5 ฟังก์ชั่น

Functions

เอกสารประกอบการสอนวิชา CP111

มหาวิทยาลัยศรีนครินทรวิโรฒ

เนื้อหา

Agenda

- รู้จักกับฟังก์ชั่น
- การสร้างฟังก์ชั่น
- การส่งผ่านตัวแปร
- ฟังก์ชั่นเรียกตัวเอง
- เอกสารอ้างอิงฟังก์ชั่นมาตราฐานของซี

รู้จักกับฟังก์ชั่น

Introduction to Functions

รู้จักกับฟังก์ชั่น

Introduction to Functions

- ฟังก์ชั่นคือชุดคำสั่งที่เราสามารถเรียกใช้งานได้
- เราได้มีการใช้ฟังก์ชั่นในบทที่ผ่านมาแล้ว
- ตัวอย่างการใช้งานฟังก์ชั่นที่พวกเราเคยใช้กัน
  - `printf("Hello World\n");`
  - `scanf("%d %d", &a, &b);`

ส่วนประกอบของฟังก์ชั่น

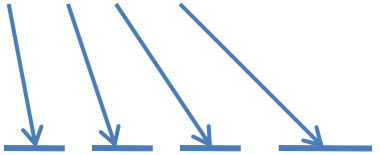
Function Components

- ชื่อฟังก์ชั่น
- ค่าที่ส่งเข้าไป หรือพารามิเตอร์ (parameters)
- ชุดคำสั่ง (statements)
- ค่าที่ส่งออกมา (return value)
  - อาจจะมีหรือไม่มีก็ได้ (optional)

ตัวอย่างฟังก์ชั่น printf

Function printf

พารามิเตอร์ตัวที่ 2 - 5

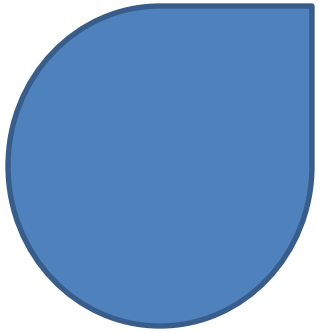


ชื่อฟังก์ชั่น

```
printf("The summation of %d %d and %d is %d\n", a, b, c, sum);
```

พารามิเตอร์ตัวแรก

(รูปแบบของการพิมพ์)



ชุดคำสั่งของ printf จะทำหน้าที่แสดงผลบนหน้าจอจากพารามิเตอร์ที่ใส่เข้าไป

ลำดับการทำงาน

Executing sequence

ex05_01.c

```
#include <stdio.h>

int max(int a, int b) {
   if (a > b)
      return a;
   else
      return b;
}

void main() {
   int b, c;
   b = 2;
   c = max(1, b);
   printf("c is %d\n", c);
}
```


อ. เอก ตังคนานนท์

ลำดับการทำงาน

Executing sequence

ex05_01.c

```
#include <stdio.h>

int max(int a, int b) {
   if (a > b)
      return a;
   else
      return b;
}

void main() {
   int b, c;
   b = 2;
   c = max(1, b);
   printf("c is %d\n", c);
}
```

พอถึงบรรทัดนี้

โปรแกรมจะทำงานในฟังก์ชั่น max(1, 2)

เสร็จแล้วจะกลับไปทำในบรรทัดต่อไป

ลำดับการทำงาน

Executing sequence

ex05_01.c

```
#include <stdio.h>

int max(int a, int b) {
   if (a > b)
      return a;
   else
      return b;
}

void main() {
   int b, c;
   b = 2;
   c = max(1, b);
   printf("c is %d\n", c);
}
```

ฟังก์ชั่นอาจจะถูกประกาศไว้นอกไฟล์นี้ก็ได้

เช่น printf ซึ่งถูกประกาศไว้ใน stdio.h

การสร้างฟังก์ชั่น

Creating a Function

การสร้างฟังก์ชั่น

Creating a Function

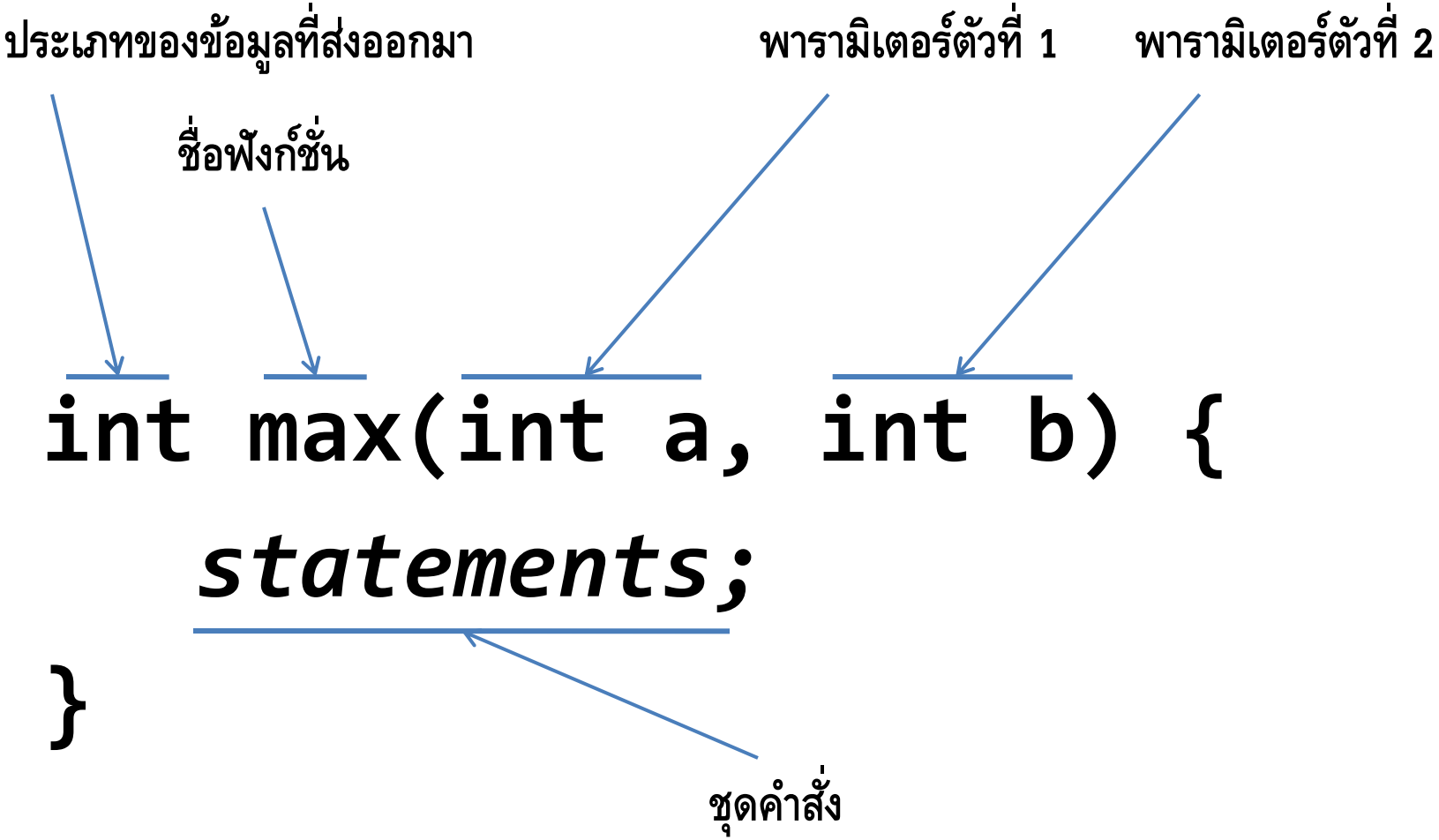
return-type function-name (***param***$_1$, ***param***$_2$, ...) {

    statements

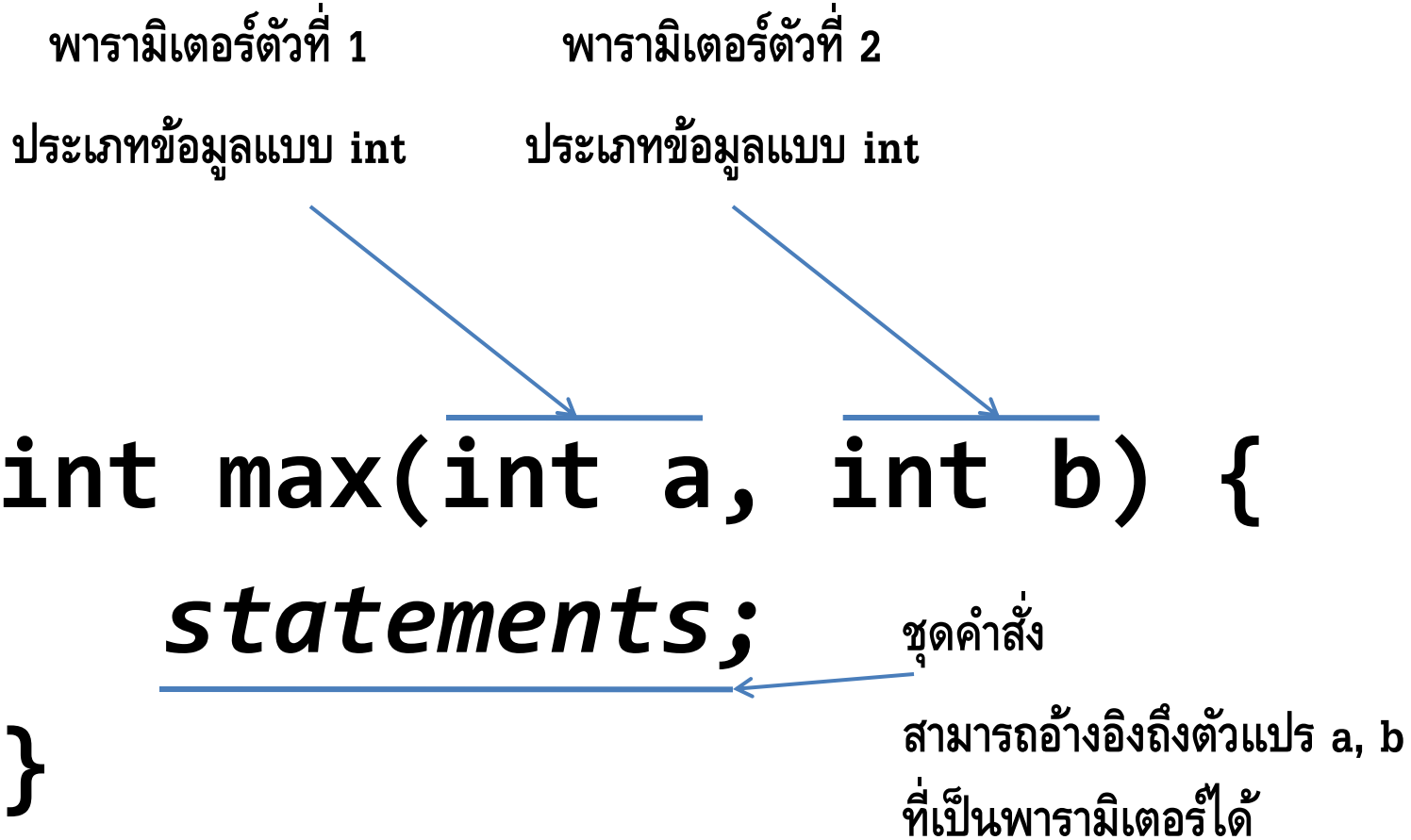
}

- **return-type** ประเภทตัวแปรของค่าที่ส่งออกจากฟังก์ชั่น
- **function-name** ชื่อฟังก์ชั่น
- **param1, param2, ...** พารามิเตอร์ที่ฟังก์ชั่นรับเข้าไป ประกอบด้วยชื่อและประเภทตัวแปร
- **statements** ชุดคำสั่ง

ตัวอย่างการสร้างฟังก์ชั่น 1

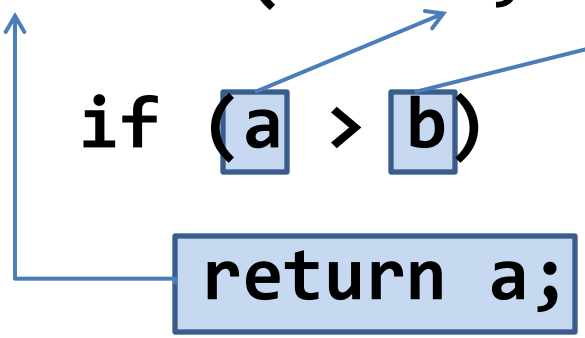


ตัวอย่างการสร้างฟังก์ชั่น 1



ตัวอย่างการสร้างฟังก์ชั่น 1

```
int max(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}
```

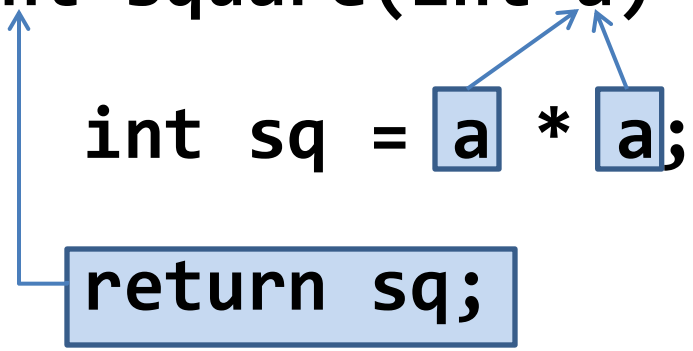


ส่งค่าออกจากฟังก์ชั่น

ต้องส่งค่าที่เป็นประเภทตัวแปร int กลับไป

ตัวอย่างการสร้างฟังก์ชั่น 2

```
int square(int a) {
    int sq = a * a;
    return sq;
}
```



ตัวอย่างการสร้างฟังก์ชั่น 3

```
int square(int a) {
    return a * a;
}
```

integer สองตัวคูณกันได้ integer

ตัวอย่างการสร้างฟังก์ชั่น 3

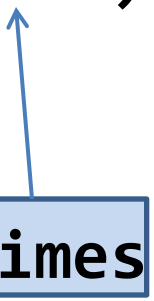
ex05_02.c

```
#include <stdio.h>

int square(int a) {
   return a * a;
}

void main() {
   int i;

   printf("=== THE SQUARES ===\n");
   for (i = 0; i < 26; i++)
      printf("sq(%d) = %d\n", i, square(i));
}
```

ตัวอย่างการสร้างฟังก์ชั่น 4

ไม่ต้องมีการส่งข้อมูลกลับออกมา

```
void greeting(int times) {
    int i;
    for (i = 0; i < times; i++) {
        printf("Hi");
    }
}
```



ตัวอย่างการสร้างฟังก์ชั่น 4

ex05_03.c

```
#include <stdio.h>

void greeting(int times) {
   int i;
   for (i = 0; i < times; i++) {
      printf("Hi\n");
   }
}

void main() {
   int i;

   printf("=== SAY HI ===\n");
   for (i = 1; i < 6; i++)
      greeting(i);
}
```

ตัวอย่างการสร้างฟังก์ชั่น 4

ex05_04.c

```
#include <stdio.h>

void greeting(int times);

void main() {
   int i;

   printf("=== SAY HI ===\n");
   for (i = 1; i < 6; i++)
      greeting(i);
}

void greeting(int times) {
   int i;
   for (i = 0; i < times; i++) {
      printf("Hi\n");
   }
}
```

ถ้าจะเขียนส่วนการทำงานไว้ด้านล่าง (บรรทัดต่ำกว่าตำแหน่งที่มีการเรียกใช้) จะต้องมีส่วนประกาศไว้ด้านบนด้วย (บรรทัดเหนือกว่าตำแหน่งที่มีการเรียกใช้)

ส่วนประกาศ

prototype, declaration

ส่วนการทำงาน

implementation, definition

การส่งผ่านตัวแปร

Parameters passing

การส่งผ่านตัวแปร

Parameters passing

- **ส่งค่า (passing by value)**
  - **การเปลี่ยนแปลงค่าในฟังก์ชั่นจะไม่กระทบตัวแปรที่ส่งเข้ามา**
- **ส่งตำแหน่งที่เก็บค่า (passing by reference)**
  - **การเปลี่ยนแปลงค่าในฟังก์ชั่นจะกระทบตัวแปรที่ส่งเข้ามา**
  - **เป็นความสามารถที่เพิ่มขึ้นมาของภาษา C++**
  - **ภาษา C ไม่รองรับ แต่สามารถทำได้โดยใช้ตัวแปรประเภทตัวชี้ (pointer)**

การส่งผ่านตัวแปรแบบส่งค่า

Parameters passing by value

- ***return-type function-name*(*param*$_1$, *param*$_2$, ...)**
- ***param*$_i$ มีลักษณะดังนี้**
  - ***type*$_i$ *var-name*$_i$**
- **ตัวอย่างที่กล่าวไปก่อนหน้านี้ทั้งหมดเป็นการส่งผ่านแบบส่งค่า**

การส่งผ่านตัวแปรแบบส่งตำแหน่ง

Parameters passing by reference

- ***return-type function-name*(*param*$_1$, *param*$_2$, ...)**
- ***param*$_i$ มีลักษณะดังนี้**
  - ***type*$_i$ & *var-name*$_i$**
- **ภาษาซีไม่รองรับการส่งผ่านตัวแปรแบบนี้**
- **ให้ดูเฉยๆ ว่าหน้าตาเป็นอย่างไร**

ฟังก์ชั่นเรียกตัวเอง

Recursive function

ฟังก์ชั่นเรียกตัวเอง

- เหมือนการเขียนฟังก์ชั่นแบบที่เคยเรียนกันทุกประการ
- เป็นรูปแบบของการเขียนฟังก์ชั่นประเภทหนึ่ง โดยฟังก์ชั่นจะมีการเรียกตัวเองซ้ำ
- สิ่งสำคัญคือฟังก์ชั่นต้องหยุด (terminate, halt)

ตัวอย่าง 1

- ลำดับฟิโบนันซี่มีนิยามว่า **f(n) = f(n-1) + f(n-2)**

  โดยที่ **f(0) = 0, f(1) = 1**

- เราจะเขียนโปรแกรมหาลำดับฟิโบนันซี่ที่ **n** ได้ดังนี้

ตัวอย่าง 1

```
int fibo(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibo(n-1) + fibo(n-2);
}
```

f(0) = 0

f(1) = 1

f(n) = f(n-1) + f(n-2)

ตัวอย่าง 1

ex05_05.c

```
#include <stdio.h>

int fibo(int n) {
   if (n == 0)
      return 0;
   else if (n == 1)
      return 1;
   else
      return fibo(n-1) + fibo(n-2);
}

void main() {
   int i;

   printf("=== FIBONANCI ===\n");
   for (i = 0; i < 21; i++)
      printf("fibo(%d) : %d", i, fibo(i));
}
```

ตัวอย่าง 2

- แฟกทอเรียล มีนิยามว่า **n! = n * (n-1)!**

  โดยที่ **0! = 1**

- เราจะเขียนโปรแกรมหา **n!** ได้ดังนี้

ตัวอย่าง 2

```
int fac(int n) {
   if (n == 0)
      return 1;
   else
      return n * fac(n-1);
}
```

0! = 1

n! = n * (n-1)!

ตัวอย่าง 2

ex05_06.c

```
#include <stdio.h>

int fac(int n) {
   if (n == 0)
      return 1;
   else
      return n * fac(n-1);
}

void main() {
   int i;

   printf("=== FACTORIAL ===\n");
   for (i = 0; i < 21; i++)
      printf("d! : %d", i, fac(i));
}
```

คำเตือน

Cautions

- การเขียนฟังก์ชั่นเรียกตัวเองช่วยให้โปรแกรมอ่านง่ายก็จริง แต่...
  - การทำงานที่สามารถเขียนในรูปแบบฟังก์ชั่นเรียกตัวเองส่วนใหญ่สามารถเขียนโดยใช้ลูปได้ ซึ่งจะทำงานได้เร็วกว่ามาก
  - การเขียนฟังก์ชั่นเรียกตัวเองที่มีการเรียกซ้ำซ้อนมากเกินไปอาจส่งผลให้โปรแกรมหยุดทำงานได้เนื่องจาก **stack overflow**
  - ถ้าเลี่ยงได้ไม่ควรเขียน ให้ใช้หลัก **dynamic programming** ช่วยแทน

เอกสารอ้างอิงฟังก์ชั่นมาตราฐานซี

C Standard Functions Reference/Documentation

แหล่งข้อมูล

Sources

- www.msdn.com
- www.cplusplus.com

- ไม่จำเป็นต้องรู้ทุกอย่าง
- แต่ต้องรู้ว่าถ้าไม่รู้แล้วจะหาข้อมูลได้จากไหน
- และก็ต้องใช้ข้อมูลที่หาได้เป็นด้วย

getch (msdn.com)

_getch, _getwch, _getche, _getwche
Get a character from the console without echo (**_getch, _getchw**) or with echo (**_getche, _getwche**).

```
int _getch( void );
wint_t _getwch( void );
int _getche( void );
wint_t _getwche( void );
```

Return Value

Returns the character read. There is no error return.

Remarks

The **_getch** and **_getwch** functions read a single character from the console without echoing. **_getche** and **_getwche** read a single character from the console and echo the character read. None of these functions can be used to read CTRL+C. When reading a function key or an arrow key, each function must be called twice; the first call returns 0 or 0xE0, and the second call returns the actual key code.

Requirements

Routine	Required header	Compatibility
_getch	<conio.h>	Win 98, Win Me, Win NT, Win 2000, Win XP
_getche	<conio.h>	Win 98, Win Me, Win NT, Win 2000, Win XP
_getwch	<conio.h> or <wchar.h>	Win 98, Win Me, Win NT, Win 2000, Win XP
_getwche	<conio.h> or <wchar.h>	Win 98, Win Me, Win NT, Win 2000, Win XP

For additional compatibility information, see Compatibility in the Introduction.

Libraries

All versions of the C run-time libraries.

rand (msdn.com)

rand

Generates a pseudorandom number. A more secure version of this function is available, see rand_s.

```
int rand( void );
```

Return Value

rand returns a pseudorandom number, as described above. There is no error return.

Remarks

The rand function returns a pseudorandom integer in the range 0 to RAND_MAX (32767). Use the srand function to seed the pseudorandom-number generator before calling rand.

Requirements

Routine	Required header
rand	<stdlib.h>

ฟังก์ชั่นพื้นฐานที่ควรรู้

- printf, puts, putchar
- scanf, gets, getchar, getch, getche
- time
- srand, rand

ดูรายละเอียดเพิ่มเติมได้ที่สไลด์ อ.สาโรช และ msdn.com

จบบทที่ 5

The End of Chapter 5

บทที่ 6 ประเภทตัวแปรขั้นสูง

แนะนำประเภทตัวแปรพื้นฐานของซีทั้งหมด ซึ่งการที่เรารู้จักตัวแปรเหล่านี้จะช่วยให้การทำงานหลายๆ อย่างสะดวกมากยิ่งขึ้น

2 November 2010

อ. เอก ตังคนานนท์