
4      struct complex {
5          double real;
6          double image;
7      } x,y;
8      x.real = 4;
9      x.image = 0.5;
10     y.real = x.real + 1;
11     y.image = x.image + 0.25;
12     printf("y= %.2f + %.2fi\n",y.real,y.image);
13     return 0;
14 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

y= 5.00 + 0.75i

### ตัวอย่างที่ 10.9 การสร้าง struct ของพนักงาน

```
1  #include <stdio.h>
2  int main()
3  {
4      struct employ {
5          char name[25];
6          int age;
7          int pay;
8      }employee;
9      employee.age=32;
10     employee.name[2] = 'X';
11     printf("Age of employee is %d\n",employee.age);
12     printf("The second letter of name is %c\n", employee.name[2]);
13     return 0;
14 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Age of employee is 32  
The second letter of name is X

จากโปรแกรมข้างต้น ส่วนของการประกาศโครงสร้างของพนักงาน สามารถประกาศได้ดังนี้

```
struct employ {
    char name[25];
    int age;
    int pay;
}employee;
```

การกำหนดค่าให้กับโครงสร้างสามารถกำหนดได้ดังนี้

```
employee.age=32;
employee.name[2] = 'X';
```

### ตัวอย่างที่ 10.10 การกำหนดค่าแบบ string ให้กับ struct

```
1  #include <stdio.h>
2  int main()
3  {
4      struct address{
5          char name[30];
6          char detail[50];
7          int age;
8          char telephone[10];
9      };
```

```

10 struct address input;
11 strcpy(input.name, "Yaya");
12 strcpy(input.detail, "1 Phayathai bangkok");
13 input.age=20;
14 strcpy(input.telephone, "212895");
15 printf("Name of employee is %s\n",input.name);
16 printf("Address of employee is%s\n", input.detail);
17 printf("Age of employee is %d\n",input.age);
18 printf("Telephone number is%s\n", input.telephone);
19 return 0;
20 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

Name of employee is Yaya
Address of employee is1 Phayathai bangkok
Age of employee is 20
Telephone number is212895

```

## 10.4 การคัดลอกค่าของตัวแปรสตรัคเจอร์

การคัดลอก(Copy) ค่าของตัวแปรสตรัคเจอร์ชนิดเดียวกัน สามารถทำได้โดยใช้เครื่องหมาย "=" เช่น

```

struct date {
    int day,month,year;
} d1, d2;

```

แทนที่จะกำหนดค่าสมาชิกทีละตัว

```

d2.day = d1.day;
d2.month = d1.month;
d2.year = d1.year;

```

สามารถกำหนดค่าเป็นตัวแปรสตรัคเจอร์ได้เลย เช่น d2 = d1;

### ตัวอย่างที่ 10.11

```

1 #include <stdio.h>
2 int main()
3 {
4     struct date {
5         int day,month,year;
6     } ;
7     struct date d1={12,01,2013};
8     struct date d2;
9
10    d2=d1;
11
12    printf("date in variable d1 is %d  and date in variable d2 is %d\n",d1.day,d2.day );
13    printf("month in variable d1 is %d  and month in variable d2 is %d\n",d1.month,d2.month );
14    printf("year in variable d1 is %d  and year in variable d2 is %d\n",d1.year,d2.year );
15
16    return 0;
17 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

```

date in variable d1 is 12  and date in variable d2 is 12
month in variable d1 is 1  and month in variable d2 is 1
year in variable d1 is 2013  and year in variable d2 is 2013

```

### ตัวอย่างที่ 10.12

```
1  #include <stdio.h>
2  int main()
3  {
4      struct date
5      {
6          int day;
7          int month;
8          int year;
9      };
10     struct person
11     {
12         char name[30];
13         struct date birthday;
14     };
15
16     struct person p1;
17
18     strcpy(p1.name, "Yaya");
19     p1.birthday.day = 10;
20     p1.birthday.month = 3;
21     p1.birthday.year = 1992;
22     printf("name in variable p1 is %s\n", p1.name );
23     printf("date in variable p1 is %d\n", p1.birthday.day );
24     return 0;
25 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```
name in variable p1 is Yaya
date in variable p1 is 10
```

ใช้เครื่องหมาย { } คร่อมค่าเริ่มต้นทั้งหมด และใช้เครื่องหมาย , คั่นระหว่างค่าของสมาชิกแต่ละตัว  
ตัวอย่างเช่น

### ตัวอย่างที่ 10.13

```
struct date
{
    int day;
    int month;
    int year;
} ;
struct date d1 = {10,5,1992};
```

### ตัวอย่างที่ 10.14

```
struct subject
{
    char name[20];
    int credit;
    char grade;
} s1 = {"Physics I",3,'A'};
```

### ตัวอย่างที่ 10.15 การแสดงข้อมูลที่อยู่ในตัวแปร struct

```
1  #include <stdio.h>
2  int main()
3  {
4      struct {
5          char   name[30];
6          char   detail[50];
7          int    age;
8          char   telephone[10];
9      } address, newAddress = {"newUser", "1 Samsennai Phayathai
10 Bangkok", 20, "212895"};
11      /* Display address of data groups */
12      printf("Address of address variable is %p \n", &address);
13      printf("Address of newAddress variable is %p \n", &newAddress);
14      printf("sizeof of newAddress = %d\n", sizeof newAddress);
15      return 0;
16  }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```
Address of address variable is 0022FEE0
Address of newAddress variable is 0022FE80
sizeof of newAddress = 96
```

## 10.5 การเปรียบเทียบค่าของตัวแปรสตรัคเจอร์

ในการเปรียบเทียบค่าของตัวแปรสตรัคเจอร์นั้น ให้เปรียบเทียบค่าของสมาชิกแต่ละตัวจะนำตัวแปรสตรัคเจอร์สตรัคเจอร์มาเปรียบเทียบกันโดยตรงไม่ได้ เช่น หากมีการตัวแปรชนิด struct date ชื่อว่า d1 และ d2

```
struct date {
    int day, month, year
} d1, d2;
```

จะนำ d1 และ d2 มาเปรียบเทียบกันโดยตรงไม่ได้

```
if(d1 == d2)
    printf("d1 is the same date as d2");
```

## 10.6 การกำหนดชนิดตัวแปรใหม่(Type definition)

ในภาษาซี สามารถกำหนดชนิดตัวแปรขึ้นมาใหม่ได้ โดยใช้คำสั่ง typedef (type definition) รูปแบบ:

typedef ชนิดตัวแปรที่มีอยู่แล้ว ชนิดตัวแปรใหม่;

เช่น typedef int my\_int;

รูปแบบการใช้ typedef ร่วมกับการนิยามสตรัคเจอร์:

```
typedef struct
{
```



```

        ชนิดตัวแปร ชื่อตัวแปรที่ 1;
        ชนิดตัวแปร ชื่อตัวแปรที่ 2;
        ...
        ชนิดตัวแปร ชื่อตัวแปรที่ n;
    } ชนิดตัวแปรใหม่;

```

#### ตัวอย่างที่ 10.16

```

typedef struct {
    int day,month,year;
} date;

```

ในการประกาศตัวแปรก็สามารถใช้ชนิดตัวแปรใหม่ได้เลย เช่น

```
date d1,d2;
```

ข้อสังเกต เมื่อกำหนดตัวแปรหลังจากการกำหนดด้วย typedef แล้ว ไม่ต้องมีคำว่า struct นำหน้าชนิดข้อมูลอีก

#### ตัวอย่างที่ 10.17

จงกำหนดชนิดตัวแปรใหม่ชื่อ Student ซึ่งมีสมาชิก 2 ตัวคือ name ใช้สำหรับเก็บชื่อซึ่งมีความยาวไม่เกิน 30 ตัวอักษร และสมาชิกตัวที่สองชื่อ faculty ใช้สำหรับเก็บชื่อคณะซึ่งมีความยาวไม่เกิน 15 ตัวอักษร

```

typedef struct
{
    char name[31];
    char faculty[16];
} Student;

```

#### ตัวอย่างที่ 10.18 การใช้ typedef เพื่อประกาศชนิดข้อมูลใหม่ที่สร้างขึ้นจากชนิดข้อมูลแบบโครงสร้าง struct

```

struct info
{
    char firstName[20];
    char lastName[20];
    int age;
};

struct info i1, i2;

typedef struct
{
    char firstName[20];
    char lastName[20];
    int age;
}

```

```

} Info;
Info j;
typedef struct info infoType;
infoType i3,i4;

```

**ตัวอย่างที่ 10.19** การใช้ typedef เพื่อประกาศชนิดข้อมูลใหม่ที่สร้างขึ้นจากชนิดข้อมูลแบบโครงสร้าง struct

```

typedef struct {
    char firstName[20];
    char lastName[20];
    int age;
} InfoT;
typedef struct {
    InfoT info;
    double salary;
} EmployeeT;
EmployeeT e1;
e1.info.age = 21;

```

## 10.7 การผ่านสตรัคเจอร์ให้กับฟังก์ชัน

การผ่านค่าของตัวแปรสตรัคเจอร์ให้กับฟังก์ชันทำได้เหมือนกับตัวแปรชนิดอื่นๆ (int, float, char) การแก้ไขของพารามิเตอร์ภายในฟังก์ชัน จะไม่มีผลต่อค่าของตัวแปรสตรัคเจอร์ที่ถูกส่งมาเป็นอาร์กิวเมนต์ ถ้าในโปรแกรมมีหลายฟังก์ชัน การนิยามสตรัคเจอร์และการกำหนดชนิดตัวแปรใหม่ให้นำมาไว้นอกฟังก์ชัน main

**ตัวอย่างที่ 10.20**

```

1  #include <stdio.h>
2  typedef struct {
3      double re;
4      double im;
5  } complex;
6
7  void display(complex a);
8
9  int main() {
10     complex x;
11     x.re = 1;
12     x.im = 0.75;
13     display(x);
14     return 0;
15 }
16 void display(complex a) {
17     printf("%.2f + %.2fi\n",a.re,a.im);
18 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

1.00 + 0.75i

### ตัวอย่างที่ 10.21

```
1 #include <stdio.h>
2 typedef struct {
3     int day,month,year;
4 } date;
5 void edit(date a);
6 int main() {
7     date d1 = {26,01,2013};
8     edit(d1);
9     printf("%d/%d/%d\n",d1.day,d1.month,d1.year);
10    return 0;
11 }
12 void edit(date a)
13 {
14     a.year = a.year + 10;
15 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

26/1/2013

### ตัวอย่างที่ 10.22\_ฟังก์ชันที่มีการส่งค่ากลับเป็นสตรัคเจอร์

```
1 #include <stdio.h>
2 typedef struct {
3     double re,im;
4 } complex;
5 complex cconst(double a,double b);
6 int main() {
7     double x = 2,y = 0.5;
8     complex cnum;
9     cnum = cconst(x,y);
10    printf("%.2f + %.2fi\n",cnum.re,cnum.im);
11    return 0;
12 }
13 complex cconst(double a,double b) {
14     complex num;
15     num.re = a; num.im = b;
16     return num;
17 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

2.00 + 0.50i

## 10.8 อาร์เรย์ของโครงสร้าง

การใช้งานโครงสร้างนอกจากใช้ในลักษณะของตัวแปรแล้ว ยังสามารถใช้งานในลักษณะของอาร์เรย์ได้อีกด้วย เช่น

การเก็บข้อมูลประวัติของพนักงาน จะมีโครงสร้างที่ใช้เก็บข้อมูลของพนักงานแต่ละคน หากใช้ในลักษณะของตัวแปรปกติจะสามารถเก็บข้อมูลของพนักงานได้เพียง 1 คน ซึ่งพนักงานทั้งบริษัทอาจจะมีหลายสิบหรือหลายร้อยคน การเก็บข้อมูลในลักษณะนี้จะใช้อาร์เรย์เข้ามาช่วย เช่น

**Person staff [STAFFSIZE];**

การอ้างโดยใช้คำสั่งต่าง ๆ

- `staff` อ้างถึงอาร์เรย์ของโครงสร้าง
- `staff[i]` อ้างถึงสมาชิกที่ `i` ในอาร์เรย์
- `staff[i].forename` อ้างถึงชื่อหน้าของสมาชิกที่ `i` ของอาร์เรย์
- `staff[i].surname[j]` อ้างถึงตัวอักษรตัวที่ `j` ในนามสกุลของสมาชิกที่ `i` ของอาร์เรย์

การเรียกใช้งานสมาชิกบางตัวในอาร์เรย์ของโครงสร้างผ่านฟังก์ชัน

- การใช้ข้อมูลสมาชิกแต่ละตัวจะอ้างถึงโดยการอ้างผ่านระบบดัชนีเหมือนอาร์เรย์ทั่วไป เช่น  
`print_person ( staff[k] );`
- รูปแบบฟังก์ชันสามารถกำหนดด้วย  
`void print_person ( Person employee )`
- หากต้องการเรียกใช้งานฟังก์ชันที่ทำงานกับทั้งอาร์เรย์ เช่น การเรียกใช้งานฟังก์ชันที่ทำการเรียงลำดับอาร์เรย์ตามชื่อหน้า เช่น  
`sort_forename ( staff, STAFFSIZE );`
- รูปแบบฟังก์ชันสามารถกำหนดด้วย  
`void sort_forename ( Person staff[ ], int size )`

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ของโครงสร้าง

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ของโครงสร้างสามารถทำได้โดย

```
Person staff[] = { { "Bloggs", "Joe", MALE, 21 },
                   { "Smith", "John", MALE, 30 },
                   { "Black", "Mary", FEMALE, 25 } };
```

การใช้อาร์เรย์กับสตรัคเจอร์

ในการเก็บข้อมูลที่ต้องใช้ตัวแปรสตรัคเจอร์จำนวนมาก สามารถแก้ปัญหาโดยใช้ตัวแปรอาร์เรย์ของสตรัคเจอร์

```
struct date {
    int day, month, year;
};
struct date date_list[3];
```

<b>day</b>		<b>day</b>		<b>day</b>	
<b>month</b>		<b>month</b>		<b>month</b>	
<b>year</b>		<b>year</b>			
<b>date_list[๓]</b>		<b>date_list[๒]</b>		<b>date_list[๑]</b>	

รูปแบบการเข้าถึงสมาชิกในแต่ละอีลีเมนต์ในอาร์เรย์ของสตรัคเจอร์

ชื่อตัวแปรอาร์เรย์ของสตรัคเจอร์[ดัชนี].ชื่อสมาชิก

**ตัวอย่างที่ 10.23** (กำหนดค่าให้กับสมาชิกในอีลีเมนต์แรก)

```
date_list[0].day = 12;
date_list[0].month = 10;
```

```
date_list[0].year = 2002;
```

#### ตัวอย่างที่ 10.24 (การรับค่าและแสดงค่าอาร์เรย์ของสตรัคเจอร์)

```
1  #include <stdio.h>
2  int main() {
3      typedef struct {
4          char name[30];
5          int age;
6      } student;
7      student stds[3];
8      int i;
9      for(i=0;i<3;i++) {
10         printf("Enter name of student %d: ",i+1);
11         //scanf("%s",stds[i].name);
12         gets(stds[i].name);
13         fflush(stdin);
14         printf("Enter age: ");
15         scanf("%d",&stds[i].age);
16         fflush(stdin);
17     }
18     printf("=====\n");
19     printf("Name          Age\n");
20     printf("=====\n");
21     for(i=0;i<3;i++){
22         printf("%-15s%d\n",stds[i].name, stds[i].age);
23     }
24     return 0;
25 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```
Enter name of student 1: Yaya
Enter age: 15
Enter name of student 2: Nadej
Enter age: 17
Enter name of student 3: Mark
Enter age: 15
=====
Name          Age
=====
Yaya          15
Nadej         17
Mark          15
```

#### ตัวอย่างที่ 10.25

```
1  #include <stdio.h>
2  typedef struct {
3      char name[30];
4      int salary;
5  } employee;
6  void display (employee emps[]);
7  int main()
8  {
9      employee emp_list[3];
10     int i;
11     for(i=0;i<3;i++)
12     {
13         printf("Enter name of employee %d: ",i+1);
14         //scanf("%s", emp_list[i].name );
15         gets(emp_list[i].name);
16         fflush(stdin);
17         printf("Enter salary: ");
18         scanf("%d", &emp_list[i].salary );
19         fflush(stdin);
```

```

20     }
21     display(emp_list);
22     return 0;
23 }
24 void display(employee emps[])
25 {
26     int i;
27     printf("-----\n");
28     printf("Name          Salary\n");
29     printf("-----\n");
30     for(i=0;i<3;i++)
31         printf("%-15s%d \n", emps[i].name, emps[i].salary);
32 }

```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```

Enter name of employee 1: Yaya
Enter salary: 15000
Enter name of employee 2: Nadej
Enter salary: 15000
Enter name of employee 3: Kimberry
Enter salary: 15000
-----
Name          Salary
-----
Yaya          15000
Nadej         15000
Kimberry      15000

```

## 10.9 การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

กรณีการส่งอาร์กิวเมนต์เป็นตัวแปร struct จะไม่เหมาะกับ struct ที่มีขนาดใหญ่ เนื่องจากทุกครั้งที่ส่งตัวแปร struct จะเป็นการสำเนาตัวแปรตัวใหม่ขึ้นมาในฟังก์ชัน ซึ่งจะทำให้ช้าและเปลืองพื้นที่หน่วยความจำ เราจะใช้พอยน์เตอร์เข้ามาช่วยแก้ปัญหานี้ โดยส่งแอดเดรสของตัวแปร struct มายังฟังก์ชัน ซึ่งรับอาร์กิวเมนต์ เป็นพอยน์เตอร์ อาร์กิวเมนต์จะชี้ไปยังแอดเดรสเริ่มต้นของตัวแปร struct จะช่วยให้การทำงานเร็วขึ้นและเปลืองหน่วยความจำน้อยลง แต่สิ่งที่ต้องระวังคือ หากมีการเปลี่ยนแปลงค่าที่อาร์กิวเมนต์พอยน์เตอร์ชี้อยู่ ค่าในตัวแปร struct ที่ส่งมายังฟังก์ชันจะเปลี่ยนตามโดยอัตโนมัติ

```
struct point origin, *pp;
```

```
pp = &origin;
```

```
printf ( "origin is (%d, %d)\n", (*pp).x, (*pp).y );
```

จะได้ตัวแปร pp ชี้ไปยังข้อมูลแบบโครงสร้างชื่อ struct point การเขียน \*pp จะเป็นการอ้างถึงโครงสร้างการอ้างถึงสมาชิกสามารถทำได้โดยอ้าง (\*pp).x หรือ (\*pp).y สิ่งที่ต้องระวังคือ (\*pp).x จะไม่เหมือนกับ \*pp.x เนื่องจากเครื่องหมาย . จะมีลำดับความสำคัญสูงกว่า \* การแปลความหมาย \*pp.x จะเหมือนกับการอ้าง \*(pp.x) ซึ่งจะก่อให้เกิดความผิดพลาดขึ้น การอ้างถึงสมาชิกอาจเขียนอีกลักษณะหนึ่งโดยใช้เครื่องหมาย -> สมมติ p เป็นพอยน์เตอร์ รูปแบบการใช้เป็นดังนี้ p->member-of-structure จะสามารถแปลงประโยคการใช้พอยน์เตอร์อ้างสมาชิกของ struct จากตัวอย่างข้างบนได้ว่า printf ( "origin is (%d, %d)\n", pp->x, pp->y);

## 10.10 ตัวอย่างการใช้ structure สำหรับการเรียงโครงสร้างข้อมูล

ตัวอย่างที่ 10.26

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdbool.h>
4  #define NUM_STU 5
5  typedef struct
6  {
7      char   name[26];
8      int    midterm[3];
9      int    final;
10     } STUDENT;
11
12 void insertionSort (STUDENT list[], int last);
13 int main ()
14 {
15     STUDENT* pStuPtr;
16     STUDENT  stuAry[NUM_STU] =
17     {
18         {"Charles, George", {85, 94, 79}, 93},
19         {"Adams, Karin",    {75, 91, 89}, 89},
20         {"Nguyen, Tuan",    {87, 88, 89}, 90},
21         {"Oh, Bill",        {78, 96, 88}, 91},
22         {"Chavez, Maria",   {83, 79, 93}, 91}
23     };
24     printf("Unsorted data:\n");
25     for (pStuPtr = stuAry; pStuPtr < stuAry + NUM_STU; pStuPtr++)
26         printf("%-26s %4d %4d %4d %4d\n",
27             pStuPtr->name,
28             pStuPtr->midterm[0],
29             pStuPtr->midterm[1],
30             pStuPtr->midterm[2],
31             pStuPtr->final);
32     printf("\n");
33     insertionSort (stuAry, NUM_STU - 1);
34     printf("Sorted data:\n");
35     for (pStuPtr = stuAry; pStuPtr < stuAry + NUM_STU; pStuPtr++)
36         printf("%-26s %4d %4d %4d %4d\n",
37             pStuPtr->name,
38             pStuPtr->midterm[0],
39             pStuPtr->midterm[1],
40             pStuPtr->midterm[2],
41             pStuPtr->final);
42     return 0;
43 }
44 void insertionSort (STUDENT list[], int last)
45 {
46     bool    located;
47     STUDENT temp;
48     STUDENT* pCurrent;
49     STUDENT* pWalker;
50     STUDENT* pLast;
51     for (pCurrent = list + 1, pLast = list + last;
52         pCurrent <= pLast;
53         pCurrent++)
54     {
55         located = false;
56         temp    = *pCurrent;
57     }
```

```

58     for (pWalker = pCurrent - 1;
59         pWalker >= list && !located;
60     )
61         if (strcmp(temp.name, pWalker->name) < 0)
62         {
63             *(pWalker + 1) = *pWalker;
64             pWalker--;
65         }
66         else
67             located = true;
68         *(pWalker + 1) = temp;
69     }
70     return;
71 }

```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

```

Unsorted data:
Charles, George      85   94   79   93
Adams, Karin         75   91   89   89
Nguyen, Tuan         87   88   89   90
Oh, Bill              78   96   88   91
Chavez, Maria        83   79   93   91

Sorted data:
Adams, Karin         75   91   89   89
Charles, George      85   94   79   93
Chavez, Maria        83   79   93   91
Nguyen, Tuan         87   88   89   90
Oh, Bill              78   96   88   91

```



## แบบฝึกหัดปฏิบัติการคาบที่ 10: Structure

1. จุดในระนาบสามารถที่จะแสดงได้โดยการใช้ระบบ Coordinate x และ y ดังนั้นเราสามารถเขียนจุดในระนาบได้โดยการใช้ตัวแปรแบบโครงสร้างที่มีสองฟิลด์ดังแสดงด้านล่าง

```
typedef struct
{
    int x;
    int y;
}POINT
```

จงเขียนโปรแกรมเพื่อทำการรับค่าข้อมูลแบบโครงสร้างของจุด (POINT) แล้วทำการเรียกฟังก์ชันเพื่อทำการคำนวณหาระยะทางระหว่างจุดสองจุดโดยใช้ระยะทางแบบยูคลิเดียน ซึ่งมีนิยามการทำงานดังนี้

$$\text{Dist (Point1, Point 2)} = \sqrt{(\text{Point1.x} - \text{Point2.x})^2 + (\text{Point1.y} - \text{Point2.y})^2}$$

หลังจากนั้นให้เรียกฟังก์ชันเพื่อระบุว่าจุดทั้งสองอยู่ Quadrant ที่เท่าไร

### ข้อมูลอินพุต

บรรทัดแรกเป็นจำนวนจุด n ( $1 \leq n \leq 100$ )

n บรรทัดถัดไปเป็นตำแหน่งของจุดในพิกัด x และ y ( $-1000 < x, y < 1000$ )

### ข้อมูลเอาต์พุต

n บรรทัดแรกเป็นผลลัพธ์ของ Quadrant มีค่าตั้งแต่ 1- 4

บรรทัดสุดท้ายเป็นผลลัพธ์ของระยะทาง

### ตัวอย่าง

อินพุต	เอาต์พุต
2	4
2 -2	1
2 2	4

2. Structure ชื่อ vector3D เป็นเวกเตอร์สามมิติ มีสมาชิกเป็นเลขทศนิยมสามตัวคือ: X, Y, Z จงเขียนโปรแกรมเพื่อรับค่าทั้งสามตัวของเวกเตอร์ จากนั้นคำนวณความยาวของเวกเตอร์โดยใช้ฟังก์ชัน FindLength ความยาวของเวกเตอร์คำนวณได้จาก  $l = \sqrt{X^2 + Y^2 + Z^2}$

**ข้อมูลอินพุต** บรรทัดแรกเป็นจำนวนจุด n ( $1 \leq n \leq 100$ )

บรรทัดถัดไปเป็นตำแหน่งของจุดในพิกัด x และ y และ z ( $-1000 < x, y, z < 1000$ )

**ข้อมูลเอาต์พุต** บรรทัดสุดท้ายเป็นผลลัพธ์

### ตัวอย่าง

อินพุต	เอาต์พุต
1	2.39
0.5 1.2 2.0	

3. ที่ร้านสะดวกซื้อแห่งหนึ่งเมื่อทำการรับเงินจากลูกค้าจะทำการแยกเงินแต่ละราคาใส่ไว้ในช่องเก็บเงินที่ประกอบด้วยชนิดของเงินแต่ละราคา คือ 1000, 500, 100, 50, 20, 10, และ 1 บาท จงเขียนโปรแกรมเพื่อที่จะรับจำนวนเงินจากลูกค้าเพื่อส่งไปยังฟังก์ชันที่ทำหน้าที่คำนวณหาจำนวนเงินแต่ละชนิดราคาหลังจากนั้นคืนค่าตัวแปรโครงสร้างที่ประกอบด้วยช่องเก็บเงินแต่ละชนิดราคา คือ 1000, 500, 100, 50, 20, 10, และ 1 บาท ตามลำดับ

### ข้อมูลอินพุต

บรรทัดแรกเป็นจำนวนเงินจากลูกค้า ( $0 \leq a \leq 1000000$ )

### ข้อมูลเอาต์พุต

บรรทัดสุดท้ายเป็นผลลัพธ์โครงสร้างที่ประกอบด้วยช่องเก็บเงินแต่ละชนิดราคา คือ 1000, 500, 100, 50, 20, 10, และ 1 บาท ตามลำดับ

### ตัวอย่าง

อินพุต	เอาต์พุต
1751	1 1 2 1 0 0 1

4. ในไพ่สำหรับหนึ่งประกอบด้วย face values และ suits

โดยที่ face values ประกอบด้วย A, 2, 3, 4, 5, 6, 7, 8, 9, J, Q, K

ส่วน suits ประกอบด้วย โพดำ (Spade) ♠ โพธิ์แดง หรือหัวใจ (Heart) ♥

ข้าวหลามตัด (Diamond) ♦ ดอกจิก (Club) ♣

จงเขียนโปรแกรมเพื่อกำหนดโครงสร้างของไพ่(Deck) ที่ประกอบไปด้วย faces และ suits หลังจากนั้นผู้ใช้กำหนดรายละเอียดของไพ่ตามจำนวนที่ผู้ใช้กำหนด หลังจากนั้นให้โปรแกรมทำการเรียงลำดับไพ่ทั้ง n ใบ ดังกล่าวจากน้อยไปหามาก แสดงผลพร้อมทั้งหาค่าผลรวมของไพ่ที่ป้อนเข้ามา

### ข้อมูลอินพุต

บรรทัดแรกเป็นจำนวนไพ่ทั้ง n ใบ ( $1 \leq n \leq 52$ )

n บรรทัดต่อไปแสดงรายละเอียดของไพ่ตามที่ใช้กำหนด

### ข้อมูลเอาต์พุต

บรรทัดต่อไปแสดงการเรียงลำดับไพ่ทั้ง n ใบดังกล่าวจากน้อยไปหามาก

บรรทัดสุดท้ายแสดงผลรวมของไพ่ที่ป้อนเข้ามา

### ตัวอย่าง

อินพุต	เอาต์พุต
3 3 C 7 S A H	A-H, 3-C, 7-S 11

5. โครงสร้างข้อมูลแบบ Stack ประกอบด้วยตัวแปรอาร์เรย์ที่ใช้ในการเก็บค่าของ Stack ขนาด N สมาชิก และตัวแปร Top สำหรับชี้ค่าบนสุดของ Stack โดยตัวแปร Top จะใช้ประกอบการเพิ่มและลบข้อมูลที่อยู่ใน Stack

การเพิ่มหรือลบข้อมูลจะทำได้ทีละค่าเฉพาะข้อมูลที่อยู่บนสุดของ Stack

ฟังก์ชันพื้นฐานของ Stack คือ

1. ฟังก์ชัน Push เป็นฟังก์ชันสำหรับเพิ่มข้อมูลเข้าไปใน stack
2. ฟังก์ชัน Pop เป็นฟังก์ชันสำหรับดึงข้อมูลที่อยู่บนสุดออกจาก stack

จงเขียนโปรแกรมเพื่อจำลองการทำงานของ Stack โดยเมื่อผู้ใช้ต้องการ Push ให้พิมพ์ P เพื่อเพิ่มข้อมูล และใส่ข้อมูลลงไป เมื่อผู้ใช้ต้องการ Pop ให้พิมพ์ X โดยโปรแกรมจะดึงข้อมูลที่อยู่บนสุดออกมา

ตัวอย่าง

Please select operation: P

Please input data: 20

Please select operation: P

Please input data: 15

Please select operation: P

Please input data: 35

Please select operation: P

Please input data: 10

Please select operation: X

Data is : 10

Please select operation: X

Data is : 35

## บทที่ 11

### แฟ้มข้อมูล (Data File)

#### จุดประสงค์

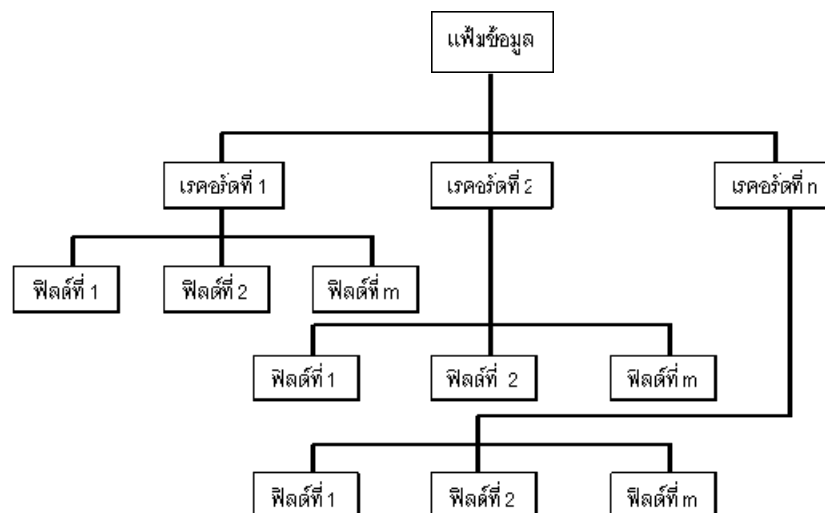
1. เพื่อให้ทราบลักษณะโครงสร้าง ความหมาย และประเภทของไฟล์
2. เพื่อให้ทราบวิธีการเปิดและปิดไฟล์
3. เพื่อให้ทราบวิธีการอ่านข้อมูลและวิธีการเขียนข้อมูลลงไฟล์
4. เพื่อให้ทราบวิธีการต่าง ๆ ในการจัดการเกี่ยวกับไฟล์

#### 11.1 ความหมายของแฟ้มข้อมูลในภาษา C

แฟ้มข้อมูล (Data file) คือ แฟ้มที่มีการเก็บข้อมูลที่มีความสัมพันธ์กันมาไว้ด้วยกัน โดยมีการเก็บข้อมูลอย่างต่อเนื่องกันไป ตั้งแต่ต้นแฟ้มข้อมูลไปจนกระทั่งจบแฟ้มข้อมูล โดยที่ผู้เขียนข้อมูลสามารถแบ่งข้อมูลที่ต้องการจัดเก็บลงในแฟ้มเป็น field หรือ record ก็ได้ หรืออาจจัดเก็บข้อมูลตามแนวขนาดเนื้อที่โดยไม่จำเป็นต้องแบ่งข้อมูลในแฟ้มเป็น field หรือ record ก็ได้ โดยปกติผู้เขียนโปรแกรมภาษา C นิยมแบ่งข้อมูลที่ต้องการลงในแฟ้มเป็น field หรือ record เพราะมีความสะดวกในการเรียกใช้ข้อมูลจากแฟ้มที่ต้องการนอกจากนี้ยังสามารถใช้โปรแกรม text editor หรือโปรแกรม word processing สร้างแฟ้มข้อมูลที่ต้องการได้อย่างสะดวกรวดเร็ว

#### 11.2 ประเภทของแฟ้มข้อมูล

Text file เป็นไฟล์ที่เก็บข้อมูลในรูปแบบของตัวอักษรและทำการแยกแต่ละบรรทัดของ text file ออกจากกันด้วย Binary file เป็นไฟล์ที่เก็บข้อมูลในรูปแบบเฉพาะของคอมพิวเตอร์ลักษณะโครงสร้างของแฟ้มข้อมูลทั่วไป



## 11.3 การประมวลผลแฟ้มข้อมูลในภาษา C (data file processing in C)

โดยปกติแล้วผู้เขียนโปรแกรมเกี่ยวกับแฟ้มข้อมูลในภาษา C จะมีความต้องการประมวลผลแฟ้มข้อมูลอยู่ 3 แบบ คือ

- 1) การบันทึกข้อมูลในแฟ้มข้อมูล (Write data into file)
- 2) การอ่านข้อมูลขึ้นจากแฟ้มข้อมูลขึ้นมาใช้งาน (read data from file)
- 3) การเพิ่มข้อมูลลงไปในแฟ้มข้อมูล (append data into file)

### 11.3.1 การบันทึกข้อมูลเก็บไว้ในแฟ้มข้อมูล (Write data into file)

1. เปิดแฟ้มข้อมูลด้วยคำสั่ง `fopen( )` ตั้งชื่อแฟ้มข้อมูล (file name) พร้อมกับระบุ mode ของการบันทึกข้อมูลลงในแฟ้มเป็น “w”
2. บันทึกข้อมูลลงในแฟ้มโดยใช้ฟังก์ชัน `putc( )` หรือ `fprintf( )` หรือ `fwrite( )` บันทึกข้อมูลลงแฟ้ม ขึ้นอยู่กับลักษณะของข้อมูลที่ต้องการบันทึกลงแฟ้มดังนี้
  - ถ้าข้อมูลที่ต้องการบันทึกเป็นตัวอักขระตัวเดียว (Single character) ให้ฟังก์ชัน `putc( )` เพราะสามารถบันทึกตัวอักขระตัวเดียวได้ดี
  - ถ้าข้อมูลที่ต้องการบันทึกเป็นตัวเลขจำนวนเต็ม (Integer) หรือตัวเลขจำนวนทศนิยม (floating point) หรือสตริง (strings) ให้ใช้ฟังก์ชัน `fprintf( )` เพราะสามารถจัดรูปแบบข้อมูลที่บันทึกได้
  - ถ้าข้อมูลที่ต้องการบันทึกเป็นข้อมูลแบบโครงสร้าง (Structures) หรือตัวแปรชุด (arrays) ให้ใช้ฟังก์ชัน `fwrite( )` เพราะสามารถกำหนดเนื้อที่และจำนวนครั้งของการบันทึกข้อมูลได้
3. หลังจากบันทึกข้อมูลลงแฟ้มเรียบร้อยแล้ว จะต้องใช้คำสั่ง `fclose( )` ปิดแฟ้มข้อมูลทุกครั้งเพื่อป้องกันความเสียหายที่อาจเกิดขึ้นได้

#### ข้อควรระวังเกี่ยวกับการบันทึกข้อมูลลงแฟ้ม

ในฟังก์ชัน `fopen( )` เมื่อใช้ mode “w” เป็นการเปิดแฟ้มข้อมูลเพื่อบันทึกข้อมูลลงในแฟ้มเท่านั้น ถ้าเป็นแฟ้มข้อมูลเก่าที่เคยเก็บข้อมูลไว้แล้ว จะมีผลทำให้ข้อมูลทั้งหมดในแฟ้มข้อมูลเก่า ถูกลบทิ้งไปโดยอัตโนมัติ แล้วสร้างแฟ้มข้อมูลใหม่ขึ้นมาแทนที่

### 11.3.2 การอ่านข้อมูลขึ้นจากแฟ้มข้อมูล (read data from file)

1. เปิดแฟ้มข้อมูลด้วยคำสั่ง `fopen( )` ตั้งชื่อแฟ้มข้อมูล พร้อมกับระบุ mode ของการบันทึกข้อมูลลงแฟ้มเป็น “r”
2. อ่านข้อมูลขึ้นจากแฟ้มโดยสามารถใช้ฟังก์ชัน `getc( )` หรือ `fscanf( )` หรือ `fread( )` อ่านข้อมูลขึ้นจากแฟ้มได้ ขึ้นอยู่กับลักษณะของข้อมูลที่ต้องการอ่านขึ้นจากแฟ้มดังนี้
  - ถ้าข้อมูลที่ต้องการอ่านขึ้นจากแฟ้มเป็นตัวอักขระตัวเดียวให้ใช้ฟังก์ชัน `getc( )` เพราะสามารถอ่านข้อมูลตัวอักขระตัวเดียวได้ดี
  - ถ้าข้อมูลที่ต้องการอ่านขึ้นจากแฟ้มเป็นตัวเลขจำนวนเต็ม หรือตัวเลขจำนวนทศนิยม หรือสตริงให้ใช้ฟังก์ชัน `fscanf( )` เพราะสามารถจัดรูปแบบข้อมูลที่อ่านขึ้นจากแฟ้มได้

- ถ้าข้อมูลที่ต้องการอ่านจากแฟ้ม เป็นข้อมูลแบบโครงสร้าง หรือตัวแปรชุดให้ใช้ฟังก์ชัน `fread( )` เพราะสามารถกำหนดขนาดเนื้อที่และจำนวนครั้งของการอ่านข้อมูลขึ้นจากแฟ้มได้

3. นำข้อมูลที่อ่านขึ้นจากแฟ้มไปประมวลผล เช่น พิมพ์ค่าออกทางจอภาพ หรือนำไปคำนวณก็ได้

4. หลังจากประมวลผลข้อมูลที่ได้จากแฟ้มเสร็จเรียบร้อยแล้ว จะต้องใช้คำสั่ง `fclose( )` ปิดแฟ้มข้อมูลทุกครั้งเพื่อป้องกันการเสียหายที่อาจเกิดขึ้นได้

#### ข้อควรระวังเกี่ยวกับการอ่านข้อมูลขึ้นจากแฟ้ม

ในฟังก์ชัน `fopen( )` เมื่อใช้ mode “r” เป็นการเปิดแฟ้มข้อมูล เพื่ออ่านข้อมูลขึ้นจากแฟ้มข้อมูลอย่างเดียว ไม่สามารถบันทึกข้อมูลเพิ่มเติมลงไปในแฟ้มได้

### 11.3.3 การเพิ่มข้อมูลลงไปในแฟ้มข้อมูล (append data into file)

1. เปิดแฟ้มข้อมูล ด้วยคำสั่ง `fopen( )` ตั้งชื่อแฟ้มข้อมูลพร้อมกับระบุ mode ของการอ่านข้อมูลจากแฟ้มเป็น “a”

2. เขียนคำสั่งเกี่ยวกับการบันทึกข้อมูลเพิ่มลงแฟ้ม โดยสามารถใช้คำสั่ง `fprintf( )` บันทึกข้อมูลลงแฟ้มหรือจะใช้คำสั่ง `fwrite( )` บันทึกข้อมูลลงแฟ้มก็ได้ ขึ้นอยู่กับความต้องการของผู้เขียนโปรแกรม

3. หลังจากบันทึกข้อมูลลงแฟ้มเสร็จเรียบร้อยแล้ว จะต้องใช้คำสั่ง `fclose( )` ปิดแฟ้มข้อมูลทุกครั้งเพื่อป้องกันความเสียหายที่อาจเกิดขึ้นได้

#### ข้อควรระวังเกี่ยวกับการเพิ่มข้อมูลลงแฟ้ม

ในฟังก์ชัน `fopen( )` เมื่อใช้ mode “a” เป็นการเปิดแฟ้มข้อมูลสำหรับบันทึกข้อมูลเพิ่มเติมลงไปในแฟ้มได้ โดยที่ข้อมูลที่เพิ่มเข้าไปจะเป็นข้อมูลชุดสุดท้ายของแฟ้มเสมอ

## 11.4 การเปิดไฟล์ (Opening Files)

File pointer คือ pointer ที่ชี้ไปยัง data type FILE ซึ่งถูกกำหนดไว้ใน `stdio.h` โดยเป็น structure ที่เป็นที่เก็บรวบรวมข้อมูลที่เกี่ยวข้องกับ file นั้น เช่น

- File descriptor
- ตำแหน่งปัจจุบันใน buffer
- Pointers ไปยัง buffers
- ในขณะนั้น file กำลังถูก read หรือ write อยู่หรือไม่ ทั้งนี้ก่อนที่ file จะสามารถถูก read หรือ write จะต้องใช้ฟังก์ชันใน C library ชื่อ `fopen()` เปิด file ไว้ก่อน ซึ่งมี limit ว่าจะสามารถเปิดได้กี่ files พร้อมๆ กัน หลังจาก that files ถูกใช้เสร็จแล้ว files จะต้องถูกปิดโดยใช้ฟังก์ชัน `fclose()` ซึ่งเป็นการเคลียร์ file buffer และตัด connection ไปยัง file นั้นออก

การเปิดหรือปิดไฟล์จะใช้ฟังก์ชัน `fopen()` ซึ่งใช้ file pointer สำหรับการเปิดปิดไฟล์ ซึ่งมีรูปแบบดังนี้

`FILE *fopen( char *path, char *mode);`

ฟังก์ชันนี้มี 2 arguments ทั้ง 2 arguments เป็น pointer ไปยัง character strings โดย pointer แรกจะเก็บค่า address ของชื่อ file ที่จะถูกเปิด และ pointer ที่ 2 เป็น address ของ access mode

### ตัวอย่างที่ 11.1 ตัวอย่างการเปิดไฟล์

```
FILE *fptr;
```

```
fptr=fopen("song.txt","w");
```

ความหมายของ access mode

access mode string	ความหมาย
"w"	เปิด file เพื่อ write ซึ่งถ้ามี file นั้นอยู่แล้วการเลือกใช้ mode นี้จะทำการลบ content ของเดิมออก
"r"	เปิด file เพื่อ read
"a"	เปิด file เพื่อ write ถ้ามี file เดิมอยู่ จะ write ข้อมูลต่อท้าย file เดิม

ถ้าฟังก์ชัน fopen() ทำงานสำเร็จจะ return ค่าของ file pointer แต่หากทำงานไม่สำเร็จ (ไม่สามารถเปิด file ได้) ฟังก์ชันจะ return NULL

access mode string	ความหมาย
"r+"	เปิด file เพื่ออ่านและเขียน
"w+"	เปิด file ใหม่เพื่ออ่านและเขียน
"a+"	เปิด file สำหรับอ่านและเขียนต่อท้าย

## 11.5 การปิด files (Closing files)

เมื่อใช้ files แล้วจะต้องปิดด้วยฟังก์ชัน fclose() ซึ่งมีรูปแบบดังนี้

```
int fclose( FILE *stream);
```

ฟังก์ชัน fclose() จะ return ค่า 0 หากทำงานสำเร็จ หากทำไม่สำเร็จจะ return ค่าเป็น EOF

### ตัวอย่างที่ 11.2 ตัวอย่างการปิดไฟล์

```
FILE *fptr;
```

```
fptr=fopen("song.txt","r");
```

```
...
```

```
fclose(fptr);
```

### ตัวอย่างที่ 11.3 การใช้ fopen() และ fclose()

```
1 #include <stdio.h>
2 #include <conio.h>
3 int main(void) {
4     FILE *fptr;
5     fptr = fopen("D:\\test.txt", "r");
6     if(fptr != NULL){
7         printf("I can open file\n");
8         fclose(fptr);
9     } else
10        printf("Error! File can't open\n");
11    getch();
12    return (0);
13 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

ฟังก์ชัน fopen() ทำงานสำเร็จจะ return ค่าของ file pointer แต่หากทำงานไม่สำเร็จ (ไม่สามารถเปิด file ได้) ฟังก์ชันจะ return NULL

## 11.6 การอ่านและเขียนตัวอักษรจากไฟล์

ในการอ่าน character จาก input stream ที่ถูกชี้โดย file pointer สามารถใช้ฟังก์ชัน getc() และ fgetc() ซึ่งมีรูปแบบของฟังก์ชัน ดังนี้

```
char fgetc(FILE *stream);
```

```
char getc(FILE *stream);
```

โดยหากฟังก์ชันทำงานสำเร็จ ฟังก์ชันจะคืนค่าเป็น character ในรูปแบบของ char และหากทำงานไม่สำเร็จจะคืนค่า EOF กลับมา ขอให้ดูวิธีการใช้ฟังก์ชันจากโปรแกรมต่อไปนี้

### ตัวอย่างที่ 11.4 การอ่านและเขียนตัวอักษรจากไฟล์

```
1 #include <stdio.h>
2 #include <conio.h>
3 int main() {
4     char ch;
5     FILE *fptr;
6     fptr = fopen("D:\\ascii.txt", "r");
7     if(fptr != NULL){
8         printf("I can successfully open my file\n");
9         while ((ch = getc(fptr))!=EOF){
10            printf("%c",ch);    }
11        fclose(fptr);
12    } else{
13        printf("Error! I can't open my file\n");
14    }
15    getch();
16    return 0;
17 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

การอ่าน character จาก input stream ที่ถูกชี้โดย file pointer สามารถใช้ฟังก์ชัน getc() และ fgetc() โดยหากฟังก์ชันทำงานสำเร็จ ฟังก์ชันจะคืนค่าเป็น character ในรูปแบบของ char และหากทำงานไม่สำเร็จจะคืนค่า EOF กลับมา



## ฟังก์ชัน `getc()`

เป็นฟังก์ชันที่ใช้อ่านข้อมูลตัวอักขระตัวเดียวขึ้นจากแฟ้มข้อมูลที่ต้องการ  
รูปแบบการใช้ฟังก์ชัน

```
getc(fp);
```

หรือ

```
single_char = getc(fp);
```

โดยที่

`single_char` คือ ตัวแปรชนิด `single character` ที่ใช้เก็บข้อมูลตัวอักขระ ตัวเดียว

`fp` คือ `file pointer` ของแฟ้มข้อมูลที่ต้องการบันทึกข้อมูลลงไป

**ตัวอย่างที่ 11.5** การอ่านและเขียนตัวอักษรจากไฟล์

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<stdlib.h>
4  int main(void)
5  {
6      FILE *fp;
7      char ch;
8      if(( fp=fopen( "D:\\file1.txt", "r" ))==NULL)
9      {
10         printf("Error in open file");
11         printf("\n007");
12         exit(1);
13     }
14     do
15     {
16         ch=getc(fp);
17         putchar(ch);
18     } while(ch !=EOF);
19     fclose(fp);
20     getch();
21     return 0;
22 }
```

**ผลลัพธ์จากการประมวลผลโปรแกรม**

อ่านข้อมูลตัวอักขระตัวเดียวขึ้นจากแฟ้มข้อมูลที่ต้องการ

## ฟังก์ชันสำหรับการเขียนตัวอักษรไปสู่ file

ฟังก์ชันสำหรับการ `write character` ไปสู่ file 2 ฟังก์ชันดังนี้

```
int fputc(int c, FILE *stream);
```

```
int putc(int c, FILE *stream);
```

ฟังก์ชันทั้ง 2 มี 2 arguments โดย argument แรกเป็น `character` ในรูปแบบของ `integer` ส่วน

argument ที่ 2 เป็น `file pointer`

ถ้าฟังก์ชันทำงานสำเร็จจะ return `character` ที่ output ไปในรูปแบบของ `integer` หากทำงานไม่สำเร็จ  
จะคืนค่า `EOF`

## ตัวอย่างที่ 11.6 การ copy file

```
1  #include <stdio.h>
2  #include <conio.h>
3  int main() {
4      char ch;
5      FILE *infile, *outfile;
6      infile = fopen("D:\\file1.txt", "r");
7      outfile = fopen("D:\\targetFile.txt", "w");
8      if((infile == NULL) || (outfile == NULL))
9      {
10         printf("File open error\n");
11     }
12     else {
13         ch = fgetc(infile);
14         while (ch != EOF) {
15             fputc((char)ch, outfile);
16             ch = fgetc(infile);
17         }
18     }
19     if (infile != NULL) fclose(infile);
20     if (outfile != NULL) fclose(outfile);
21     getch();
22     return 0;
23 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

อ่านข้อมูลตัวอักขระตัวเดียวขึ้นจากแฟ้มข้อมูลที่ต้องการ file1.txt และนำไปเก็บในอีกแฟ้มข้อมูล targetFile.txt

ฟังก์ชัน putc( )

เป็นฟังก์ชันที่ใช้บันทึกข้อมูลตัวอักขระตัวเดียวลงไปในแฟ้มข้อมูลที่ต้องการ

รูปแบบการใช้ฟังก์ชัน

putc(single\_char,fp);

โดยที่

single\_char คือ ค่าคงที่ชนิดตัวอักขระตัวเดียว หรือตัวแปรชนิด single\_character

fp คือ file pointer ของแฟ้มข้อมูลที่ต้องการบันทึกข้อมูลลงไป

## ตัวอย่างที่ 11.7 ฟังก์ชันสำหรับการเขียนตัวอักษรไปสู่ file

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<stdlib.h>
4  int main(void)
5  {
6      FILE *fp;
7      char ch;
8      if(( fp=fopen("D:\\file1.txt", "w"))==NULL)
9      {
10         printf("Error in open file");
11         printf("\007");
12         exit(1);
13     }
14     printf("Please press <Enter> to quit program.\n");
```

15	printf("\nEnter your sentence : ");
16	do {
17	ch=getche( );
18	putc(ch, fp);
19	} while(ch != '\r');
20	fclose(fp);
21	getch();
22	return 0;
23	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
อ่านข้อมูลตัวอักษรจากผู้ใช้ แล้วนำไปเก็บในแฟ้มข้อมูลที่ต้องการ file1.txt	

## 11.7 การอ่านและเขียน String ลงแฟ้มข้อมูล

มีฟังก์ชัน ที่เกี่ยวข้องดังต่อไปนี้

int fputs(const char \*s, FILE \*stream);

char \*fgets(char \*s, int size, FILE \*stream);

fprintf() และ fscanf() ซึ่งทำงานเหมือนกับ printf() และ scanf() ยกเว้นมี arguments ตัวแรกเป็น file pointer รูปแบบของฟังก์ชันมีดังนี้

int fscanf( FILE \*stream, const char \*format, ...);

int fprintf( FILE \*stream, const char \*format, ...);

**ตัวอย่างที่ 11.8** การอ่านและเขียน String ลงแฟ้มข้อมูล

1	#include <stdio.h>
2	#include <conio.h>
3	#define STRSIZE 1000
4	#define READ "r"
5	int main() {
6	FILE *fp;
7	int wc = 0;
8	int checkread;
9	char word[STRSIZE];
10	fp = fopen("D:\\file1.txt",READ);
11	if (fp == NULL)
12	fprintf(stderr, "Can't open file");
13	else {
14	checkread = fscanf(fp,"%s",word);
15	while (checkread != EOF) {
16	wc++;
17	checkread = fscanf(fp,"%s",word);
18	}
19	fprintf(stdout, "Number of words %d\n",wc);
20	fclose(fp);
21	}
22	getch();
23	return 0;
24	}
ผลลัพธ์จากการประมวลผลโปรแกรม	
อ่านข้อมูลจากแฟ้มที่กำหนด และทำการนับจำนวนคำที่อยู่ในแฟ้มนั้น ๆ	

### ฟังก์ชัน fprintf( )

เป็นฟังก์ชันที่ใช้บันทึกข้อมูล (write) ลงแฟ้มโดยสามารถจัดรูปแบบข้อมูลที่ต้องการบันทึกได้คล้ายกับฟังก์ชัน printf( ) แตกต่างกันตรงที่ printf( ) เป็นฟังก์ชันที่ใช้พิมพ์ผลลัพธ์ออกทางจอภาพแต่ฟังก์ชัน fprintf( ) ใช้บันทึกข้อมูลลงแฟ้ม

#### รูปแบบการใช้

fprintf (fp,control string,variable list);

#### โดยที่

fp คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล

control string คือ รหัสรูปแบบข้อมูลและรหัสควบคุมใช้เหมือนฟังก์ชัน printf( ) เช่น สามารถระบุชนิดของข้อมูลที่ต้องการบันทึกลงแฟ้มเป็น %d, %c, %f, %s หรือใช้รหัสควบคุม \n หรือ \t ก็ได้

variable list คือ ค่าคงที่ ตัวแปร หรือนิพจน์ที่จะเขียนลงแฟ้มข้อมูล ถ้าเป็นค่าคงที่สตริงต้องเขียนอยู่ภายในเครื่องหมาย “...”

#### ตัวอย่างที่ 11.9 แสดงการใช้ฟังก์ชัน fprintf( )

```
int x=15; float f=4.562; char str[20]="Computer";  
FILE *fptr;  
fprintf(fptr, "%d \t %f \t %s \n",x,f,str);
```

#### โดยที่

fptr คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล (file pointer)

x คือ ตัวแปร int เก็บข้อมูลตัวเลข 15 ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 1

f คือ ตัวแปร float เก็บข้อมูลตัวเลข 4.562 ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 2

str คือ ตัวแปร string เก็บข้อความว่า “Computer” ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 3

\t คือ รหัสควบคุมที่สั่งให้ tab ไป 1 ครั้ง ก่อนที่จะบันทึกข้อมูล field ต่อไป

\n คือ รหัส new line ใช้สั่งให้ขึ้นบรรทัดใหม่ในแฟ้มข้อมูล

### ฟังก์ชัน fscanf( )

เป็นฟังก์ชันที่ใช้อ่านข้อมูล (read) ขึ้นจากแฟ้มข้อมูลแล้วนำมาเก็บไว้ในตัวแปรที่ต้องการได้โดยมีการทำงานคล้ายกับฟังก์ชัน scanf( ) แตกต่างกันตรงที่ฟังก์ชัน fscanf( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลจากแฟ้มข้อมูลแต่ฟังก์ชัน scanf( ) เป็นฟังก์ชันที่รับข้อมูลจากคีย์บอร์ดแล้วไปเก็บไว้ในตัวแปรที่ต้องการ

#### รูปแบบการใช้

fscanf( fp, control string, variable list);

#### โดยที่

fp คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล

control string คือ รหัสรูปแบบข้อมูลใช้เหมือนฟังก์ชัน scanf( ) เช่น สามารถระบุชนิดของข้อมูล

ต้องการอ่านข้อมูลจากแฟ้มเป็น %d,%c,%f,%s และยังสามารถใช้รหัส new line หรือ \n ได้

variable list คือ ชื่อตัวแปรที่ใช้เก็บข้อมูลที่อ่านมาจากแฟ้มข้อมูล โดยจะต้องระบุเครื่องหมาย & (ampersand) นำหน้าชื่อตัวแปรด้วย ยกเว้นตัวแปรสตริงเท่านั้นที่ไม่ต้องมีเครื่องหมาย &

#### ตัวอย่างที่ 11.10 แสดงการใช้ฟังก์ชัน fscanf( )

```
int i; float f; char str[80];
FILE *fptr;
fscanf( fptr, "%d %f %s \n", &i, &f, str );
```

โดยที่

i คือ ตัวแปรชนิด int ใช้เก็บข้อมูลที่อ่านจากแฟ้ม field ที่ 1

f คือ ตัวแปรชนิด float ใช้เก็บข้อมูลที่อ่านจากแฟ้ม field ที่ 2

str คือ ตัวแปรสตริง ใช้เก็บข้อมูลที่อ่านจากแฟ้ม field ที่ 3

fptr คือ file pointer ใช้ชี้ตำแหน่งข้อมูลในแฟ้ม

#### ตัวอย่างที่ 11.11 ตัวอย่างการใช้ fprintf

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<conio.h>
4  int main(void)
5  {
6      FILE *fptr;
7      int x=5;
8      float f=4.5;
9      char str[80]="C and C++ Programming";
10     char fname[80];
11     printf("Enter your file name ( file format is *.txt ) :");
12     gets(fname);
13     if( (fptr=fopen(fname, "w")) == NULL )
14     {
15         printf("Error in open file ");
16         printf("\007");
17         exit(1);
18     }
19     fprintf(fptr, "%d \t %.2f \t %s\n", x, f, str);
20     printf("Successful to write data to file");
21     fclose(fptr);
22     getch();
23     return 0;
24 }
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

จะให้เปิดแฟ้มข้อมูลขึ้นมาเพื่อเขียนโดยให้ผู้ใช้งานตั้งชื่อแฟ้มเอง แต่ให้มีนามสกุล .txt เช่น D:\\test.txt คือ ตั้งชื่อแฟ้ม test.txt เก็บไว้ที่ไดร์ฟ D ถ้าเปิดไม่ได้ก็จะพิมพ์ข้อผิดพลาด ซึ่งถ้าเปิดได้แล้วจะนำค่าของตัวแปร x, f และ ตัวแปร str เขียนลงในแฟ้มข้อมูลดังกล่าว 1 record ตามคำสั่ง ซึ่งการเขียนข้อมูลจะมีการเว้นช่องว่างในแต่ละฟิลด์ เนื่องจากมีการใช้รหัส \t และ \n ทำให้เกิดความสับสนในการอ่านข้อมูล

### ตัวอย่างที่ 11.12 ตัวอย่างการใช้ฟังก์ชัน fprintf

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<conio.h>
4  int main(void)
5  {
6      FILE *fptr;
7      int x=1;
8      float f=8.5;
9      char str[80]="C and C++ Programming";
10     int j;
11     char fname[80];
12     printf("Enter your file name ( file format is *.txt ) :");
13     gets(fname);
14     if( (fptr=fopen(fname, "w")) == NULL )
15     {
16         printf("Error in open file ");
17         printf("\007");
18         exit(1);
19     }
20     for(j=1; j<=5; j++){
21         fprintf(fptr, "%d \t %.2f \t %s\n ", x, f, str);
22         x= x+1;
23         f= f+1.5;
24     }
25     fclose(fptr);
26     getch();
27     return 0;
28 }

```

ผลลัพธ์จากการประมวลผลโปรแกรม

Enter your file name ( file format is \*.txt ):Test1.txt

เมื่อผู้ใช้ป้อนชื่อไฟล์เรียบร้อยแล้วโปรแกรมจะนำผลลัพธ์ที่ได้ไปเก็บในไฟล์ตามที่ผู้ใช้ระบุโดยมีการเก็บดังนี้

1	8.50	C and C++ Programming
2	10.00	C and C++ Programming
3	11.50	C and C++ Programming
4	13.00	C and C++ Programming
5	14.50	C and C++ Programming

### ฟังก์ชัน fwrite( )

ฟังก์ชัน fwrite( ) เป็นฟังก์ชันที่ใช้เก็บข้อมูล ลงแฟ้มโดยที่การเก็บข้อมูลแต่ละครั้ง สามารถกำหนดขนาดของข้อมูลที่ต้องการบันทึกได้

รูปแบบการใช้ fwrite(&var, size, n, fp);

โดยที่

&var คือ ตำแหน่งของตัวแปรที่เก็บข้อมูลที่ต้องการนำไปเก็บไว้ในแฟ้ม

size คือ ขนาดของข้อมูลที่ต้องการ write ลงแฟ้มในแต่ละครั้ง ซึ่งสามารถหาได้จากฟังก์ชัน sizeof(data type); หรือ sizeof(variable name); เช่น sizeof(int); หรือ sizeof(j);

n คือ จำนวนครั้งที่ต้องการ write ข้อมูลลงแฟ้ม

fp คือ ตัวชี้ตำแหน่งข้อมูลในแฟ้ม (file pointer)

ข้อควรจำ ฟังก์ชัน fwrite( ) จะให้ค่าเลขจำนวนเต็มเท่ากับ n (จำนวนครั้งที่ write ข้อมูลลงแฟ้ม) เมื่อไม่เกิดข้อผิดพลาดในการเขียนข้อมูล และจะให้ค่าน้อยกว่า n เมื่อมีข้อผิดพลาดเกิดขึ้น

แสดงการใช้ฟังก์ชัน fwrite( )

เช่น

1)

```
FILE *fptr;
```

```
double x=1.2345678912345;
```

```
fwrite(&x,sizeof(double x),1,fptr);
```

เป็นการบันทึกข้อมูลที่เก็บไว้ในตัวแปร x ลงแฟ้มข้อมูล โดยทำการบันทึก 1 ครั้ง (n=1) ขนาดของข้อมูลที่บันทึกลงแฟ้มในแต่ละครั้งมีขนาด 8 bytes ตามชนิดตัวแปร x

2)

```
FILE *fptr; int j;
```

```
int a[10]={10,20,30,40,50,60,70,80,90,100};
```

```
for (j=0;j<10;j++)
```

```
fwrite(&a[j],sizeof(int a[j]),1,fptr);
```

**ตัวอย่างที่ 11.13** ตัวอย่างการเขียนข้อมูลลงไฟล์

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<conio.h>
4  #include<ctype.h>
5  int main(void)
6  {
7      struct
8      {
9          char name[30];
10         char id[20];
11         float gpa;
12     }student;
13
14     char numstr[80], fname[80];
15     FILE *fptr;
16     printf("Enter your file name (file format is *.txt) :");
17     gets(fname);
18     if( (fptr=fopen(fname, "wb")) == NULL )
19     {
20         printf("Error in open file");
21         exit(1);
22     }
23     do {
24         printf("\nEnter student name :");
25         gets( student.name );
26         printf("Enter student id :");
27         gets( student.id );
28         printf("Enter student gpa :");
29         gets( numstr );
```

30	student.gpa = atof(numstr);
31	fwrite(&student, sizeof(student), 1, fptr);
32	printf("Add another student(y/n)?:");
33	} while( tolower(getche()) == 'y' );
34	fclose(fptr);
35	getch();
36	return 0;
37	}
<b>ผลลัพธ์จากการประมวลผลโปรแกรม</b> Enter your file name (file format is *.txt) :Student.txt  Enter student name :Yaya Enter student id :58001 Enter student gpa :3.89 Add another student(y/n)?:y Enter student name :Nadej Enter student id :58002 Enter student gpa :3.20 Add another student(y/n)?:y Enter student name :Kimberly Enter student id :58003 Enter student gpa :3.54 Add another student(y/n)?:n และนำไปเก็บในแฟ้มข้อมูล Student.txt	

### ฟังก์ชัน fread( )

ฟังก์ชัน fread( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลจากแฟ้ม โดยที่แต่ละครั้งสามารถกำหนดขนาด (size) ของข้อมูลที่ต้องการอ่านได้

รูปแบบการใช้ fread(&var,size,n,fp);

โดยที่

&var คือ ตำแหน่งของตัวแปรที่เก็บข้อมูลที่ต้องการนำไปเก็บไว้ในแฟ้ม

size คือ ขนาดของข้อมูลที่ต้องการ read ขึ้นจากแฟ้มในแต่ละครั้ง ซึ่งสามารถหาได้จากฟังก์ชัน sizeof(data type); หรือ sizeof(variable name); เช่น sizeof(int); หรือ sizeof(j);

n คือ จำนวนครั้งที่ต้องการ read ข้อมูลจากแฟ้ม

fp คือ ตัวชี้ตำแหน่งข้อมูลในแฟ้ม (file pointer)

ข้อควรจำ ฟังก์ชัน fread( ) จะให้ค่าตัวเลขจำนวนเต็ม = n เมื่อไม่เกิดข้อผิดพลาดในการอ่านข้อมูล และจะให้ค่าเป็นศูนย์ (0 หรือ NULL) เมื่อมีข้อผิดพลาดในการอ่านข้อมูลจากแฟ้มหรือสิ้นสุดไฟล์ (EOF)

ตัวอย่าง แสดงการใช้ฟังก์ชัน fread( )

เช่น



1)

```
FILE *fptr;  
double x;  
fread(&x,sizeof(double x),1,fptr);
```

เป็นการอ่านข้อมูลในแฟ้มข้อมูลมาเก็บไว้ที่ตัวแปร x โดยทำการอ่านข้อมูล 1 ครั้ง (n=1) ขนาดของข้อมูล  
ที่อ่านจากแฟ้มในแต่ละครั้งมีขนาด 8 bytes ตามชนิดตัวแปร x

2)

```
FILE *fptr;  
int a[10]; int j;  
for (j=0;j<10;j++)  
fread(&a[j],sizeof(int a[j]),1,fptr);
```

เป็นการอ่านข้อมูลในแฟ้มมาเก็บไว้ในตัวแปรชุด a โดยทำการอ่านข้อมูล 10 ครั้ง โดยที่ขนาดของข้อมูล  
ที่อ่านจากแฟ้มในแต่ละครั้งมีขนาด 2 bytes ตามชนิดตัวแปร a[j]

#### ตัวอย่างที่ 11.14 การอ่านข้อมูลจากไฟล์

```
1  #include<stdio.h>  
2  #include<conio.h>  
3  #include<stdlib.h>  
4  int main(void)  
5  {  
6      struct {  
7          char name[30];  
8          char id[20];  
9          float gpa;  
10     } student;  
11     char fname[80];  
12     FILE *fptr;  
13     printf("Enter your file name (file format is *.txt) :");  
14     gets(fname);  
15     if( (fptr=fopen(fname, "rb")) == NULL )  
16     {  
17         printf("Error in open file");  
18         exit(1);  
19     }  
20     while( fread(&student, sizeof(student),1, fptr) == 1 )  
21     {  
22         printf("\nName = %s\n", student.name);  
23         printf("Id = %s\n", student.id);  
24         printf("GPA = %.2f\n", student.gpa);  
25     }  
26     fclose(fptr);  
27     getch();  
28     return 0;  
29 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Enter your file name (file format is \*.txt) :Student.txt

Name = Yaya

Id = 58001

GPA = 3.89

Name = Nadej

Id = 58002

GPA = 3.20

Name = Kimberry

Id = 58003

GPA = 3.54

## 11.8 ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้มข้อมูล

การควบคุมตำแหน่งของ fp (file pointer) ในแฟ้มข้อมูล นิยมใช้กันมากในการประมวลผลแฟ้มข้อมูลแบบสุ่ม (random file access) ซึ่งสามารถให้ fp ไปยังตำแหน่งเริ่มต้นของแฟ้มข้อมูล (BOF =beginning of file) หรือ ตำแหน่งใดตำแหน่งหนึ่งในแฟ้มข้อมูลได้

ฟังก์ชันที่ใช้ควบคุมตำแหน่ง file pointer มีดังนี้

ฟังก์ชัน rewind( )

เป็นฟังก์ชันที่ใช้ย้ายตำแหน่งของ file pointer ไปยังตำแหน่งเริ่มต้นของแฟ้ม

รูปแบบการใช้ rewind(fp);

ฟังก์ชัน fseek( )

เป็นฟังก์ชันที่ใช้ย้ายตำแหน่งของ file pointer ไปยังตำแหน่งที่ต้องการในแฟ้มข้อมูลโดยจะต้องกำหนดจุดเริ่มต้น (origin) ของ file pointer และค่า offset

รูปแบบการใช้ fseek(fp, offset, origin);

โดยที่ offset คือ ระยะห่างจากตำแหน่งจุดเริ่มต้น มีหน่วยเป็น byte

origin คือ จุดที่ file pointer ชี้อยู่ มีอยู่ 3 สถานะ ดังนี้คือ

SEEK\_SET 0 file pointer อยู่ที่ต้นแฟ้ม BOF

SEEK\_CUR 1 file pointer อยู่ที่ตำแหน่งปัจจุบัน

SEEK\_END 2 file pointer อยู่ที่ท้ายไฟล์ EOF

ตัวอย่าง

แสดงการใช้งาน fseek(fp,10,0);

หรือคำสั่ง fseek(fp,10, SEEK\_SET);

หมายความว่าให้ย้าย file pointer ถัดจากตำแหน่งต้นไฟล์ (BOF) ไปอีก 10 byte

ข้อควรจำ ฟังก์ชัน fseek( ) จะให้ค่าไม่เท่ากับศูนย์ เมื่อไม่สามารถย้าย file pointer ได้

ฟังก์ชัน ftell( )

เป็นฟังก์ชันที่ใช้บอกตำแหน่งของ file pointer ว่าปัจจุบันกำลังอยู่ที่ตำแหน่งใดในแฟ้มข้อมูล โดยฟังก์ชันนี้จะให้ค่ากลับเป็นตัวเลขจำนวนเต็ม

รูปแบบการใช้ ftell( fp );

ตัวอย่าง แสดงการใช้งาน int position;

position = ftell(fp);

printf("Position is %d Byte",position);

**ตัวอย่างที่ 11.15** ตัวอย่างการอ่านข้อมูล ณ ตำแหน่งไบต์ที่ผู้ใช้กำหนด

```
1  #include <stdio.h>
2  void main(){
3      long loc;
4      FILE *Ptr;
5      if ((Ptr=fopen("binary.txt","rb"))==NULL){
6          printf("Error opening file!");
7          return;
8      }
9      printf("Enter byte to seek:");
10     scanf("%d",&loc);
11     if (fseek(Ptr, loc, SEEK_SET)){
12         printf("Error on seeking file!");
13         return;
14     }
15     printf("data at location %ld is %c", loc, getc(Ptr));
16     fclose(Ptr);
17 }
```

**ผลลัพธ์จากการประมวลผลโปรแกรม**

อ่านข้อมูลจากแฟ้มข้อมูล **binary.txt** ที่ตำแหน่งไบต์ที่ต้องการ และแสดงผลข้อมูล ณ ตำแหน่งนั้น ๆ

## แบบฝึกหัดปฏิบัติการคาบที่ 11: FILE

1. จงเขียนโปรแกรมเพื่อบันทึกข้อมูลของดีวีดีเพลงลงไฟล์ โดยในดีวีดีหนึ่งแผ่นจะประกอบด้วยข้อมูลดังต่อไปนี้

รหัส ชื่ออัลบั้ม ชื่อศิลปิน วันเดือนปีที่ออกอัลบั้ม บริษัทผู้ผลิต ราคา

กำหนดให้ผู้ใช้สามารถป้อนชื่อไฟล์ที่ต้องการบันทึกได้ และสามารถบันทึกข้อมูลจนกระทั่งผู้ใช้ป้อน n

ตัวอย่างผลการรันโปรแกรม

```
Welcome to CS-KMUTNB MUSIC Shop
Please enter name of file: music.txt
Please enter the product code: JP001
Please enter the product title: 35xxxv
Please enter the name of artist: ONE OK ROCK
Please enter the issue date: 2015.11.23
Please enter the company: SONY-MUSIC-JAPAN
Please enter the price: 500
Do you want to continue ('y/n'): y

Please enter the product code: JP002
Please enter the product title: Koko ga Rhodes da, Koko de Tobe
Please enter the name of artist: AKB48
Please enter the issue date: 2015.01.21
Please enter the company: AKS
Please enter the price: 500
Do you want to continue ('y/n'): y

Please enter the product code: JP003
Please enter the product title: Kiss Kiss Kiss
Please enter the name of artist: KAT-TUN
Please enter the issue date: 2015.03.11
Please enter the company: Johnny Entertainment
Please enter the price: 500
Do you want to continue ('y/n'): n
THANK YOU.
THE CS-KMUTNB MUSIC Shop IS CLOSING.
NOW, WE ARE WRITING THE REMAINING GOODS FOR TOMORROW!
```

2. สร้างไฟล์ input.txt ซึ่งมีข้อมูลภายในเป็นรหัสสินค้า ชื่อสินค้า จำนวนสินค้าดังนี้

100	Pencil	25.0	20
101	Pen	90.0	5
103	Eraser	12.0	10

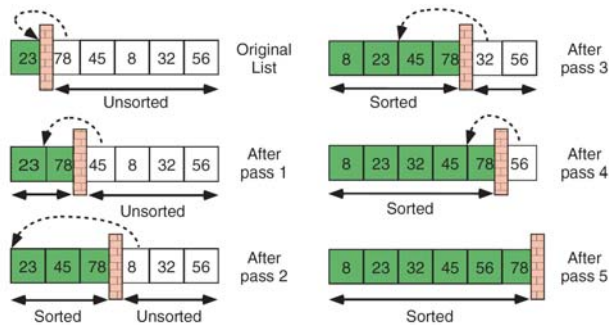
จงเขียนโปรแกรมอ่านไฟล์นี้มาเก็บในอาร์เรย์ของ code name price และ quantity โดยให้ชื่อสินค้ามีความยาวไม่เกิน 15 ตัวอักษร จากนั้นให้ผู้ใช้เลือกสินค้าที่ต้องการพร้อมระบุจำนวนที่ต้องซื้อ จากนั้นให้โปรแกรมทำการคำนวณราคา พร้อมบันทึกข้อมูลสินค้าคงเหลือลงในไฟล์

ตัวอย่างผลการรันโปรแกรม

```
Welcome to KMUTNB Shop
The list of product:
100 Pencil 25.0 20
101 Pen 90.0 5
103 Eraser 12.0 10
Please enter the product code: 101
You have chosen Pen with Price 90.00.
```

How many? 2  
 Your total price is 180  
 THANK YOU.  
 Do you want to continue ('y/n'): n  
 THE SHOP IS CLOSING.  
 NOW, WE ARE WRITING THE REMAINING QTY OF GOODS FOR TOMORROW!

### 3. ให้ศึกษารูปแบบของการเรียงลำดับด้วยวิธี Insertion sort ต่อไปนี้



```

1  /* ===== insertionSort =====
2  Sort list using Insertion Sort. The list is divided
3  into sorted and unsorted lists. With each pass, first
4  element in unsorted list is inserted into sorted list.
5  Pre list must contain at least one element
6  last contains index to last element in list
7  Post list has been rearranged
8  */
9  void insertionSort (int list[], int last)
10 {
11 // Local Declarations
12 int walk;
13 int temp;
14 bool located;
15
16 // Statements
17 // Outer loop
18 for (int current = 1; current <= last; current++)
19 {
20 // Inner loop: Select and move one element
21 located = false;
22 temp = list[current];
23 for (walk = current - 1; walk >= 0 && !located;)
24 if (temp < list[walk])
25 {
26 list[walk + 1] = list[walk];
27 walk--;
28 } // if
29 else
30 located = true;
31 list [walk + 1] = temp;
32 } // for
33 return;
34 } // insertionSort

```

ให้อ่านข้อมูลต่อไปนีเพื่อทำการเรียงลำดับโดยวิธี Insertion sort จากมากไปน้อยแล้วทำการบันทึกข้อมูลที่เรียงลำดับแล้วลงในไฟล์ตามที่ใช้ระบุ

#### ตัวอย่างผลการรันโปรแกรม

Please enter number of digit for sorting: 10

```

1 2 3 4 5 6 7 8 9 10
Please enter file name to save: random.txt
-----
Now, you have the following sorted number:
10 9 8 7 6 5 4 3 2 1
Please enter file name: sorted.txt
FILE IS ALREADY SAVED.

```

4. จงเขียนโปรแกรมเพื่ออ่านไฟล์ เพื่อนับจำนวนสระแล้วแสดงผลจำนวนสระแต่ละตัวในไฟล์ใหม่ โดยในไฟล์ที่เก็บคำตอบให้เก็บในรูปแบบดังนี้ หลังจากนั้นให้ดึงข้อมูลในไฟล์ที่บันทึกดังกล่าวขึ้นมาแสดงผล

```

a: 10
e: 0
i: 5
o: 2
u: 0

```

ตัวอย่างผลการรันโปรแกรม

```

Please enter file name to read: random1.txt
Please enter file name to save: save.txt
-----
FILE IS ALREADY SAVED.
-----
Number of Vowels in random1.txt are
a: 10
e: 0
i: 5
o: 2
u: 0

```

5. จงเขียนโปรแกรมเพื่อเก็บข้อมูลของคนจำนวน 10 คน โดยเก็บชื่อ นามสกุล อายุ อาชีพ เพศ ที่อยู่ในไฟล์ แล้วค้นหาคนที่มียุมากกว่า 20 ปีจากข้อมูลในไฟล์ที่เก็บไว้ แล้วแสดงผล

ตัวอย่างผลการรันโปรแกรม

```

Please enter name of file: person.txt
Please enter ID: 001
Please enter name: Yaya
Please enter surname: Sperbund
Please enter occupation: Actress
Please enter age: 20
Please enter sex: F
Please enter address: 1/268 Dusit Bangkok THAILAND
Do you want to continue ('y/n'): y
Please enter ID: 002
Please enter name: Kimberly
Please enter surname: Sperbund
Please enter occupation: Actress
Please enter age: 21
Please enter sex: F
Please enter address: 2/268 Dusit Bangkok THAILAND
Do you want to continue ('y/n'): n
-----
The list of person whose age over 20 are
001 Yaya Sperbund Actress 20 F
002 Kimberly Sperbund Actress 21 F
-----

```

## บทที่ 12

### การโปรแกรมเชิงวัตถุเบื้องต้น

#### (Introduction to Object Oriented Programming)

##### จุดประสงค์

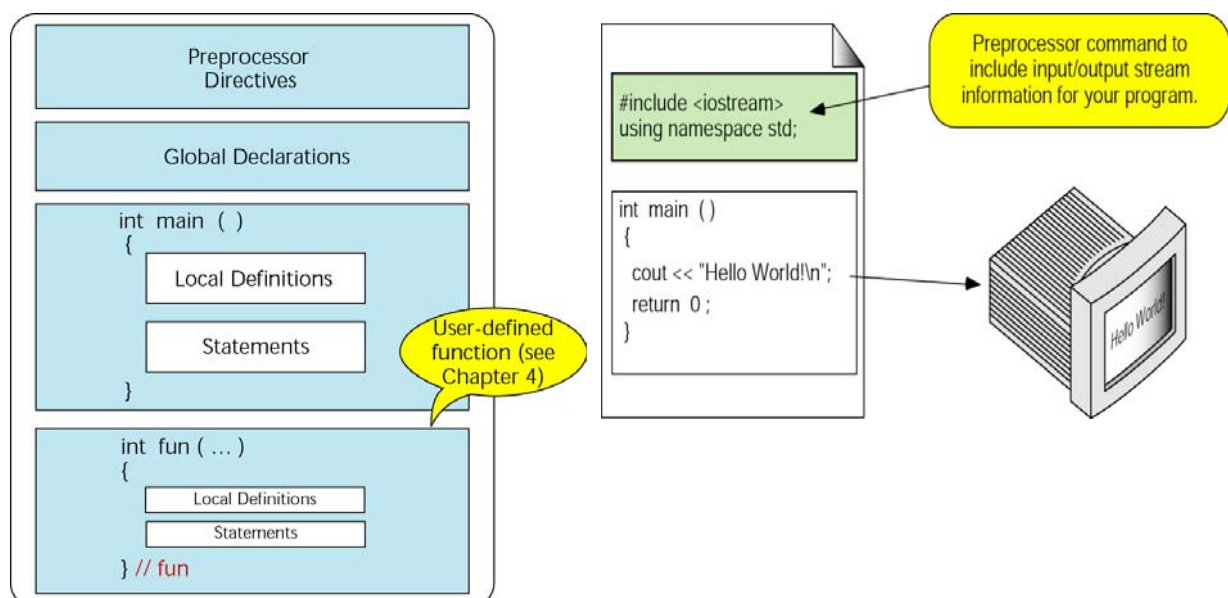
1. เพื่อให้นักศึกษาทราบหลักการการโปรแกรมเชิงวัตถุเบื้องต้น
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมเชิงวัตถุเบื้องต้นได้

#### 12.1 ความรู้เบื้องต้นเกี่ยวกับภาษา C++

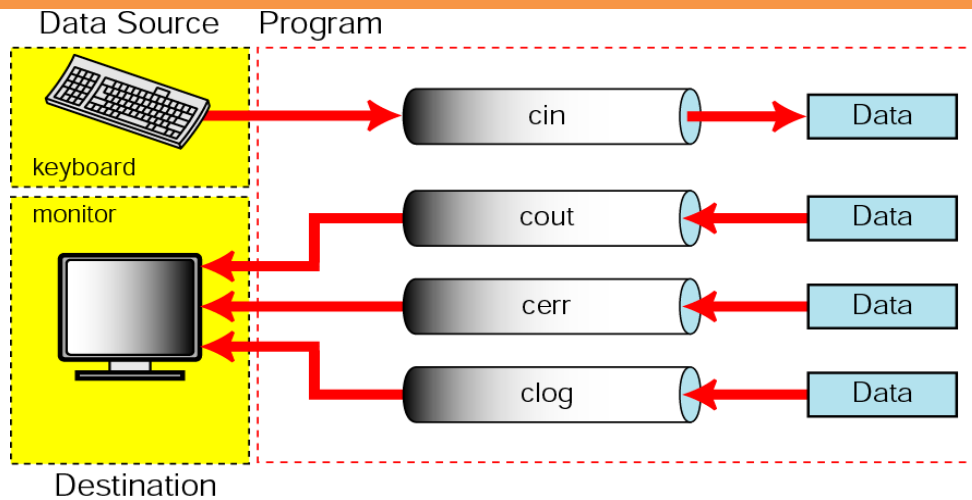
ภาษา C++ เป็นภาษาที่ได้รับการพัฒนา ณ ห้องปฏิบัติการเบลล์ โดย สโตรสตรัพ(Bjarne Stroustrup) ใน ค.ศ. 1980 เพื่อสนับสนุนงานวิจัยจำลองเหตุการณ์ของเขาโดยได้ออกแบบภาษา C++ ให้มีความสามารถสนับสนุนการเขียนโปรแกรมเชิงวัตถุ ในขณะเดียวกันก็ยังคงความสามารถครอบคลุมภาษา C ทั้งหมด ภาษา C++ จึงถือว่าเป็นซูเปอร์เซตของภาษา C โปรแกรมที่เขียนด้วยภาษา C จึงสามารถ ผ่านการแปลโปรแกรมด้วย C++ ได้ แต่อย่างไรก็ตามอาจจะมีการแก้ไขบ้างเล็กน้อย

คุณสมบัติของ C++ สนับสนุนการนิยามคลาสของวัตถุ การสืบทอด การทำ Dynamic binding

#### 12.2 Structure of a C++ Program



## 12.3 การนำเข้าและแสดงผลข้อมูลใน C++



การเขียนโปรแกรมรับและแสดงผลในภาษา C++ บนอุปกรณ์มาตรฐาน สามารถเขียนได้โดยใช้รูปแบบเช่นเดียวกับภาษา C เนื่องจากตัวภาษา C++ เองได้นำเอารูปแบบหรือความสามารถของ C ทั้งหมดมารวมเอาไว้ด้วยกัน สิ่งที่เพิ่มเติมจากการเขียนโดยใช้ภาษา C คือการใช้คลาส iostream ซึ่งเป็นคลาสที่ทำหน้าที่เป็นตัวกลางระหว่าง สตรีม กับอุปกรณ์มาตรฐาน (stream=พื้นที่บริเวณใดๆ ในหน่วยความจำ สำหรับเก็บข้อมูลที่จะรับหรือส่งไปมาระหว่างอุปกรณ์มาตรฐานในลักษณะต่อเนื่องกันเป็นสายตามลำดับการเกิด) การใช้คลาส stream ทุกครั้งจะต้องมีการประกาศ `#include <iostream.h>` ก่อนเสมอ โดยเป็นคลาสที่เก็บอยู่ในแฟ้ม iostream.h ซึ่งในแฟ้มนี้จะมีคลาสต่างๆ ที่เกี่ยวข้องกับการรับข้อมูล และการส่งข้อมูลไปมาระหว่าง สตรีมกับอุปกรณ์มาตรฐานได้แก่คลาส ios คลาส istream คลาส ostream คลาส iostream ความสัมพันธ์ระหว่างคลาสดังกล่าวคือ คลาส istream สืบทอดมาจาก ios คลาส ostream สืบทอดจาก ios คลาส iostream สืบทอดมาจากคลาส istream ostream การรับส่งข้อมูลจะเกี่ยวกับวัตถุ cin cout cerr clog โดยแต่ละ object มีลักษณะดังนี้

cin เป็นวัตถุของคลาส istream\_withassign ซึ่งทำหน้าที่เกี่ยวกับส่วนนำเข้ามาตรฐาน เพื่อรับข้อมูลจากอุปกรณ์มาตรฐานมาเก็บไว้ใน stream แล้วนำจาก stream ไปเก็บไว้ในตัวแปร

cout เป็นวัตถุของคลาส ostream\_withassign ซึ่งทำหน้าที่เกี่ยวกับส่วนแสดงผลมาตรฐาน เพื่อนำเอาค่าข้อมูลที่จะแสดงใส่ลงไปใน stream แล้วนำจาก stream ส่งต่อไปยังอุปกรณ์มาตรฐานอีกทีหนึ่ง

cerr เป็นวัตถุที่ทำหน้าที่เกี่ยวกับส่วนแสดงข้อผิดพลาดมาตรฐานที่ต้องใช้บัฟเฟอร์

clog เป็นวัตถุที่ทำหน้าที่เกี่ยวกับส่วนแสดงข้อผิดพลาดมาตรฐาน ที่ไม่ต้องใช้ buffer

### 12.3.1 คำสั่งการนำเข้า และ แสดงผล

คำสั่ง cin เป็นคำสั่งที่รับข้อมูลผ่านทางแป้นพิมพ์เข้าไปใช้งานในโปรแกรม มีรูปแบบคำสั่งดังนี้

```
cin >> variable;
```

คำสั่ง cout เป็นคำสั่งในการแสดงค่าออกทางจอภาพ มีรูปแบบคำสั่งดังนี้



```
cout << variable;
```

### ตัวอย่างที่ 12.1 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อแสดงข้อความทางหน้าจอ

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {   char letter1, letter2;
5      string lastName;
6      cout << "Enter 2 initials and last name: ";
7      cin >> letter1 >> letter2 >> lastName;
8      cout << "Hello " << letter1 << ". " << letter2 << ". "
9          << lastName << "! ";
10     cout << "We hope you enjoy studying C++." << endl;
11     system("PAUSE");
12     return 0;
13 }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Enter 2 initials and last name: SP Prasomphan

Hello S. P. Prasomphan! We hope you enjoy studying C++.

### ตัวอย่างที่ 12.2 ตัวอย่างการแสดงข้อความทางหน้าจอ

```
1  #include <iostream>
2  using namespace std;
3  int main ()
4  {
5      //Statements
6      cout << 24      << "\n";           // Print an integer
7      cout << 12.3    << "\n";           // Print a float
8      cout << 'A'     << "\n";           // Print a character
9      cout << "Hello" << "\n";           // Print a string
10     system("PAUSE");
11     return 0;
12 } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

24

12.3

A

Hello

#### 1.2.2 ฟังก์ชันสมาชิกของวัตถุ cout

ภายในวัตถุ cout จะมีฟังก์ชันสำหรับการทำงาน ดังนี้

flush() ทำหน้าที่กวาดข้อมูลจากสตริมที่ค้างอยู่ทั้งหมดส่งไปให้อุปกรณ์ส่งออกมาตราฐาน เช่น

cout<<flush; หรือ cout.flush();

put() ทำหน้าที่ส่งข้อมูลไปยังสตริมเพื่อส่งต่อออกไปยังอุปกรณ์ส่งออกมาตราฐานครั้งละ 1 อักขระ เช่น

```
cout<<'\\n'; หรือ cout.put('\\n');
write() ทำหน้าที่ส่งข้อความไปให้สตรีนเพื่อส่งต่อไปยังอุปกรณ์มาตรฐาน จำนวนที่ส่งไปแสดงจะ
เท่ากับขนาดที่ระบุอาร์กิวเมนต์ตัวที่ 2 ตัวอย่างการใช้งานเช่น
char temp [] ="string test";
cout.write(temp,1);
```

## 12.4 แนวความคิดเชิงวัตถุ

ภาษา C++ เป็นภาษาเชิงวัตถุเพราะสามารถสนับสนุนหลักการเขียนโปรแกรมเชิงวัตถุซึ่งประกอบไปด้วย

1. การกำหนดข้อมูลนามธรรม(Abstraction)
2. การทำให้ข้อมูลอยู่ในขอบเขตจำกัด (Encapsulation)
3. การสืบทอด(Inheritance)
4. การทำหลายรูปแบบ(Polymorphism)

### 12.3.1 นามธรรม(Abstraction)

การแก้ปัญหาโดยพิจารณาตามหลักการเขียนข้อมูลเชิงวัตถุ จะต้องตีความหมายโจทย์หรือ นามธรรมเป็นการกำหนด Model ของโดเมนของปัญหา คือ เป็นการแทนปัญหาต่าง ๆ ให้อยู่ในลักษณะที่สามารถอธิบายเป็นแบบแผนหรือโมเดลได้ การเขียนนิยามปัญหาตามกรรมวิธีเชิงวัตถุจะใกล้เคียงกับความเป็นจริงมากที่สุด โดยเป็นการนำเอาเรื่องของพฤติกรรมหรือคุณสมบัติของวัตถุแต่ละชนิดมารวมกัน โดยการแทนที่ลักษณะนี้จะอยู่ในรูปของคลาส (class)

### 12.3.2 การทำให้ข้อมูลอยู่ในขอบเขตจำกัด (Encapsulation)

คือ การซ่อนข้อมูล (data hiding) และ พฤติกรรมต่าง ๆ ของวัตถุเอาไว้ภายใน โดยอาศัยหลักการเข้าถึง (access) โดยมีตัวระบุการเข้าถึงของข้อมูลเป็นตัวกำหนดลักษณะการเข้าถึง ประกอบไปด้วย 3 ตัว คือ

- Private access
- Protected access
- Public access

โดยมีหลักการว่า การเข้าถึงหรือการทำอะไรกับ Attribute ของวัตถุจะต้องเรียกผ่าน method ของวัตถุเท่านั้น ไม่สามารถเข้าถึงจากภายนอกโดยตรงได้ ดังนั้นจึงเปรียบเสมือนมีสิ่งมาห่อหุ้มวัตถุเอาไว้ วิธีการแบบนี้จะช่วยให้ข้อมูลมีความปลอดภัย ถ้ามีข้อผิดพลาดเกิดขึ้นนักเขียนโปรแกรมสามารถหาจุดที่เป็นสาเหตุของการเกิดข้อผิดพลาดได้ คุณสมบัติแบบนี้ทำให้วัตถุมีลักษณะของกล่องดำ คือ มีการปิดบังรายละเอียดภายในเอาไว้ ผู้ใช้เพียงรู้วิธีการติดต่อกับวัตถุนั้นว่าทำได้อย่างไรก็เพียงพอ ไม่จำเป็นต้องรู้วิธีการทำงานภายใน

### 12.3.3 การสืบทอด(Inheritance)

การสืบทอดเป็นคุณสมบัติที่สำคัญในการเขียนโปรแกรมในเชิงวัตถุหลังจากที่เราได้นิยามปัญหาในรูปแบบของคลาสแล้ว คลาสที่สร้างใหม่สามารถนำกลับมาใช้ได้ อีก การสืบทอดจะสืบทอดทั้งคุณสมบัติและวิธีการดำเนินงานให้กับคลาสใหม่ คลาสต้นแบบที่จะให้คลาสอื่นมาสืบทอดว่า คลาสพ่อ หรือคลาสบรรพบุรุษ

(Ascendant) คลาสที่สืบทอดมาจากคลาสอื่นเรียกว่า คลาสลูก (descendant) การนำ Code ที่ใช้แล้วกลับมาใช้อีกครั้ง จะพิจารณาได้เป็น 2 กรณี

- 1.การสืบทอดจากคลาสหลัก(Base class) มาใช้ หรือเรียกว่า Inheritance ส่วนคลาสที่สืบทอดมาเรียกว่า คลาสสืบทอด (Derived class) เช่น โปรแกรมวาดรูปหนึ่งสามารถวาดรูปได้หลายแบบ เช่น สี่เหลี่ยม วงกลม เส้นตรง เราสามารถสร้างคลาสรูปภาพ Picture ซึ่งภายในประกอบด้วย ตำแหน่งพิกัดที่จะวาด เช่น x y color เป็นคลาสหลัก หลังจากนั้นเราสามารถสืบทอดคลาสดังกล่าวเป็นคลาสวงกลม หรือคลาสเส้นตรง คลาสสี่เหลี่ยม โดยภายในมีลักษณะเหมือนกับคลาสรูปภาพได้
- 2.การนำ class มาเป็น object ของ คลาสใหม่ (Composition) เช่น หุ่นยนต์ประกอบด้วย แขนขา ตา ดังนั้น คลาส Robot จึงประกอบไปด้วยคลาสArms คลาส Legs คลาสEyes คุณสมบัตินี้มีข้อดีคือ ช่วยลดระยะเวลาในการเขียนโปรแกรม และลดจำนวนคำสั่งได้เป็นอันมาก แต่การสืบทอดไม่ควรมีหลายระดับ

#### 12.3.4 การทำหลายรูปแบบ (Polymorphism)

เป็นคุณสมบัติที่ระบบมีความสามารถในการตัดสินใจได้ว่าจะเลือกทำวิธีการดำเนินการของวัตถุใดในระหว่างการประมวลผล เช่น คลาสShape เป็นคลาสรูปร่าง มีวิธีการดำเนินการ calculateArea () สำหรับคำนวณพื้นที่ คลาสนี้มีคลาสลูกคือ คลาส Circle คลาส Square โดยวัตถุของคลาสทั้งสองมีวิธีการดำเนินการเหมือนกันคือ calculateArea () แต่มีวิธีการทำงานต่างกัน ดังนั้นเมื่อมีการเรียกใช้ผ่านวัตถุของคลาส Shape จะมีการตัดสินใจว่าจะเลือก calculateArea () ว่าเป็นของวัตถุใดในการประมวลผล เราเรียกรูปแบบการทำงานแบบนี้โดยอาศัยหลักการ Late binding หรือเรียกอีกอย่างว่า dynamic binding การทำหลายรูปแบบด้วยภาษา C++ บางครั้งจะต้องทำให้คลาสหลักเป็น Abstract class ซึ่งจะไม่สามารถสร้าง object ได้ โดยการสร้างคำสงวน virtual

#### แนวความคิดเชิงวัตถุ (Object-Oriented Programming Concept)

เป็นการเขียนโปรแกรมเพื่อให้เป็นไปตามหลักการเชิงวัตถุโดยผู้เขียนโปรแกรมต้องพยายามมองรูปแบบวัตถุให้ออกก่อน ถ้าต้องการสร้างจักรยานขึ้นมาสักคัน ถ้าคิดในแง่อุตสาหกรรมจะต้องมีขบวนการออกแบบตัวจักรยาน ผู้ออกแบบต้องวาดแบบจักรยานลงกระดาษเขียนแบบมีการกำหนดขนาด มีคำอธิบายลักษณะจักรยาน จนได้แบบที่ดีที่สุดของจักรยาน จากนั้นนำไปทำสำเนาทำเป็นแบบพิมพ์เขียว ซึ่งใช้สำหรับนำไปสร้างตัวจักรยาน โดยทีมงานผู้สร้าง เมื่อผู้สร้างอ่านแบบพิมพ์เขียวจะรู้ทันทีว่าผู้ออกแบบต้องการจักรยานมีหน้าตาเป็นอย่างไรจากนั้นก็จัดสร้างตามแบบเป็นจักรยานตามจำนวนที่ต้องการ เช่น 10 100 หรือ 500 คัน เป็นต้น

#### คลาส

การเขียนโปรแกรมเชิงวัตถุใน C++ คลาสถือว่าเป็นหัวใจสำคัญ โดยจะนิยามคลาสเป็นชนิดข้อมูลนามธรรม ซึ่งถือว่าเป็นชนิดข้อมูลแบบหนึ่งที่มีคุณสมบัติและวิธีการดำเนินงานอยู่ภายใน คำศัพท์เฉพาะของคลาสประกอบด้วย

- คุณสมบัติของคลาสแทนด้วย สมาชิกข้อมูล(Data member)
- วิธีการดำเนินการ แทนด้วย ฟังก์ชันสมาชิก(Member function)

- คลาสลูก แทนด้วย คลาสอนุพันธ์(Derived class)
- คลาสพ่อ หรือ คลาสบรรพบุรุษ แทนด้วย คลาสฐาน(Base class)

#### ความหมาย

คลาสเป็นคำที่ใช้แทนกลุ่มของวัตถุที่มีคุณสมบัติ และพฤติกรรมที่จัดแล้วว่าอยู่ในกลุ่มเดียวกัน สิ่งที่เกิดขึ้นให้วัตถุทำงานเรียกว่า Message หรือข่าวสาร พฤติกรรมที่เขียนเป็นขั้นตอนที่อธิบายถึงการตอบสนองของวัตถุต่อสิ่งเร้าเรียกว่า ฟังก์ชันสมาชิก (Member function) ชื่อของฟังก์ชันสมาชิกและอาร์กิวเมนต์ที่ส่งไปให้ให้วัตถุถือเป็นข่าวสารอันหนึ่ง คลาสถือเป็นแม่แบบที่จะใช้สร้างวัตถุ (object) ใดๆ วัตถุที่เป็นสมาชิกของคลาสเดียวกันจะมีโครงสร้างทั่ว ๆ ไปเหมือนกัน ดังนั้นเมื่อเอาคลาสเขียนโปรแกรมเชิงวัตถุก็เปรียบเสมือนกับชนิดข้อมูลพิเศษชนิดหนึ่งนั่นเอง

#### ตัวอย่างที่ 12.3 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

1	<code>#include &lt;iostream&gt;</code>
2	<code>using namespace std;</code>
3	<code>class Fraction</code>
4	<code>{</code>
5	<code>private:</code>
6	<code>int numerator;</code>
7	<code>int denominator;</code>
8	<code>public:</code>
9	<code>void store (int numer, int denom);</code>
10	<code>void print ();</code>
11	<code>}; // Fraction</code>
12	<code>void Fraction :: store (int numer, int denom)</code>
13	<code>{</code>
14	<code>numerator = numer;</code>
15	<code>denominator = denom;</code>
16	<code>return;</code>
17	<code>} // Fraction store</code>
18	<code>void Fraction :: print()</code>
19	<code>{</code>
20	<code>cout &lt;&lt; numerator &lt;&lt; "/" &lt;&lt; denominator;</code>
21	<code>return;</code>
22	<code>} // Fraction print</code>
ผลลัพธ์จากการประมวลผลโปรแกรม	
คลาส Fraction	

## วัตถุ(Class object)

การสร้างวัตถุจากคลาสเรียกว่า Instantiation สิ่งที่ได้จากการสร้างของคลาสว่า instance ของคลาส กลุ่มของ instance เรียกว่า object หรือวัตถุ เพราะฉะนั้นจึงถือว่า วัตถุเป็นสมาชิกตัวหนึ่งภายใต้คลาส เช่น จากคลาส Fraction หากต้องการสร้างวัตถุเพื่อนำไปใช้งานเรามีวิธีการสร้างคล้ายกับการประกาศตัวแปรธรรมดาทั่วไป คือ

```
Fraction fr;
```

วิธีการเรียกใช้ฟังก์ชันสมาชิกหรือข้อมูลมีรูปแบบในการเรียกใช้คือ

```
objected.memberID;
```

เช่น

```
Fraction fr;  
fr.store(3,4);  
fr.print();
```

หรือหากต้องการระบุให้ชัดเจนว่าฟังก์ชันสมาชิกเป็นของคลาสใด อาจใช้

```
fr.Fraction::print();
```

### ตัวอย่างที่ 12.4 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

```
1  #include <iostream>  
2  using namespace std;  
3  class Fraction  
4  {  
5      private:  
6          int numerator;  
7          int denominator;  
8      public:  
9          void store(int numer, int denom);  
10         void print();  
11     }; // Fraction  
12     void Fraction :: store (int numer, int denom)  
13     {  
14         numerator    = numer;  
15         denominator = denom;  
16         return;  
17     } // Fraction store  
18     void Fraction :: print()  
19     {  
20         cout << numerator << "/" << denominator;  
21         return;  
22     } // Fraction print  
23     int main ()  
24     {  
25         Fraction fr;  
26         fr.store(3,4);  
27         fr.print();  
28         system("PAUSE");  
29         return 0;  
30     } // main
```

ผลลัพธ์จากการประมวลผลโปรแกรม

3/4

### ตัวอย่างที่ 12.5 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

```
1  #include <iostream>
2  using namespace std;
3  void getData (int& numer, int& denom);
4  class Fraction
5  {
6      private:
7          int numerator;
8          int denominator;
9      public:
10         void store(int numer, int denom);
11         void print();
12     }; // Fraction
13     void Fraction :: store (int numer, int denom)
14     {
15         numerator    = numer;
16         denominator  = denom;
17         return;
18     } // Fraction store
19     void Fraction :: print()
20     {
21         cout << numerator << "/" << denominator;
22         return;
23     } // Fraction print
24     int main ()
25     {
26         cout << "This program creates a fraction\n\n";
27         int numer;
28         int denom;
29         getData (numer, denom);
30         Fraction fr;
31         fr.store(numer,denom);
32         fr.print();
33         system("PAUSE");
34         return 0;
35     } // main
36     void getData (int& numer, int& denom)
37     {
38         cout<<"please enter the numerator:";
39         cin >> numer;
40         cout<<"please enter the denomirator:";
41         cin >> denom;
42         return;
43     }
```

ผลลัพธ์จากการประมวลผลโปรแกรม

This program creates a fraction

please enter the numerator:5

please enter the denomirator:6

5/6

Press any key to continue . . .

## คอนสตรัคเตอร์(Constructor)

เป็นฟังก์ชันสมาชิกที่มีชื่อเดียวกับชื่อของคลาส และเป็นฟังก์ชันสมาชิกที่ไม่มีการคืนค่า constructor จะทำงานโดยอัตโนมัติทันทีหลังจากสร้างวัตถุเสร็จ หน้าที่ของ Constructor คือ การกำหนดค่าเริ่มต้นให้สมาชิกของวัตถุ วิธีการเช่นนี้จะช่วยรับประกันได้ว่าสมาชิกบางตัวของวัตถุได้มีการกำหนดค่าไว้เรียบร้อยแล้ว รูปแบบของ Constructor โดยทั่ว ๆ ไป จะมีลักษณะ 2 ประการ คือ

1. ชื่อของ Constructor จะเป็นชื่อเดียวกับชื่อของคลาส
2. ไม่มีการคืนค่ากลับ ไม่ต้องการประกาศ return type ยกเว้น void

รูปแบบคือ

```
className(parameter list); //declaration
className::className (parameter list) //definition
{
    ...
} //class name
```

## ดีสทริกเตอร์(Destructor)

มีลักษณะการทำงานตรงกันข้ามกับคอนสตรัคเตอร์ คือ จะทำงานก่อนที่วัตถุจะถูกทำลาย และเป็นฟังก์ชันที่ไม่มีการคืนค่า การเขียนนิยามคลาสอาจจะมี destructor หรือไม่มีก็ได้ แต่มีข้อแตกต่างจากคอนสตรัคเตอร์ คือ การเขียนฟังก์ชันสมาชิกของ destructor จะเขียนได้อันเดียวเท่านั้น ซึ่งตรงกันข้ามกับคอนสตรัคเตอร์จะเขียนกี่ฟังก์ชันก็ได้ โดยปกติมักจะใช้ destructor เพื่อคืนค่าหน่วยความจำที่จองในระหว่างประมวลผลเพื่อคืนค่าให้แก่ระบบ รูปแบบของการเขียน คือ

```
~destructorName ( )
{
    ...
}
```

## ฟังก์ชันสมาชิก

วิธีการเขียนฟังก์ชันสมาชิกของคลาส สามารถเขียนได้ในรูปแบบต่อไปนี้

รูปแบบที่ 1

```
class <className>{
    ...
    <return-type><functionName>([arg1[,arg2,...]]){
        // รายละเอียดฟังก์ชันสมาชิก
    }
}
```

รูปแบบที่ 2

```
class <className>{
    ...
    <return-type><functionName>([arg1[,arg2,...]])[const];
    ...
};
<return-type><className>::<functionName>([arg1[,arg2,...]]) {
    // รายละเอียดฟังก์ชันสมาชิก
}
```

รูปแบบที่ 1 แสดงการเขียนฟังก์ชันสมาชิกแบบภายในตัวคลาสเอง (inline) การเขียนนิยามแบบนี้จะเขียนภายในนิยามของคลาสทั้งหมด

รูปแบบที่ 2 จะเขียนเพียงต้นแบบของฟังก์ชันในนิยามของคลาสเท่านั้น ส่วนนิยามของฟังก์ชันที่จะนำมาเขียนเอาไว้ภายนอก

รูปแบบการเขียนต้นแบบของฟังก์ชันภายนอกวงเล็บของคลาส มีรูปแบบดังนี้

```
<return-type><className>::<functionName>([arg1,arg2,...])
```

การอ้างถึงสมาชิกของวัตถุ

objectName.dataName;

objectName.functionName([arg1,arg2,...]);

objectPointer->dataName;

objectPointer-> functionName([arg1,arg2,...]);

objectName หมายถึง วัตถุ(ตัวแปร)

dataName แทนสมาชิกข้อมูลของวัตถุ

functionName คือ ชื่อฟังก์ชันในวัตถุนั้น arg1 เป็นอาร์กิวเมนต์ ถ้ามีมากกว่าหนึ่งตัวให้คั่นด้วย

เครื่องหมายจุลภาค การใส่อาร์กิวเมนต์ต้องสอดคล้องกับฟังก์ชันสมาชิกที่นิยามไว้ในคลาส ตัว

ดำเนินการอ้างสมาชิก ‘.’ เป็นตัวบ่งบอกว่าเป็นสมาชิกของวัตถุ objectName

objectPointer ตัวชี้วัตถุต้องใช้เครื่องหมาย “->” แทนการใช้จุด เพื่อบอกว่าเป็นสมาชิกของวัตถุที่

objectPointer ชี้ ข้อสำคัญการอ้างถึงทั้งสองแบบนี้มักจะใช้เรียกสมาชิกของวัตถุภายนอกนิยามของคลาส

### ฟังก์ชันโอเวอร์โหลดดิ้ง (Function overloading)

การเขียนโปรแกรมเชิงวัตถุโดยส่วนใหญ่จะมีคุณสมบัติการทำโอเวอร์โหลดดิ้ง โดยเฉพาะกับฟังก์ชัน คุณสมบัติโอเวอร์โหลดดิ้งของฟังก์ชัน การที่ภาษานั้น ๆ อนุญาตให้โปรแกรมเมอร์สามารถตั้งชื่อซ้ำกันได้ ตัวแปลภาษาจะทำหน้าที่พิจารณาความแตกต่างของฟังก์ชันเอง การพิจารณาว่าฟังก์ชันใดมีความแตกต่างจากฟังก์ชันอื่นหรือไม่ ทำได้ด้วยการนำเอาชื่อของฟังก์ชัน และชนิดข้อมูลของอาร์กิวเมนต์มาพิจารณาร่วมกันด้วย ถ้ามีชื่อต่างกันก็ถือว่าเป็นอีก 1 ฟังก์ชัน แต่ถ้าชื่อซ้ำกันก็พิจารณาต่อไปว่าจำนวน และชนิดข้อมูลของอาร์กิวเมนต์มีความแตกต่างกันหรือไม่ ถ้าใช่แสดงว่าเป็นอีกฟังก์ชันหนึ่งที่ไม่ซ้ำกับฟังก์ชันอื่นถึงแม้ว่าชื่อของฟังก์ชันอาจซ้ำกันก็ตาม

### ตัวอย่างที่ 12.6 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

```
1  #include <iostream>
2  using namespace std;
3  class Funny
4  {
5      private:
6          int num;
7      public:
8          Funny (int x); // A constructor
9  }; // Funny
10 Funny :: Funny (int x)
11 {
12     num = x;
```



```

13 } // Initializtion Constructor
14 int main ()
15 {
16     Funny funny1 (10); // Constructor called
17     Funny funny2; // Error: no default constructor
18     return 0 ;
19 } // main

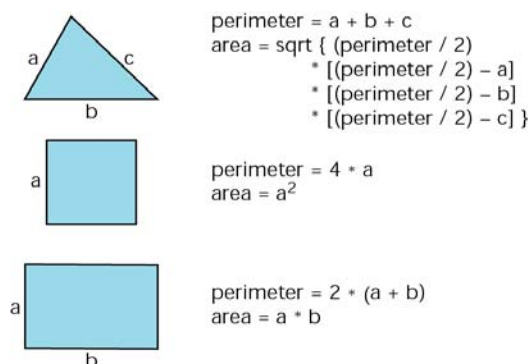
```

ผลลัพธ์จากการประมวลผลโปรแกรม

Error: no default constructor

## การสืบทอด

สิ่งสำคัญประการหนึ่งที่จะทำให้คลาสมีประสิทธิภาพมากยิ่งขึ้นคือการที่จะสามารถขยายคลาสเพื่อที่จะสร้างคลาสใหม่ได้ ในขณะที่เดียวกันคลาสใหม่นั้นก็ยังคุณสมบัติคลาสเดิมทุกประการได้ ซึ่งเรียกว่าการสืบทอด (Inheritance) มีคำศัพท์ที่เกี่ยวข้องกับการสืบทอด อยู่ 2 คำ คือ คลาสที่ทำหน้าที่เป็นต้นแบบให้คลาสอื่นที่สืบทอดเรียกว่า คลาสฐาน หรือ Base class สำหรับคลาสที่สืบทอดมาจากคลาสดั้งเดิมเรียกว่า คลาสอนุพันธ์ (Derived class) เป็นคลาสที่สร้างขึ้นโดยคงคุณสมบัติของคลาสเดิมเอาไว้ทุกประการ และอาจจะมี Attribute และ Method ของคลาสใหม่นั้นเพิ่มขึ้นได้ คุณสมบัติของ Inheritance จะมีคุณสมบัติเป็นแบบ is a คือ คลาสอนุพันธ์เป็นคลาสดั้งเดิมประเภทหนึ่ง ที่มีรายละเอียดเพิ่มเติม ทำให้เราสามารถสร้างความสัมพันธ์ระหว่างคลาสต่าง ๆ ได้ เช่น คลาสพนักงานเป็นคลาสดั้งเดิม ส่วนคลาสพนักงาน Part time เป็นคลาสอนุพันธ์ ดังนั้นเราสามารถกล่าวได้ว่า พนักงาน Part time เป็น (is a) พนักงาน ตัวอย่างของการสืบทอด



กำหนดให้ Polygon เป็นวัตถุในรูปแบบ 2 มิติที่สามารถมีด้าน 3- 4 ด้าน ตัวอย่างของ Polygon คือ สามเหลี่ยม สี่เหลี่ยมจัตุรัส สี่เหลี่ยมผืนผ้า โดยทุก ๆ Polygon จะมี attribute 2 แอททริบิวต์คือ พื้นที่และเส้นรอบวง ซึ่งรูป Polygon แต่ละประเภทก็จะมีการคำนวณพื้นที่หรือเส้นรอบวงที่แตกต่างกันไป จากข้อมูลข้างต้นทำให้เราสามารถสร้างคลาส Polygon ได้ โดยประกอบไปด้วย แอททริบิวต์ 2 ตัว คือ area และ perimeter และเมทอดอีก 2 ตัว คือ constructor และ destructor และเมทอดที่ช่วยแสดงพื้นที่และเส้นรอบวงคือ printArea และ printPeri

**ตัวอย่างที่ 12.7** ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

```

1 class Polygons
2 {
3     protected:
4         double area;
5         double perimeter;
6         void printArea () const;

```

```

7         void printPeri () const;
8     }; // Class Polygons

```

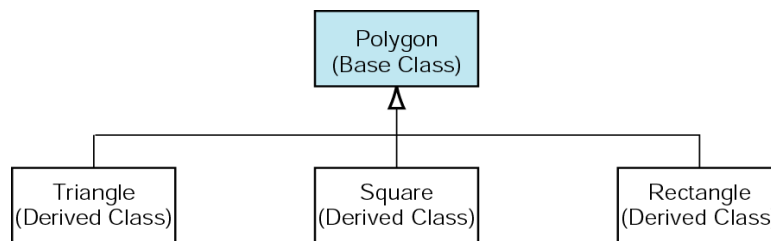
#### ผลลัพธ์จากการประมวลผลโปรแกรม

สร้างคลาส Polygon ได้ โดยประกอบไปด้วย แอทริบิวต์ 2 ตัว คือ area และ perimeter และเมทอดอีก 2 ตัว คือ constructor และ destructor และเมทอดที่ช่วยแสดงพื้นที่และเส้นรอบวงคือ printArea และ printPeri

### Base and derived classes

จากตัวอย่างดังกล่าวคลาส Polygon เป็นคลาสฐาน เพราะมีข้อมูลพื้นฐานที่สามารถให้คลาสอื่นสืบทอดไปได้ โดยคลาสอนุพันธ์ที่สืบทอดไปจะมีการคำนวณพื้นที่และเส้นรอบวงที่แตกต่างกันขึ้นอยู่กับประเภทของคลาสนั้น ๆ หรือรูปแบบ ของ Polygon รูปนั้น เรียกคลาสที่สืบทอดข้อมูลจากคลาสฐานเรียกว่า คลาสอนุพันธ์ (Derive class) โดยสามารถเข้าถึงแอทริบิวต์และฟังก์ชันในคลาสฐานได้ ในขณะเดียวกันตัวคลาสอนุพันธ์เองก็ยังคงมีแอทริบิวต์และฟังก์ชันเพื่อคำนวณข้อมูลได้ด้วย แสดงได้ดังรูป

#### ตัวอย่างของการสืบทอด



ตัวอย่างที่ 12.8 ตัวอย่างการเขียนโปรแกรมอย่างง่ายเพื่อสร้างคลาส

```

1  class Triangle : public Polygons
2  {
3      private:
4          double sideA;
5          double sideB;
6          double sideC;
7          void calcArea ();
8          void calcPeri ();
9  }; // Class Triangle
10 class Square : public Polygons
11 {
12     private:
13         double side;
14         void calcArea ();
15         void calcPeri ();
16 }; // Class Square
17 class Rectangle : public Polygons
18 {
19     private:
20         double sideA;
21         double sideB;
22         void calcArea ();
23         void calcPeri ();
24 }; // Class Rectangle

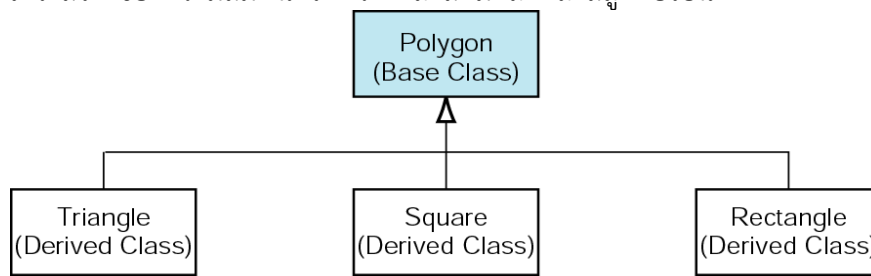
```

#### ผลลัพธ์จากการประมวลผลโปรแกรม

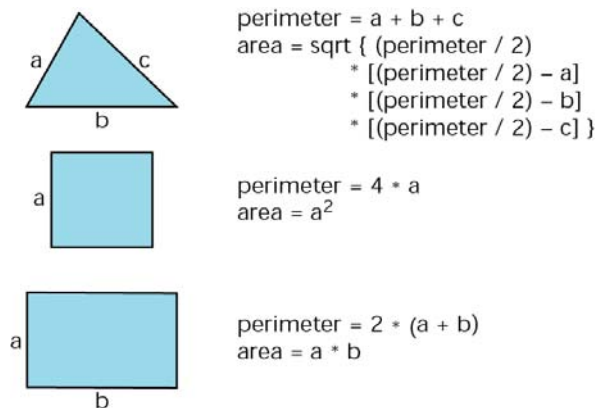
คลาสอนุพันธ์ที่ Triangle Square Rectangle สืบทอดไปจากคลาส Polygons จะมีการคำนวณพื้นที่และเส้นรอบวงที่แตกต่างกันขึ้นอยู่กับประเภทของคลาสนั้น ๆ

1. จงให้นิยามของคลาส และ object พร้อมยกตัวอย่างประกอบ
2. แนวคิดเชิงวัตถุประกอบด้วย 4 concept หลัก ๆ ได้แก่อะไรบ้าง
3. เมื่อวัตถุถูกสร้างจากคลาสจะเรียกใช้ method ใดเป็นอันดับแรก เพื่อวัตถุประสงค์อะไร
4. หลังการเลิกใช้งานวัตถุ method ใดจะถูกเรียก เพื่อวัตถุประสงค์อะไร
5. เทคนิคที่ method ที่มีชื่อเหมือนกันใน 1 คลาสแต่มีพารามิเตอร์ต่างกันเรียกว่า
6. หากต้องการให้คลาสลูกสามารถมองเห็นตัวแปรที่คลาสแม่ จะต้องกำหนดรูปแบบการเข้าถึงแบบใดให้กับตัวแปร
7. ให้สร้างคลาสที่ทำหน้าที่เก็บหน่วยของเวลา ชั่วโมง นาที และวินาที โดยในคลาสดังกล่าวสามารถตั้งเวลาเริ่มต้นรีเซ็ตเวลา แสดงชั่วโมง นาที และวินาทีที่หน้าจอ

8. จากโครงสร้างของความสัมพันธ์ระหว่างคลาสแม่และคลาสลูกต่อไปนี้



ให้เขียนโปรแกรมเชิงวัตถุเพื่อคำนวณหาพื้นที่และเส้นรอบวงของ Polygon แต่ละชนิด โดยสูตรการคำนวณของ Polygon แต่ละชนิดมีดังนี้



9. จงเขียนโปรแกรมเพื่อสร้างคลาส complex (จำนวนเชิงซ้อน) ที่มีฟังก์ชัน add, subtract, และ multiply มีฟังก์ชันสำหรับป้อนค่าเพื่อสร้าง complex ตัวใหม่

10. ประกาศคลาส circle ที่มีตัวแปรและฟังก์ชันสมาชิกดังต่อไปนี้

- มีตัวแปรเก็บสามค่าคือ a b r โดย a b เก็บค่าพิกัด x y ของจุดศูนย์กลาง และ r เป็นเลขจำนวนจริงเก็บรัศมี
- มีฟังก์ชัน ChangeCenter(a1,b1) สำหรับกำหนดค่า a b ของวงกลม เป็น a1 และ b1 ตามลำดับ
- มีฟังก์ชัน ChangeRadius(r1) สำหรับกำหนดค่า r ของวงกลม เป็น r1 ตามลำดับ
- มีฟังก์ชัน Area() ส่งกลับพื้นที่วงกลม

## เอกสารอ้างอิง

1. Behrouz A. Forouzan and Richard F. Bilberg “Computer Science A Structure Programming Approach Using C”, Brooks/Cole Thomson Learning , 2001
2. Harvey M. Deitel, Paul J. Deitel. “C How to Program Introducing C++ and Java” 4th Edition, Prentice Hall, 2004.
3. อรพิน ประวัติดิสสุทธิ์, “คู่มือเรียนภาษาซี”, โปรวิชั่น, 2547.
4. พนิดา พานิชกุล, “การโปรแกรมภาษาซี”, เคทีพี คอม แอนด์ คอนซัลท์, 2549.
5. อุมพร ศิริธรรานนท์, กัลยาณี บรรจงจิตร, นววรรณ สุนทรภิชช์, “การเขียนโปรแกรมคอมพิวเตอร์ ภาษาซี”, มูลนิธิ สอวน., 2549.
6. นิรุธ อำนวยศิลป์ คู่มือการเขียนโปรแกรมด้วยภาษาซี กรุงเทพฯ บริษัทโปรวิชั่น จำกัด พ.ศ. 2546
7. ประภาพร ช่างไม้ คู่มือการเขียนโปรแกรม ภาษา ซี ฉบับผู้เริ่มต้น นนทบุรี สำนักพิมพ์ อินโฟเพรส พ.ศ.2545