02204171 Structured Programming

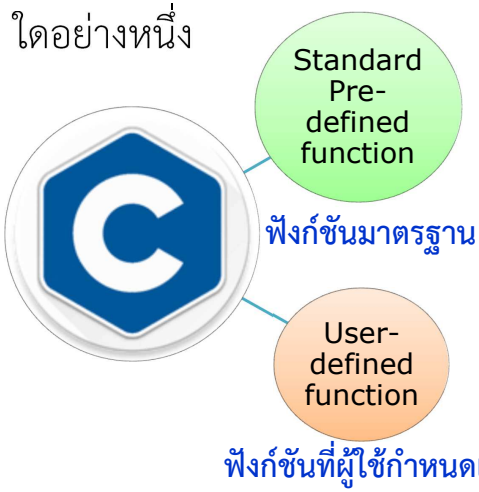
Chapter 11 : Function (part 2)

Outline

- Review: Function
- Variable Scope & Life time
- Advanced parameter passing
- Functions and Arrays

1. Review: Function

- **ฟังก์ชัน** คือ ส่วนย่อยของโปรแกรมที่เขียนขึ้นมาเพื่อทำงานอย่างใดอย่างหนึ่ง



ฟังก์ชันสามารถถูกเรียกใช้ภายใน main() หรือ ภายในฟังก์ชันอื่นก็ได้

1. Review: Function

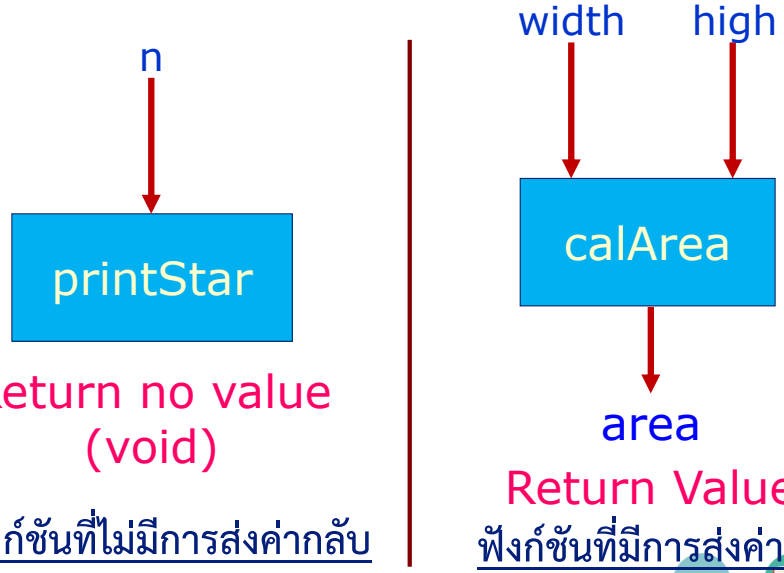
```c
#include <stdio.h>
void printStar(int n)
{
   int i;
   for(i=1;i<=n;i++)
      printf("*");
}
int main()
{
   int num=5;
   printStar(num);
   return 0;
}
```

1. Review: Function

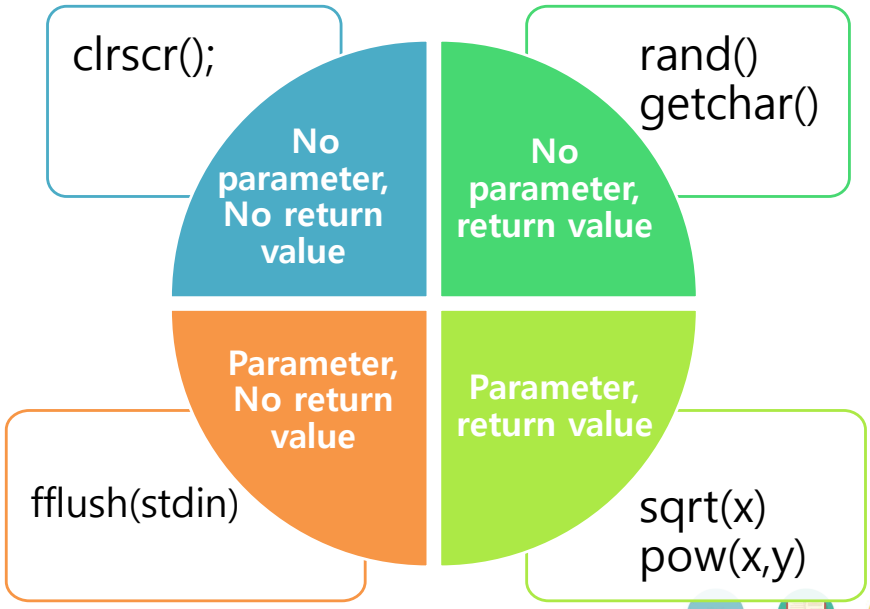
```c
#include <stdio.h>
float calArea(int w,int h)
{
    return 0.5*w*h;
}
int main()
{
   int width=5, high=10;
   float area;
   area=calArea(width,high);
   printf("%.2f",area);
   return 0;
}
```


1. Review: Function

Return vs. No Return Value Function



1. Review: Function



1. Review: Function

การส่งค่ากลับในกรณีคำสั่งเลือกทำ

```c
bool IsTriangle(int angle1, int angle2, int angle3)
{
        if(angle1+angle2+angle3==180)
                return 1;
}
```

```c
bool IsTriangle(int angle1, int angle2, int angle3)
{
        if(angle1+angle2+angle3==180)
                return 1;
        return 0;
}
```

1. Review: Function

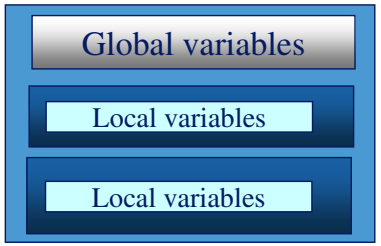
การส่งค่ากลับในกรณีคำสั่งเลือกทำ

```
#include<stdio.h>
#include<stdbool.h>
bool comparator(double x, double y)
{
      if(x>=y)
         return true;
      else
         return false;
}
int main()
{
      double x,y;
      printf("Enter x and y:" );
      scanf("%lf %lf", &x , &y);

      printf("%d",comparator(x,y));

      return 0;
}
```


2. Variable Scope & Life time

ขอบเขตของตัวแปรมี 2 แบบด้วยกัน คือ

- **ตัวแปรภายใน (Local variables)**
  - ตัวแปรที่ถูกประกาศขึ้น**ภายใน main()** หรือ **ภายในฟังก์ชัน**ใดๆที่สร้างขึ้นหรือใน block ใด
  - เรียกใช้ได้**เฉพาะภายใน block** ที่ประกาศมันเท่านั้น
- **ตัวแปรภายนอก (Global variables)**
  - ตัวแปรที่ถูกประกาศ**ภายนอก**ฟังก์ชันหรือไม่อยู่ใน block ใดเลย
  - สามารถมองเห็น และเรียกใช้งาน **ในทุก block หลังจากการประกาศมัน**



2. Variable Scope & Life time

Example 1: Local vs. Global Variable

```
int sum = 0;
void test()
{
    printf("In function sum= %d\n",sum);
    printf( "In function temp=%d",temp);
}

void main()
{
   int temp = 5;
   printf("In main sum=%d\n",sum);
   printf("In main temp=%d\n",temp);
   test();
}
```


2. Variable Scope & Life time

Example 2: Local vs. Global Variable

```
void test()
{
    printf("In function sum=%d\n", sum);
}

int sum = 0;
void main()
{
   scanf("%d",&sum);
   printf("In main sum=%d\n", sum);
   test();
}
```

2. Variable Scope & Life time

Example 3: Local vs. Global Variable

```
double c;
void sum(double a, double b)
{
   c = a + b;
}
int main()
{
   double a = 5,b = 10;
   sum(a,b);
   printf("c = %.2f",c);
   return 0;
}
```

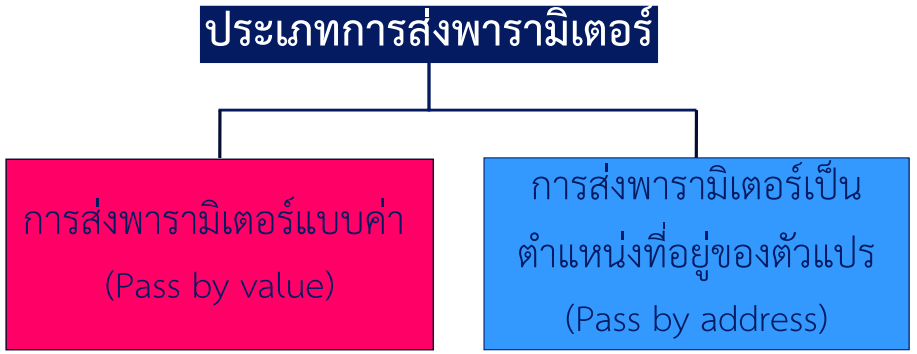

Quick check1: แสดงผลทางจอภาพของโปรแกรมนี้

```
int c=0;
void sum(int a, int b)
{
   int c;
   c = a + b;
   printf("In function c=%d\n", c);
   a = 0;
   b = 0;
}
int main()
{
  int a=5,b=10;
  sum(a,b);
  printf("%d %d\n",a, b);
  printf("In main c=%d\n", c);
  return 0;
}
```


Quick check2: แสดงผลทางจอภาพของโปรแกรมนี้

```
const int a=17;
int b,c;
void someFunc(float c)
{
   float b=2.3;
   printf("a=%d  b= %.1f  c= %.1f\n",a,b,c);
}
int main()
{
    b=4;
    c=6;
    someFunc(42.8);
    printf("a=%d  b=%d  c=%d\n",a,b,c);
  return 0;
}
```


3. Advanced parameter passing



- By value : ส่ง **ค่า**ของตัวแปร/ค่าคงที่ ที่ส่งเข้าฟังก์ชัน
- By address : ส่ง**ตำแหน่งที่อยู่**ของตัวแปรที่ส่งเข้าฟังก์ชัน

3. Advanced parameter passing

- **การส่งพารามิเตอร์แบบค่า**

⇨ Copy ค่าตัวแปร/ค่าคงที่ ที่ส่งเข้าฟังก์ชัน (เสมือนว่าตัวแปรที่กำหนดเป็นพารามิเตอร์นั้นเป็นตัวแปรตัวใหม่ที่ใช้ในฟังก์ชัน)

⇨ การเปลี่ยนค่าพารามิเตอร์ตัวนั้นในฟังก์ชันจึงไม่มีผลกระทบกับตัวแปร ที่ส่งค่าให้ฟังก์ชัน

```
void square(int x) {
  printf("%d\n",x);
  x = x*x;
}
void main(){
  int a;
  a=5;
  square(a);
  printf("%d\n",a);
}
```


3. Advanced parameter passing

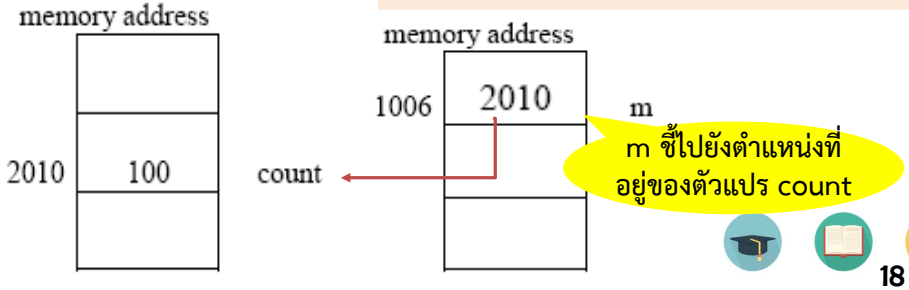
- **การส่งพารามิเตอร์เป็นตำแหน่งที่อยู่ของตัวแปร**

⇨ Copy ที่อยู่ในหน่วยความจำ (address) ของตัวแปรที่ส่งเข้าฟังก์ชัน

⇨ ใช้ตัวดำเนินการ & และ * ในการส่งและรับค่าฟังก์ชัน ตามลำดับ

```
int *m;
int count =100;

m= &count;
```

- **พอยน์เตอร์ (ตัวชี้)** คือ ตัวแปรหรือค่าคงที่ที่เก็บตำแหน่งที่อยู่ของหน่วยความจำ (address)
- **ตัวดำเนินการ &**: เป็นตัวดำเนินการที่จะเก็บค่าตำแหน่งที่อยู่ของหน่วยความจำ (address) ของตัวถูกดำเนินการ(operand)



3. Advanced parameter passing

- **การส่งพารามิเตอร์เป็นตำแหน่งที่อยู่ของตัวแปร**

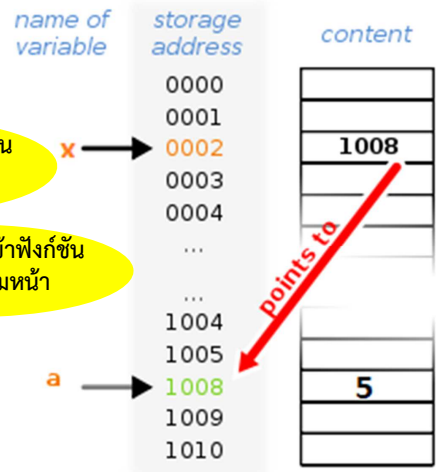
⇨ Copy ที่อยู่ในหน่วยความจำ (address) ของตัวแปรที่ส่งเข้าฟังก์ชัน

⇨ ใช้ตัวดำเนินการ & และ * ในการส่งและรับค่าฟังก์ชัน ตามลำดับ

```
void square(int *x) {
  printf("%d\n",*x);
   *x = *x * *x;
}
void main(){
  int a;
  a=5;
  square(&a);
  printf("%d\n",a);
}
```

พารามิเตอร์ในฟังก์ชัน ต้องมี * เติมหน้า

การส่งตัวแปรเข้าฟังก์ชัน ต้องมี & เติมหน้า



3. Advanced parameter passing

- **การส่งพารามิเตอร์เป็นตำแหน่งที่อยู่ของตัวแปร**

⇨ Copy ที่อยู่ในหน่วยความจำ (address) ของตัวแปรที่ส่งเข้าฟังก์ชัน

⇨ การเปลี่ยนแปลงค่าของตัวแปรที่อยู่ในฟังก์ชัน (Called function) ทำให้ค่าของตัวแปรที่ส่งเข้าฟังก์ชัน (Calling function) เปลี่ยนแปลงด้วย

```
void square(int *x) {
  printf("%d\n",*x);
   *x = *x * *x;
}
void main(){
  int a;
  a=5;
  square(&a);
  printf("%d\n",a);
}
```

การเปลี่ยนแปลงค่าที่ตัวแปร x ชี้ ก็คือ **การเปลี่ยนค่าตัวแปร a**

3. Advanced parameter passing

Example 4: Pass by value/ Pass by address

```
void sum1(int a, int b){
   int c;
   c = a + b;
   a = 0;
   b = 0;
   printf("%d %d\n",a,b);
}
int main()
{
  int a=5,b=10;
  sum1(a,b);
  printf("%d %d\n",a,b);
  return 0;
}
```


3. Advanced parameter passing

Example 5: Pass by value/ Pass by address

```
void sum2(int *a, int b){
   int c;
   c = *a + b;
   b = 0;
   *a = 0;
   printf("%d %d\n",*a,b);
}
int main()
{
  int a=5,b=10;
  sum2(&a,b);
  printf("%d %d\n",a,b);
  return 0;
}
```


Quick check3: แสดงผลทางจอภาพของโปรแกรมนี้

```
int test(int *a, int b) {
   int c;
   c = *a + b;
   *a *= b;
   b += c;
   printf("%d %d\n",*a,b);
   return c;
}
int c = 0;
int main() {
  int a=2,b=3;
  ---------------
 printf("%d %d %d\n",a,b,c);
 return 0;
}
```

2 1 3

(1) คำสั่งเรียกฟังก์ชัน	(2) ผลลัพธ์ทางจอภาพ	(3) ผลลัพธ์ทางจอภาพ
test(&a,b);		
c = test(&b,a);		
b = test(c,5);		
c=test(2,3);		
b = test(&c,5);		

3. Advanced parameter passing

- Summary

	Pass by value	Pass by address
กลไกการทำงาน (Passing Mechanism)	คัดลอกค่า (value) ของค่าตัวแปร/ค่าคงที่ ที่ส่งเข้าฟังก์ชัน	คัดลอกตำแหน่ง (address) ของตัวแปร ที่ส่งเข้าฟังก์ชัน
สิ่งที่ส่งผ่านฟังก์ชัน	ตัวแปร /ค่าคงที่	ตัวแปร **เท่านั้น**
ชนิดข้อมูลที่ส่งเข้าฟังก์ชัน	ชนิดข้อมูลพื้นฐาน	ชนิดข้อมูลพื้นฐาน (ใส่เครื่องหมาย &) / อาเรย์
การนำไปใช้ (Parameter Data Flow)	ส่งพารามิเตอร์ที่เป็น input ของฟังก์ชัน (Incoming)	ส่งพารามิเตอร์ที่เป็น input/output ของฟังก์ชัน (Incoming/Outgoing)

4. Function and Array

- การส่งค่าผ่านค่าตัวแปรอะเรย์ทั้งชุด (Array) ไปสู่ฟังก์ชัน เป็นการส่งพารามิเตอร์แบบ **Pass by address**
  - การเปลี่ยนแปลงค่าอะเรย์ในฟังก์ชัน ก็ส่งผลถึงอะเรย์ที่ส่งผ่านฟังก์ชัน
  - ดังนั้น ไม่จำเป็นต้องส่งอะเรย์กลับจากฟังก์ชัน (Return)
- ตัวอย่าง: การเขียนฟังก์ชัน (Function declaration)

```
void printArray(int data[ ], int nElements)
{
   ...
}
```

พารามิเตอร์สำหรับบอกขนาดของอะเรย์

การส่งผ่านอะเรย์สู่ฟังก์ชัน ส่วนใหญ่จะใช้ พารามิเตอร์อย่างน้อย 2 ตัว คือ อะเรย์ และ ขนาดของอะเรย์

4. Function and Array

Example 6: Pass array to Function

```
#include <stdio.h>
void inc(int [], int);
void printArray(int [], int);

void main() {
   int a1[] = {8, 4, 5, 3, 2};

   printArray(a1, 5);
   inc(a1, 5);
   printArray(a1, 5);
}
```

การส่งตัวแปรอะเรย์ทั้งชุดเข้าฟังก์ชันไม่ต้องมี & เติมหน้า

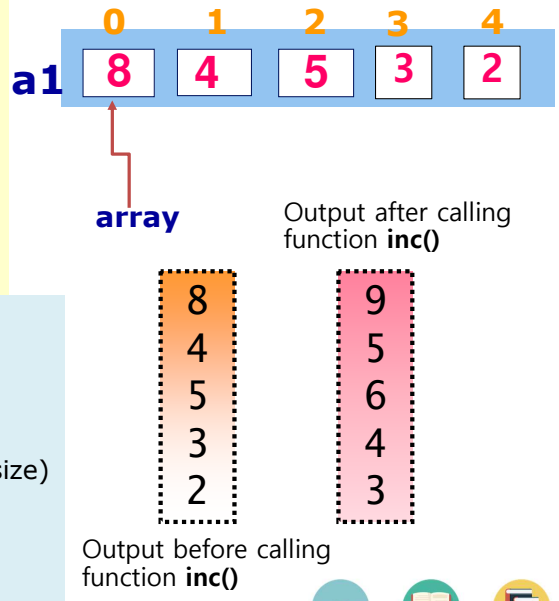
```
void inc(int array[], int size) {
   int i;
   for (i = 0; i < size; i++)
      array[i]++;
}

void printArray(int array[], int size)
{
   int i;
   for (i = 0; i < size; i++)
      printf("%d\n", array[i]);
}
```


4. Function and Array

Example 6: Pass array to Function

```
#include <stdio.h>
void inc(int [], int);
void printArray(int [], int);

void main() {
   int a1[] = {8, 4, 5, 3, 2};
   printArray(a1, 5);
   inc(a1, 5);
   printArray(a1, 5);
}
```

```
void inc(int array[], int size) {
   int i;
   for (i = 0; i < size; i++)
      array[i]++;
}
void printArray(int array[], int size)
{
   int i;
   for (i = 0; i < size; i++)
      printf("%d\n", array[i]);
}
```



Quick check4: แสดงค่าของตัวแปร x และ arr

```
void f(int z){
    z=z+1;
}
void g(int a[]){
    a[2]=8;
}
void h(int *z){
    *z = *z+1;
}
void main(){
   int x=6,arr[5];
   for (int i=0;i<5;i++)
      arr[i]=i;

   f(x);
   g(arr);
   h(&x);
   h(&(arr[4]));
}
```