

C PROGRAMMING

Chapter 5 - chapter 8



Goals

5.

Loop
การเขียนโปรแกรมแบบวนซ้ำ

6.

Conditional Loop
การเขียนโปรแกรมแบบวนซ้ำ

7.

Function and Variable scope
ฟังก์ชันเบื้องต้นและขอบเขตของตัวแปร

8.

Array and Structure
ตัวแปรแถวลำดับและตัวแปรแบบโครงสร้าง

The background features several organic, flowing shapes in shades of pink, peach, and brown. In the top left, there are small brown dots and a larger brown shape. A large, light orange oval with a dashed white border is centered on the page. A large, dark brown question mark is positioned at the bottom right of this oval. The text "Chapter 5" and "Loop" is centered within the oval.

Chapter 5

Loop

Chapter 5

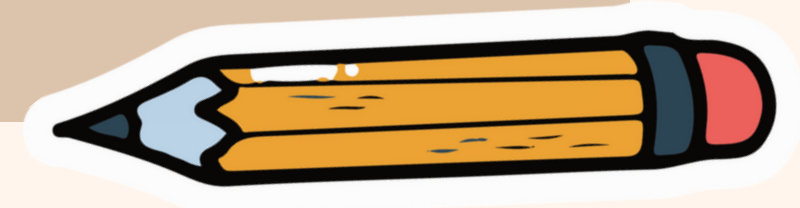


Loop คือ การสั่งให้คอมพิวเตอร์ทำงานซ้ำ ๆ เช่น สั่งให้คอมพิวเตอร์ทำการวนซ้ำ 10,000 รอบ เป็นต้น

Loop เหมาะกับ >>



- เหตุการณ์ที่เกิดขึ้นหลาย ๆ รอบ ซ้ำ ๆ
- เหตุการณ์ที่เกิดขึ้นหลาย ๆ รอบ โดยมี
การเปลี่ยนแปลงค่าบ่อย ๆ หรือวนซ้ำ
ตามเงื่อนไขที่กำหนดขึ้น





คำสั่ง While

วนซ้ำเรื่อย ๆ จนกว่า
เงื่อนไขจะไม่เป็นจริง

```
while (เงื่อนไข) {  
    คำสั่ง  
    คำสั่ง  
    ....  
}
```



คำสั่ง do-while

วนซ้ำเหมือน while
แต่จะมีการทำงานอย่าง
น้อย 1 ครั้ง

```
do {  
    คำสั่ง  
    คำสั่ง  
    ....  
} while (เงื่อนไข)
```



คำสั่ง for

วนซ้ำตามจำนวนรอบที่
แน่นอน

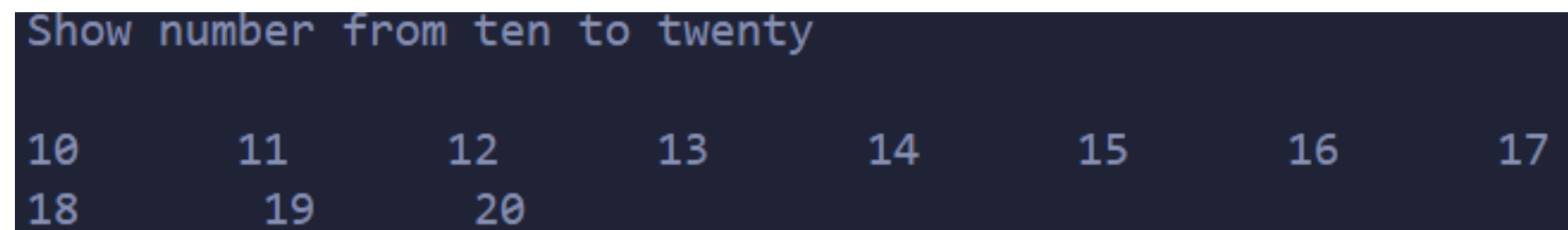
```
for (ค่าเริ่มต้น,เงื่อนไข  
    ,เพิ่ม/ลด ค่า) {  
    คำสั่ง  
    คำสั่ง  
    ....  
}
```





ตัวอย่างโปรแกรมแสดงผลด้วยตัวเลข 10 - 20 ด้วย **while**

```
#include <stdio.h>
int main ( )
{
    int count = 10;
    printf ("Show number from ten to twenty\n\n");
    while ( count <= 20 ) {
        printf ( "%d\t" , count );
        count++;
    }
    return 0;
}
```



```
Show number from ten to twenty
10      11      12      13      14      15      16      17
18      19      20
```





ตัวอย่างโปรแกรมหาผลรวม 1 ถึง 50 ด้วย for

```
#include <stdio.h>
int main ( )
{
    int sum = 0, count;
    for ( count = 1 ; count <= 50 ; count++) {
        sum = sum + count ;
    }
    printf ( "Summation of 1 to 50 = %d" , sum );
    return 0;
}
```

```
Summation of 1 to 50 = 1275
```





ตัวอย่างโปรแกรมหาผลรวม 1 ถึง 50 ด้วย do-while

```
#include <stdio.h>
int main ( )
{
    int count = 1, sum = 0;
    do {
        sum = sum + count;
        count++;
    } while ( count <= 50 );
    printf ( "%d" , sum );
    return 0;
}
```

1275





Chapter 6

Condition Loop

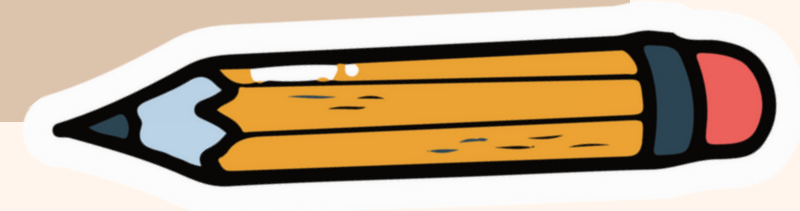
Chapter 6



Condition Loop คือ การนำการทำงานซ้ำ ๆ (Loop) มารวมกับการเขียนโปรแกรมแบบมีเงื่อนไข เช่น **if**, **if....else**, **if.....else if.....else** โดยการเพิ่มเงื่อนไขในการวนซ้ำหรือมีการตรวจสอบเงื่อนไขว่าให้วนซ้ำแบบใด

Condition Loop >>
ประกอบด้วยดังนี้

- คำสั่ง **while** (chapter 5)
- คำสั่ง **while + if**
- คำสั่ง **for + if**
- คำสั่ง **do-while + if**





โครงสร้างคำสั่ง while

```
while (เงื่อนไข) {  
    คำสั่ง  
    คำสั่ง  
    ....  
}
```

โปรแกรมแสดง 0...100 ใช้ while

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

← เงื่อนไข
วนซ้ำ

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 2  
2 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4  
1 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 6  
0 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 7  
9 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 9  
8 99 100
```





โครงสร้างคำสั่ง while + if

```
while (เงื่อนไข) {  
    if (เงื่อนไข) {  
        คำสั่ง  
    }  
    คำสั่ง  
    คำสั่ง  
    .....  
}
```

โปรแกรมแสดงเลขคู่ 0...100 ใช้ while + if

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

← เงื่อนไข
แสดงเลขคู่

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40  
42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78  
80 82 84 86 88 90 92 94 96 98 100
```





โครงสร้างคำสั่ง **for** + **if**

```
for (ค่าเริ่มต้น,เงื่อนไข,เพิ่ม/ลด ค่า) {  
    if (เงื่อนไข){  
        คำสั่ง  
    }  
    คำสั่ง  
    คำสั่ง  
    ....  
}
```



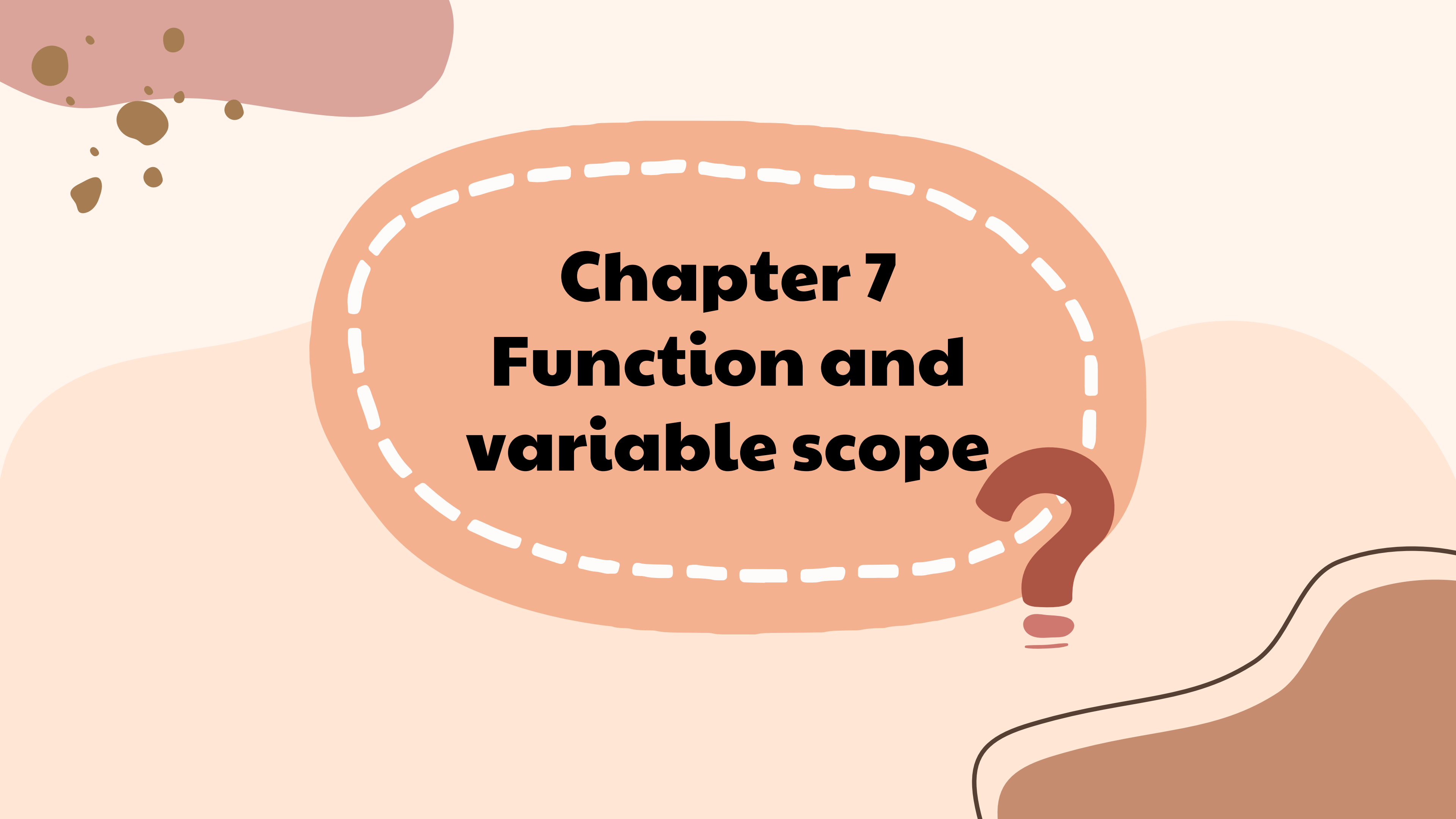
โปรแกรมตรวจสอบจำนวนสระใช้ for + if

```
#include <stdio.h>
#include <conio.h>
int main () {
    int vowel = 0, alphabet = 0, count;
    char letter;
    for ( count = 0; count < 10; count++ ) {
        printf ( "\nEnter letter a-z : " );
        letter = getch ();
        if ( ( letter == 'a' ) || ( letter == 'e' ) || ( letter == 'i' ) || ( letter == 'o' ) || ( letter == 'u' ) ) {
            vowel++;
        }
        else {
            alphabet++;
        }
    }
    printf ( "\n***Result***\n" );
    printf ( "Vowel ( a, e, i, o, u) = %d\n", vowel );
    printf ( "Other letter = %d", alphabet );
    return 0;
}
```

เรียกใช้ฟังก์ชัน getch() จาก library นี้

เงื่อนไข

```
Enter letter a-z : c
Enter letter a-z : o
Enter letter a-z : m
Enter letter a-z : p
Enter letter a-z : r
Enter letter a-z : o
Enter letter a-z : c
Enter letter a-z : o
Enter letter a-z : o
Enter letter a-z : l
***Result***
Vowel ( a, e, i, o, u) = 4
Other letter = 6
```

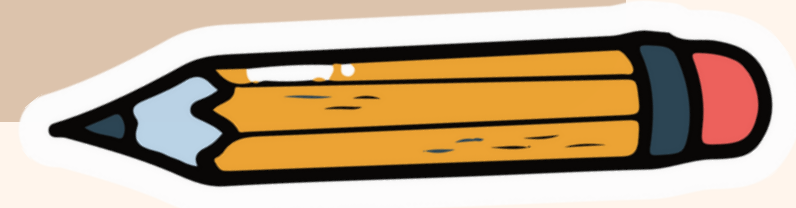


Chapter 7

Function and variable scope

Chapter 7

Function คือ การเขียนคำสั่งหลาย ๆ คำสั่งมารวมกันไว้ภายในเครื่องหมาย **{ }** เพื่อให้สามารถพัฒนาโปรแกรมได้โดยง่าย โดยไม่ต้องรู้การทำงานของฟังก์ชัน รู้แค่เพียงผลลัพธ์ของการทำงาน และไม่ต้องเขียนโปรแกรมที่ซับซ้อนหลาย ๆ ครั้ง





ฟังก์ชันแบ่งออกเป็น 2 ประเภท

- ฟังก์ชันไลบรารีมาตรฐาน (Standard Library Function) เป็นฟังก์ชันที่มีอยู่แล้วเก็บไว้ใน Library ในการใช้งานต้อง include directives หรือ header file เข้ามาในโปรแกรมของเรา

ตัวอย่าง

<code>#include <stdio.h></code>	<code>=> printf () , scanf ()</code>
<code>#include <math.h></code>	<code>=> sin () , cos () , log ()</code>
<code>#include <string.h></code>	<code>=> strcpy () , strcat ()</code>

- ฟังก์ชันที่สร้างขึ้นเอง (User Defined function)





ฟังก์ชันการคำนวณทางคณิตศาสตร์

```
#include <math.h>
```

```
sin (var);  
cos (var);  
tan (var);  
sqrt (var);           //รากที่ 2 เช่น sqrt (25) => 5.0  
pow (var1 , var2);    // var1 ยกกำลัง var2 เช่น pow (2,5) => 32.0  
log (var);             // log ฐาน e เช่น log (2) => 0.693147  
log10 (var);           // log ฐาน 10 เช่น log (100) => 2.0  
exp (var);             // e ยกกำลัง var เช่น exp (2) => 7.389056  
fabs (var);           // ค่าสัมบูรณ์ เช่น fabs (-5) => 5.0
```





ฟังก์ชันสำหรับข้อความ

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

```
strcpy (str1 , str2); // copy ข้อความจาก str1 ไปไว้ที่ str2  
strcat (str1 , str2); // ต่อข้อความ str2 หลัง str1  
strcmp (str1, str2); // เปรียบเทียบข้อความ str1 กับ str2 ว่าเหมือนไหม  
strlen (str); // หาความยาวของข้อความ
```

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

```
tolower (str1 , str2); // แปลงข้อความให้เป็นพิมพ์เล็ก  
toupper (str2 , str1); // แปลงข้อความให้เป็นพิมพ์ใหญ่
```





ฟังก์ชันที่สร้างขึ้นเอง (User-defined Function)

```
type function_name (type v1,type v2,...,type v){  
    คำสั่ง  
    คำสั่ง  
    ....  
}
```

type

คือ ชนิดของฟังก์ชันที่บอกว่าฟังก์ชันจะส่งข้อมูล
ชนิดใดกลับ เช่น `int,char,float,void,char []`

function_name

คือ ชื่อฟังก์ชัน

type v

คือ ชนิดข้อมูลที่รับเข้ามาใช้งานในฟังก์ชัน





ฟังก์ชันที่สร้างขึ้นเอง (User-defined Function)

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

```
void show_star (int n) {  
    int i;  
    for ( i=1 ; i<=n ; i++){  
        printf ("*");  
    }  
}
```

ฟังก์ชันที่
สร้างขึ้น

```
int main () {  
    int num = 9;  
    show_star(num);  
    printf ("\n* kmitl *\n");  
    show_star (num);  
    return 0;  
}
```

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

```
void show_star (int);
```

```
int main () {  
    int num = 9;  
    show_star(num);  
    printf ("\n* kmitl *\n");  
    show_star (num);  
    return 0;  
}
```

ต้องประกาศ
ชื่อฟังก์ชัน
ก่อน

```
void show_star (int n) {  
    int i;  
    for (i = 1; i <= n; i++)  
        printf ("*");  
}
```

ฟังก์ชันที่
สร้างขึ้น





ฟังก์ชันที่ไม่มีมีการส่งค่ากลับ

```
void sum_int (int n,int n2)
{
    int sum = 0;
    sum = n1 + n2;
    printf ("sum : %d", sum);
}
```

ฟังก์ชันที่มีการส่งค่ากลับ

```
int sum_int (int n1,int n2) {
    int sum = 0;
    sum = n1 + n2;
    return sum;
}
```

ต้องมี return

```
int x;
x = sum_int (2,5);
หรือ
printf ("sum : %d", sum_int (2,5));
```



ขอบเขตตัวแปรของการทำงานในฟังก์ชัน

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

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

ขอบเขตตัวแปรของการใช้งานในฟังก์ชัน

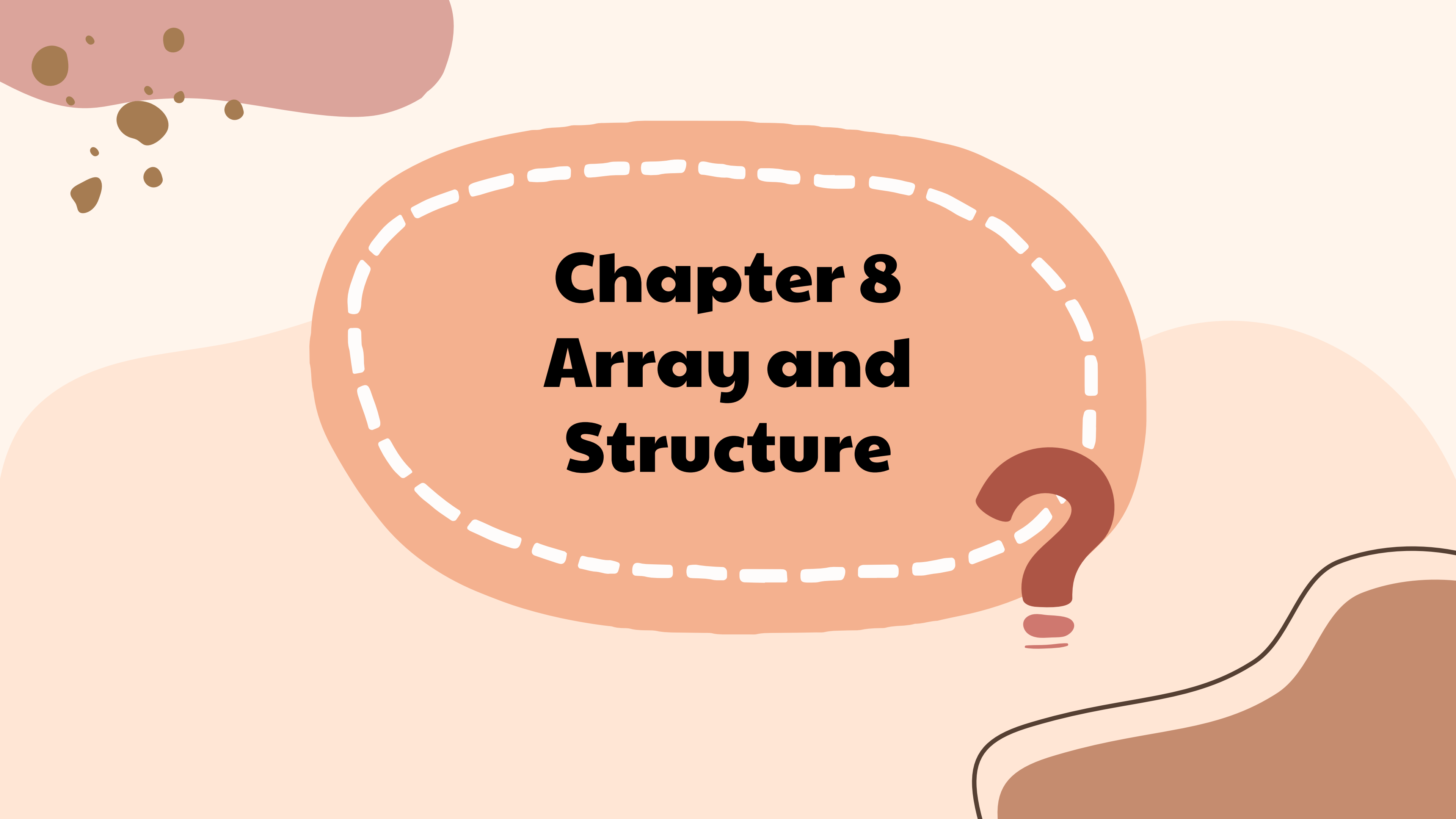
```
#include <stdio.h>
void test(void);    /*Function Prototype*/
int main ( )
{
    int num1;        // ตัวแปร local num1 ใน main ( )
    num1 = 19;
    printf ("line1 (main) : num1 = %d\n",num1);
    test ( );
    printf ("line2 (main) : num1 = %d\n",num1);
    return 0;
}
void test()
{
    int num1;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

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


ขอบเขตตัวแปรของการทำงานในฟังก์ชัน

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

```
variable_scope3.c:14:6: error: 'num1' undeclared
      ction)
      14 |         num1 = 26;
          |         ^~~~
```



Chapter 8

Array and Structure

Chapter 8



Array คือ ประเภทของข้อมูลที่สามารถเก็บข้อมูลประเภทเดียวกันแบบเป็นลำดับได้โดยข้อมูลนั้นจะอยู่ในตัวแปรตัวเดียวกันที่เรียกว่า **ตัวแปรอาร์เรย์** ใช้แก้ปัญหากรณีที่ต้องการเก็บข้อมูลประเภทเดียวกันหลาย ๆ ตัว

Array 1 มิติ

Index

2001

2544

2003

1998

2000

2565

0

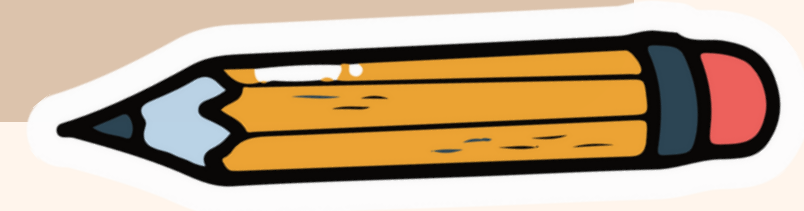
1

2

3

4

5





โครงสร้างคำสั่ง Array 1 มิติ

```
type array_name [n];
```

Type	ชนิดของตัวแปร
array_name	ชื่อของอาร์เรย์
n	ขนาดของอาร์เรย์

กำหนดค่าเริ่มต้น

```
int number [3] = {2001, 2015, 4000};
```

```
#include <stdio.h>
int main ( ) {
    int number[3] = {23, -186, 43};
    float value_2[5]={0.98,43.213,-3.47,52.08,-0.987};
    char vowel[5] = { 'a', 'e', 'i', 'o', 'u' };
    char vowel2[6] = "aeiou" ;
    char name[9] = { 'E', 'n', 'g', 'i', 'n', 'e', 'e', 'r', '\0' };
    printf ( "%s\n" , vowel );
    printf ( "%c\n" , vowel[0] );
    printf ( "%d\n" , number[1] );
    printf ( "%f\n" , value_2[2] );
    return 0;
}
```

```
aeiouHßz?⌘⌘,B{ᵍ^L∞QP¼|ᵗᵗ
a
-186
-3.470000
```





Array 2 မှတ်

		0	1	2	3	4
Index	0	15	16	2003	1998	2000
	1	200	400	90	199	400
	2	2001	2544	33	34	35
		←-----→				
		Column				





โครงสร้างคำสั่ง Array 2 มิติ

```
type array_name [n][m];
```

type ชนิดของตัวแปร
array_name ชื่อของอาร์เรย์
n จำนวนแถวของอาร์เรย์
m จำนวนคอลัมน์ของอาร์เรย์

กำหนดค่าเริ่มต้น

```
int number [n][m] = { 10, 11, 12,  
                      13, 14, 15 };
```

```
#include <stdio.h>
int main () {
    int number[2][3] = { 23, -186, 43,
                        11, 128, 300 };
    char name[2][20] = { "Computer",
                        "Programmimg" };

    printf ( "%d\n" , number[0][2] );
    printf ( "%d\n" , number[1][2] );
    printf ( "%s\n" , name[0] );
    printf ( "%c\n" , name[1][3] );
    return 0;
}
```

```
43
300
Computer
g
```





Structure ตัวแปรโครงสร้าง

Structure คือ การจัดกลุ่มข้อมูลตัวแปรหลาย ๆ ตัวไว้ในตัวแปรตัวเดียว และตัวแปรที่อยู่ภายใน structure มีชนิดตัวแปรที่แตกต่างกันได้

Person



name = "jame"
age = 30
gender = "male"



name = "Julia"
age = 21
gender = "female"



name = "two"
age = 27
gender = "male"





โครงสร้างคำสั่ง **struct**

```
Struct Person {  
    char name[20];  
    int age;  
    char gender[20];  
};
```

รูปแบบที่ 1

// การประกาศตัวแปร struct

```
Struct Person p1, p2;
```

// การเรียกใช้ข้อมูล

```
p1.name = "two";
```

```
p1.age = 27;
```

```
p1.gender = "male";
```

```
p2 = { "somying", 24, "female" };
```

```
Struct Person {  
    char name[20];  
    int age;  
    char gender[20];  
} p1, p2;
```

รูปแบบที่ 2

// การเรียกใช้งานข้อมูลที่อยู่ใน Struct

```
p1.name = "jame";
```

```
p1.age = 30;
```

```
p1.gender = "male";
```

```
p2.name = "julia";
```

```
p2.age = 21;
```

```
p2.gender = "female";
```



ประกาศตัวแปร struct แบบ Array

โปรแกรมข้อมูลนักเรียน 10 คน

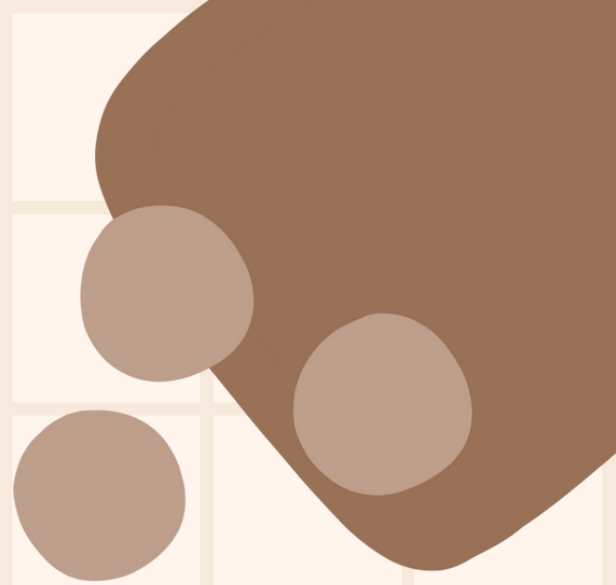
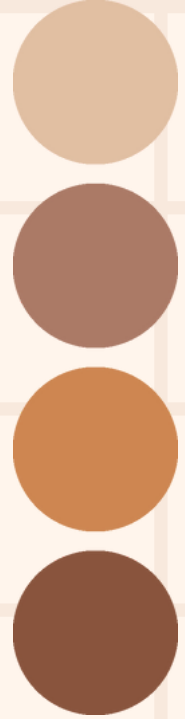
```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main() {
    int i;
    struct profile {
        char name[20];
        int age;
    } s[4];
```

← Array
ของ Struct

```
for ( i = 0 ; i < 10 ; i++) {
    printf( "Student[%d]\n" , i );
    printf( "\t name: " );
    scanf( "%s" ,s[i].name );
    printf( "\t age:" );
    scanf( "%d" , &s[i].age );
}
for( i = 0; i < 4; i++)
    if( s[i].age > 20 ) {
        printf( "\n %s,%d" ,s[i].name,s[i].age);
    }
return 0;
}
```

```
Student[0]
    name: game
    age:23
Student[1]
    name: love
    age:13
Student[2]
    name: lulu
    age:21
Student[3]
    name: jame
    age:23

game,23
lulu,21
jame,23
```



THANK YOU
SO MUCH!

