บทที่ 2

พื้นฐานภาษา C/C++

ในบทนี้ เราจะขอกล่าวถึงความรู้พื้นฐานเกี่ยวกับภาษา C/C++ ที่จำเป็นในการเขียนโปรแกรมเพื่อให้สามารถใช้งานอุปกรณ์อิเล็กทรอนิกส์และเซนเซอร์ต่าง ๆ ผ่านระบบไมโครคอนโทรลเลอร์

2.1 ตัวแปรในภาษา C/C++ และคำสั่งพื้นฐาน

ในภาษา C/C++ เราจะต้องประกาศชื่อและชนิดตัวแปรก่อนจะนำไปใช้เก็บข้อมูลหรือประมวลผล โดยตัวแปรในภาษา C/C++ สามารถแบ่งได้เป็น 4 ชนิดหลัก ๆ ได้แก่ ตัวอักษร (character) ตัวเลขจำนวนเต็ม (integer) ตัวเลขทศนิยม (floating point number) และค่าความจริง (true/false) โดยอาจแบ่งย่อยลงไปอีก ขึ้นกับการกำหนดในแต่ละระบบปฏิบัติการ (16, 32 หรือ 64 บิต) ตารางที่ 2.1 แสดงชนิดของตัวแปรในภาษา C/C++ โดยตัวแปรพื้นฐานที่ใช้งานได้แก่ ตัวอักษร, ตัวเลขจำนวนเต็มและตัวเลขทศนิยม โดยคำสำคัญที่ใช้ประกาศตัวแปรทั้ง 3 คือ char, int และ float ตามลำดับ ผู้ศึกษาควรจำคำเหล่านี้เพื่อใช้ในการเขียนโปรแกรม

ตัวอย่างการตั้งชื่อตัวแปร	ตัวอย่างการตั้งชื่อตัวแปรที่ผิด
int Relay1;	int 1relay; ผิด ชื่อตัวแปรขึ้นต้นด้วยตัวเลขไม่ได้
char buff;	char bool; ผิดเพราะว่า bool เป็นคำสงวน
float volt_LDR;	float volt LDR; ผิดเพราะชื่อตัวแปรมีเว้นวรรคไม่ได้
long distance;	

ตารางที่ 2.1 ชนิดตัวแปรในภาษา C/C++

ชนิดตัวแปร	คำสั่งสำหรับการประกาศ	ขนาด (บิต)	ช่วงของข้อมูล
ตัวอักษร*	char	8	-128 ถึง 127
ตัวอักษร	unsigned char	8	0 ถึง 255
จำนวนเต็ม	int	16	-32768 ถึง 32767
จำนวนเต็ม	unsigned int	16	0 ถึง 65535
จำนวนเต็ม	long	32	-2^{31} ถึง $2^{31}-1$
เลขทศนิยม	float	32	-3.4E+38 ถึง 3.4E+38**
เลขทศนิยม	double	64	-1.8E+308 ถึง 1.8E+308
ค่าความจริง	bool	8	True (= 1) หรือ False (= 0)

* เก็บด้วยรหัสแอสกี (ASCII Code)

**เราใช้ E แทนการเขียนด้วยสิบยกกำลัง (เช่น 3E+2 = 3×10^2)

สำหรับในภาษา C/C++ นั้นชื่อตัวแปรที่มีตัวพิมพ์เล็ก-ใหญ่ต่างกัน ถือเป็นตัวแปรคนละตัว (case sensitive) เช่นตัวแปรชื่อ relay เป็นคนละตัวแปรกับ Relay หรือ ReLay หรือ RELAY

โดยทุกคำสั่งในในภาษา C/C++ จะต้องลงท้ายด้วยเครื่องหมายเซมิโคลอน ; (หรือเครื่องหมายอัฒภาค) เพื่อบอกให้ภาษา C/C++ ทราบว่าเป็นการสิ้นสุดคำสั่งนั้น

เราสามารถใส่ คำหรือข้อความที่เราไม่ต้องการให้โปรแกรมประมวล แต่เขียนเพื่อบันทึกช่วยในการอ่านโค้ดนั้นได้ โดยทำได้ 2 วิธี คือ

1. ใส่เครื่องหมาย /* */ ครอบข้อความที่เราเขียน เช่น

```
/*  ข้อความนี้ไม่ถูกประมวลผลด้วยโปรแกรม  */
```

2. ใส่เครื่องหมาย // ไว้ด้านหน้าของข้อความ ทำให้โปรแกรมไม่ประมวลข้อความที่ตามมาทั้งบรรทัด เช่น

```
int x = 2;                  // ประกาศตัวแปร x ให้มีค่าเท่ากับ 2
```

สำหรับสตริง (String) ซึ่งเป็นข้อมูลที่สร้างได้จากอาเรย์ของตัวอักษร จะสามารถสร้างได้จากการประกาศอาเรย์ของตัวอักษร โดยในการใช้งานเบื้องต้นนั้น เราจะกำหนด ตัวอักษรที่อยู่ใน เครื่องหมายอัญประกาศเดี่ยว ‘ ’ เป็นชนิดตัวอักษรตัวเดียว (character) และตัวอักษรในเครื่องหมายอัญประกาศ “ ” เป็นข้อมูลชนิดสตริง เช่น “ABC”, “Hello”

สำหรับตัวอักษรพิเศษต่าง ๆ เช่น การกั้นหน้า (tab, \t) ที่มีการกำหนดในรหัสแอสกีมาตรฐานและไม่สามารถเขียนแทนด้วยตัวอักษรภาษาอังกฤษได้ เราก็จะใช้ เครื่องหมาย \ นำหน้า เพื่อใช้แจ้งให้โปรแกรมภาษา C/C++ ทราบ ‘\n’ คือ ตัวอักษรขึ้นบรรทัดใหม่ (newline character) และเมื่อโปรแกรมต้องแสดงผลตัวอักษรนี้ ก็จะแสดงโดยการเลื่อนเคอ-เซอร์ไปขึ้นบรรทัดใหม่

ในการเขียนภาษา C/C++ คำสั่งพื้นฐานจะมีลักษณะเป็นฟังก์ชัน โดยฟังก์ชันที่ถูกเรียกใช้บ่อย ๆ คือฟังก์ชันหรือคำสั่ง printf() ซึ่งฟังก์ชันนี้จะใช้เพื่อให้โปรแกรมพิมพ์แสดงข้อความออกมาทางจอภาพ (ฟังก์ชันนี้ถูกกำหนดไว้ในไลบรารีมาตรฐานของภาษา C คือใน stdio.h) รูปแบบการใช้งานคือ

printf(สตริง);

เช่นคำสั่ง printf(“Hello World.”); จะแสดงข้อความ Hello World. บนจอ โดยไม่แสดงเครื่องหมายคำพูดที่เป็นตัวกำหนดชนิดข้อมูล ซึ่งสตริงนี้อาจกำหนดให้มีข้อมูลที่เป็นตัวแปรและแทนที่ด้วยค่าตัวแปรในขณะแสดงผลได้ เช่นคำสั่ง printf(“value of x = %d”, x); จะแสดงข้อความ value x = ค่าตัวเลขจำนวนเต็มที่เก็บอยู่ในตัวแปรชื่อ x โดย %d หมายถึงการแสดงค่าตัวแปรที่ตามมานั้นเป็นค่าในเลขฐานสิบ (decimal value)

การดำเนินการในการเขียนโปรแกรมพื้นฐานคือการกำหนดค่า (assignment operator) โดยใช้เครื่องหมายเท่ากับ (=) นั่นคือ

ตัวแปร = ผลของการดำเนินการ

โดยเครื่องหมายเท่ากับนี้จะต่างกับเครื่องหมายเท่ากับในทางคณิตศาสตร์ เพราะ ในการเขียนโปรแกรม เครื่องหมายนี้จะหมายถึงการนำค่าที่ได้จากผลการดำเนิน (หรือผลลัพธ์จากการเรียกฟังก์ชัน) การลงไปเก็บในตัวแปรที่อยู่ด้านซ้านมือเท่านั้น

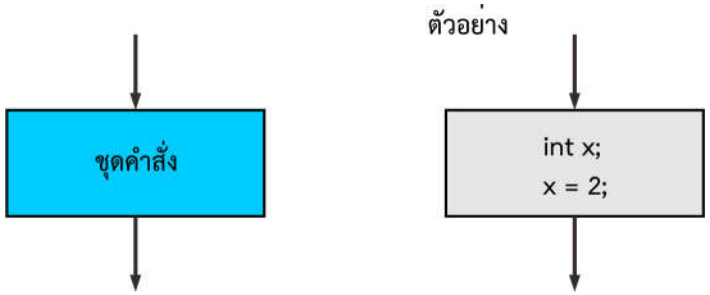
สำหรับการดำเนินการพื้นฐาน แบ่งได้เป็น 4 กลุ่ม ซึ่งสรุปไว้ในตารางที่ 2.2 สำหรับตัวดำเนินการที่มิใช่ตัวดำเนินการพื้นฐาน เช่น การหาค่ารากที่สอง การหาค่าจากการคำนวณเลขยกกำลัง x^y ภาษา C/C++ ก็สามารถดำเนินการได้โดยเรียกใช้คำสั่งหรือฟังก์ชันเฉพาะ เช่น sqrt(), pow() โดยอาจจะต้องเรียกใช้ไลบรารีเพิ่มเติมในตอนต้นของโปรแกรม โดยใช้คำสั่ง #include เช่น #include <math.h> เป็นการเพิ่มไลบรารีที่มีชุดคำสั่งในการคำนวณทางคณิตศาสตร์เพิ่มเติมเข้าไปในโปรแกรม โดยคำสั่งที่เพิ่ม จะมีแสดงอยู่ในไฟล์ math.h

ตารางที่ 2.2 ตัวดำเนินการในภาษา C/C++

กลุ่มตัวดำเนินการ	ตัวดำเนินการ	คำอธิบาย
ตัวดำเนินการคำนวณ	+ (บวก), – (ลบ) * (คูณ), และ / (หาร)	เครื่องหมาย * และ / จะถูกกระทำก่อนเครื่องหมาย + และ -
ตัวดำเนินการเชิงความสัมพันธ์ (relational operator)	> (มากกว่า), < (น้อยกว่า) >= (มากกว่าหรือเท่ากับ), <= (น้อยกว่าหรือเท่ากับ), == (เท่ากับ) และ != (ไม่เท่ากับ)	ผลลัพธ์จากตัวดำเนินการนี้จะเป็นค่าความจริง (จริงหรือเท็จ) เท่านั้น เช่น 2 => 1 ให้ค่าความจริงเป็นจริง และ 1 != 7/7 ให้ค่าความจริงเป็นเท็จ
ตัวดำเนินการทางตรรก (logical operator)	&& (และ) || (หรือ) และ ! (นิเสธ)	ใช้ในการกำหนดเงื่อนไขการทำงานของโปรแกรมโดยดำเนินการกับผลลัพธ์ที่ได้จากตัวดำเนินการความสัมพันธ์
ตัวดำเนินการประกอบ (compound operator)	++ (เพิ่มค่าอีกหนึ่งแล้วแทนที่) -- (ลดค่าลงหนึ่งแล้วแทนที่) += (เพิ่มค่าแล้วแทนที่) -= (ลดค่าแล้วแทนที่)	เช่น i++ หมายถึง i = i +1; j += 2 หมายถึง j = j + 2; โดยตัวดำเนินการนี้ทำให้สามารถเขียนโปรแกรมได้กระชับขึ้น

2.2 คำสั่งควบคุม

ในการเขียนโปรแกรมภาษา C/C++ เพื่อควบคุมการทำงานของคอมพิวเตอร์นั้น เราจะต้องกำหนดการดำเนินการคือการเขียนชุดคำสั่งให้โปรแกรมทำงานเป็นลำดับ ในกรณีอย่างง่ายที่สุดคือการทำงานตามลำดับดังแสดงเป็นแผนผังการทำงานได้ดังรูปที่ 2.1



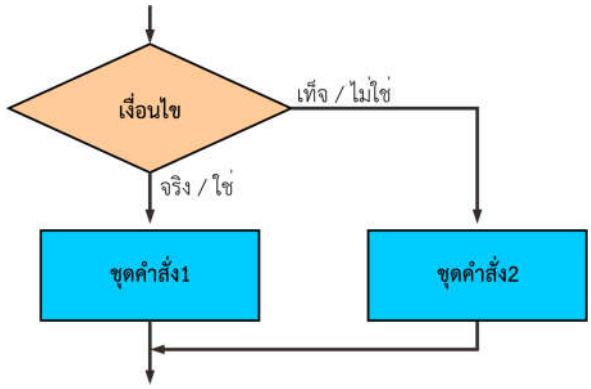
รูปที่ 2.1 แผนผังการทำงานตามลำดับและ
ตัวอย่างคำสั่งประกาศตัวแปร x และกำหนดค่า x = 2

ชุดคำสั่งควบคุมที่จะเป็นตัวกำหนดลำดับการทำงานแบ่งได้เป็น 2 ประเภท คือ แบบเลือกทางใดทางหนึ่งโดยใช้คำสั่ง if-else และ แบบวนลูปโดยการตรวจสอบเงื่อนไข คือคำสั่ง for โดยคำสั่งควบคุมทั้งสองประเภทจะใช้ผลลัพธ์จากการดำเนินการเชิงความสัมพันธ์ เป็นตัวตัดสินใจ

- ชุดคำสั่ง if-else มีรูปแบบคือ

```
if (เงื่อนไข) {
        ชุดคำสั่ง1
} else {
        ชุดคำสั่ง2
}
```

ซึ่งเงื่อนไขจะให้ผลลัพธ์ออกมาเป็นจริง (True) หรือเท็จ (False) ได้เท่านั้น โดยเมื่อเงื่อนไขเป็นจริง โปรแกรมก็จะทำชุดคำสั่ง1 และหากเป็นเท็จโปรแกรมก็จะทำชุดคำสั่ง2 โดยชุดคำสั่ง if-else นี้ สามารถเขียนแสดงได้ในรูปแผนผังการทำงาน ดังรูปที่ 2.2

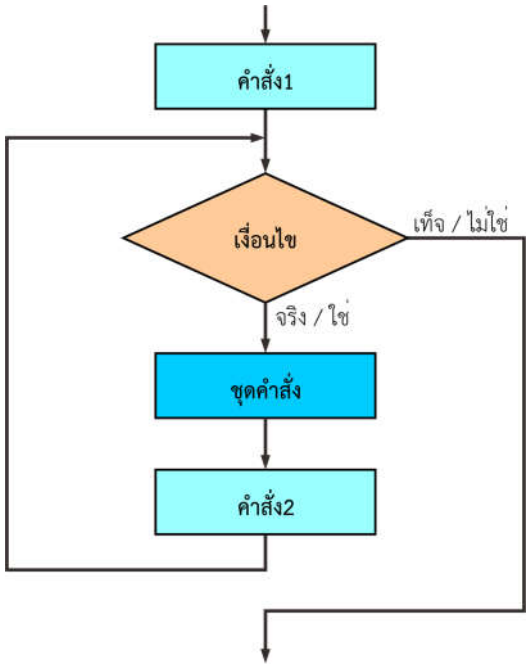


รูปที่ 2.2 แผนผังการทำงานของชุดคำสั่ง if-else

- ชุดคำสั่ง for เป็นชุดคำสั่งที่ใช้ในการวนลูป มีรูปแบบคือ

```
for (คำสั่ง1; เงื่อนไข; คำสั่ง2) {
        ชุดคำสั่ง
}
```

ในการทำงานของคำสั่ง for นี้ ก็จะเริ่มจากการทำคำสั่ง1 แล้วตรวจสอบเงื่อนไข จากนั้นจึงทำชุดคำสั่งภายในลูป 1 ครั้ง จึงทำคำสั่ง2 ก่อนตรวจสอบเงื่อนไข เพื่อดำเนินการในรอบถัดไป โดยหากเงื่อนไขที่ตรวจสอบให้ผลลัพธ์เป็นเท็จ โปรแกรมก็จะออกจากคำสั่ง for นี้แล้วดำเนินการในลำดับถัดไป ชุดคำสั่ง for นี้ สามารถเขียนแสดงได้ในรูปแผนผังการทำงาน ดังรูปที่ 2.3



รูปที่ 2.3 แผนผังการทำงานของชุดคำสั่ง for

ตัวอย่างการใช้งานชุดคำสั่ง for ที่พบเห็นได้บ่อยคือ

for (int i=0; i < 10; i++) { ชุดคำสั่ง }

ซึ่งเป็นการ กำหนดตัวแปรชื่อ i เป็นตัวแปรที่ใช้ในการวนลูป โดยเริ่มต้นที่ i = 0 แล้วจากนั้นให้ตรวจสอบว่า i มีค่าน้อยกว่า 10 หรือไม่ หากใช่ ก็จะทำงานในชุดคำสั่งในลูป for นี้ โดยเมื่อทำงานเสร็จ โปรแกรมก็จะเพิ่มค่า i อีก 1 ด้วยคำสั่ง i++ ซึ่งหมายความว่าให้ i+1 กลายเป็น i (i = i + 1) ดังนั้น ชุดคำสั่งในลูปนี้ก็จะถูกนำมาประมวล 10 ครั้ง

- ชุดคำสั่ง while เป็นชุดคำสั่งที่ใช้ในการวนลูปโดยตรวจสอบเงื่อนไขก่อน (คล้าย for) มีรูปแบบคือ

```
while (เงื่อนไข) {
        ชุดคำสั่ง
}
```

ในการทำงานของคำสั่ง while นี้ ก็จะตรวจสอบเงื่อนไข โดยหากผลการตรวจสอบเป็นจริง โปรแกรมก็จะทำชุดคำสั่งภายในลูป 1 ครั้ง แล้วจึงกลับมาตรวจสอบเงื่อนไขอีกที แผนผังการทำงานของชุดคำสั่ง while จะคล้ายกับของชุดคำสั่ง for แต่จะไม่มีคำสั่ง1 และคำสั่ง2 ในรูปแผนผัง (รูปที่ 2.3)

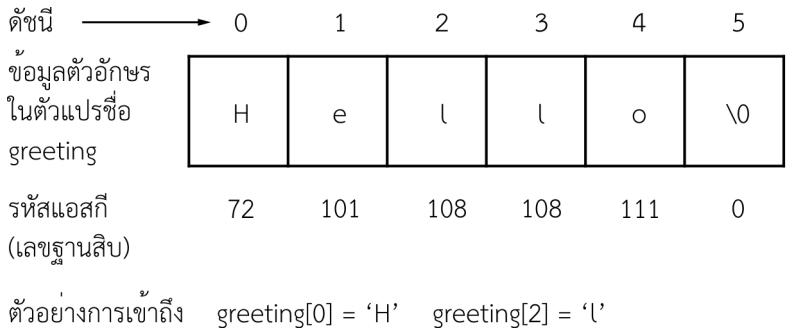
นอกจากคำสั่งควบคุมพื้นฐาน if-else, for และ while ที่กล่าวถึงในหัวข้อนี้ โปรแกรมภาษา C/C++ ยังมีคำสั่งควบคุมอีกหลายตัว ได้แก่ ชุดคำสั่ง switch-case และชุดคำสั่ง do-while นอกจากนี้ยังมีคำสั่งที่ใช้เปลี่ยนเส้นทางการดำเนินการคือคำสั่ง continue, break, goto และ return ที่มิได้กล่าวถึงในรายละเอียดในที่นี้

2.3 อาเรย์และฟังก์ชัน

อาเรย์ (array) เป็นตัวแปรชุดหรือตัวแปรลำดับ ที่เป็นข้อมูลชนิดเดียวกัน รูปแบบอาเรย์ที่มีใช้งานกันมากคืออาเรย์ของข้อมูล 1 มิติ (โดยในทางคณิตศาสตร์จะเรียกว่าเวกเตอร์) และอาเรย์ของข้อมูล 2 มิติ (ที่เรียกว่าเมทริกซ์) ในภาษา C/C++ เราสามารถประกาศตัวแปรอาเรย์ได้โดยใช้เครื่องหมายวงเล็บ [*N*] ตามหลังชื่อของอาเรย์โดย *N* คือขนาดของอาเรย์ ตัวอย่างเช่น ตัวแปรสตริงซึ่งเป็นอาเรย์ 1 มิติของตัวอักษร สามารถประกาศและกำหนดค่าได้ด้วยคำสั่ง

```
char greeting[6] = “Hello”;
```

ตัวอักษร 6 ตัวในข้อความข้างต้น จะถูกกำหนดลงในอาเรย์ของตัวแปรตัวอักษรที่ชื่อว่า greeting โดยมีลำดับการเก็บข้อมูลภายในหน่วยความจำแสดงได้ดังรูปที่ 2.4 โดยสัญลักษณ์ \0 ซึ่งเป็นตัวอักษรสุดท้ายที่ซ่อนอยู่ เป็นตัวอักษณะพิเศษที่ ใช้ในการบอกจุดสิ้นสุดของอาเรย์ข้อความ โดยอาเรย์ในภาษา C/C++ จะมีเลขดัชนี (index) เป็นจำนวนเต็มบวกที่เริ่มจาก 0 เสมอ และเราอาจไม่ใส่ขนาดของอาเรย์ก็ได้หากเรากำหนดข้อมูลที่เก็บในอาเรย์ทันที นั่นคือคำสั่ง char greeting[] = “Hello”; จะให้ผลเหมือนกับผลของตัวอย่างข้างต้น



รูปที่ 2.4 ลักษณะการเก็บข้อมูลแบบอาเรย์ 1 มิติ

บ่อยครั้งในโปรแกรมภาษา C/C++ ที่เราเขียนขึ้นที่เรามิได้ประกาศตัวแปรสตริง แต่นำมาใช้ เช่นการแสดงข้อความผ่านฟังก์ชัน printf ทั้งนี้เพราะว่าเรามิได้ต้องการเปลี่ยนแก้ไขหรือประมวลข้อความเหล่านั้น โดยสำหรับการดำเนินการเกี่ยวกับอาเรย์สตริงนั้น ก็สามารถทำได้ โดยภาษา C/C++ มีไลบรารี string.h ที่รวบรวมคำสั่งมากมายสำหรับการจัดการสตริง เช่น การคัดลอก (strcpy) การรวมสตริง (strcat) การหาความยาวสตริง (strlen) เป็นต้น

อาเรย์ของข้อมูลประเภทต่าง ๆ สามารถประกาศได้ในลักษณะเดียวกันกับอาเรย์ของตัวอักษร ตัวอย่างเช่น

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

เป็นการประกาศตัวแปรอาเรย์ 1 มิติ คือ a[0], a[1], และ a[2] โดยการเข้าถึงค่าของสมาชิกแต่ละตัวอาจทำโดยการใช้ชุดคำสั่ง for เช่น

```
int a[3];
for (int i = 0; i < 3; i++) {
    a[i] = 5*i;
}
```

โดยชุดคำสั่งข้างต้นนี้ จะกำหนดค่าให้กับตัวแปร a[0], a[1], และ a[2] ตามลำดับ

สำหรับอาเรย์สองมิติ ก็สามารถกำหนดได้โดยใช้วงเล็บ 2 ชุด ตัวอย่างเช่น

```
float M[3][2];
```

เป็นการประกาศตัวแปรอาเรย์ 2 มิติ ชื่อ M โดยมีสมาชิก 6 ตัว คือ M[0][0], M[0][1], M[1][0], M[1][1], M[2][0], และ M[2][1] ซึ่งการเข้าถึงหรือการเรียกใช้ค่าในอาเรย์นี้ ก็สามารถทำได้โดยใช้ชุดคำสั่ง for ซ้อนกัน เช่น

```
float M[3][2];
for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
                M[i][j] = 3.0*i + j;
                printf("i = %d, j = %d, M[%d][%d] = %f", i, j, i, j, M[i][j]);
        }
}
```

สำหรับการเข้าถึงอาเรย์อีกวิธีหนึ่ง คือการใช้พอยเตอร์ (pointer) ที่เป็นเสมือนตัวแปรที่เก็บข้อมูลของที่อยู่ของข้อมูลอีกทีหนึ่ง การใช้งานตัวแปรพอยเตอร์จะมีประโยชน์มากในการสร้างโครงสร้างข้อมูล (data structure) บางประเภท เช่น ลิงค์สิสต์ (linked list) และการจัดการกับข้อมูล เช่น การเรียงลำดับ (sorting) ผู้สนใจในส่วนนี้ สามารถศึกษาเพิ่มเติมได้จากหนังสือเกี่ยวกับการเขียนโปรแกรมภาษา C/C++ โดยตรง

ฟังก์ชัน (function) หรือคำสั่งในภาษา C/C++ เป็นสิ่งที่ผู้เขียนโปรแกรมนำมาใช้ โดยฟังก์ชันอาจแบ่งเป็นสองประเภท คือฟังก์ชันภายในตัวภาษา (built-in function) และฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (external function) ในหัวข้อนี้เราจะกล่าวถึงการสร้างฟังก์ชันขึ้นเอง สำหรับฟังก์ชันพื้นฐานภายในภาษา C/C+ และฟังก์ชันจากไลบรารีมาตรฐาน จะมีจำนวนมาก และผู้ใช้สามารถศึกษาได้จากเอกสารออนไลน์ได้โดยง่าย โดยการค้นหาคำที่เกี่ยวข้อง

รูปแบบของการประกาศฟังก์ชันในภาษา C/C++ คือ

```
ชนิดข้อมูลที่ส่งคืน ชื่อฟังก์ชัน (ชนิดและชื่อข้อมูลที่รับเข้า)
{
        ชุดคำสั่ง
        return( ชื่อตัวแปรที่ส่งค่าคืน );
}
```

โดยคำสั่ง return อาจใช้หรือไม่ใช้ก็ได้ ขึ้นกับว่า ฟังก์ชันที่เรียกมีการคืนค่าหรือไม่ หากไม่มีการคืนค่า จะใช้ คำว่า **void** บ่งบอกชนิดข้อมูลที่ส่งคืนว่า ไม่มีการส่งคืน (void แปลว่า ว่างเปล่า) ตัวอย่างเช่น

ตัวอย่างที่ 1

```
void setup() { }
```

คือฟังก์ชันชื่อ setup ที่ไม่มีการรับค่าใด ๆ เข้าและไม่มีการคืนค่ากลับออกไป

ตัวอย่างที่ 2

```
float cal_area(float x, float y)   {
        float area;
        area = x * y;
        return(area);
}
```

คือฟังก์ชันชื่อ cal_area ที่มีการรับค่าตัวแปร x และ y เข้าและมีการคืนค่าของตัวแปรชื่อ area กลับออกไป โดยตัวแปรที่ประกาศในฟังก์ชันนี้จะมีค่าเฉพาะภายในฟังก์ชันนี้เท่านั้น (เรียกว่า local variable) เมื่อโปรแกรมทำการคำนวณเสร็จ ก็จะล้างหน่วยความจำของตัวแปรภายในออกไป โดยหากเราต้องการให้ตัวแปรนั้นมีลักษณะที่ใช้ได้ทั่วไปตลอดทั้งโปรแกรม (เรียกว่า global variable) เราจะต้องทำการประกาศตัวแปรไว้ภายนอกฟังก์ชัน ซึ่งภาษา C/C++ จะทราบว่าตัวแปรที่ประกาศไว้ภายนอก จะเป็นตัวแปรที่สามารถใช้ได้ในทุกฟังก์ชันที่อยู่ภายในโปรแกรม

2.4 รูปแบบการเขียนโปรแกรมเชิงวัตถุเบื้องต้น

ในเชิงประวัติศาสตร์ ภาษา C++ เป็นภาษาที่พัฒนามาจากภาษา C โดยมีการเพิ่มความสามารถในการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) เข้าไปด้วย ซึ่งทำให้ภาษา C++ เป็นภาษาที่มีประสิทธิภาพสูง ผู้ใช้สามารถเขียนโปรแกรมได้หลากหลายรูปแบบ

การเขียนโปรแกรมเชิงวัตถุเป็นแนวคิดในการเขียนโปรแกรมที่กำหนดให้ “วัตถุ” (instance) มีทั้งข้อมูลและกระบวนการอยู่ภายใน โดยข้อมูลนั้นจะอยู่ในฟิลด์ (field) ที่ประกาศไว้ในตอนที่สร้างตัววัตถุ และ กระบวนการที่อยู่ภายในวัตถุจะถูกเรียกว่า เมธอด (method) ซึ่งเมธอดนี้คือฟังก์ชันประเภทหนึ่งที่กำหนดภายในโครงสร้างของ “วัตถุ” และใช้ข้อมูลที่เก็บในตัววัตถุนี้ในการประมวลผลด้วย โดยในการเขียนโปรแกรมขั้นต้น เราไม่จำเป็นต้องสร้างวัตถุขึ้นเอง แต่เราจะต้องเข้าใจรูปแบบของการเขียนโปรแกรมเชิงวัตถุนี้เพื่อให้สามารถนำ “วัตถุ” ที่ผู้อื่นกำหนดมาใช้ได้อย่างถูกต้อง

การใช้งานไลบรารีที่เขียนโปรแกรมเชิงวัตถุนี้ เริ่มต้นจากการประกาศชื่อตัวแปร “วัตถุ” ตามต้นแบบที่กำหนดอยู่ภายในไลบรารีที่เรียกใช้ ตัวอย่าง (นำมาจากบทที่ 6) เช่น ชุดคำสั่ง

```
#include <SH1106.h>
SH1106 display(0x3c, D1, D2);
```

เป็นการเพิ่มไลบรารี SH1106.h เข้ามาในโปรแกรม จากนั้นจึงประกาศตัวแปรวัตถุหรือ instance ที่ให้ชื่อว่า display ซึ่งจะนำโครงสร้างของวัตถุนี้มาจากคลาส (class) ชื่อ SH1106 มาใช้ โดยการสร้างวัตถุที่ชื่อว่า display นี้ถูกกำหนดให้เชื่อมต่อกับหน่วยความจำตำแหน่ง 0x3c และต่อกับขา D1 และ D2 โดย D1 และ D2 เป็นชื่อขาของไมโครคอนโทรลเลอร์ ที่มีการกำหนดตำแหน่งและวิธีการเข้าถึงในหน่วยความจำของเครื่องอยู่ก่อนแล้ว

ผู้ใช้สามารถเข้าถึงเมธอดหรือฟังก์ชันที่กำหนดในไลบรารีได้โดยการเปิดไฟล์ไลบรารีดูได้โดยตรง และการเรียกใช้งานเมธอดเหล่านั้นก็สามารถทำได้โดยการใช้จุด . ซึ่งคำสั่งจะมีรูปแบบคือ

วัตถุ.เมธอด(พารามิเตอร์รับเข้า) หรือ instance.method(parameters);
ตัวอย่างเช่น

display.setFont(ArialMT_Plain_16);

เป็นการเรียกใช้เมธอดชื่อ setFont ซึ่งประกาศในวัตถุนี้ โดยมีพารามิเตอร์ คือ ArialMT_Plain_16 ซึ่งเป็นกำหนดชนิดของตัวอักษรหรือฟอนต์ที่แสดงด้วยวัตถุชื่อ display โดยหากว่าเรามีวัตถุหลาย ๆ ตัว การกำหนดนี้ก็จะไม่ส่งผลใด ๆ ต่อวัตถุอื่น เพราะว่า ตัวแปรที่ถูกเปลี่ยนแปลงค่า ซ่อนอยู่ภายในวัตถุชื่อ display อีกทีหนึ่ง

แนวคิดการเขียนโปรแกรมเชิงวัตถุนี้อาจจะดูซับซ้อนสำหรับผู้ฝึกหัดเขียนโปรแกรม แต่หากศึกษาให้ลึกลงไปจะพบว่า การเขียนโปรแกรมขนาดใหญ่ เช่นโปรแกรมระบบปฏิบัติการ จะมีความซับซ้อนมากหากใช้วิธีการเขียนโปรแกรมแบบดั้งเดิม เนื่องจากจะพบความยุ่งยากในการจำกัดเขตของการกำหนดค่าของตัวแปรต่าง ๆ และการส่งค่าของข้อมูลระหว่างฟังก์ชันจะมีความซับซ้อนมาก การเขียนโปรแกรมเชิงวัตถุนี้ เป็นแนวคิดที่ทำให้การพัฒนาโปรแกรมทำงานได้ง่ายขึ้น โดยผู้ที่นำไลบรารีที่มีการเขียนโปรแกรมลักษณะนี้มาใช้จำเป็นจะต้องศึกษาเพียงฟังก์ชัน (หรือเมธอด) ที่อยู่ภายในเพื่อให้สามารถใช้งานได้ตามต้องการ สำหรับการนำแนวคิดของการเขียนโปรแกรมเชิงวัตถุมาใช้กับการพัฒนาฮาร์ดแวร์ด้านไมโครคอนโทรลเลอร์จะทำให้เราสามารถมองเสมือนว่า ฮาร์ดแวร์ต่าง ๆ ที่นำมาต่อ เป็นเสมือนกับวัตถุหนึ่ง ๆ ที่เราสามารถเขียนโปรแกรมสั่งการได้ตามความต้องการโดยตรง โดยการเรียกเมธอดที่มีอยู่มาใช้งาน