Dataset ข้อมูลพัฒนาChat bot เพื่อส่งเสริมการเรียนการสอนภาษา C เบื้องต้น

บทที่ 1

แนะนำภาษาC และโปรแกรม

1.1 ประวัติและความสำคัญของภาษา C

ภาษาซีเกิดขึ้นในปี ค.ศ. 1972 ผู้คิดค้นคือ Dennis Rittchie โดยพัฒนามาจากภาษาB และ ภาษา BCPL แต่ยังไม่มีการใช้งานอย่างกว้างขวางนัก ในปี ค.ศ. 1978 Brain Kernighan ได้ร่วมกับ Dennis Ritchie มาพัฒนามาตรฐานของภาษาซี เรียกว่า K&R ทำให้มีผู้ผู้สนใจเกี่ยวกับภาษาซีมากขึ้น จึงเกิดภาษาซี อีกหลายรูปแบบเพราะยังไม่มีการกำหนดรูปแบบภาษาซีที่เป็นมาตรฐาน และในปี 1988 Ritchie จึงได้ กำหนดมาตรฐานของภาษาซี เรียกว่า ANSI C เพื่อใช้เป็นตัวกำหนดมาตรฐานในการสร้างภาษาซีรุ่นต่อไป ภาษาซี เป็นภาษาระดับกลาง เหมาะสมสำหรับการเขียนโปรแกรมแบบโครงสร้าง เป็นภาษาที่มีความยืดหยุ่น มากคือใช้งานได้กับเครื่องต่างๆ ได้ และปัจจุบันภาษาซีเป็นภาษาพื้นฐานของภาษาโปรแกรม รุ่นใหม่ๆ เช่น C++

ภาษา C เริ่มมีคนสนใจมากขึ้นในปี ค.ศ.1978 เมื่อ Brain Kernighan ร่วมกับ Dennis Ritchie พัฒนามาตรฐานของภาษาซีขึ้นมา คือ K&R (Kernighan & Ritchie) และทั้งสองยังได้แต่งหนังสือชื่อว่า “The C Programming Language” หนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น ในเวลาต่อมาภาษาซีได้รับความนิยมสูง สถาบัน ANSI (American National Standards Institute) ได้สร้างมาตรฐานภาษาซีขึ้นมา เพื่อรับรองให้เป็นสากล ภายใต้ชื่อว่า ANSI-C ตั้งแต่ปี ค.ศ.1983 ในปัจจุบันได้มีการพัฒนาภาษาซีให้มีประสิทธิภาพมากยิ่งขึ้น เป็นเวอร์ชันต่าง มากมายมีการพัฒนาต่อยอดเป็นภาษาซีพลัสพลัส (C++) หรือภาษาซีชาร์ป (C#)ซึ่งมีการเพิ่มชุดคำสั่งที่สนับสนุนการพัฒนาโปรแกรมเชิงวัตถุ (Object-Oriented Programming) และยังคงรองรับชุดคำสั่งมาตรฐานของภาษาซี คือ ANSI-C อยู่ด้วย ภาษาซีเป็นภาษาแบบโครงสร้างที่สามารถศึกษา และทำความเข้าใจได้ไม่ยาก อีกทั้งยังสามารถเป็นพื้นฐานในการเขียนโปรแกรมภาษาอื่นๆ ได้อีก เช่น C++, Perl, JAVA, PHP, Python, Ruby เป็นต้น ซึ่งส่วนใหญ่ได้รับความนิยมนำมาใช้เขียนโปรแกรม จึงเป็นเหตุผลหนึ่งที่ทำให้ภาษาซีเป็นภาษาแรกในการเริ่มต้น โดยเฉพาะการทำงานที่มีข้อจำกัดในเรื่องหน่วยความจำและการประมวลผล

1.1.1 ลักษณะเด่นของภาษาซี

1.1.1.2 สามารถใช้งานบนสภาพแวดล้อมที่แตกต่างกันได้ ซอร์สโค้ดภาษาซี สามารถรันอยู่บนคอมพิวเตอร์ได้หลากหลายระดับ ตั้งแต่ระดับเมนเฟรมไปจนถึงไมโครเฟรม โดยไม่ต้องเปลี่ยนแปลงชุดคำสั่งใดๆ หรือเปลี่ยนแปลงเพียงเล็กน้อย

1.1.1.3 มีประสิทธิภาพสูง ชุดคำสั่งมีความกะทัดรัดและกระชับมาก และยังมีความใกล้ชิดกับฮาร์ดแวร์มากกว่าภาษาอื่นๆ

1.1.1.4 ความสามารถในการโปรแกรมโมดูล ภาษาซีมีการแบ่งโมดูลเพื่อคอมไพล์ได้ และยังสามารถลิงก์เชื่อมโยงเข้าด้วยกันได้อีก ภาษาซีประกอบด้วยฟังก์ชัน โดยโมดูลต่างๆ จะเขียนอยู่ในรูปของฟังก์ชันทั้งสิ้น

1.1.1.5. มีความยืดหยุ่นสูง สามารถใช้งานร่วมกับภาษาอื่นๆได้ มีการกล่าวว่า "ภาษา C เป็นภาษาที่อยู่ กึ่งกลางระหว่างภาษาระดับต่ำและภาษาระดับสูง"

1.1.1.6. ตัวอักษรตัวพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่แตกต่างกัน ตามปกติภาษาระดับสูงทั่วไป ตัวแปรที่ตั้งขึ้นด้วยตัวอักษรพิมพ์เล็กและตัวพิมพ์ใหญ่ สามารถนำมาใช้ร่วมกันได้ แต่ในภาษาซีถือว่าแตกต่างกัน เช่น NUM ไม่เท่ากับ num

1.1.2 ขั้นตอนการทำโปรแกรมด้วยภาษาซี

1.1.2.1. ขั้นตอนที่ 1 เขียนโปรแกรม (source code) ใช้ editorเขียนโปรแกรมภาษาซีและทำการบันทึกไฟล์ให้มีนามสกุลเป็น .c เช่น work.c เป็นต้น editor คือ โปรแกรมที่ใช้สำหรับการเขียนโปรแกรม โดยตัวอย่างของ editor ที่นิยมนำมาใช้ใน การเขียนโปรแกรมได้แก่ Notepad,Edit ของ Dos,TextPad และ EditPlus เป็นต้น ผู้เขียนโปรแกรม สามารถเลือกใช้โปรแกรมใดในการเรียนโปรแกรมก็ได้ แล้วแต่ความถนัดของแต่ละบุคคล

1.1.2.2. ขั้นตอนที่ 2 คอมไพล์โปรแกรม (compile) นำ source code จากขั้นตอนที่ 1 มาทำการคอมไพล์ เพื่อแปลจากภาษาซีที่มนุษย์เข้าใจไปเป็น ภาษาเครื่องที่คอมพิวเตอร์เข้าใจได้ ในขั้นตอนนี้คอมไพเลอร์จะทำการตรวจสอบ source code ว่าเกิด ข้อผิดพลาดหรือไม่ หากเกิดข้อผิดพลาด จะแจ้งให้ผู้เขียนโปรแกรมทราบ ผู้เขียนโปรแกรมจะต้องกลับไปแก้ไข โปรแกรม และทำการคอมไพล์โปรแกรมใหม่อีกครั้งหากเกิดข้อผิดพลาด จะแจ้งให้ ผู้เขียนโปรแกรมทราบ ผู้เขียนโปรแกรมจะต้องกลับไปแก้ไข โปรแกรม และทำการคอมไพล์โปรแกรมใหม่อีกครั้ง ขึ้นไปเป็นหากไม่พบข้อผิดพลาด คอมไพเลอร์จะแปลไฟล์ source code จาก

ภาษาซีไปเป็นภาษาเครื่อง (ไฟล์นามสกุล .obj) เช่นถ้าไฟล์ source code ชื่อ work.c ก็จะถูกแปลไปเป็นไฟล์ work.obj ซึ่งเก็บ ภาษาเครื่องไว้เป็นต้น

- compile เป็นตัวแปลภาษารูปแบบหนึ่ง มีหน้าที่หลักคือการแปลภาษาโปรแกรมที่มนุษย์เขียน - ภาษาเครื่อง โดยคอมไพเลอร์ของภาษาซี คือ C Compiler ซึ่งหลักการที่คอมไพเลอร์ ใช้ เรียกว่า คอมไพล์ (compile) โดยจะทำการอ่านโปรแกรมภาษาซีทั้งหมดตั้งแต่ต้นจนจบ แล้วทำการแปลผลที่เดียว

นอกจากคอมไพเลอร์แล้ว ยังมีตัวแปลภาษาอีกรูปแบบหนึ่งที่เรียกว่า อินเตอร์พรีเตอร์ การอ่าน และแปลโปรแกรมทีละบรรทัด เมื่อแปลผลบรรทัดหนึ่งเสร็จก็จะทำงานตามคำสั่งในบรรทัดนั้น แล้วจึงทำ การแปลผลตามคำสั่งในบรรทัดถัดไป หลักการที่อินเตอร์พรีเตอร์ใช้เรียกว่า อินเตอร์เพรต (interpret)

ข้อดีและข้อเสียของตัวแปลภาษาทั้งสองแบบมีดังนี

	ข้อดี	ข้อเสีย
คอมไพเลอร์	-ทำงานได้เร็ว เนื่องจากทำการแปลผลทีเดียวแล้วจึงทำงานตามคำสั่งของโปรแกรมในภายหลัง -เมื่อทำการแปลผลแล้ว ในครั้งต่อไปไม่จำเป็นต้องทำการแปลผลใหม่อีกเนื่องจากภาษาเครื่องที่แปลได้จะถูกเก็บไว้ที่หน่วยความจำสามารถเรียกใช้งานได้ทันที	- เมื่อเกิดข้อผิดพลาดขึ้นกับโปรแกรมจะตรวจสอบหาข้อผิดพลาดได้ยาก เพราะทำการแปลผลทีเดียวทั้งโปรแกรม
อินเตอร์พรีเตอร์	- หาข้อผิดพลาดของโปรแกรมได้ง่าย เนื่องจากทำ การแปลผลทีละบรรทัด เนื่องจากทำงานทีละบรรทัดดังนั้นจึงสั่งให้โปรแกรมทำงานตามคำสั่ง เฉพาะจุดที่ต้องการได้ ไม่เสียเวลารอการแปลโปรแกรมเป็นเวลานาน	-ช้าเนื่องจากทำงานที่ละบรรทัด

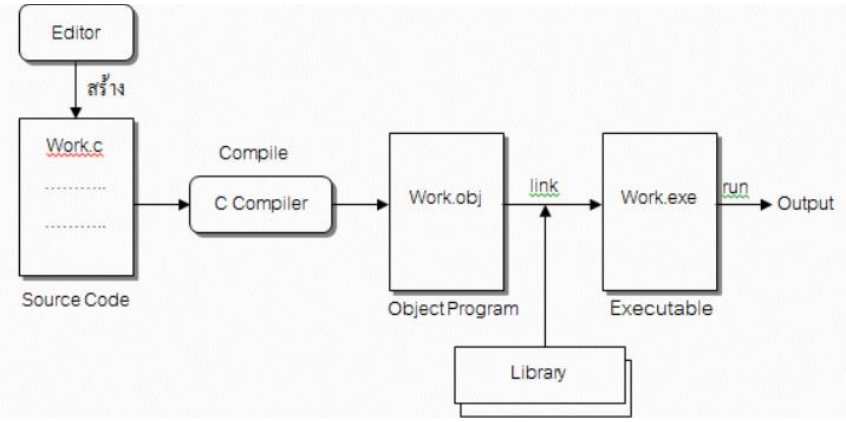
1.1.2.3 ขั้นตอนที่ 3 เชื่อมโยงโปรแกรม (link) การเขียนโปรแกรมภาษาซีนั้นผู้เขียนโปรแกรมไม่จำเป็นต้องเขียนคำสั่งต่าง ๆ ขึ้นใช้งานเอง เนื่องจากภาษาซีมีฟังก์ชั่นมาตรฐานให้ผู้เขียนโปรแกรมสามารถเรียกใช้งานได้ เช่น การเขียนโปรแกรม แสดงข้อความ "Lumpangkanyanee" ออกทางหน้าจอ ผู้เขียนโปรแกรมสามารถ

เรียกใช้ ฟังก์ชั่น print) ซึ่งเป็นฟังก์ชั่น มาตรฐานของภาษาซีมาใช้งานได้ โดยส่วนการประกาศ (declaration) ของฟังก์ชันมาตรฐานต่าง ๆ จะถูกจัดเก็บอยู่ในเฮดเดอร์ไฟล์แต่ละตัว แตกต่างกันไปตามลักษณะการใช้งาน

ด้วยเหตุนี้ภาษาครื่องที่ให้จากขั้นตอนที่ 2 จึงมีไม่สามกรถไปได้ แต่ต้องนำมาเชื่อเชื่อมโยงเข้ากับ library ก่อน ซึ่งผลจากการเชื่อมโยงจะทำให้ได้ executable program (ไฟล์นามสกุล.exe เช่น work.exe) ที่สามารถนำไปใช้งานได้

1.2.2.4 ขันตอนที่ 4 ประมวลผล (run)

เมื่อนำ executable prograrn จากขั้นตอนที่ 3 มาประมวลผลก็จะได้ผลลัพธ์ (output) ของ โปรแกรมออกมา



ภาพที่ 1 ขั้นตอนการทำงานโปรแกรมด้วยภาษาซี

1.2 โครงสร้างพื้นฐานของภาษาซี

1.2.1 ข้อความสั่งตัวประมวลผลก่อน (preprocessor statements) ข้อความสั่งตัวประมวลผลก่อนขึ้นต้นด้วยเครื่องหมาย # เช่น #include<stdio.h> หมายความว่าให้ตัวประมวลผลก่อนไปอ่านข้อมูลจากแฟ้ม stdio.กซึ่งเป็นแฟ้มที่มีอยู่ในคลังเมือ โปรแกรมมีการใช้ข้อความสั่งอ่านและบันทึกข้อความสั่งตัวประมวลผลก่อนจะต้องเขียนไว้ตอนต้นขอ โปรแกรม

1.2.2 รหัสต้นฉบับ (source code) รหัสต้นฉบับหมายถึงตัวโปรแกรมที่ประกอบด้วยข้อความสั่งและตัวฟังก์ชั่นต่างๆ

1.2.3 ข้อความสั่งประกาศครอบคลุม (global declaration statements) ข้อความสั่งประกาศครอบคลุมใช้ประกาศตัวแปรส่วนกลางโดยที่ตัวแปรส่วนกลางนั้นจะสามารถถูก เรียกใช้จากทุกส่วนของโปรแกรม

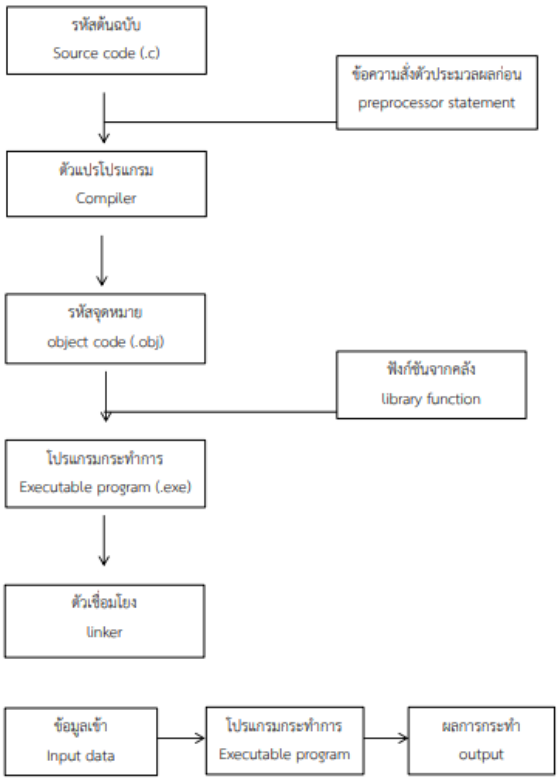
1.2.4 ต้นแบบฟังก์ชัน (function prototype) ต้นแบบฟังก์ชันใช้ประกาศฟังก์ขันเพื่อบอกให้ตัวแปลโปรแกรมทราบถึงชนิดของค่าที่ส่งกลับและ ชนิดของค่าต่างๆที่ส่งไปกระทำการในฟังก์ชัน

1.2.5 ฟังก์ชันหลัก (main function) เมื่อสั่งให้กระทำการโปรแกรมฟังก์ชันหลักจะเป็นจุดเริ่มต้นของการกระทำการภายในฟังก์ชันหลัก จะประกอบด้วยข้อความสั่งและข้อความสั่งที่เรียกใช้ฟังก์ชั่น

1.2.6 ฟังก์ชัน(function) ฟังก์ชันหมายถึงกลุ่มของข้อความสั่งที่ทำงานใดงานหนึ่งโดยเป็นอิสระจากฟังก์ชันหลักแต่อาจมีการ รับส่งค่าระหว่างฟังก์ชันและฟังก์ชันหลัก

1.2.7 ข้อความสั่งประกาศตัวแปรเฉพาะที่ (local declaration statements) ข้อความสั่งประกาศตัวแปรเฉพาะที่ใช้ประกาศตัวแปรเฉพาะที่โดยที่ตัวแปรเฉพาะที่จะสามารถถูก

1.2.8 การแปลและกระทำการโปรแกรม (program compilation and execution) เมื่อได้เขียนและป้อนข้อความสั่งตัวประมวลผลก่อนและรหัสต้นฉบับลงในโปรแกรมอิดิเตอร์เสร็จ แล้วจะต้องเรียกตัวแปรโปรแกรมมาเพื่อให้แปลภาษาชีให้เป็นภาษาเครื่องหากโปรแกรมนั้นเขียนได้ถูกต้อง ตรงตามกฎของภาษาชีตัวแปรโปรแกรมจะแปลโปรแกรมภาษาชีให้เป็นภาษาเครื่องแล้วนาไปเก็บไว้ในแฟ้ม ชื่อเดียวกันแต่มีนามสกุลเป็น .objจากนั้นตัวเชื่อมโยง (linker) จะต้องนาฟังก์ชันจากคลัง (ibrary function) ต่างๆที่โปรแกรมได้เรียกใช้มารวมเข้ากับแฟ้ม .objแล้วนำไปเก็บไว้ในแฟ้มชื่อเดิมแต่มีนามสกุล 43 ไฟล์เป็น .exe เมื่อต้องการกระทำการโปรแกรมก็สามารถป้อนข้อมูลเข้า (input data) ให้กับโปรแกรมซึ่ง จะได้ผลการกระทำ (output)



ภาพที่ 2 โครงสร้างพื้นฐานภาษาของซี

ทำงานได้เร็ว เนื่องจากทำการแปลผลทีเดียว - เมื่อเกิดข้อผิดพลาดขึ้นกับ แล้วจึงทำงานตามคำสั่งของโปรแกรมในภายหลัง- โปรแกรมจะตรวจสอบหา - เมื่อทำการแปลผลแล้ว ในครั้งต่อไปไม่ ข้อผิดพลาดได้ยาก เพราะทำการ

จำเป็นต้องทำการแปลผลใหม่อีก เนื่องจาก แปลผลทีเดียวทั้งโปรแกรม ภาษาเครื่องที่แปลได้จะถูกเก็บไว้ที่หน่วยความจำ สามารถเรียกใช้งานได้ทันที อินเตอร์พรีเตอร์ - หาข้อผิดพลาดของโปรแกรมได้ง่าย เนื่องจากทำ - ช้า เนื่องจากที่ทำงานทีละบรรทัด การแปลผลทีละบรรทัด - เนื่องจากทำงานทีละ บรรทัดดังนั้นจึงสั่งให้โปรแกรมทำงานตามคำสั่ง เฉพาะจุดที่ต้องการได้ ไม่เสียเวลารอการแปล โปรแกรมเป็นเวลานาน คอมไพเลอร์

ตัวอย่างที่ 1.1

แสดงโครงสร้างของโปรแกรมโดยโปรแกรมนี้ประกอบด้วยฟังก์ชัน main() และฟังก์ชัน sum() ฟังก์ชัน main() ทำหน้าที่รับค่ามาเก็บไว้ในตัวแปร a และตัวแปร b แล้วส่งค่าของตัวแปรทั้ง สองไปยังฟังก์ชัน sumา() เพื่อคำนวณหาผลรวมเมื่อคำนวณผลรวมแล้วจะส่งค่าของผลรวมกลับไปยัง ฟังก์ชัน main() จากนั้นฟังก์ชัน main) จะแสดงค่าของผลรวม

```
#include<stdio.h>      //คำสั่งตัวประมวลผลก่อน
int a, b, c;           //คำสั่งประกาศครอบคลุม
int sum(int x, int y); //คำสั่งแบบฟังก์ชัน
void main()            //ฟังก์ชัน main()
{                       //เริ่มต้นฟังก์ชันmain()
scanf("%d", &a);       //คำสั่งรับค่าตัวแปร
 scanf("%d", &b);      //คำสั่งรับค่าตัวแปร
c = sum(a, b);         //เรียกฟังก์ชัน sum()
printf("Vn96d + %d = %d",a, b, c); );   //แสดงผล
}                      //จบฟังก์ชันmain()
int sum (int x, int y) //ฟังก์ชัน sum()
{                      //เริ่มต้นฟังก์ชันsuml)
return (x + y);        //คำสั่งรวมค่าและส่งค่ากลับ
}                      //จบฟังก์ชันsum()
```

1.3 ตัวอย่างโปรแกรม Hello World

```
#include <stdio.h>


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

ภาพที่ 3 ตัวอย่างโปรแกรม Hello World

คำอธิบายโค้ด

1. #include <stdio.h>: เป็นการรวมไลบรารี stdio.h (Standard Input/Output) ซึ่งจำเป็นสำหรับฟังก์ชันการรับและแสดงข้อมูล เช่น printf().

2. int main(): คือฟังก์ชันหลักของโปรแกรม C ซึ่งเป็นจุดเริ่มต้นของการทำงานของโปรแกรม.

3. printf("Hello, World!\n");: คือคำสั่งที่ใช้แสดงข้อความ "Hello, World!" บนหน้าจอ. \n (newline character) ใช้เพื่อขึ้นบรรทัดใหม่หลังจากการแสดงข้อความ.

4. return 0;: บ่งบอกว่าโปรแกรมทำงานเสร็จสมบูรณ์และสิ้นสุดโดยไม่พบข้อผิดพลาด.

บทที่ 2

ตัวแปรและชนิดข้อมูล

2.1 ตัวแปร (variables)

คอมพิวเตอร์มีส่วนประกอบที่สำคัญส่วนหนึ่งคือหน่วยความจำซึ่งเปรียบได้กับสมองของมนุษย์ ทาหน้าที่เก็บข้อมูลในขณะที่ประมวลผลในการประมวลผลแต่ละครั้งมักต้องใช้ข้อมูลจำนวนมากซึ่ง จำเป็นจะต้องเก็บไว้ในหน่วยความจำเป็นเก็บแล้วจะต้องทราบตำแหน่งที่นำข้อมูลเข้าไปเก็บไว้ภายใน ของหน่วยความจำด้วยเพื่อให้สามารถนาข้อมูลเหล่านั้นกลับมาประมวลผลได้ดังนั้นตัวแปรจึงมีหน้าที่ สำคัญที่ช่วยในการเก็บข้อมูลแต่ละประเภทที่ผู้ใช้ป้อนเข้าสู่โปรแกรม

2.2 ชนิดข้อมูล (data types)

ข้อมูลที่ใช้ในโปรแกรมมีหลายชนิดซึ่งนักเขียนโปรแกรมต้องเลือกใช้ตามความเหมาะสมกับการ ใช้งานข้อมูลมีขนาดที่แตกต่างกันไปตามชนิดข้อมูลนอกจากนี้แล้วชนิดข้อมูลยังอาจมีขนาดที่แตกต่างกันโดยขึ้นกับเครื่องคอมพิวเตอร์และตัวแปลโปรแกรมที่ใช้ในการประมวลผลแต่โดยทั่วไปแล้วในไมโครคอมพิวเตอร์ชนิดข้อมูลมีการใช้ในโปรแกรมและขนาดดังนี้

ชนิดข้อมูล	การใช้โปรแกรม	คำสั่งในโปรแกรม	ขนาดข้อมูล (ไบต์)	ช่วงข้อมูล
character	char	%c	1	-128 ถึง 127
integer	int	%d	2	-32768 ถึง 32767
Long integer	long	%ld	4	-2147483648 ถึง 2147483647
Unsigned character	Unsigned char	%c	1	0 ถึง 255
Unsigned integer	Unsigned int	%d	2	0 ถึง 65535
Unsigned long	Unsigned long	%ld	4	0 ถึง 4294967295
Single-precision Floating-point	Float	%f	4	1.2 x 10-38 ถึง 3.4 X 1038

ตารางที่ 1 ชนิดข้อมูลการใช้งานภาษา C

2.2.1 กฎการตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรมีข้อกำหนดดังนี้

-ประกอบด้วย a ถึง z, o ถึง 9 และ _ เท่านั้น

-อักขระตัวแรกต้องเป็น a ถึง z และ _

-ห้ามใช้ชื่อเฉพาะ

-ตัวพิมพ์ใหญ่ตัวพิมพ์เล็กมีความหมายที่แตกต่างกัน

-มีความยาวได้สูงสุด 31 ตัวอักษร

2.2.2 การประกาศตัวแปร

การประกาศตัวแปรทำได้โดยเขียนข้อความสั่งขึ้นต้นด้วยชนิดข้อมูลตามด้วยชื่อตัวแปรและจบข้อความสั่งประกาศตัวแปรด้วยเครื่องหมายอัฒภาค (;) ดังนี้ ชนิดข้อมูลชื่อตัวแปร; ถ้าต้องการประกาศตัวแปรชนิดเดียวกันหลายตัวต้องคั่นระหว่างตัวแปรด้วยเครื่องหมายจุลภาค (,) และจบด้วยเครื่องหมายอัฒภาค (:) ดังนี้ ชนิดข้อมูลชื่อตัวแปร1, ชื่อตัวแปร2; เช่น

```
int count;        //ประกาศตัวแปรชื่อ count ใช้เก็บข้อมูลชนิด integer
int m, n;         //ประกาศตัวแปรชื่อ m และ n ใช้เก็บข้อมูลชนิด integer
intnum = 10;   //ประกาศตัวแปรชื่อกบกาและเก็บค่า 10 ไว้ในตัวแปรดังกล่าว
charstr = 'a';     //ประกาศตัวแปรชื่อstrและเก็บค่าอักขระ a ไว้ในตัวแปรดังกล่าว
```

2.3 การแสดงผลและการรับค่า

2.3.1 ฟังก์ชั่นprintf() เป็นฟังก์ชันจากคลังที่มาพร้อมกับตัวแปลโปรแกรมภาษาซีใช้สาหรับการ แสดงผลมีรูปแบบ ดังนี้ printf ("สายอักขระควบคุม", ตัวแปร);

โดยที่สายอักขระควบคุมประกอบด้วย 3 ส่วนคือ

-ตัวอักขระที่จะแสดง

-รูปแบบการแสดงผลขึ้นต้นด้วยเครื่องหมายเปอร์เซ็นต์ (96)

– ลำดับหลัก (escape sequence)

ตัวแปรคือชื่อของตัวแปรที่จะแสดงผล

รูปแบบการแสดงผล (format specifiers)

การกำหนดรูปแบบการแสดงผล

-ขึ้นต้นด้วยเครื่องหมายเปอร์เซ็น (%)

ตามด้วยอักขระ 1 ตัวหรือหลายตัวโดยที่อักขระนั้นมีความหมายดังนี้

อักขระ	ชนิดข้อมูล	รูปแบบการแสดงผล
C	char	อักขระเดียว
D O x	int	จำนวนเต็มสิบ จำนวนเต็มฐานแปด จำนวนเต็มฐานสิบหก
f	float	จำนวนที่มีทศนิยมในรูปฐานสิบ

ตารางที่ 2 การแสดงผลและการรับค่า

2.3.2 ลำดับหลัก (escape sequence) ในการแสดงผลบางสิ่งบางอย่างที่จะแสดงอาจไม่ใช่ตัวอักษรจึงไม่สามารถที่จะเขียนสิ่งที่จะแสดงไว้ในโปรแกรมได้เช่นต้องการเขียนโปรแกรมให้ส่งเสียงหรือต้องการให้เลื่อนขึ้นบรรทัดใหม่ก่อนแสดงข้อความดังนั้นในการเขียนโปรแกรมเพื่อแสดงผลสิ่งที่ไม่ใช่ตัวอักขระปกติจะต้องใช้ลาดับหลีกเพื่อ ช่วยในการกำหนดอักขระพิเศษหรือสิ่งที่ไม่ใช่อักขระที่ต้องการให้โปรแกรมแสดง ลำดับหลีกจะเขียนขึ้นต้นด้วยเครื่องหมาย (back slash) แล้วตามด้วยอักขระในการทำงาน เครื่องหมายทับกลับหลังจะบอกให้เครื่องคอมพิวเตอร์ทราบว่าให้หลีกเลียงการตีความอักขระที่ตามหลัง มานี้ในลักษณะปกติเพราะอักขระเหล่านี้จะมีความหมายพิเศษแตกต่างออกไป

ลำดับหลัก	ผลกระบวนทำการ
\n	ขึ้นบรรทัดใหม่
\t	เลื่อนไป1แถบ
\a	เสียงกระดิ่ง
\b	ถอยไปหนึ่งบรรทัด
\f	ขั้นหน้าใหม่
\\	แสดงเครื่องหมาย\
\’	แสดงเครื่องหมายฝนทอง
\”	แสดงเครื่องหมายฝันหนู

ตารางที่ 3 การแสดงลำดับกระบวนการ

2.3.3 ฟังก์ชัน scanf() เป็นฟังก์ชันที่ใช้ในการรับข้อมูลจากแป้นพิมพ์โดยจะบอกเลขที่อยู่ของตัวแปรในหน่วยความจำแล้วจึงนาค่าที่รับมาเก็บไว้ตามที่อยู่นั้นโดยมีรูปแบบดังนี้ scanf ("%รูปแบบ", &ตัวแปร); โดยที่&ตัวแปรหมายถึงเลขที่อยู่ (address) ของตัวแปรที่จะรับค่ามาเก็บในหน่วยความจำ

บทที่ 3

ตัวดำเนินการและนิพจน์

3.1 นิพจน์ (Expression)

นิพจน์ คือ การนำข้อมูลซึ่งอาจจะอยู่ในรูปของค่าคงที่ หรือตัวแปร มาดำเนินการโดยใช้ เครื่องหมายต่างๆ เป็นตัวสั่งงาน สำหรับนิพจน์ที่เราพบเห็นกันทั่วไปในชีวิตประจำวัน ยกตัวอย่าง ดังต่อไปนี้

```
-score = midterm + final
-ax2 + bx +c
-ans = 100 - 50
```

ภาพที่ 3 ตัวอย่างข้อมูลนิพจน์

3.1.1 นิพจน์ทางคณิตศาสตร์ (Arithmetic Expression)

การเขียนนิพจน์ทางคณิตศาสตร์ในภาษาซี จะเหมือนกับการเขียนนิพจน์ทางคณิตศาสตร์ตามปกติ เพียงแต่เปลี่ยนมาใช้เครื่องหมายทางคณิตศาสตร์ของภาษาซีแทน ตัวอย่างเช่น เครื่องหมาย * แทนการคูณ (X) หรือการหารจะใช้เครื่องหมาย / แทน

ตัวอย่างการเขียนนิพจน์ทางคณิตศาสตร์ในภาษาซีแสดงดังต่อไปนี้

นิพจน์ทางคณิตศาตร์ในภาษาซี
X+y-Z
2*x*y+4*z
X*x+2*x+1
(a - b) / (c + d)
(x*x)/(x* (x * y + 2)

ตารางที่ 4 ตัวอย่างนิพจน์ทางคณิตศาสตร์

3.1.2 นิพจน์ทางตรรกศาสตร์ (Logical Expression)

การเขียนนิพจน์ทางตรรกศาสตร์ในภาษาซี ก็คือ การเขียนนิพจน์โดยใช้เครื่องหมาย การดำเนินการทางตรรกศาสตร์ในภาษาซี (88, II, !) เป็นการสั่งงาน ซึ่งส่วนแล้วนิพจน์ทาง ตรรกศาสตร์จะอยู่ร่วมกับนิพจน์ประเภทอื่น ๆตัวอย่างนิพจน์ทางตรรกศาสตร์ พร้อมทั้งผลลัพธ์จากการดำเนินการ แสดงตังตารางต่อไปนี้ โดยกำหนดให้ตัวแปร a = 30, b = -100 และ c =0

นิพจน์ทางตรรกศาสตร์	การดำเนินการ	ผลลัพธ์
(c>=a) && (a<=b)	F && F	F
(b >= c) || (c <= a)	F | T	T
(a > b) && (c <= a)	T && T	T
(!c) || (a==b)	T | F	T
(a + 5) > (b - 100)	35 > -200	T

ตารางที่ 5 ตัวอย่างนิพจทน์ทางตรรกศาสตร์

3.2. ตัวดำเนินการ (Operators)

ในภาษาซี มีตัวทำเนินการหลากหลายชนิด ในที่นี้จะกล่าวถึงตัวตัวตัวทำเนินการพื้นฐานที่สำคัญดังต่อไปนี้

-ตัวดำเนินการคณิตศาสตร์

-ตัวดำเนินการยูนารีตัว

-ดำเนินการเปรียบเทียบ

-ตัวดำเนินการตรรกะ

-ตัวดำเนินการกำหนดค่าแบบผสม

-ตัวดำเนินการเงื่อนไข

3.2.1 ตัวดำเนินการทางคณิตศาสตร์ คือตัวดำเนินการที่ใช้เพื่อกระทำการดำเนินการทางคณิตศาสตร์ระหว่างตัวแปรหรือค่าคงที่ เช่น การบวก การลบ การคูณ และการหาร สำหรับในการเขียนโปรแกรมในภาษา C นั้นจะมีตัวดำเนินการสำหรับการหารเอาเศษ (Modulo) เพิ่มเข้ามา

ตารางข้างล่างนี้คือตัวดำเนินการทางคณิตศาสตร์ในภาษา C

Symbol	Name	Example
+	Addition	c = a + b
-	Subtraction	c = a - b
*	Multiplication	c = a * b
/	Division	c = a / b
%	Modulo	c = a % b

ตารางที่ 6 ตัวดำเนินการทางคณิตศาสตร์

จากในตารางของตัวดำเนินการข้างบน ในการเรียนในวิชาคณิตศาสตร์ นอกจากตัวดำเนินการ Modulo ที่เป็นการหารเอาเศษ ต่อไปมาดูตัวอย่างการใช้ตัวดำเนินการทางคณิตศาสตร์ ในการเขียนโปรแกรมในภาษา C

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 10;
    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n", a - b);
    printf("a * b = %d\n", a * b);
    printf("a / b = %d\n", a / b);
    printf("a % b = %d\n", a % b);
    return 0;
}
```

ภาพที่ 4 ตัวอย่างโค้ดตัวดำเนินทางคณิตศาตร์

ในตัวอย่าง เป็นการใช้งานตัวดำเนินการประเภทต่างๆ ในภาษา C เราได้ประกาศตัวแปร a และ b และพร้อมกับกำหนดค่าให้กับมันและใช้ตัวดำเนินการต่างๆ ซึ่งคุณคงจะคุ้นเคยดี และในตัวดำเนินการ Modulo นั้นเป็นการหารเอาเศษซึ่งคำตอบที่ได้จะเป็นเศษของการหาร เมื่อรันโปรแกรม มันจะได้ผลลัพธ์ดังนี้

```
a + b = 15
a - b = -5
a * b = 50
a / b = 0
a % b = 5
```

ภาพที่ 5 ผลลัพธ์การตัวดำเนินทางคณิตศาตร์ในภาษา C

3.2.2 Compound assignment คือตัวดำเนินการที่ใช้เพื่ออัพเดทหรือแก้ไขค่าปัจจุบันของตัวแปรโดยการกระทำทางคณิตศาสตร์และใช้ตัวดำเนินการกำหนดค่าร่วมด้วย ซึ่งตัวดำเนินการแบบ Compound assignment มักจะใช้เป็นรูปแบบสั้นของตัวดำเนินการทางคณิตศาสตร์และตัวดำเนินการระดับบิตเพื่อให้การเขียนสั้นลง ข้างล่างนี้เป็นตารางของ Compound assignment operators ในภาษา C

Operator	Example	Equivalent to
+=	a += 2;	a = a + 2
-=	a -= 2;	a = a - 2
*=	a *= 2;	a = a * 2
/=	a /= 2;	a= a / 2
%=	a %= 2;	a = a % 2
>>=	a >>= 2;	a = a >> 2
<<=	a <<= 2	a = a << 2
&=	a & = 2;	a = a & 2
^=	a ^= 2;	a= a ^ 2

ตารางที่ 7 Compound assignment

ตัวอย่างของการใช้งานของตัวดำเนินการ Compound assignment

```c
#include <stdio.h>

int main()
{
    int x = 10;
    int y = 2;
    x += 10; //equivalent to x = x + 10
    y -= 2;  //equivalent to y = y – 2
    printf(“x = %d, y = %d”, x, y);
    return 0;
}
```

ภาพที่ 6 ตัวอย่างโค้ด Compound assignment

ในตัวอย่าง นั้นเป็นรูปแบบอย่างสั้นในการดำเนินการทางคณิตศาสตร์ของมัน โดยการทำงานของตัวดำเนินการจะอ้างอิงจากค่าเดิม เช่น ในตัวแปร x นั้นเป็นการบวกค่าเข้าไปในตัวแปรเดิมอีก 10 และในตัวแปร y นั้นเป็นการลบค่าออกจากตัวแปรเดิมออก 2 และมันจะได้ผลลัพธ์ดังนี้

```
x = 20, y = 0
```

ภาพที่ 7 ผลลัพธ์การดำเนินการ Compound assignment

3.2.3 ตัวดำเนินการเพิ่มและลดค่า คือตัวดำเนินการที่ใช้เพื่อบวกหรือลบค่าออกจากตัวแปรโดย 1 โดยการเพิ่มเครื่องหมาย ++ หรือ-- ใส่ข้างหน้าหรือข้างหลังตัวแปร ซึ่งมีรูปแบบการใช้ดังนี้

ตัวอย่างของการใช้ตัวดำเนินการเพิ่มและลดค่า

```c
#include <stdio.h>

int main()
{
    int a = 1;
    int b = 10;
    a++;
    b--;
    printf("a=%d, b=%d", a, b);
    return 0;
}
```

ภาพที่ 8 ตัวอย่างโค้ดการทำงานเพิ่มและลดค่า

ผลลัพธ์ที่ได้

```
a=2, b=9
```

ภาพที่ 9 ผลลัพธ์การเพิ่มและลดค่า

ในตัวอย่าง จะป็นรูปแบบอย่างสั้นของ a = a + 1 และ b = b - 1 มักจะใช้กับคำสั่ง for loop เพื่อเพิ่มของการรัน Index หรือตำแหน่ง Index ของอาเรย์ และยังมีรูปแบบอื่นของตัวดำเนินการนี้คือ Prefix เช่น ++a --b โดยมันหมายถึงจะมีการเพิ่มหรือลดค่าก่อนที่จะมีการประมวลผลคำสั่งปัจจุบัน

3.2.4 ตัวดำเนินการความสัมพันธ์และเปรียบเทียบ คือตัวดำเนินการที่ถูกใช้เพื่อประเมินค่า true และ false ระหว่างสองค่าถูกดำเนินการ ซึ่งขึ้นกับเงื่อนไขและความสัมพันธ์

ตารางของตัวดำเนินการ Relational และตัวดำเนินการเปรียบเทียบ

Operator	Example	Result
==	a == b	true if `a` equal to `b`, otherwise false
!=	a != b	true if `a` not equal to `b`, otherwise false
<	a < b	true if `a` less than `b`, otherwise false
>	a > b	true if `a` greater than `b`, otherwise false
<=	a <= b	true if `a` less than or equal to `b`, otherwise false

ตารางที่ 8 ตัวดำเนินการ Relational

ตัวดำเนินการเหล่านี้ถูกใช้ เช่น เพื่อเปรียบเทียบค่าระหว่างตัวแปรว่าเท่ากันหรือไม่ หรือเปรียบการมากกว่าน้อยกว่าหรือค่าในตัวแปร และผลลัพธ์ของการเปรียบเทียบจะเป็น true และ false

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 10;

    if (a == b)
    {
        printf ("a and b are equal");
    }
    else
    {
        printf ("a and b are not equal");
    }

    return 0;
}
```

ภาพที่ 10 ตัวอย่างโค้ดการใช้งานตัวดำเนินการเปรียบเทียบ

ในตัวอย่าง เป็นโปรแกรมในการเปรียบเทียบค่าในตัวแปร a และ b โดยการใช้คำสั่ง if ซึ่งถ้าหากค่าของตัวแปรทั้งสองเท่ากันซึ่งจะทำให้เงื่อนไขเป็นจริง จะทำให้โปรแกรทำงานในบล็อคของคำสั่ง if ไม่เช่นนั้นโปรแกรมจะทำในบล็อคของคำสั่ง else แทน

```
a and b are not equal
```

ภาพที่ 11 ผลลัพธ์การใช้งานตัวดำเนินการเปรียบเทียบ

ผลลัพธ์การทำงานของโปรแกรม จะเห็นว่าโปรแกรมทำงานในบล็อคของคำสั่ง else เพราะว่าค่าในตัวแปรทั้งสองนั้นไม่เท่ากัน ให้คุณลองเปลี่ยนค่าในตัวแปรให้เท่ากันแล้วลองรันโปรแกรมอีกครั้งเพื่อดูผลลัพธ์

3.2.5 Logical operators คือ ตัวดำเนินการทางตรรกศาสตร์ถูกใช้เพื่อประเมิน Expression ย่อยหลายๆ Expression ให้เหลือเพียงอันเดียว โดยผลลัพธ์สุดท้ายนั้นจะเป็นจริงหรือเท็จ

ตารางรายการตัวดำเนินการทางตรรกศาสตร์ในภาษา C

Name	Symbol	Example
not	!	!a
and	&&	a && b
or	||	a || b

ตารางที่ 9 Logical operators

ในการทำงานของตัวดำเนินการนั้น ตัวดำเนินการ ! (not) จะกลับค่าของ Boolean expression ตัวดำเนินการ && (and) ทำการเชื่อมสอง Expression เข้าด้วยกัน โดยจะได้ผลลัพธ์เป็นจริงหากทั้งสองค่าเป็นจริง ไม่เช่นนั้นจะเป็นเท็จ และตัวดำเนินการ || (or) เชื่อมสอง Expression เข้าด้วยกัน โดยจะได้ผลลัพธ์เป็นจริงหากมีอย่างน้อยหนึ่ง Expression ที่เป็นจริง ไม่เช่นนั้นจะเป็นเท็จ ต่อไปมาดูตัวอย่างการใช้งานของตัวดำเนินการตรรกศาสตร์

```c
#include <stdio.h>

int main()
{
    int a = 10;
    bool b = 1;
    bool c = 0;

    // using "not" operator
    if (!(a == 10))
```

```
        printf("a is not equal to 10.\n");
    else
        printf("a is equal to 10.\n");

    // using "and" operator
    if (b && c)
        printf("True.\n");
    else
        printf("False.\n");

    // using "or" operator
    if (b || c)
        printf("True.\n");
    else
        printf("False.\n");

    return 0;
}
```

ภาพที่ 12 ตัวอย่างโค้ดการทำงานของตัวดำเนินการ Logical operators

ในตัวอย่างเรามีตัวแปรสามตัว ตัวแปร a เป็น integer ในขณะที่ตัวแปร b และ c เป็น boolean ในภาษา C ค่าของ boolean ที่เป็นจริงนั้นจะมีค่าเป็น 1 และเท็จจะเป็น 0

ผลลัพธ์ของโปรแกรม

```
a is equal to 10.
False.
True.
```

ภาพที่ 13 ผลลัพธ์ของโปรแกรม

3.2.6 Bitwise operators นั้นถูกใช้ในการดำเนินการระดับบิตของตัวแปรหรือข้อมูล มันมีประโยชน์มากในการเขียนโปรแกรมระดับต่ำ ตัวดำเนินการ Bitwise นั้นใช้หลักการทำงานเหมือนกับตัวดำเนินการทางตรรกศาสตร์ โดยใช้ 1 สำหรับค่าจริงและ 0 สำหรับค่าเท็จ

ตารางตัวดำเนินการ Bitwise ในภาษา C

Symbol	Name	Description
&	Bitwise AND	1 ถ้าบิตทั้งคู่เป็น 1, ไม่เช่นนั้นเป็น 0
|	Bitwise inclusive OR	1 ถ้าอย่างน้อยหนึงบิตเป็น 1, ไม่เช่นนั้นเป็น 0
^	Bitwise exclusive OR	1 ถ้าทั้งสองบิตแตกต่างกัน, ไม่เช่นนั้นเป็น 0
~	bit inversion	กลับบิตจาก 1 เป็น 0 และในทางตรงข้าม
<<	Shift bits left	เลื่อนบิตไปทางซ้าย เติมบิต 0 ทางขวา
>>	Shift bits right	เลื่อนบิตไปทางขวา เติมบิต 0 ทางซ้าย

ตารางที่ 10 ตัวดำเนินการ Bitwise ในภาษา C

จากตารางข้างบน เป็นตัวดำเนินการประเภทต่างๆ ในภาษา C ที่ใช้ดำเนินการกับข้อมูลในระดับบิต ยกตัวอย่าง เช่น เมื่อเรากำหนดค่าให้กับตัวแปรนั้น คอมพิวเตอร์จะเก็บข้อมูลในรูปแบบของเลขฐานสอง (Binary) ตัวดำเนินการเหล่านี้ใช้สำหรับจัดการกับข้อมูลดังกล่าว มาดูตัวอย่างการใช้งานในภาษา C

```c
#include <stdio.h>

int main()
{
    int a = 2; // 00000010
    int b = 5; // 00000101
    printf("a & b = %d\n", a & b);    // 00000000 = 0
    printf("a & b = %d\n", a | b);    // 00000111 = 7
    printf("~a = %d\n", ~a);          // 11111101 = -1
    printf("a >> 1 = %d\n", a >> 1);  // 00000001 = 1
    printf("a << 1 = %d\n", a << 1);  // 00000100 = 8
    return 0;
}
```

ภาพที่ 14 ตัวอย่างโค้ด

ในตัวอย่าง เป็นโปรแกรมในการใช้งานตัวดำเนินการระดับบิตกับการจัดการข้อมูลประเภท Integer เราได้ประกาศตัวแปร a และ b และได้คอมเมนต์ค่าของ Binary ที่ถูกเก็บไว้ในหน่วยความจำ การทำงานของตัวดำเนินการ & และ | นั้นทำงานกับคู่ของแต่ละบิตของตัวแปรทั้งสอง ส่วนอีกสามตัวดำเนินการที่เหลือนั้นกระทำกับตัวแปรเดียว ในตัวดำเนินการ ~ นั้นเป็นการกลับบิต และสำหรับตัวดำเนินการ Bit shift นั้นเมื่อเลื่อนบิตไปทางซ้ายจะทำให้ค่าเพิ่มขึ้นสองเท่า และเมื่อเลื่อนบิตไปทางขวาก็จะทำให้ค่าลดลงสองเท่า

ผลลัพธ์การทำงานของโปรแกรมของการใช้งานตัวดำเนินการระดับบิตในภาษา C

```
a & b = 0
a & b = 7
~a = -3
a >> 1 = 1
a << 1 = 4
```

ภาพที่ 15 ผลลัพธ์การทำงานของโปรแกรมตัวเนินการภาษา C

3.2.7 ลำดับความสำคัญของตัวดำเนินการ (Operator precedence) ใช้เพื่อกำหนดว่าตัวดำเนินการใช้ที่จะถูกทำงานก่อน ยกตัวอย่างเช่น ในการทำงานของ Expression 6 + 3 * 4 นั้นจะได้ผลลัพธ์เป็น 18 เพราะว่าตัวดำเนินการ * นั้นมีความสำคัญมากกว่าตัวดำเนินการ + คุณสามารถบังคับให้การบวกทำงานก่อนได้โดยใช้วงเล็บ (6 + 3) * 4 เพื่อให้การบวกสามารถทำงานก่อนได้ ซึ่งจะได้ผลลัพธ์เป็น 36 เป็นต้น

ตารางแสดงลำดับความสำคัญของตัวดำเนินการในภาษา C

Operator Name	Associativity	Operators
Primary	left to right	()[]
Unary	ขวาไปซ้าย	++--+-!~&*(*type_name*)sizeof new delete
Multiplicative	ซ้ายไปขวา	*/%
Additive	ซ้ายไปขวา	+-
Bitwise Shift	ซ้ายไปขวา	<<>>
Relational	ซ้ายไปขวา	<><=>=
Equality	ซ้ายไปขวา	==!=
Bitwise AND	ซ้ายไปขวา	&
Bitwise Exclusive OR	ซ้ายไปขวา	^
Bitwise Inclusive OR	ซ้ายไปขวา	|
Logical AND	ซ้ายไปขวา	&&
Logical OR	ซ้ายไปขวา	||
Conditional	ขวาไปซ้าย	?:
Assignment	ขวาไปซ้าย	=+=-=*= /=<<=>>= %= &=^=|=
Comma	ซ้ายไปขวา	,

บทที่ 4

คำสั่งแบบควบคุมแบบทางเลือก

4.1การควบคุมทิศทางแบบทางเลือก

การเขียนโปรแกรมให้มีการตัดสินใจ สามารถเลือกได้ว่าจะทำหรือไม่ทำตามคำสั่ง ขึ้นอยู่กับกับเงื่อนไขที่กำหนดขึ้นมา โดยเงื่อนไขทางเลือกที่เขียนจะอยู่ในรูปของนิพจน์เปรียบเทียบ (relational expression) หรือนิพจน์ตรรกะ (boolean expression) ซึ่งใช้ตัวดำเนินการเปรียบเทียบ หรือตัวดำเนินการตรรกะเป็นตัวดำเนินการของนิพจน์

ตัวดำเนินการเปรียบเทียบของภาษาซีประกอบด้วย

ตัวดำเนินการ	ความหมาย	ตัวอย่างการใช้งาน
==	เท่ากัน	x==y
!=	ไม่เท่ากัน	x != y
<	น้อยกว่า	x < y
<=	น้อยกว่า หรือเท่ากับ	x < = y
>	มากกว่า	x>y
>=	มากกว่า หรือเท่ากับ	x>=y

โดยคำสั่งสำหรับการควบคุมทิศทางแบบทางเลือก ภาษาซีแบ่งออกเป็น 4 ประเภท คือ if,if else, if-else if และ switch-case

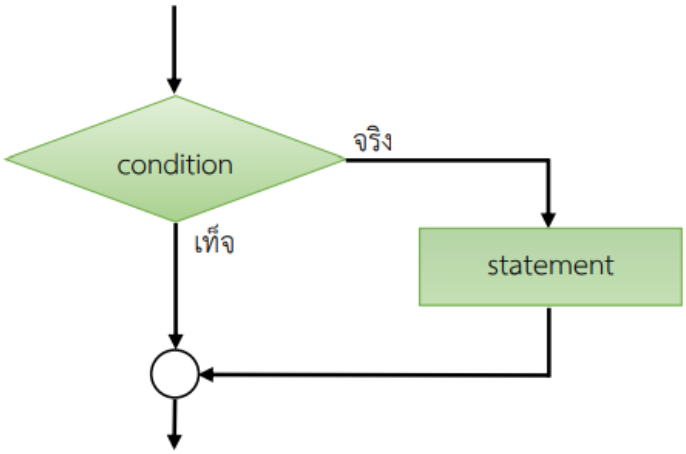
4.2 คำสั่งเงื่อนไข IF

การสร้างทางเลือกโดยใช้คำสั่ง if จะใช้ในกรณีที่มีทางเลือกให้ทำงานอยู่เพียงทางเลือกเดียว ผลจากการตรวจสอบเงื่อนไขก็คือ ทำ หรือไม่ทำตามคำสั่งนั้น รูปแบบการเขียนคำสั่ง if แสดง

```
if (condition)
    statement;
```

Condition = เงื่อนไขทางเลือก คือเงื่อนไขที่กำหนดขึ้นเพื่อใช้พิจารณาว่าจะทำ หรือไม่ ทำตามคำสั่ง โดยเงื่อนไขอาจอยู่ในรูปของนิพจน์การคำนวณและเปรียบเทียบ หรือเป็นค่าของ ตัวแปรก็ได้ statement = คำสั่ง คือคำสั่งที่ต้องทำงานเมื่อเงื่อนไขที่กำหนดเป็นจริง

จากรูปแบบคำสั่ง if สามารเขียนผังงาน (Flowchart) ได้ดังนี้

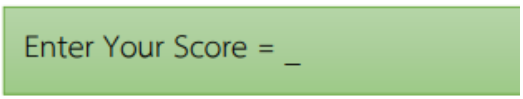


สำหรับการเขียนโปรแกรมโดยใช้คำสั่ง if ตรวจสอบเงื่อนไขเพื่อเลือกทำงานแสดง ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรมแจ้งผลการทดสอบจากคะแนนที่ป้อนเข้าไป เทียบกับค่าคะแนนที่ตั้งไว้ โดยถ้าผ่านเกณฑ์ที่กำหนดจะแสดงข้อความให้ทราบบนหน้าจอ

```
#include<stdio.h>
main()
{
        int score;
        printf("Enter Your Score = ");          // แสดงผลข้อความให้ป้อนคะแนน
        scanf("%d",&score);                     // รอรับค่าคะแนนมาเก็บในตัวแปร
        score if(score>=50)         // ตรวจสอบคะแนนในตัวแปร score ว่ามากกว่า หรือเท่ากับ 50 หรือไม่
       printf("Congratulation! --> Your PASS");   // ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความ
}
```

เมื่อรันโปรแกรมจะปรากฏผลลัพธ์และให้ป้อนคะแนน ดังนี้



ถ้าคะแนนที่ป้อนเข้าไปในโปรแกรมมากกว่า หรือเท่ากับ 50 ซึ่งทำให้เงื่อนไขของคำสั่ง if เป็นจริง โปรแกรมจะแสดงข้อความ Congratulation! --> Your PASS บนหน้าจอ ดังนี้

```
Enter Your Score = 75
Congratulation! --> Your PASS
```

รูปภาพที่ ตัวอย่างคำตอบ

ตัวอย่าง โปรแกรมเกมทายตัวเลข จำนวน 3 หลัก

```
#define TARGET 238 // กำหนดรหัสตัวเลขที่ต้องการ
#include<stdio.h>
main()
{
        int pw;
        printf("Enter Number (3 digits) = ");         // แสดงผลข้อความให้ป้อนรหัสตัวเลข 3 หลัก
        scanf("%d",&pw);        // รอรับค่ารหัสตัวเลขมาเก็บในตัวแปร pw
        if(pw==TARGET)       // ตรวจสอบรหัสตัวเลขในตัวแปร pw ว่าเท่ากันกับรหัสที่ตั้งไว้หรือไม่
              printf("\nPassword is Correct");       // ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความว่าถูกต้อง
        printf("\nGood Bye.");     // แสดงข้อความก่อนจบโปรแกรม
}
```

เมื่อสั่งรันโปรแกรม ให้ป้อนรหัสตัวเลขจำนวน 3 หลัก ในกรณีที่เงื่อนไขของคำสั่ง if เป็น จริงจะปรากฏผลลัพธ์ดังนี้

```
Enter Number (3 digits) = 238
Password is Correct
Good Bye.
```

และในกรณีที่เงื่อนไขของคำสั่ง if เป็นเท็จ จะปรากฏผลลัพธ์ ดังนี้

```
Enter Number (3 digits) = 123
Good Bye.
```

สรุป การทำงานของคำสั่งการตรวจสอบเงื่อนไข แบบ if ทางเลือกเดียว จะทำการตรวจสอบ เงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำงานตามประโยคคำสั่งภายในวงเล็บปีกกา แต่ถ้าเป็นเท็จจะข้ามไปทำ ชุดคำสั่งถัดไป

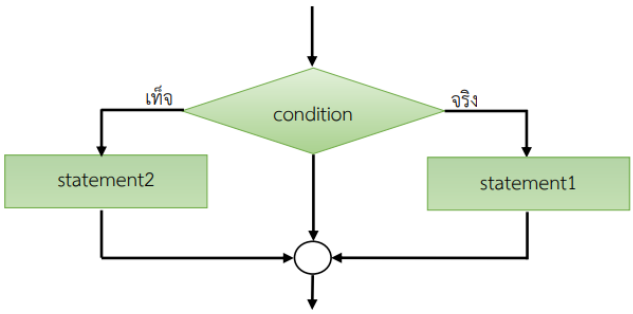
4.3 คำสั่งเงื่อนไข If-Else

การสร้างทางเลือกโดยใช้คำสั่ง if-else จะใช้ในกรณีที่มีทางเลือกให้ทำงาน 2 ทางเลือก เป็นคำสั่งที่ใช้กำหนดให้โปรแกรมตัดสินใจเลือกทำคำสั่งอย่างใดอย่างหนึ่งจาก 2 ทางเลือก โดยการ ตรวจสอบเงื่อนไขที่กำหนดว่าเป็นจริง หรือเท็จ ถ้าเงื่อนไขที่กำหนดให้เป็นจริง (true) โปรแกรมจะ ทำตามคำสั่งที่อยู่ภายใต้คำสั่ง if แต่ถ้าเงื่อนไขที่กำหนดให้เป็นเท็จ (false) โปรแกรมจะทำตามคำสั่งที่ อยู่ภายใต้คำสั่ง else ซึ่งรูปแบบการเขียนคำสั่ง ร if-else ในภาษาชีแสดงดังต่อไปนี้

```
if (condition)
        statement_1;
else
        statement_2;
```

condition = เงื่อนไขทางเลือก คือเงื่อนไขที่กำหนดขึ้นเพื่อใช้พิจารณาว่าจะทำ หรือไม่ ทำตามคำสั่ง โดยเงื่อนไขอาจอยู่ในรูปของนิพจน์การคำนวณและเปรียบเทียบ หรือเป็นค่าของ ตัวแปรก็ได้ statement = คำสั่งคือคำสั่งที่ต้องทำงานเมื่อเงื่อนไขที่กำหนดเป็นจริง

จากรูปแบบคำสั่ง if สามารถเขียนผังงาน (Flowchart) ได้ดังนี้



การทำงานจะเริ่มจากการตรวจสอบ condition โดยถ้าเป็นจริงจะกระทำ statement1 แต่ถ้าเป็นเท็จจะกระทำ statennent2 และเช่นเดียวกัน statement1 ที่ตามหลัง if และ statement2 ที่ตามหลัง else จะต้องเป็น staternent เดียวเท่านั้นสำหรับการเขียนโปรแกรมโดยใช้คำสั่ง if-else ตรวจสอบเงื่อนไขเพื่อเลือกทำงานแสดงดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรมแจ้งผลการทดสอบจากคะแนนที่ป้อนเข้าไปทางคีย์บอร์ด ถ้าคะแนนมากกว่าหรือเท่ากับ 50 ซึ่งทำให้เงื่อนไขของคำสัง f เป็นจริง จะแสดงข้อความว่า PASS แต่ถ้าเงื่อนไขของคำสัง if เป็นเท็จจะแสดงข้อความว่า FAIL ปรากฏออกทางหน้าจอ

```
#include<stdio.h>
 main()
{
      int score; printf("Enter Your Score = ");   // แสดงผลข้อความให้ป้อนคะแนน
      scanf("%d",&score);        // รอรับค่าคะแนนมาเก็บในตัวแปร score
      If(score>=50)        // ตรวจสอบคะแนนในตัวแปร score ว่ามากกว่า หรือเท่ากับ 50 หรือไม่
      printf("Congratulation! --> PASS");       // ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความนี้
      else          // ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานตามคำสั่งที่อยู่ถัดจากค าสั่ง else
      printf("Sorry! --> FAIL");       // กรณีที่เงื่อนไขเป็นเท็จให้แสดงข้อความนี้
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลลัพธ์และให้เราปอนคะแนน ดังนี้

```
Enter Your Score = _
```

ถ้าป้อนคะแนนมากกว่า หรือเท่ากับ 50 ซึ่งทำให้เงื่อนไขของ if เป็นจริง จะปรากฏข้อความ Congratulation! -> PASS ซึ่งจะปรากฏผลลัพธ์ทางหน้าจอภาพ ดังนี้

```
Enter Your Score = 75
Congratulation! --> PASS
```

ถ้าป้อนคะแนนน้อยกว่า 50 ซึ่งทำให้เงื่อนไขของ if เป็นเท็จ จะปรากฏข้อความ Sorry! -> FAIL ซึ่งจะปรากฏผลลัพธ์ทางหน้าจอภาพ ดังนี้

```
Enter Your Score = 39
Sorry! --> FAIL
```

ตัวอย่าง โปรแกรมคำนวณค่าบัตรผ่านประตูเข้าชมพิพิธภัณฑ์ โดยพิจารณาจากส่วนสูงของผู้เข้าชมตามเงื่อนไขดังนี้

1. ถ้าส่วนสูงน้อยกว่าหรือเท่ากับ 130 เซนติเมตร คิดค่าบัตรผ่านประตู 50 บาท
2. ถ้าส่วนสูงมากกว่า 130 เซนติเมตรขึ้นไป คิดค่าบัตรผ่านประตู 100 บาท

```
#include<stdio.h>
main()
{
        int height; printf("Enter Your height = ");   // แสดงผลข้อความให้ป้อนส่วนสูงเข้ามา
        scanf("%d",&height);    // รอรับตัวเลขมาเก็บในตัวแปร
        height if(height<=130)    // กำหนดเงื่อนไข ถ้าค่าของตัวแปร height น้อยกว่า หรือเท่ากับ 130
        printf("Ticket = 50 Bath");  // ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความนี้
        else   // ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานตามคำสั่งที่อยู่ถัดจากคำสั่ง else
        printf("Ticket = 100 Bath");   // ถ้าเงื่อนไขเป็นเท็จให้แสดงข้อความนี้
}
```

เมื่อสังรันโปรแกรมจะปรากฏผลลัพธ์และให้เราปอนความสูง ดังนี้

```
Enter Your height = _
```

ถ้าป้อนความสูงน้อยกว่า หรือเท่ากับ 130 ซึ่งทำให้เงื่อนไขของ if เป็นจริง โดยสมมุติว่า ป้อนความสูงเท่ากับ 110 ซึ่งจะทำให้เงื่อนไขของ if เป็นจริง จะปรากฏผลลัพธ์ ดังนี้

```
Enter Your height = 110
Ticket = 50 Bath
```

จากโปรแกรมตัวอย่างใช้เงื่อนไขของ if คือ (height<=130) แต่เรายังสามารถเขียนเงื่อนไข ของ if ได้ในอีกรูปแบบหนึ่ง คือ (height>130) โดยยังได้ผลลัพธ์แบบเดิม แต่จำเป็นต้องเปลี่ยนคำสั่ง ที่อยู่ถัดจาก if และ else ใหม่ตามไปด้วย ดังนี้

```
#include<stdio.h>
 main()
{
        int height;
        printf("Enter Your height = ");       // แสดงผลข้อความให้ป้อนส่วนสูงเข้ามา
        scanf("%d",&height);     // รอรับตัวเลขมาเก็บในตัวแปร height
        if(height>130)    // กำหนดเงื่อนไข ถ้าค่าของตัวแปร height มากกว่า 130
            printf("Ticket = 100 Bath");    // ถ้าเงื่อนไขเป็นจริงให้แสดงข้อความนี้
        else      // ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานตามคำสั่งที่อยู่ถัดจากคำสั่ง else
            printf("Ticket = 50 Bath");    // ถ้าเงื่อนไขเป็นเท็จให้แสดงข้อความนี้
}
```

ตัวอย่าง โปรแกรมคำนวณค่าบัตรผ่านประตูเข้าชมพิพิธภัณฑ์ โดยพิจารณาจากส่วนสูงและอายุของ ผู้เข้าชมตามเงื่อนไขดังนี้

1. ถ้าส่วนสูงน้อยกว่าหรือเท่ากับ 130 เซนติเมตร คิดค่าบัตรผ่านประตู 50 บาท
2. ถ้าส่วนสูงมากกว่า 130 เซนติเมตรขึ้นไป คิดค่าบัตรผ่านประตู 100 บาท
3. ถ้าอายุเท่ากับ หรือมากกว่า 60 ปี จะได้รับส่วนลดค่าบัตรผ่านประตู 30 บาท

```
#include<stdio.h>
main()
{
        int height, ticket, age;
        printf("Enter Your height = ");
        scanf("%d",&height);
        if(height<=130)
            ticket=50;
```

```
else
       ticket=100;
     printf(“Enter Your Age = ”);
     scanf(“%d”,&age);
     if(age>=60)
         ticket=ticket-30;
     printf(“Ticket = %d Bath”,ticket);
}
```

ลักษณะการเขียนโปรแกรมประกอบไปด้วย คำสั่ง if-else ตามด้วยคำสั่ง if โดย คำสั่ง if- else ตัวแรก ทำหน้าที่คำนวณหาค่าค่าบัตรผ่านประตูเข้าชมพิพิธภัณฑ์ คำตอบที่ได้จะเก็บไว้ในตัวแปร ticket ส่วนคำสั่ง if ตัวที่สองทำหน้าที่ตรวจสอบอายุ ซึ่งถ้าเงื่อนไขเป็นจริงจะลดค่าบัตรผ่านประตูเข้า ชมลง 30 แล้วจึงนำค่าสุดท้ายไปแสดงผลบนหน้าจอ

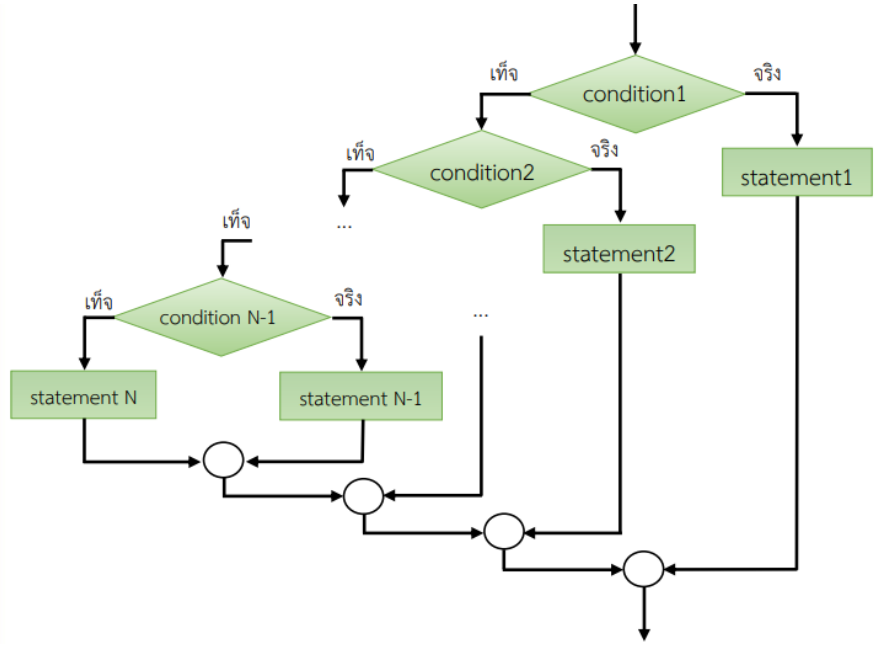
สรุป การทำงานของคำสั่ง if-else สองทางเลือก จะทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไข เป็นจริงจะทำงานตามประโยคคำสั่งชุดที่ 1 ถ้าเป็นเท็จจะทำงานตามประโยคคำสั่งชุดที่ 2 ที่อยู่หลัง else

4.4 คำสั่งเงื่อนไข if-else-if

คำสั่ง if-else if เป็นคำสั่งที่ใช้กำหนดให้โปรแกรมตัดสินใจเลือกทำคำสั่งอย่างใดอย่างหนึ่ง จากทางเลือกที่มีมากกว่า 2 ทางเลือก และในแต่ละทางเลือกจะมีการกำหนดเงื่อนไขของแต่ละ ทางเลือกไว้ด้วย โดยโปรแกรมจะทำการตรวจสอบเงื่อนไขของแต่ละทางเลือก หากพบว่าทางเลือกใด มีเงื่อนไขเป็นจริง (true) ก็จะทำงานที่ชุดคำสั่งภายในทางเลือกนั้น โดยจะไม่สนใจทางเลือกอื่นที่ยัง ไม่ได้ทำการตรวจสอบ แต่ในกรณีที่โปรแกรมตรวจสอบเงื่อนไขแล้วพบว่าเป็นเท็จ (false) ก็จะทำการ ตรวจสอบเงื่อนไขถัดไปเรื่อยๆ หากพบว่าทุกกรณีเงื่อนไขเป็นเท็จ โปรแกรมจะทำงานที่ชุดคำสั่ง ภายใต้คำสั่ง else ลำดับสุดท้าย (statetent Nt N) ซึ่งรูปแบบการเขียนคำสั่ง if-else if ในภาษาซี แสดงดังต่อไปนี้

```
if (condition1)
        statement1;
else  if (condition2)
        statement2;
...
...
else  if (condition N-1)
        statement N-1;
else
        statement N;
```

จากรูปแบบคำสั่ง if-else if สามารถเขียนผังงาน (Flowchart) ได้ดังนี้



ตัวอย่าง โปรแกรมป้อนตัวเลข 0-9 เข้ามาทางแป้นพิมพ์ แล้วให้แสดงผลเป็นคำภาษาอังกฤษ สามารถเขียนโปรแกรมภาษาซีได้ดังนี้

```
#include<stdio.h>
main()
{
        int number;
        printf(“Enter Number (0-9) : ”);
        scanf(“%d”,&number);
        if(number==0)
           printf(“0 : ZERO\n”);
        else if(number==1)
           printf(“1 : ONE\n”);
        else if(number==2)
           printf(“2 : TWO\n”);
        else if(number==3)
           printf(“3 : THREE\n”);
        else if(number==4)
           printf(“4 : FOUR\n”);
        else if(number==5)
           printf(“5 : FIVE\n”);
        else if(number==6)
           printf(“6 : SIX\n”);
        else if(number==7)
           printf(“7 : SEVEN\n”);
        else if(number==8)
           printf(“8 : EIGHT\n”);
        else if(number==9)
           printf(“9 : NINE\n”);
        else
           printf(“Number is invalid : Please Enter Number 0-9\n”);
}
```

เมื่อสั่งรันโปรแกรมจะปรากฏลัพธ์และให้เราป้อนตัวเลข ดังนี้

```
Enter Number (0-9) :
```

ถ้าป้อนตัวเลขเท่ากับ 7 ซึ่งจะทำให้เงื่อนไขของ if==7 เป็นจริง จะปรากฏข้อความ 7 : SEVEN ออกทางหน้าจอ ดังนี้

```
Enter Number (0-9) : 7
7 : SEVEN
```

จากโปรแกรม กรณีเมื่อพบเงื่อนไขที่เป็นจริงแล้ว โปรแกรมจะไม่สนใจเงื่อนไขลำดับถัดไป อีกเลย ถึงแม้ว่าเงื่อนไขที่อยู่ลำดับถัดไปจะเป็นจริงก็ตาม ดังตัวอย่างเช่น

```
else  if(number==7)
      printf("7 : SEVEN\n");
else  if(number==8)
      printf("8 : EIGHT\n");
```

---------->

```
else  if(number==7)
      printf("7 : SEVEN\n");
else  if(number==7)
      printf("8 : EIGHT\n");
```

เมื่อรันโปรแกรมแล้วป้อนตัวเลขเท่ากับ 7 จะปรากฏข้อความ 7 : SEVEN ออกทางหน้าจอเพราะเป็นเงื่อนไขแรกที่เป็นจริง โดยโปรแกรมจะไม่สนใจเงื่อนไขอื่นที่อยู่ถัดไปทั้งหมด

ตัวอย่าง โปรแกรมรับจำนวนตัวเลข 2 จำนวน และมีเมนูแสดงผลให้เลือกว่าต้องการจะคำนวณ ตัวเลขทั้ง 2 จำนวนนั้นอย่างไรเพื่อหาคำตอบ โดยมีเงื่อนไขดังนี้

เมนูที่ 1 คือ นำจำนวนทั้ง 2 มาบวกกัน

เมนูที่ 2 คือ นำจำนวนทั้ง 2 มาลบกัน

เมนูที่ 3 คือ นำจำนวนทั้ง 2 มาคูณกัน

```
#include
main()
{
        int num1, num2, operator, ans;
        printf(“Enter Number 1 = ”);
        scanf(“%d”,&num1);
        printf(“Enter Number 2 = ”);
        scanf(“%d”,&num2);
        printf(“\n\n Menu for Select”);
        printf(“\n Menu 1 : Add (+)”);
        printf(“\n Menu 2 : Subtract (-)”);
        printf(“\n Menu 3 : Multiply (*)”);
        printf(“\nPlease Enter Number (For Select Menu) : ”);
        scanf(“%d”,&operator);
        if(operator==1) {
             ans=num1+num2;
             printf(“%d + %d = %d”,num1,num2,ans);
         } else if(operator==2) {
              { ans=num1-num2;
              printf(“%d - %d = %d”,num1,num2,ans);
       } else if(operator==3) {
            ans=num1*num2;
            printf(“%d x %d = %d”,num1,num2,ans);
      } else
          printf(“Good Bye”);
 }
```

ถ้าป้อนตัวเลขที่หนึ่งเท่ากับ 9 และป้อนตัวเลขที่สองเท่ากับ 5 แล้วป้อนตัวเลขสำหรับเลือก เมนูการคำนวณเป็นเลข 1 ซึ่งก็คือการ บวก จะได้คำตอบเท่ากับ 14 และจะปรากฏข้อความออกทาง หน้าจอ ดังนี้ข

```
Enter Number 1 = 9
Enter Number 2 = 5

Menu for Select
  Menu 1 : Add (+)
  Menu 2 : Subtract (-)
  Menu 3 : Multiply (*)
Please Enter Number (For Select Menu) : 1
9 + 5 = 14
```

คำสั่งใน if แต่ละตัวถ้ามีมากกว่า 1 statement ต้องใช้เครื่องหมาย { } เพื่อปรับให้เป็น single statement มิฉะนั้นโปรแกรมจะเกิดข้อผิดพลาดไม่สามารถทำงานได้ตามที่ต้องการ

สรุป การทำงานของคำสั่ง if-else if หลายทางเลือกจะทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไข เป็นจริงจะทำงานตามประโยคคำสั่งชุดที่ 1 ถ้าเป็นเท็จจะทำการตรวจสอบเงื่อนไขต่อไป ถ้าเงื่อนไข ชุดที่ 2 เป็นจริงจะทำงานตามประโยคคำสั่งชุดที่ 2 แต่ถ้าเป็นเท็จอีกก็จะตรวจสอบเงื่อนไขชุดที่ 3 ต่อไป จนถึงเงื่อนไขสุดท้ายถ้าตรงกับเงื่อนไขใดก็จะทำงานตามประโยคคำสั่งของชุดเงื่อนไขนั้น

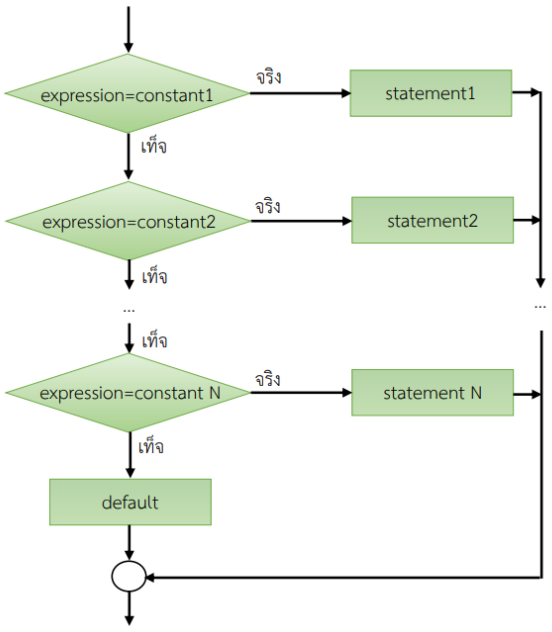
4.5 คำสั่งเงื่อนไข Switch

คำสั่ง switch เป็นคำสั่งตัดสินใจที่มีลักษณะการทำงานเช่นเดียวกับคำสั่ง if-else if คือ เลือกทางใดทางหนึ่ง จากทางเลือกที่มีมากกว่า 2 ทางเลือก ในแต่ละทางเลือกจะมีการกำหนดเงื่อนไข ของแต่ละทางเลือก โดยตรวจสอบเงื่อนไข หากพบว่าทางเลือกใดมีเงื่อนไขเป็นจริง (true) ก็จะทำงาน ที่ชุดคำสั่งภายในทางเลือกนั้น โดยจะไม่สนใจทางเลือกอื่นที่ยังไม่ได้ทำการตรวจสอบ รูปแบบการ เขียนคำสั่ง switch ในภาษาซีแสดงดังต่อไปนี้

```
switch(integer_expression) {
        case  constant_1    :  statement_1;
                               break;
        case  constant_2    :  statement_2;
                               break;
        ...
        ...
        case  constant_N    :  statement_N;
                               break;
        default             :  statement;
```

โดยที่ integer expression คือนิพจน์ของค่าจำนวนเต็มที่นำมาใช้ตรวจสอบเงื่อนไข case คือเงื่อนไขที่อยู่ในแต่ละกรณี ที่เป็นไปตาม constant นั้นๆ constant คือค่าคงที่ที่ใช้ตรวจสอบกับ integer_expression statement คือชุดคำสั่งที่ต้องการให้ทำงาน เมื่อเงื่อนไขในกรณีนั้นเป็นจริง default กรณีที่เงื่อนไขตรวจสอบไม่อยู่ในทุกกรณี ก็จะตกอยู่ในกรณี ของ default โดยอัตโนมัติ (อาจจะมีหรือไม่ก็ได้) break แต่ละ case จำเป็นต้องใส่คำสั่ง break ลงไปด้วยทุกครั้ง เพื่อให้หลุดออกจาก case มิฉะนั้นจะมีการตรวจสอบใน case ถัดไป ส่งผลให้ผลลัพธ์ผิดพลาด แต่คำสั่ง break ไม่จำเป็น ต้องใส่ลงในส่วนกรณีของ default

จากรูปแบบคำสั่ง switch สามารถเขียนผังงาน (Flowchart) ได้ดังนี้



การทำงานของคำสั่ง switch โปรแกรมจะตรวจสอบค่าจากตัวแปรที่อยู่ภายในคำสั่ง switch ซึ่งอาจเป็นสมการทางคณิตศาสตร์ หรือการคำนวณทางตรรกะใด ๆ ก็ได้ ว่าตรงกับ case ใด โปรแกรมก็จะเข้าทำงานในชุดคำสั่งเฉพาะที่อยู่ใน case นั้น ๆ เมื่อทำงานเสร็จจะพบคำสั่ง break เพื่อให้โปรแกรมไปทำงานนอกคำสั่ง switch ทันที แต่ในกรณีที่ไม่มีค่าใดใน case ตรงกับเงื่อนไข โปรแกรมก็จะเข้ามาทำงานที่ชุดคำสั่งภายในคำสั่ง default ทันที

ตัวอย่าง โปรแกรมป้อนตัวเลข 0-9 เข้ามามาทางแป้นพิมพ์ แล้วให้แสดงผลเป็นคำภาษาอังกฤษ โดยใช้รูปแบบการเขียนโปรแกรมด้วยคำสั่ง switch สามารถเขียนโปรแกรมภาษาซีได้ ดังนี้

```
#include<stdio.h>
main()
{
        int number;
        printf("Enter Number (0-9) : ");
        scanf("%d",&number);
        switch(number)
        {
          case 0 : printf("0 : ZERO\n"); break;
          case 1 : printf("1 : ONE\n"); break;
          case 2 : printf("2 : TWO\n"); break;
          case 3 : printf("3 : THREE\n"); break;
          case 4 : printf("4 : FOUR\n"); break;
          case 5 : printf("5 : FIVE\n"); break;
          case 6 : printf("6 : SIX\n"); break;
          case 7 : printf("7 : SEVEN\n"); break;
          case 8 : printf("8 : EIGHT\n"); break;
          case 9 : printf("9 : NINE\n"); break;
          default : printf("Number is invalid : Please Enter Number 0-9\n");
        }
}
```

```
Enter Number (0-9) : 5
5 : FIVE
```

ตัวอย่าง โปรแกรมคำนวณค่าจ้างพนักงาน โดยกำหนดให้รับข้อมูลประเภทของพนักงานและจำนวน ชั่วโมงทำงาน ซึ่งพนักงานแต่ละประเภทจะได้รับค่าจ้างต่อชั่วโมงต่างกัน ดังนี้

ประเภทของพนักงาน	อัตราค่าจ้าง/ชั่วโมง
a	30
b	40
c	45

```
#include <stdio.h>
main()
{
        int workHour;
        float workRate,totalPay;
        char type;
        printf("Enter type (a-c) : ");
        scanf("%s",&type);
        switch(type)
       {
        case 'a' : workRate=30.0f;
                break;
        case 'b' : workRate=40.0f;
                break;
        default : workRate=45.0f;
}
       printf("Enter Hours = ");
       scanf("%d",&workHour);
       totalPay=workRate*workHour;
       printf("Rate is %.2f \nTotal pay = %.2f",workRate,totalPay);
 }
```

เมื่อสั่งรันโปรแกรมจะปรากฏผลลัพธ์และให้เราป้อนประเภทของพนักงาน โดยทดลองป้อน เป็นประเภท a และป้อนจำนวนชั่วโมงทำงาน 8 ชั่วโมง จะปรากฏผลลัพธ์ออกทางหน้าจอ ดังนี้

```
Enter type (a-c) :  a
Enter Hours = 8
Rate is 30.00
Total pay = 240.00
```

สรุป คำสั่ง switch จะทำการตรวจสอบตัวแปรว่ามีค่าเท่ากับ หรือตรงกับ case ใด ถ้าตรง กับ case ใด ก็จะทำงานตามประโยคคำสั่งของ case นั้น สามารถนำมาใช้งานได้ดีกับโปรแกรมที่มี รายการเมนูให้เลือกและไม่สามารถนำมาใช้ตรวจสอบเงื่อนไขที่ใช้ตัวแปรและเลขจำนวนจริงได้

บทที่ 5

ฟังก์ชันและอาร์เรย์

5.1 ฟังก์ชัน (Function)

ฟังก์ชัน (Function) คือ ส่วนหนึ่งของโปรแกรมที่ถูกออกแบบมาให้ทำงานเฉพาะอย่าง โดยการเขียนคำสั่งที่สามารถนำกลับมาใช้ซ้ำได้ ช่วยให้โปรแกรมมีโครงสร้างที่เป็นระเบียบ ลดความซับซ้อนและแก้ไขได้ง่าย

ในภาษา C มีฟังก์ชันต่างๆ มากมายที่เราสามารถใช้ได้ เพื่อทำหน้าที่ต่างๆ ที่ต้องการ ในการใช้ฟังก์ชันโดยปกติต้อง include header file ของฟังก์ชันนั้นมาด้วย ตารางข้างล่างเป็นตัวอย่างฟังก์ชันบางส่วนในภาษา C

Function name	Library	Usage
scanf	stdio.h	getting keyboard input
printf	stdio.h	display text to screen
getch	conio.h	get character input
sqrt	math.h	get square root of number
floor	math.h	floor floating point of number to int
strlen	stdio.h	get characters' length

5.2 ความสำคัญของฟังก์ชันก์

- ช่วยลดความซับซ้อนของโปรแกรม
- สามารถนำกลับมาใช้ซ้ำได้
- เพิ่มความเป็นระเบียบในการพัฒนาโปรแกรม
- ลดความผิดพลาดจากการเขียนโค้ดซ้ำ

5.3 ส่วนประกอบของฟังก์ชัน

5.3.1 ชื่อฟังก์ชัน (Function Name)

เป็นชื่อที่ใช้ เรียกใช้งานฟังก์ชัน

ชื่อควรสื่อความหมายของงานที่ฟังก์ชันทำ เช่น add สำหรับการบวก, multiply สำหรับการคูณ

ตัวอย่าง:

```
int add(int a, int b) {  // ชื่อฟังก์ชันคือ add
    return a + b;
}
```

5.3.2 ตัวแปรนำเข้า (Parameters)

เป็น ตัวแปรที่รับค่าจากภายนอก เมื่อเรียกใช้ฟังก์ชัน

ทำให้ฟังก์ชัน ยืดหยุ่น สามารถทำงานกับข้อมูลหลายชุดได้

ตัวอย่าง:

```
int multiply(int a, int b) {  // Parameters: a, b
    return a * b;
}
```

5.3.3 คำสั่งที่อยู่ภายในฟังก์ชัน (Function Body)

เป็น บล็อกคำสั่ง ที่ฟังก์ชันจะทำงาน

สามารถมีตัวแปรภายในฟังก์ชัน (Local Variables) เพื่อใช้ประมวลผล

ตัวอย่าง:

```
int subtract(int a, int b) {
    int result = a - b;  // คำสั่งใน Function Body
    return result;
}
```

5.3.4 ค่าที่คืนกลับ (Return Value) เป็น ค่าที่ฟังก์ชันส่งกลับ ให้ส่วนที่เรียกใช้งาน

ช่วยให้ฟังก์ชันสามารถประมวลผลและส่งผลลัพธ์ออกไปได้

ตัวอย่าง:

```
int add(int a, int b) {
    return a + b;  // Return Value คือ a + b
}

int main() {
    int sum = add(5, 3);  // sum รับค่าที่ฟังก์ชันคืนกลับ
    printf("ผลรวม = %d\n", sum);
    return 0;
}
```

5.4 ประเภทของฟังก์ชัน

ฟังก์ชันในภาษา C สามารถแบ่งตามการส่งค่าไปยังฟังก์ชันและการคืนค่ากลับได้เป็น 4 ประเภท ดังนี้

5.4.1 ฟังก์ชันที่ไม่มีการส่งค่าและไม่มีการคืนค่า (No Parameter, No Return)

ลักษณะ: ไม่รับค่าใด ๆ และไม่ส่งค่ากลับ

ใช้สำหรับทำงานเฉพาะ เช่น แสดงข้อความ หรือทำงานซ้ำภายในฟังก์ชัน

ตัวอย่าง:

```
#include <stdio.h>
void greet() {  // ไม่มีพารามิเตอร์, ไม่มี return
    printf("สวัสดีครับ!\n");
}
int main() {
    greet();  // เรียกใช้งานฟังก์ชัน
    return 0;
}
```

5.4.2 ฟังก์ชันที่ไม่มีการส่งค่าแต่มีการคืนค่า (No Parameter, Return Value)

ลักษณะ: ไม่รับค่าใด ๆ แต่ส่งค่ากลับ

ใช้สำหรับประมวลผลข้อมูลภายในฟังก์ชันแล้วส่งผลลัพธ์

ตัวอย่าง:

```
#include <stdio.h>


int getRandomNumber() {  // ไม่มีพารามิเตอร์, มี return
    return 7;  // ส่งค่ากลับ
}
int main() {
    int num = getRandomNumber();
    printf("เลขสุ่ม = %d\n", num);
    return 0;
}
```

5.4.3 ฟังก์ชันที่มีการส่งค่าแต่ไม่มีการคืนค่า (Parameter, No Return)

ลักษณะ: รับค่าจากภายนอกแต่ไม่ส่งค่ากลับ

ใช้สำหรับประมวลผลข้อมูลและแสดงผล หรือเปลี่ยนสถานะบางอย่าง

ตัวอย่าง:

```
#include <stdio.h>
void printSum(int a, int b) {  // รับค่าพารามิเตอร์, ไม่มี return
    int sum = a + b;
    printf("ผลรวม = %d\n", sum);
}
int main() {
    printSum(5, 3);
    return 0;
}
```

5.4.4 ฟังก์ชันที่มีการส่งค่าและมีการคืนค่า (Parameter, Return Value)

ลักษณะ: รับค่าจากภายนอกและส่งค่ากลับ

ใช้สำหรับการประมวลผลข้อมูลและคืนผลลัพธ์ให้ส่วนที่เรียกใช้งาน

ตัวอย่าง:

```
#include <stdio.h>

int multiply(int a, int b) {  // รับพารามิเตอร์, ส่งค่ากลับ
    return a * b;
}
int main() {
    int result = multiply(4, 5);
    printf("ผลคูณ = %d\n", result);
    return 0;
}
```

5.5 การประกาศฟังก์ชัน

เนื้อหาที่เราจะเน้นในบทนี้คือฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง ในภาษา C นั้น คุณสามารถสร้างฟังก์ชันของคุณเองเพื่อที่จะให้ทำงานตามที่ต้องการได้ แนวคิดของฟังก์ชันก็คือนำโค้ดกลับมาใช้ใหม่โดยที่ไม่ต้องเขียนโปรแกรมใหม่ทั้งหมด ก่อนที่เราจะใช้ฟังก์ชันมันจำเป็นต้องถูกประกาศก่อน ซึ่งรูปแบบการประกาศฟังก์ชันในภาษา C

type เป็นประเภทของฟังก์ชันที่ต้องการประกาศ ซึ่งขึ้นกับค่าผลลัพธ์ของฟังก์ชันเป็นอะไร คุณสามารถประกาศ function type ได้เหมือนกับ primitive datatype เช่น int, float, char และอื่นๆ สำหรับฟังก์ชันที่ไม่มีการส่งค่ากลับจะมี type เป็น void

name เป็นชื่อของฟังก์ชันที่คุณต้องการสร้าง ในการตั้งชื่อฟังก์ชันมันมีกฎเช่นเดียวกันเหมือนกับการตั้งชื่อตัวแปร ชื่อของฟังก์ชันเป็นสิ่งที่เราจะใช้เมื่อต้องการใช้งานฟังก์ชัน

parameters เป็นตัวแปรที่ส่งเข้ามาในฟังก์ชัน พารามิเตอร์เป็นทางเลือกซึ่งสามารถมีหรือไม่ก็ได้

statement เป็นคำสั่งของโปรแกรมเพื่อที่จะให้ฟังก์ชันทำงานและได้ผลลัพธ์ที่ต้องการ

5.6 ฟังก์ชันพารามิเตอร์ (Function Parameters)

พารามิเตอร์ คือ ตัวแปรที่ถูกกำหนดใน ฟังก์ชัน เพื่อรับค่าจากภายนอก (Argument) เมื่อฟังก์ชันถูกเรียกใช้งาน พารามิเตอร์ช่วยให้ฟังก์ชันทำงานกับข้อมูลที่เปลี่ยนแปลงได้โดยไม่ต้องแก้ไขโค้ดภายในฟังก์ชัน

5.6.1 ประเภทของพารามิเตอร์

5.6.1.1 พารามิเตอร์ตำแหน่ง (Positional Parameters)

การส่งค่าเรียงตามตำแหน่งของฟังก์ชันที่ประกาศ

```
#include <stdio.h>

int add(int a, int b) {
    return a + b;   // a + b
}

int main() {
    int result = add(5, 3); // 5 -> a, 3 -> b
    printf("%d\n", result); // 8
    return 0;
}
```

อธิบาย

a = พารามิเตอร์ตำแหน่งตัวแรก รับค่า 5

b = พารามิเตอร์ตำแหน่งตัวที่สอง รับค่า 3

ผลลัพธ์คือ 5+3 = 8

5.6.1.2 พารามิเตอร์ค่าเริ่มต้น (Default Parameters)

ใน C ไม่มีค่าเริ่มต้นโดยตรง (ต่างจาก Python / C++)

แต่สามารถจำลองได้โดย เช็กค่า ที่ส่งเข้ามา

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

```
void greet(char name[ ]) {
    if (name[0] == '\0') { // ถ้าไม่มีค่า
        printf("สวัสดี Guest\n");
    } else {
        printf("สวัสดี %s\n", name);
    }
}

int main() {
    greet("");          // สวัสดี Guest
    greet("Alice");     // สวัสดี Alice
    return 0;
}
```

อธิบาย

name = พารามิเตอร์สำหรับรับชื่อ

ถ้าไม่ได้ส่งค่า ⟶ ใช้ "Guest"

ถ้าส่งค่า "Alice" ⟶ แสดง "สวัสดี Alice"

5.6.1.3 พารามิเตอร์แบบไม่ระบุชื่อ (Keyword Arguments)

C ไม่มี keyword arguments (เหมือน Python)

แต่ใช้ ตำแหน่งบังคับ เช่นเดียวกับข้อ 1.9.1

```
#include <stdio.h>

void student_info(char name[ ], int age) {
    printf("ชื่อ: %s, อายุ: %d\n", name, age);
}

int main() {
    student_info("Bob", 18);
    return 0;
```

```
}
```

อธิบาย

name = รับค่า "Bob"

age = รับค่า 18

แสดงผล "ชื่อ: Bob, อายุ: 18"

5.6.1.4 พารามิเตอร์ไม่จำกัดจำนวน (Variable-length Parameters)

ใน C ใช้ stdarg.h เพื่อรับค่าพารามิเตอร์ไม่จำกัดจำนวน

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

int sum_all(int count, ...) {
    va_list args;
    va_start(args, count);
    int total = 0;

    for (int i = 0; i < count; i++) {
        total += va_arg(args, int); // ดึงค่าทีละตัว
    }

    va_end(args);
    return total;
}

int main() {
    printf("%d\n", sum_all(4, 1, 2, 3, 4)); // 10
    return 0;
}
```

count = จำนวนพารามิเตอร์ที่จะส่ง

... = ตัวบอกว่ามีพารามิเตอร์ไม่จำกัด

va_arg = ดึงค่าพารามิเตอร์ทีละตัว

รวมผลลัพธ์ได้ 10

5.6.1.5 พารามิเตอร์แบบอ้างอิง (Reference Parameters)

C ไม่มี reference แบบ C++ แต่ใช้ pointer แทน

```
#include <stdio.h>

void addTen(int *x) {
    *x = *x + 10; // เปลี่ยนค่าตัวจริง
}

int main() {
    int num = 5;
    addTen(&num);
    printf("%d\n", num); // 15
    return 0;
}
```

อธิบาย

*x = พารามิเตอร์ pointer ที่ชี้ไปยัง num

&num = ส่ง address ของ num

ค่าของ num เปลี่ยนจาก 5 ⟶ 15

5.6.1.6 การใช้งานพารามิเตอร์กับอาร์เรย์

ฟังก์ชันสามารถรับอาร์เรย์เป็นพารามิเตอร์

```
#include <stdio.h>

int sum_array(int arr[ ], int size) {
    int total = 0;
    for (int i = 0; i < size; i++) {
        total += arr[i];
    }
    return total;
}

int main() {
    int numbers[ ] = {10, 20, 30, 40};
    int result = sum_array(numbers, 4);
    printf("%d\n", result); // 100
    return 0;
}
```

อธิบาย

arr[ ] = พารามิเตอร์อาร์เรย์ที่รับข้อมูลเข้ามา

size = ขนาดของอาร์เรย์

วนลูปบวกทุกค่า ⟶ ผลรวม 100

5.6.1.7 การใช้งานพารามิเตอร์กับอาร์เรย

ฟังก์ชันสามารถรับ **อาร์เรย์เป็นพารามิเตอร์** เพื่อประมวลผลข้อมูลจำนวนมากได้ เช่น

```
def sum_array(arr):
    total = 0
    for num in arr:
        total += num
    return total
```

```
numbers = [10, 20, 30, 40]
print(sum_array(numbers))  # 100
```

ส่วนนี้คือการประกาศฟังก์ชันชื่อ sum_array

```
def sum_array(arr):  ประกาศฟังก์ชัน รับพารามิเตอร์ชื่อ arr ซึ่งคาดว่าจะเป็น ลิสต์ (list) หรืออาร์เรย์ของตัวเลข
total = 0  กำหนดตัวแปร total สำหรับเก็บผลรวม เริ่มต้นที่ 0
for num in arr:  วนลูปเพื่อดึงสมาชิกแต่ละตัวจาก arr มาเก็บในตัวแปร num
total += num  บวกค่าของ num เข้าไปใน total ทีละรอบ
return total  ส่งค่าผลรวมทั้งหมดกลับไป
```

5.7 ฟังก์ชันอาร์กิวเมนต์

ฟังก์ชันอาร์กิวเมนต์ เป็นค่าของตัวแปรที่จะส่งเข้าไปในฟังก์ชัน ซึ่งจะต้องสอดคล้องกับฟังก์ชันพารามิเตอร์ ในตัวอย่างของคำสั่ง area(width, height) สังเกตุว่าฟังก์ชันอาร์กิวเมนต์จะมีสองตัว และประเภทของฟังก์ชันจะตรงกับลำดับของฟังก์ชันพารามิเตอร์ที่เราได้สร้างไว้

ค่า return ของฟังก์ชัน

ในการเขียนฟังก์ชันนั้น ส่วนมากจำเป็นต้องมีการส่งค่ากลับ โดยการใช้คำสั่ง return โดยในการประกาศฟังก์ชันประเภทของฟังก์ชันบ่งบอกถึงประเภทค่าที่ต้องการส่งกลับ จากในตัวอย่าง

```
float area(float, float);
```

สังเกตุว่า function type จะเป็น float นั่นหมายความว่าฟังก์ชันนี้จะมีค่าที่ส่งกลับเป็นเลขจำนวนจริง

นอกจากนี้ ในภาษา C เรายังสามารถสร้างฟังก์ชันที่ไม่มีค่า return ได้ โดยให้มี function type เป็น void มาดูตัวอย่าง

```
#include <stdio.h>
void introduce(char s[])
{
    printf("My name is %s\n", s);
}
void sayHello(char s[])
{
```

```
    printf("Hello %s\n", s);
}
int main()
{
    char name[] = "Mateo";

    introduce(name);
    sayHello("Tommy");
    return 0;
}
```

อธิบายทีละส่วน

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

เป็นการเรียกใช้ไลบรารีมาตรฐาน stdio.h

ไลบรารีนี้มีฟังก์ชันสำหรับ รับ–ส่งข้อมูล เช่น printf() และ scanf()

```
void introduce(char s[ ]) { ... }
```

เป็นการประกาศ ฟังก์ชัน introduce

void หมายถึงฟังก์ชันนี้ ไม่มีการคืนค่า

พารามิเตอร์ char s[ ] คือ อาร์เรย์ตัวอักษร (string) ที่ส่งเข้ามา

ภายในฟังก์ชันใช้ printf("My name is %s\n", s); เพื่อแสดงข้อความพร้อมชื่อที่ส่งมา

```
void sayHello(char s[ ]) { ... }
```

เป็นการประกาศ ฟังก์ชัน sayHello

มีพารามิเตอร์ char s[ ] เช่นเดียวกัน ใช้เก็บข้อความที่ส่งเข้ามา

ภายในฟังก์ชันใช้ printf("Hello %s\n", s); เพื่อพิมพ์ข้อความทักทาย

```
int main() { ... }
```

เป็นฟังก์ชันหลักที่โปรแกรมเริ่มทำงาน

char name[ ] = "Mateo"; สร้างตัวแปรอาร์เรย์ตัวอักษรเก็บชื่อ "Mateo"

introduce(name); เรียกฟังก์ชัน introduce โดยส่ง "Mateo" เข้าไป แสดงผล My name is Mateo

sayHello("Tommy"); เรียกฟังก์ชัน sayHello โดยส่ง "Tommy" เข้าไป แสดงผล Hello Tommy

return 0; ส่งค่า 0 กลับเพื่อบอกว่าการทำงานของโปรแกรมเสร็จสิ้นสมบูรณ์

5.8 อาร์เรย์ (Array)

อาร์เรย์ (Array) คือ โครงสร้างข้อมูลแบบลำดับ (Data Structure) ที่ใช้เก็บ ข้อมูลหลายค่าในตัวแปรเดียว ข้อมูลในอาร์เรย์จะมี ชนิดเดียวกัน เช่น ตัวเลขจำนวนเต็ม (int), ตัวอักษร (char), ตัวเลขทศนิยม (float)การเข้าถึงค่าภายในอาร์เรย์จะใช้ ดัชนี (Index) โดยค่าดัชนีเริ่มต้นที่ 0

5.8.1 คุณสมบัติสำคัญของอาร์เรย์

5.8.1.1 เก็บข้อมูลหลายค่าในตัวแปรเดียวช่วยให้จัดการข้อมูลจำนวนมากได้ง่ายขึ้น

5.8.1.2 ชนิดข้อมูลเดียวกันทั้งหมดช่วยให้การประมวลผลข้อมูลเป็นไปอย่างถูกต้อง

5.8.1.3 เข้าถึงข้อมูลด้วยดัชนี (Index)สามารถดึงหรือแก้ไขค่าข้อมูลใด ๆ ได้อย่างรวดเร็ว

5.8.1.4 จัดเก็บแบบเรียงลำดับต่อเนื่องในหน่วยความจำทำให้การวนลูปประมวลผลอาร์เรย์เร็ว

5.8.2 ความสำคัญของอาร์เรย์

อาร์เรย์เป็นโครงสร้างข้อมูลที่มีความสำคัญต่อการเขียนโปรแกรม เนื่องจากช่วยให้การจัดการข้อมูลเป็นระบบและมีประสิทธิภาพ โดยความสำคัญสามารถสรุปได้ดังนี้

5.8.2.1ลดจำนวนการประกาศตัวแปร

เมื่อโปรแกรมต้องเก็บข้อมูลจำนวนมาก การประกาศตัวแปรหลายตัวจะยุ่งยากและซับซ้อน

อาร์เรย์ช่วย เก็บค่าหลายค่าในตัวแปรเดียว ทำให้โค้ดสั้นลงและอ่านง่าย

ตัวอย่าง:

```
#include <stdio.h>

int main() {

    int numbers[5] = {10, 20, 30, 40, 50};  // เก็บค่าจำนวน 5 ค่าในตัวแปรเดียว

    for(int i = 0; i < 5; i++) {

        printf("numbers[%d] = %d\n", i, numbers[i]);

    }

    return 0;
```

}

อธิบายทีละส่วน

#include <stdio.h>

เรียกใช้ไลบรารีมาตรฐาน stdio.h เพื่อใช้ฟังก์ชัน printf สำหรับแสดงผล

int numbers[5] = {10, 20, 30, 40, 50};

ประกาศอาร์เรย์ชื่อ numbers ขนาด 5 ช่อง

เก็บค่า [10, 20, 30, 40, 50] ไว้ในแต่ละช่อง

index ของอาร์เรย์เริ่มที่ 0 ถึง 4

for(int i = 0; i < 5; i++)

ใช้ลูป for เพื่อวนซ้ำ 5 รอบ

ค่าของ i จะเป็น 0, 1, 2, 3, 4 ตามลำดับ

printf("numbers[%d] = %d\n", i, numbers[i]);

แสดงผลข้อความในรูปแบบ:

%d ตัวแรก = index (ตำแหน่ง)

%d ตัวที่สอง = ค่าของอาร์เรย์ในตำแหน่งนั้น

return 0;

บอกว่าการทำงานของโปรแกรมเสร็จสมบูรณ์

5.8.3 จัดเก็บข้อมูลจำนวนมากได้อย่างมีระบบ

อาร์เรย์เก็บข้อมูล เป็นลำดับเรียงต่อเนื่อง ทำให้สามารถจัดการข้อมูลจำนวนมากได้สะดวก ข้อมูลที่จัดเก็บในอาร์เรย์สามารถนำไปใช้งาน เช่น การคำนวณผลรวม การค้นหาค่ามากที่สุด หรือ ตรวจสอบเงื่อนไขต่าง ๆ

ตัวอย่าง:

```
#include <stdio.h>

int main() {

    int scores[5] = {80, 90, 75, 85, 95};

     int total = 0;

    for(int i = 0; i < 5; i++) {
```

```
total += scores[i];
    }
    printf("ผลรวมคะแนน = %d\n", total);
    return 0;
}
```

อธิบายทีละข้อ

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

เป็นการเรียกใช้ไลบรารี stdio.h เพื่อใช้ฟังก์ชันมาตรฐาน เช่น printf()

```
int scores[5] = {80, 90, 75, 85, 95};
```

ประกาศ อาร์เรย์ (array) ขนาด 5 ตัว เพื่อเก็บคะแนน

ค่าในอาร์เรย์ คือ {80, 90, 75, 85, 95}

```
int total = 0;
```

ประกาศตัวแปร total เพื่อเก็บค่าผลรวม เริ่มต้นที่ 0

```
for(int i = 0; i < 5; i++) { total += scores[i]; }
```

ใช้ ลูป for เพื่อบวกค่าทุกตำแหน่งในอาร์เรย์เข้ากับ total

รอบที่ 1: total = 0 + 80 = 80

รอบที่ 2: total = 80 + 90 = 170

รอบที่ 3: total = 170 + 75 = 245

รอบที่ 4: total = 245 + 85 = 330

รอบที่ 5: total = 330 + 95 = 425

สุดท้ายได้ total = 425

```
printf("ผลรวมคะแนน = %d\n", total);
```

แสดงผลรวมคะแนนออกทางหน้าจอ

ผลลัพธ์คือ

ผลรวมคะแนน = 425

```
return 0;
```

บอกว่าฟังก์ชัน main ทำงานสำเร็จและจบโปรแกรม

5.8.4 ทำให้การเข้าถึงและประมวลผลข้อมูลสะดวกการเข้าถึงสมาชิกในอาร์เรย์ทำได้โดยใช้ ดัชนี (Index)สามารถประมวลผลข้อมูลทั้งหมดในอาร์เรย์ได้ง่าย เช่น ใช้ ลูป ทำการวนรอบประมวลผล

ตัวอย่าง:

```
#include <stdio.h>
int main() {
    int numbers[5] = {2, 4, 6, 8, 10};
    // คำนวณผลรวมของอาร์เรย์
    int sum = 0;
    for(int i = 0; i < 5; i++) {
        sum += numbers[i];
    }
    printf("ผลรวมของอาร์เรย์ = %d\n", sum);
    return 0;
}
```

อธิบายทีละข้อ

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

ใช้สำหรับเรียกไลบรารีมาตรฐาน stdio.h เพื่อใช้ฟังก์ชัน printf()

```
int numbers[5] = {2, 4, 6, 8, 10};
```

ประกาศ อาร์เรย์ (array) ขนาด 5 ตำแหน่ง

ค่าในอาร์เรย์คือ {2, 4, 6, 8, 10}

```
int sum = 0;
```

ประกาศตัวแปร sum สำหรับเก็บผลรวม เริ่มจาก 0

```
for(int i = 0; i < 5; i++) { sum += numbers[i]; }
```

ใช้ลูป for วิ่งตั้งแต่ i = 0 ถึง i < 5 (รวมทั้งหมด 5 รอบ)

แต่ละรอบจะเอาค่าของ numbers[i] มาบวกกับ sum

การทำงานของลูป

รอบที่ 1: sum = 0 + 2 = 2

รอบที่ 2: sum = 2 + 4 = 6

รอบที่ 3: sum = 6 + 6 = 12

รอบที่ 4: sum = 12 + 8 = 20

รอบที่ 5: sum = 20 + 10 = 30

ผลรวมสุดท้ายคือ sum = 30

printf("ผลรวมของอาร์เรย์ = %d\n", sum);

แสดงค่าผลรวมออกทางหน้าจอ

ผลลัพธ์ที่ได้คือ

ผลรวมของอาร์เรย์ = 30

return 0;

บอกว่าฟังก์ชัน main ทำงานเสร็จสมบูรณ์

5.8.5 ประโยชน์ของอาร์เรย์

อาร์เรย์ในภาษา C เป็นเครื่องมือสำคัญสำหรับการจัดการและประมวลผลข้อมูลที่เป็นกลุ่ม ซึ่งมีประโยชน์หลายด้าน ดังนี้

5.8.5.1เก็บข้อมูลจำนวนมากในตัวแปรเดียว

อาร์เรย์ช่วยเก็บ ข้อมูลหลายค่าในตัวแปรเดียวทำให้โค้ดอ่านง่ายและจัดการข้อมูลได้สะดวก

ตัวอย่าง:

```
#include <stdio.h>
int main() {
    int scores[5] = {85, 90, 78, 92, 88};  // เก็บคะแนนนักเรียน 5 คน
    for(int i = 0; i < 5; i++) {
        printf("คะแนนนักเรียนคนที่ %d = %d\n", i+1, scores[i]);
    }
    return 0;
}
```

อธิบายทีละข้อ

#include <stdio.h>

เรียกใช้ไลบรารีมาตรฐาน stdio.h เพื่อให้ใช้ฟังก์ชัน printf() ได้

int scores[5] = {85, 90, 78, 92, 88};

ประกาศ อาร์เรย์ (array) ชื่อ scores ขนาด 5 ตำแหน่ง

เก็บค่าคะแนนของนักเรียน 5 คน คือ

คนที่ 1 = 85

คนที่ 2 = 90

คนที่ 3 = 78

คนที่ 4 = 92

คนที่ 5 = 88

```
for(int i = 0; i < 5; i++) { ... }
```

ใช้ลูป for เพื่อวนซ้ำ 5 รอบ (ตั้งแต่ i = 0 ถึง i = 4)

ในแต่ละรอบจะดึงค่าคะแนนจากอาร์เรย์ scores[i] มาแสดงผล

```
printf("คะแนนนักเรียนคนที่ %d = %d\n", i+1, scores[i]);
```

แสดงข้อความว่า "คะแนนนักเรียนคนที่ ... = ..."

ใช้ i+1 เพราะในอาร์เรย์ index เริ่มจาก 0 แต่ลำดับนักเรียนเริ่มจาก 1

เช่น รอบแรก i=0 ⟶ แสดง "คะแนนนักเรียนคนที่ 1 = 85"

```
return 0;
```

บอกว่าโปรแกรมทำงานเสร็จสมบูรณ์

5.8.6 ประมวลผลข้อมูลเป็นกลุ่ม (Batch Processing) อาร์เรย์ช่วย คำนวณหรือประมวลผลข้อมูลหลายค่าได้พร้อมกันสามารถใช้ลูปหรือฟังก์ชันร่วมกับอาร์เรย์เพื่อหาผลรวม, หาค่ามากที่สุด, หาค่าเฉลี่ย เป็นต้น

ตัวอย่าง:

```
#include <stdio.h>
int main() {
    int scores[5] = {85, 90, 78, 92, 88};
    int sum = 0;
    for(int i = 0; i < 5; i++) {
        sum += scores[i];  // ประมวลผลผลรวม
    }
    float average = sum / 5.0;
    printf("คะแนนเฉลี่ย = %.2f\n", average);
    return 0;
}
```

#include <stdio.h>

เรียกใช้ไลบรารีมาตรฐาน stdio.h เพื่อให้ใช้ฟังก์ชัน printf() ได้

int scores[5] = {85, 90, 78, 92, 88};

ประกาศอาร์เรย์ scores ขนาด 5 ตำแหน่ง เก็บคะแนนนักเรียน 5 คน

int sum = 0;

ประกาศตัวแปร sum สำหรับเก็บผลรวมของคะแนน เริ่มต้นที่ 0

for(int i = 0; i < 5; i++) { sum += scores[i]; }

ใช้ลูป for วนตั้งแต่ i=0 ถึง i=4

ในแต่ละรอบเอาค่าคะแนนจาก scores[i] มาบวกเข้ากับ sum

เมื่อครบ 5 รอบ sum จะเท่ากับผลรวมคะแนนทั้งหมด

float average = sum / 5.0;

คำนวณค่าเฉลี่ย โดยเอาผลรวม sum หารด้วย 5.0 (ใช้ 5.0 เพื่อให้ผลลัพธ์เป็น ทศนิยม)

printf("คะแนนเฉลี่ย = %.2f\n", average);

แสดงผลค่าเฉลี่ยออกมาในรูปแบบทศนิยม 2 ตำแหน่ง (%.2f)

return 0;

จบการทำงานของโปรแกรม

5.8.7 ใช้งานกับรายการหรือชุดข้อมูลต่าง ๆอาร์เรย์สามารถใช้ เก็บรายการสินค้า, ชื่อ, รหัสนักเรียน, หรือชุดข้อมูลอื่น ๆทำให้สามารถ ค้นหา, แก้ไข, หรือลบข้อมูล ได้ง่าย

ตัวอย่าง:

```
#include <stdio.h>
int main() {
    char products[3][20] = {"น้ำดื่ม", "ขนมขบเคี้ยว", "นม"}; // อาร์เรย์เก็บชื่อสินค้า
    for(int i = 0; i < 3; i++) {
        printf("สินค้ารายการที่ %d: %s\n", i+1, products[i]);
    }
    return 0;
}
```

#include <stdio.h>

เรียกใช้ไลบรารีมาตรฐาน stdio.h เพื่อใช้ฟังก์ชัน printf()

char products[3][20] = {"น้ำดื่ม", "ขนมขบเคี้ยว", "นม"};

ประกาศอาร์เรย์ 2 มิติสำหรับเก็บ ข้อความ (string)

3 หมายถึงเก็บชื่อสินค้าได้ 3 รายการ

20 หมายถึง แต่ละชื่อสินค้ามีความยาวได้สูงสุด 19 ตัวอักษร (บวก \0 ปิดท้าย)

กำหนดค่าเริ่มต้นให้เป็น "น้ำดื่ม", "ขนมขบเคี้ยว", "นม"

for(int i = 0; i < 3; i++)

ใช้ลูป for เพื่อวนแสดงชื่อสินค้า 3 รายการ (จาก i = 0 ถึง i = 2)

printf("สินค้ารายการที่ %d: %s\n", i+1, products[i]);

แสดงผลชื่อสินค้าแต่ละรายการ

%d ใช้แสดงตัวเลข (แสดง i+1 เพื่อให้เริ่มนับเป็น 1, 2, 3)

%s ใช้แสดงข้อความ (ชื่อสินค้าใน products[i])

return 0;

จบการทำงานของโปรแกรม

5.9. การประกาศอาร์เรย์ (Array Declaration)

การประกาศอาร์เรย์ต้องระบุ ชนิดข้อมูล และ ขนาดของอาร์เรย์

รูปแบบ: data_type array_name[array_size];

ตัวอย่าง:

int numbers[5]; // ประกาศอาร์เรย์จำนวน 5 ค่า ชนิด int

5.9.1 การกำหนดค่าให้กับอาร์เรย์ (Array Initialization)

สามารถกำหนดค่าเริ่มต้นได้เมื่อประกาศอาร์เรย์ หรือหลังจากประกาศแล้ว

วิธีที่ 1: กำหนดค่าเมื่อประกาศ

```
#include <stdio.h>
int main() {
    int numbers[5] = {10, 20, 30, 40, 50}; // กำหนดค่าเริ่มต้น
    for(int i = 0; i < 5; i++) {
        printf("numbers[%d] = %d\n", i, numbers[i]);
```

```
    }
    return 0;
}
```

วิธีที่ 2: กำหนดค่าทีหลัง

```
#include <stdio.h>
int main() {
    int numbers[5];  // ประกาศอาร์เรย์
    numbers[0] = 10;
    numbers[1] = 20;
    numbers[2] = 30;
    numbers[3] = 40;
    numbers[4] = 50;


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

ตัวอย่างภาษา C:

```
#include <stdio.h>
int main() {
    int numbers[5] = {10, 20, 30, 40, 50};

    printf("numbers[0] = %d\n", numbers[0]); // แสดงค่า 10
    printf("numbers[2] = %d\n", numbers[2]); // แสดงค่า 30
```

```
    return 0;
}
#include <stdio.h>
int main() {
    int matrix[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    printf("matrix[1][2] = %d\n", matrix[1][2]); // แสดงค่า 6
    return 0;
}
#include <stdio.h>
```

เรียกใช้ไลบรารีมาตรฐาน stdio.h เพื่อใช้ printf()

```
int matrix[3][3] = { ... };
```

ประกาศอาร์เรย์สองมิติชื่อ matrix ขนาด 3 x 3 (3 แถว 3 คอลัมน์)

ค่าในอาร์เรย์คือ

แถวที่ 0 ⟶ {1, 2, 3}

แถวที่ 1 ⟶ {4, 5, 6}

แถวที่ 2 ⟶ {7, 8, 9}

การเข้าถึงค่าในอาร์เรย์ 2 มิติ

ใช้รูปแบบ matrix[แถว][คอลัมน์]

matrix[1][2] หมายถึง แถวที่ 1 (บรรทัดที่สอง) และ คอลัมน์ที่ 2 (ตัวที่สามในบรรทัด)

ค่าที่อยู่ตรงนั้นคือ 6

```
printf("matrix[1][2] = %d\n", matrix[1][2]);
```

แสดงผลค่าในตำแหน่ง [1][2] ออกมาทางหน้าจอ

%d ใช้แสดงตัวเลขชนิดจำนวนเต็ม (int)

```
return 0;
```

ตัวอย่างโค้ดอาร์เรย์

การใช้ฟังก์ชันกับอาร์เรย์ใน C

สามารถส่ง อาร์เรย์เป็นพารามิเตอร์ ให้ฟังก์ชันประมวลผลข้อมูล

ตัวอย่างการยกกำลังสองสมาชิกในอาร์เรย์:

```
#include <stdio.h>
void square_array(int arr[], int size, int result[]) {
    for(int i = 0; i < size; i++) {
        result[i] = arr[i] * arr[i];
    }
}

int main() {
    int numbers[5] = {1, 2, 3, 4, 5};
    int squared[5];


    square_array(numbers, 5, squared);


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

อธิบายทีละข้อ

ฟังก์ชัน square_array

```
void square_array(int arr[], int size, int result[])
```

ประเภทฟังก์ชัน void ⟶ ไม่มีค่าคืนกลับ

พารามิเตอร์:

int arr[ ] ⟶ อาร์เรย์ต้นฉบับที่ต้องการหากำลังสอง

int size ⟶ ขนาดของอาร์เรย์

int result[] ⟶ อาร์เรย์เก็บผลลัพธ์

ฟังก์ชันนี้ วนลูปทุกตำแหน่งใน arr แล้วเก็บ $arr[i]^2$ ลงใน result[i]

ประกาศอาร์เรย์ใน main()

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

int squared[5];

numbers ⟶ เก็บค่าต้นฉบับ

squared ⟶ อาร์เรย์ว่างสำหรับเก็บผลลัพธ์

เรียกใช้ฟังก์ชัน

square_array(numbers, 5, squared);

ส่ง numbers (อาร์เรย์), 5 (ขนาด), squared (เก็บผลลัพธ์) เข้าไปในฟังก์ชัน

แสดงผลค่ากำลังสอง

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

วนลูปแสดงค่าจากอาร์เรย์ squared ทีละตัว

จบโปรแกรม

return 0;

บอกว่าการทำงานของ main() เสร็จสมบูรณ์