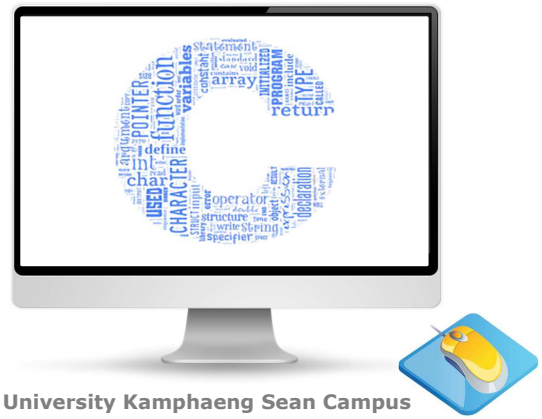
02204171 Structured Programming

Chapter 11 : Function (part 1)



Outline

- Function Overview
- Variable scope
- Two Types of Function
  - Pre-defined function
  - User-defined function
- Operation of a Function
  - Calling a function
  - Defining a function
- Function prototype

Preface: Lines of Code

1. Function Overview

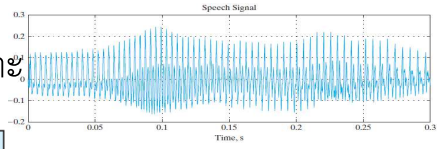
- โปรแกรมคอมพิวเตอร์ส่วนใหญ่ที่ใช้งานจริง มักจะเป็นโปรแกรมขนาดใหญ่ ซึ่งใหญ่เกินกว่าที่จะเขียนส่วนการทำงานทั้งหมดไว้ในที่เดียวกัน หรือในไฟล์เดียวกัน
- แนวปฏิบัติในการเขียนโปรแกรม ควรแบ่งการทำงานออกเป็นโปรแกรมย่อยๆ (Module) เมื่อสร้างและทดสอบโปรแกรมย่อยๆ นั้นแล้ว จึงนำมาประกอบรวมเป็นโปรแกรมที่สมบูรณ์

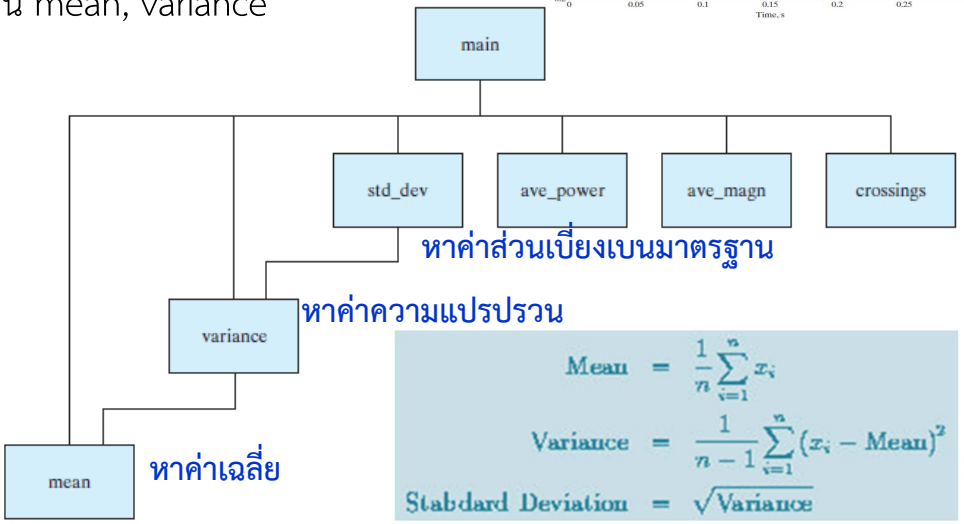
Divide and conquer

1. Function Overview

ตัวอย่าง: โปรแกรมวิเคราะห์สัญญาณเสียง แบ่งเป็นส่วนโปรแกรมย่อยๆ ที่มีหน้าที่เฉพาะ เช่น mean, variance

Structure Chart
Speech Signal Analysis





1. Function Overview

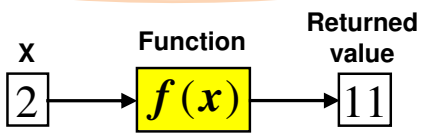
- ในภาษา C **ส่วนย่อยของโปรแกรม**ที่ถูกเขียนขึ้นมาเพื่อทำงานอย่างใดอย่างหนึ่ง เรียกว่า **ฟังก์ชัน (Function)**
  - ลดความซับซ้อนของโปรแกรม ง่ายต่อการตรวจหาข้อผิดพลาด
  - สามารถนำไปใช้ทำงานได้หลายๆ ครั้งในโปรแกรมดียวกัน หรือต่างโปรแกรม โดยไม่จำเป็นต้องเขียนส่วนโปรแกรมที่ทำงานแบบเดียวกันขึ้นใหม่ (re-use of code)
  - ส่วนต่างๆ ของโปรแกรม สามารถพัฒนาได้โดยเป็นอิสระต่อกัน (Independent development of code)

1. Function Overview

- **ฟังก์ชัน** คือ ส่วนย่อยของโปรแกรมที่เขียนขึ้นมาเพื่อทำงานอย่างใดอย่างหนึ่ง โดยการอ้างถึงด้วยชื่อที่กำหนดขึ้น เช่น
  - sqrt() -- ใช้คำนวณค่ารากที่สองของจำนวนใดๆ
  - sin() -- ใช้คำนวณค่า sin ของจำนวนใดๆ

Function: ในมุมมองคณิตศาสตร์

$$f(x) \overset{\Delta}{=} x^2 + 2x + 3$$



$$f(2) \Rightarrow (2)^2 + 2(2) + 3 \Rightarrow 4 + 4 + 3 \Rightarrow 11$$

$$\therefore f(2) \text{ is } 11$$

1. Function Overview

```
#include <stdio.h>
 int main ( )
{
    printf ("Hello World!\n");
    return 0 ;
}
```

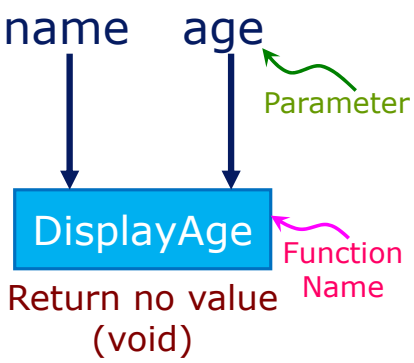
ฟังก์ชันมาตรฐานของภาษา C

การเรียกฟังก์ชัน (Function call)

- ภายในโปรแกรมประกอบด้วยฟังก์ชันอย่างน้อยหนึ่งฟังก์ชันเสมอ คือ main() ซึ่งเป็นฟังก์ชันหลักที่โปรแกรมภาษา C จะเริ่มทำงานจากจุดนี้
- โดยฟังก์ชัน main() อาจจะมีการเรียกใช้ฟังก์ชันอื่นๆ ต่อไปอีกได้

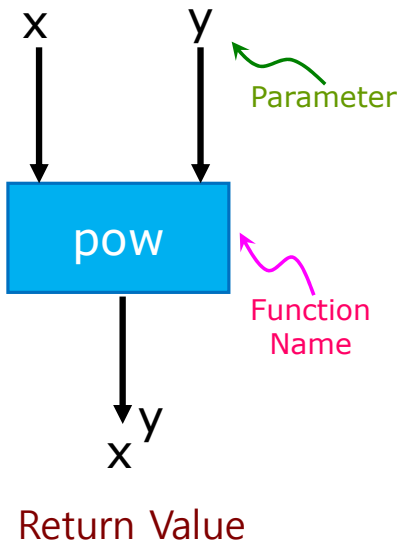
1. Function Overview: Function Component

- **องค์ประกอบของฟังก์ชัน**
  - ชื่อฟังก์ชัน
  - พารามิเตอร์ (parameters)
  - การส่งค่ากลับ (return values)
- **ตัวอย่าง**
  - DisplayAge("Harry", 17)



1. Function Overview: Function Component

- **องค์ประกอบของฟังก์ชัน**
  - ชื่อฟังก์ชัน
  - พารามิเตอร์ (parameters)
  - การส่งค่ากลับ (return values)
- **ตัวอย่าง**
  - Z=pow(2.0, 3)



2. Variable Lifetime

- ตัวแปรมีขอบเขตการใช้งานของมัน
- วงจรชีวิตของตัวแปร
  - ประกาศตัวแปร → เริ่มต้นชีวิต
    - หากตัวแปรยังไม่ถูกประกาศ = ยังไม่มีชีวิต → ใช้งานไม่ได้
  - ใช้งาน
  - ตัวแปรใช้งานได้ภายในบล็อกของมันเท่านั้น
- ทำลายอัตโนมัติเมื่อมีการปิดบล็อกที่มันถูกประกาศไว้

2. Variable Lifetime

```
int main() {
  int num = 5;
  for (int i = 1; i <= num; i++)
  {
    printf("%d",i);
  }
  printf("%d",i+1);
  if (num > 0) {
    float x = 2.5;
    for (int i = num; i >= 0; i--) {
       printf("%f",x+i);
    }
    printf("%.2f",x);
    printf("%d",i);
  }
  printf("%.2f",x);
  printf("%d",num);
}
```

2. Variable Lifetime

```
void main()
{
   int x;
   x = 5;
   printf("%c",c);
}

void func1(char c)
{
   c = c + x;
   printf("%c",c);
}
```


3. Two Types of Function

1. ฟังก์ชันมาตรฐาน (Standard Pre-defined function in C)
   - pow(x,y)
   - sqrt(25)
   - sizeof()
2. ฟังก์ชันที่ผู้ใช้กำหนดเอง (User-defined function)
   - sayhi(5)
   - add(x,y)

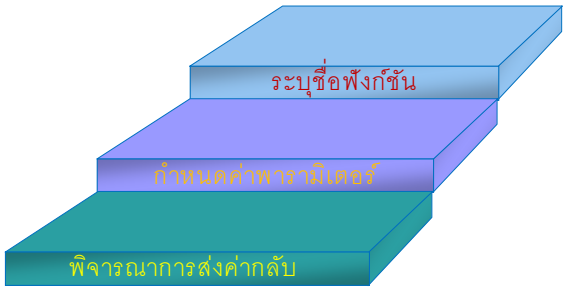
3.1 Standard Pre-defined function

- ฟังก์ชันมาตรฐานในภาษาซี เป็นฟังก์ชันที่มีมาให้พร้อมกับตัวแปลภาษา (Compiler) ให้โปรแกรมเมอร์ใช้งานได้ทันที
  - สำหรับใช้ในงานด้านต่างๆ โดยเน้นงานพื้นฐาน เช่น ฟังก์ชันคำนวณทางคณิตศาสตร์ ฟังก์ชันสำหรับจัดการข้อความ ฟังก์ชันเวลา เป็นต้น ซึ่งจะทำให้โปรแกรมเมอร์เขียนโปรแกรมได้ง่ายขึ้น
- ฟังก์ชันมาตรฐานถูกรวบรวมในไลบรารีภาษาซี (C Standard Library) ซึ่งการใช้งานฟังก์ชันประเภทนี้จะต้อง**ระบุชื่อของไลบรารีที่เกี่ยวข้อง**

3.1 Standard Pre-defined function

- y = sqrt(x);
- y = sqrt(b * b - 4 * a * c);
- z = pow(3.3, a);
- w = toupper('d');

จำนวนและชนิดของพารามิเตอร์ต้องตรงกันกับจำนวนที่ระบุไว้ในฟังก์ชัน



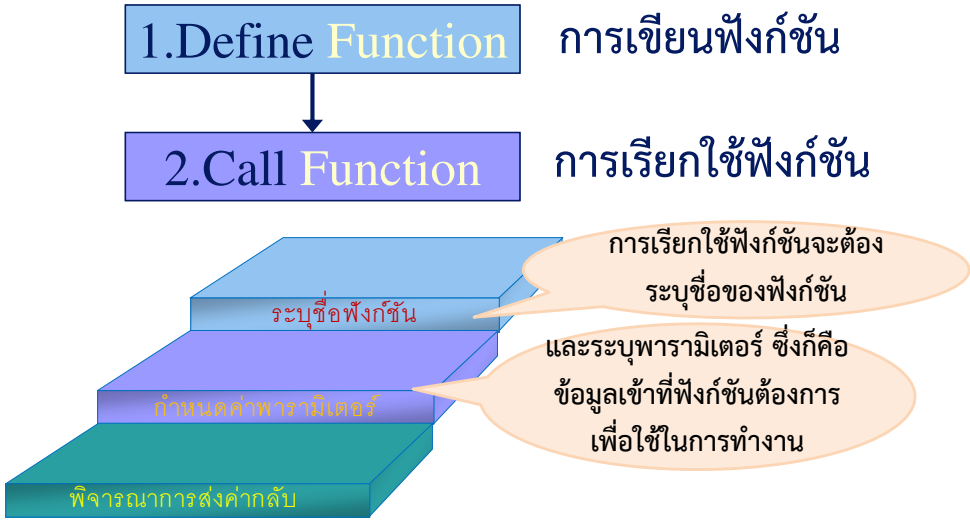
การเรียกใช้ฟังก์ชัน (Call Function)

3.2 User-defined function

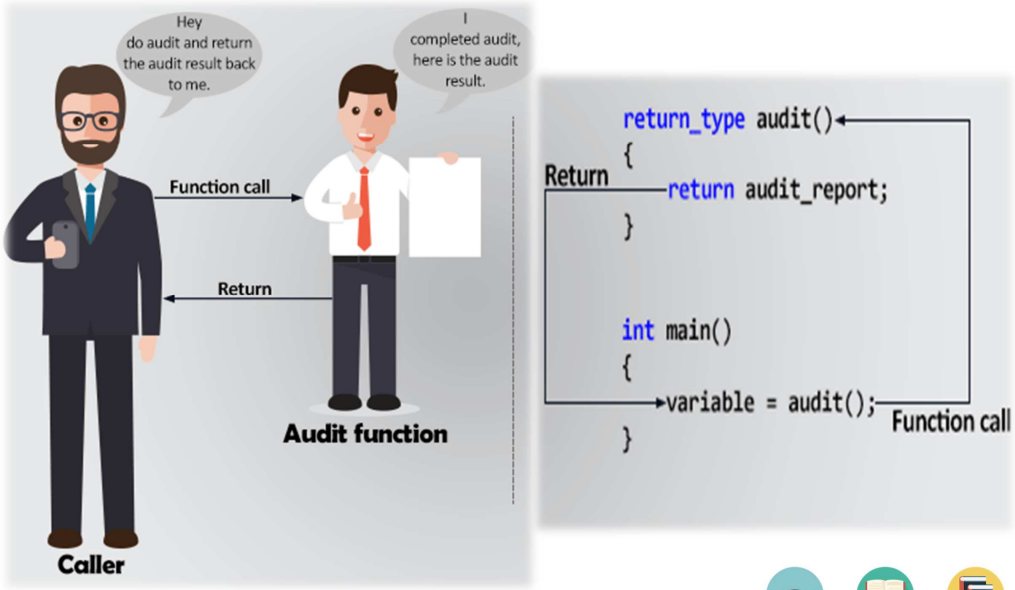
- **ฟังก์ชันที่โปรแกรมเมอร์สร้างขึ้นเอง**

โปรแกรมเมอร์สามารถเขียนฟังก์ชัน (ส่วนย่อยของโปรแกรม) เพื่อนิยามการทำงานเฉพาะ ซึ่งสามารถเรียกใช้ในส่วนต่างๆ ของโปรแกรม โดยฟังก์ชันการทำงานดังกล่าวจะถูกเขียนไว้ในฟังก์ชันเพียงครั้งเดียวเท่านั้น แต่สามารถเรียกใช้งานได้หลายครั้ง

4. Operation of a function



4. Operation of a function (Cont.)



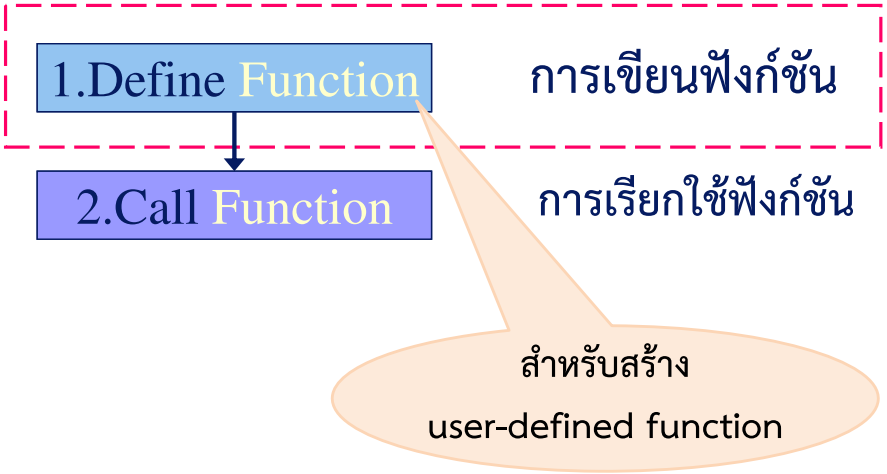

```
return_type audit()
{
    return audit_report;
}

int main()
{
    variable = audit();
}
```

Return

Function call

4.1 Defining a function



4.1 Defining a function

รูปแบบการเขียนฟังก์ชัน (Function Definition Format)

```
<return-type> function-name(<parameter list>)
{
   const/variable declaration;
   statements;
}
```

- **ชื่อฟังก์ชัน** (function_name): ตั้งชื่อตามกฎการตั้งชื่อเหมือนตัวแปร
- **พารามิเตอร์** (จะมีหรือไม่มีก็ได้) : เขียนในรูปของการประกาศตัวแปรทีละตัว (คั่นด้วย semi-colon ‘,’) เช่น (int a, int b)
- **ชนิดของตัวแปรที่ส่งค่ากลับ** (return_type)
  - ถ้าส่งค่าใดๆ กลับ ให้ระบุชนิดข้อมูล เช่น int, double เป็นต้น
  - ถ้าส่งค่ากลับโดยไม่ให้ค่าใดๆ ให้ระบุข้อมูลชนิด void

4.1 Defining a function

```
#include <stdio.h>
#include <math.h>
void displayBox()
{
   printf("*\n");
   printf("**\n");
   printf("***\n");
}
int main()
{
   displayBox();
   printf("%.2f",sqrt(100));
   return 0;
}
```


4.1 Defining a function

- **ตัวอย่าง: การเขียนฟังก์ชัน**

```
void displaybox() {
}
```

ฟังก์ชันที่ไม่มีการส่งค่ากลับ

```
int calage() {
}
```

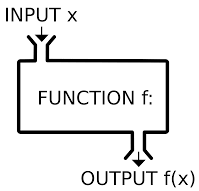
ต้องระบุชนิดข้อมูลของพารามิเตอร์แต่ละตัว

```
int calbox(int x,int y) {
}
```


4.1 Defining a function

Return vs. No Return Value Function

1. ฟังก์ชันที่ไม่มีการส่งค่ากลับ(function with no return value)
   - ตัวอย่าง: DisplayBox()
2. ฟังก์ชันที่มีการส่งค่ากลับ (function with return value)
   - ตัวอย่าง: x=sqrt(9)



4.1.1 Function with No Return Value

- **ฟังก์ชันที่ไม่มีการส่งค่ากลับ** ฟังก์ชันแบบนี้จะถูกกำหนดด้วยคำว่า void ที่หน้าชื่อของฟังก์ชัน

```
void function-name(<parameter list>)
{
   const/variable declaration;
   statements;
}
```


4.1.1 Function with No Return Value

Example1: No Return Value without Parameter Function

```
#include <stdio.h>
int main() {
int i,j,k;
for(k=1; k<=3;k++) {
  for(i=1;i<=3;i++){
    for(j=i;j<3;j++)
       printf(" ");
    for(j=1;j<=2*i-1;j++)
       printf("*");
    printf("\n");
  }
 }
}
```

```
  *
 ***
*****
  *
 ***
*****
  *
 ***
*****
```


4.1.1 Function with No Return Value

Example1: No Return Value without Parameter Function

```
#include <stdio.h>
void displayStar() {
   int i,j;
   for(i=1;i<=3;i++){
      for(j=i;j<3;j++)
         printf(" ");
      for(j=1;j<=2*i-1;j++)
         printf("*");
      printf("\n");
   }
}
int main() {
int k;
for(k=1; k<=3;k++)
  displayStar();
}
```


4.1.1 Function with No Return Value

Example2: No Return Value with Parameter Function

- แสดงรูปสามเหลี่ยมพีระมิดของสัญลักษณ์ต่างๆ

```
#include <stdio.h>
int main() {
int i,j;
for(i=1;i<=3;i++){
    for(j=i;j<3;j++)
       printf(" ");
    for(j=1;j<=2*i-1;j++)
       printf("*");
    printf("\n");
  }
}
```

```
  *
 ***
*****
  +
 +++
+++++
  X
 XXX
XXXXX
```

4.1.1 Function with No Return Value

Example2: No Return Value with Parameter Function

```
#include <stdio.h>
void displayStar(char c) {
   int i,j;
   for(i=1;i<=3;i++){
      for(j=i;j<3;j++)
         printf(" ");
      for(j=1;j<=2*i-1;j++)
         printf("%c",c);
      printf("\n");
   }
}
int main() {
 displayStar('*');
 displayStar('+');
 displayStar('x');
}
```

c '*'

Quick check 1

- ให้นิสิตเขียนโปรแกรมเพื่อคำนวณหาพื้นที่วงกลมโดยการสร้าง**ฟังก์ชันแบบไม่ส่งค่ากลับ** โดยการป้อนค่าพารามิเตอร์คือ รัศมีของวงกลม ไปให้ฟังก์ชันชื่อ circle_area()
- ตัวอย่างการรันของโปรแกรม

```
Radius of circle: 3
The circle area is 28.26
```


Quick check 1

```
int main()
{
 double r;
 double area, pi = 3.14;
 printf("Radius of circle: ");
 scanf("%lf",&r);
 area = pi*r*r;
 printf("The circle area is %0.2f\n",area);
 return 0;
}
```


Quick check 1

```
_____ circle_area(          )
{
   double area, pi = 3.14;
   area = pi*r*r;
   printf("The circle area is %0.2f\n",area);
}

int main() {
 double r;
 printf("Radius of circle: ");
 scanf("%lf",&r);
 ____________________________
 return 0;
}
```

4.1.2 Function with Return Value

- **ฟังก์ชันที่มีการส่งค่ากลับ** ฟังก์ชันแบบนี้จะกำหนดชนิดของค่าที่ต้องการส่งกลับที่หน้าชื่อของฟังก์ชัน เช่น int, float, bool

```
<return type> function-name(<parameter list>)
{
  const/variable declaration;
  statements;
  return value;
}
```

ต้องตรงกับ ส่วนนี้ของฟังก์ชัน

ชนิดค่าตัวแปรหรือค่าคงที่ ที่ส่งค่ากลับ

4.1.2 Function with Return Value

Example3: Return Value without Parameter Function

```
#include <stdio.h>
int current_AD() {
   return 2560-543;
  }
int main() {
 printf("This year is %d",current_AD());
 return 0;
}
```


4.2 Defining a function

Example4: Return Value with Parameter Function

```
#include <stdio.h>
int BE_to_AD(int year) {
   return year-543;
  }
int main() {
int yr =2543;
printf("This year is %d\n",BE_to_AD(2560));
printf("Y2K is %d\n",BE_to_AD(yr));
return 0;
}
```


Quick check 2

- ให้นิสิตเขียนโปรแกรมเพื่อคำนวณหาพื้นที่ผิวข้างของทรงกระบอกโดยการ**สร้างฟังก์ชันแบบส่งค่ากลับ** โดยการป้อนค่าพารามิเตอร์ คือ ความสูงและรัศมีของทรงกระบอกไปให้ฟังก์ชันชื่อ cylinder_surface
- ตัวอย่างการรันของโปรแกรม

```
Radius of cylinder: 3
Height of cylinder: 5
The cylinder surface area is 94.20
```

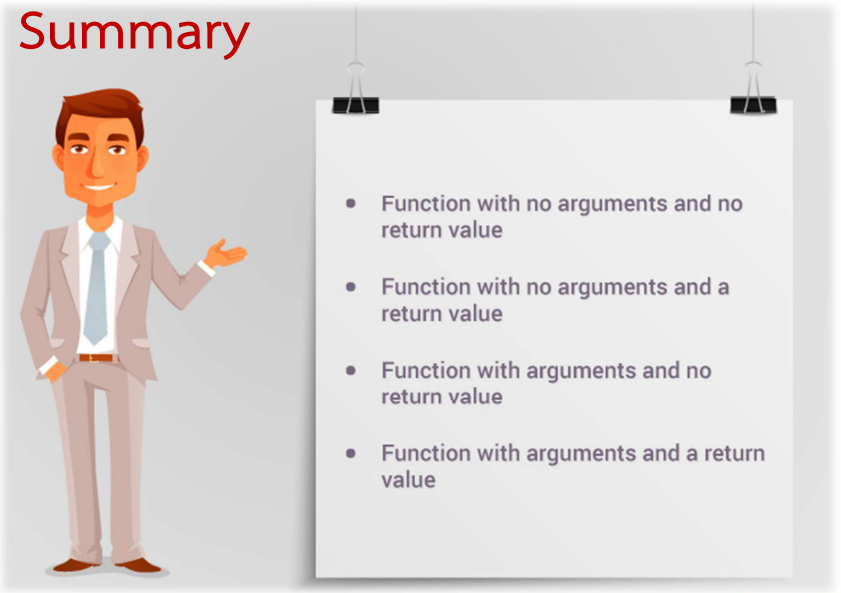
Quick check 2

```
_____cylinder_surface(_____)
{


}
int main()
{
   double h,r,area;
   printf("Radius of cylinder: ");
   scanf("%lf",&r);
   printf("Height of cylinder: ");
   scanf("%lf",&h);

   printf(" The cylinder surface area is %0.2f\n",area);
  return 0;
}
```


4.1 Defining a function



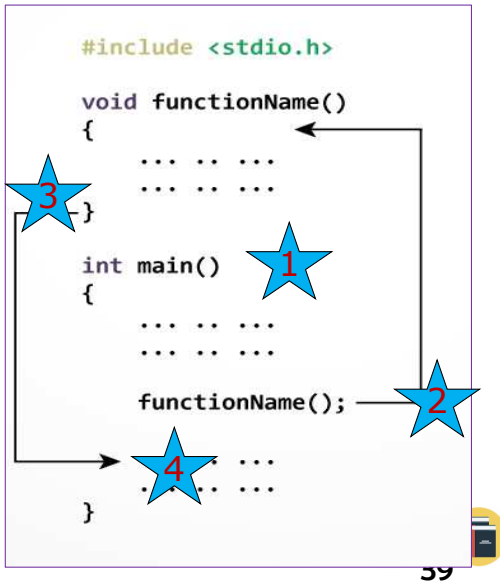
https://www.programiz.com/c-programming/types-user-defined-functions

4.2 Calling a function

- การเรียกใช้ function ได้อย่างถูกต้องสามารถดูได้จาก header
  - ชื่อ function
  - ค่า parameter ตามลำดับ

  พร้อมทั้งชนิดที่ถูกต้อง
- เมื่อมีการเรียกใช้ฟังก์ชัน

โปรแกรมจะไปประมวลผลคำสั่งในฟังก์ชันจนเสร็จ แล้วจึงกลับมาทำงานคำสั่งที่ต่อจากที่ที่เรียกฟังก์ชันนั้น



4.2 Calling a function

- การเรียกใช้ตามประเภทของ function
  - function ที่ไม่มีการส่งค่ากลับ
    - *<function -name>* (*<param>*);
  - function ที่มีการส่งค่ากลับ
    - <var> = *<function -name>* (*<param>*);
    - หรือ เทียบเท่า คือ มักเขียนคู่กับคำสั่งอื่น
    - มักไม่เขียนโดดๆ แบบ function ที่ไม่มีการส่งค่ากลับ

Quick Check 3

- พิจารณาคำสั่งเรียกใช้ฟังก์ชันต่อไปนี้ ว่าเป็นคำสั่งที่ถูกต้องหรือไม่

```
int Hello(void) {

}
void Hi(int a,char b){

}
int main(){
  int x,y;
  char s,t;
  -- ✡ ---
}
```

```
1. x = Hello();
2. Hello("girls");
3. Hello();
4. Hi(s);
5. y = Hi(3, t);
6. Hi(x,'K');
```


5 Function prototype

- ฟังก์ชันจะถูกเรียกใช้ได้ก็ต่อเมื่อมีการ define (เขียนฟังก์ชัน) → เราเขียนฟังก์ชันที่ถูกเรียกใช้ไว้ก่อน main()

```
void say_hi() {
printf("Hi...How are you doing?");
}
int main() {
   say_hi();
   return 0;
}
```

```
void A()
{ … }
int B()
{ … }
void C(int x, char c)
{ … }

int main() {
   A();
   C( B(), 'Q');
   return 0;
}
```

- หาก main() เรียกใช้หลายฟังก์ชัน เราจะต้องเขียน main() ล่างสุด ซึ่งอาจทำให้หาฟังก์ชัน main() ได้ยาก

5 Function prototype

- ภาษา C จึงสร้างการประกาศส่วนหัวของฟังก์ชัน (function prototype) ขึ้นมา ซึ่งทำให้เราสามารถแก้ปัญหาข้างต้นได้
- Syntax

```
<return-type> function_name(param);
```

เขียนเฉพาะ header ของฟังก์ชัน แล้วตามด้วย ;

การประกาศฟังก์ชัน เปรียบเสมือนการแนะนำให้รู้จักฟังก์ชันที่ถูกประกาศโดยยังไม่จำเป็นต้องสนใจว่าฟังก์ชันนั้นจะทำงานอย่างไร

5 Function prototype

- การประกาศ function prototype

```
#include<stdio.h>
void say_hi(void);        // ประกาศฟังก์ชันที่หัวของโปรแกรม
int main()
{
  say_hi();
  return 0;
}
void say_hi()
{
  printf("Hi...How are you doing?");
}
```

Quick check4

- จงเขียนฟังก์ชัน Power4() ที่มี n เป็นพารามิเตอร์ และส่งค่า n^4 กลับเป็นผลลัพธ์ // Power4(2) → 16

```
#include <stdio.h>

int main()
{



  return 0;
}
________ Power4(________________)
{


}
```