w11-Lec

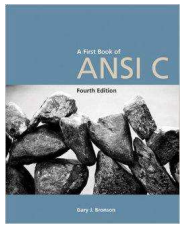
Functions Part I

Assembled for 204111

by Ratsameetip Wita

Topics

- Introduction to C Function
  - Prototype and Definition
- C Standard Library Functions
  - Function Prototype
- User-defined Function
  - Function Placing
  - Calling a Function
  - Returning a Value



Introduction to C Functions

- ฟังก์ชัน คือการรวบรวมชุดคำสั่งที่ทำหน้าที่เฉพาะไว้ด้วยกัน
- มีการตั้งชื่อชุดคำสั่ง ซึ่งเรียกว่าชื่อฟังก์ชัน (Function Name)
- โดยในการเรียกใช้งาน จะเรียกผ่านคำสั่งที่เป็นชื่อฟังก์ชัน
  - เช่น `printf();` `scanf();` `pow(x,y);` เป็นต้น
- ฟังก์ชันในภาษา C แบ่งออกเป็น 2 ประเภท ได้แก่
  - ฟังก์ชันมาตรฐานของภาษา (Standard Library Functions)
  - ฟังก์ชันที่ผู้เขียนโปแกรมสร้างเอง (User-Defined Functions)

C Standard Library Functions

- ฟังก์ชันมาตรฐานของภาษา C เป็นฟังก์ชันที่เตรียมให้ผู้เขียนโปรแกรมได้เลือกใช้งาน
- โดยชุดคำสั่งที่เป็นรายละเอียดของฟังก์ชัน จะถูกเก็บไว้ใน header file (*.h) ตามกลุ่มของฟังก์ชัน
  - เช่น `stdio.h`, `stdlib.h`, `math.h`, `string.h` เป็นต้น
- ในการเรียกใช้งาน จึงต้องมีการ include header ไฟล์ก่อนการใช้งานฟังก์ชันต่าง ๆ
- Standard Library: มี 15 header files
  - http://www.tutorialspoint.com/c_standard_library/

C Standard Library Functions (2)

- ก่อนที่จะเรียกใช้ฟังก์ชัน จะต้องรู้สิ่งเหล่านี้ก่อน (Function Prototype) คือ
  - มีฟังก์ชันอะไรให้เรียกใช้บ้าง (Function Name)
  - แต่ละฟังก์ชันมี arguments อะไรบ้าง (Parameter)
  - Data Type ของข้อมูลที่ส่งกลับเป็นอะไร (Return Type)
  - แต่ละฟังก์ชันทำหน้าที่อะไร (Function Description)
  - ฟังก์ชันนั้น ๆ อยู่ใน header file ไหน และจะต้อง include header file ดังกล่าว โดยมี General Syntax คือ

```
#include <header-file-name>
#include <stdio.h>
#include <math.h>
```


Function Prototype

- Function Prototype: การประกาศฟังก์ชัน ประกอบด้วย
  - Return DataType: เพื่อประกาศว่าฟังก์ชันจะส่งข้อมูลประเภทไหนกลับมาให้ผู้ที่เรียกใช้ฟังก์ชัน เช่น int, float, double, void (ไม่คืนค่าใด ๆ)
  - Function Name: เพื่อประกาศชื่อฟังก์ชัน
  - Argument / Input Parameters: เพื่อประกาศว่าจะส่ง ข้อมูลอะไรและประเภทไหนบ้าง เข้าไปประมวลผลในฟังก์ชัน

```
returnDataType functionName(datatypes argument);
double pow(double x, double y);
int rand();
```


Mathematical Library Functions

Table 6.1 Mathematica ns (require the math.h header file)

Arguments

Return Type

Function Name

Prototype	Description
double fabs(double)	Returns the absolute value of its double-precision argument. (Note, the function int abs(int) is defined in the stdlib.h header file.)
double ceil(double)	Returns a floating-point value that is the smallest integer that is greater than or equal to its argument value.
double floor(double)	Returns a floating-point value that is the largest integer that is less than or equal to its argument value.
double fmod(double, double)	Returns the remainder of its first argument divided by its second argument.
double exp(double)	Returns e raised to its double-precision argument.
double log(double)	Returns the natural logarithm (base e) of its argument.
double log10(double)	Returns the common logarithm (base 10) of its argument.

Mathematical Library Functions (2)

Table 6.1 Mathematical Library Functions (require the math.h header file)(continued)

Prototype	Description
double sqrt(double)	Returns the square root of its argument.
double pow(double, double)	Returns its first argument raised to the power of its second argument.
double sin(double)	Returns the sine of its argument. The argument must be in radians.
double cos(double)	Returns the cosine of its argument. The argument must be in radians.
double tan(double)	Returns the tangent of its argument. The argument must be in radians.
double asin(double)	Returns the angle (in radians) whose sine is the argument.
double acos(double)	Returns the angle (in radians) whose cosine is the argument.
double atan(double)	Returns the angle (in radians) whose tangent is the argument.
double atan2(double, double)	Returns the angle (in radians) whose tangent is the first argument divided by the second argument.
double sinh(double)	Returns the hyperbolic sine of its argument.
double cosh(double)	Returns the hyperbolic cosine of its argument.
double tanh(double)	Returns the hyperbolic tangent of its argument.

Character Processing Functions

Table 6.2 Character Functions (require the header file `ctype.h`)

Prototype	Description	Example
`int isalnum(int)`	Returns a non-0 number if the argument is a letter or a digit; otherwise it returns a 0.	`isalnum('9');`
`int isalpha(int)`	Returns a non-0 number if the argument is a letter; otherwise, it returns 0.	`isalpha('a')`
`int iscntrl(int)`	Returns a non-0 number if the argument is a control argument; otherwise, it returns 0.	`iscntrl('a')`
`int isdigit(int)`	Returns a non-0 number if the argument is a digit (0–9); otherwise, it returns 0.	`isdigit('a')`
`int isgraph(int)`	Returns a non-0 value if the argument is a printable character other than a space; otherwise it returns a 0.	`isgraph('@')`
`int islower(int)`	Returns a non-0 number if the argument is lowercase; otherwise, it returns 0.	`islower('a')`

Conversion Functions

Table 6.3 String Conversion Functions (require the header file `stdlib.h`)

Prototype[8]	Description	Example
`int atoi(string)`	Converts an ASCII string to an integer. Conversion stops at the first noninteger character.	`atoi("1234")`
`double atof(string)`	Converts an ASCII string to a double-precision number. Conversion stops at the first character that cannot be interpreted as a double.	`atof("12.34")`
`string itoa(int)`	Converts an integer to an ASCII string. The space allocated for the returned string must be large enough for the converted value.	`itoa(1234)`

User-Defined Functions

- ภาษา C อนุญาตให้ผู้เขียนโปรแกรมสามารถสร้างฟังก์ชันขึ้นเองได้
- ใช้สำหรับแบ่งปัญหาออกเป็นปัญหาย่อย ๆ
- สามารถเรียกใช้คำสั่งโดยไม่ต้องเขียนโค้ดซ้ำ ๆได้
- สามารถตรวจสอบความถูกต้องได้สะดวกกว่า

User-Defined Functions

- **Temperature Conversion Pseudocode**
  - **Input Temp in Fahrenheit**
  - **Calculate Celsius from c = (5/9) * (f- 32)**
  - **Display Result in Celsius**

Temperature Conversion Revisited

```c
#include <stdio.h>
#include <stdlib.h>
// ต้องการสร้างตารางการแปลงอุณหภูมิจากองศาฟาเรนไฮต์เป็นองศาเซลเซียส
int main(){
    double temp_f;
    double temp_c;
    temp_f = 60;

    temp_c = (5.0/9.0) * (temp_f - 32);
    printf("%.2f degree F is degree %.2f C\n", temp_f, temp_c);
    return 0;
}
```

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

double convertFtoC(double fahrenheit){
     return (5.0/9.0) * (fahrenheit - 32);
}
int main() {
    double temp_f = 60;
    printf("%.2f degree F is degree %.2f C\n", temp_f, convertFtoC(temp_f));
    return 0;
}
```

ส่วนที่ใช้ในการคำนวณ

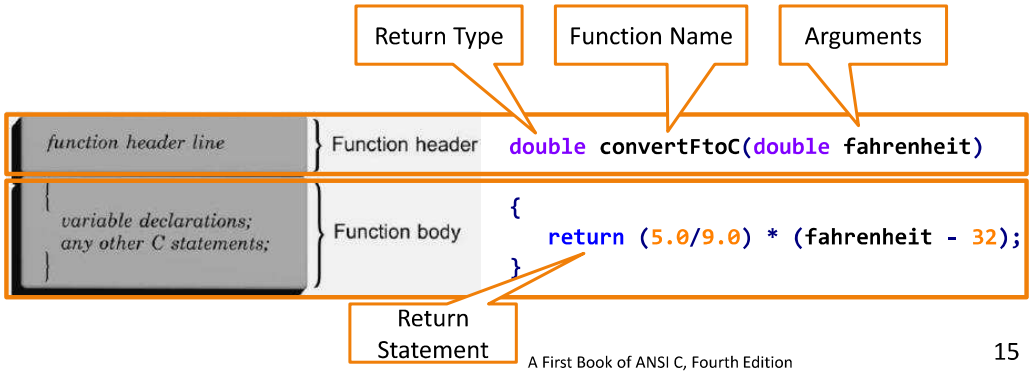
เรียกฟังก์ชันการคำนวณ

Function Definition

- คำนิยามของฟังก์ชัน ประกอบด้วย 2 ส่วนหลัก คือ
- **Function header** ประกอบด้วย
  - **Return Type** กำหนด data type ของผลลัพธ์ของฟังก์ชัน หากไม่ส่งค่ากลับให้ระบุว่า void
  - **Function Name** กำหนด ชื่อฟังก์ชัน สำหรับเรียกใช้งาน
  - **Arguments/input parameters** ตัวแปรที่ต้องใส่เมื่อเรียกฟังก์ชัน
- **Function body** ประกอบด้วย ชุดคำสั่งที่ใช้ประมวลผลข้อมูล และคำสั่งที่ใช้ส่งค่าข้อมูลกลับไปยังผู้เรียกใช้ฟังก์ชัน (หากมีการกำหนด return type)

Function Definition (2)

- เราสามารถวาง **Function Definition** ไว้ก่อน หรือ หลัง **main()** ฟังก์ชัน
- ห้ามวาง **Function Definition** ซ้อนไว้ในฟังก์ชันอื่น **(Nested functions are not permitted)**



Function Placement

- ในภาษา C เมื่อใช้ตัวแปร หรือค่าคงที่ใด ๆ จะต้องมีการประกาศ (Declare) ก่อนใช้งาน
- โดยโปรแกรมหลัก จะอยู่ในส่วน ฟังก์ชัน main()
- การเขียนฟังก์ชันย่อยใด ๆ สามารถวางโค้ดของฟังก์ชันไว้ก่อน main() หรือหลัง main() ก็ได้

Function Placement (2)

- หากวาง function ไว้ต่อท้าย main () function จำเป็นต้องมีการประกาศ Function Prototype
- Function Prototype สามารถประกาศไว้ได้ทั้งก่อน main() และใน main () function

```
preprocessor directives
symbolic constants
function prototypes can be placed here
int main()
{
   function prototypes can be placed here
   variable declarations;
   other executable statements;
   return value;
}
```

Not Recommended

A First Book of ANSI C, Fourth Edition

Function Placement (3)

มี ; ที่ prototype

ไม่มี ; ที่ function header

- Before main()

```c
#include <stdio.h>
#include <stdlib.h>
double convertFtoC(double fahrenheit){
    double local_temp_C;
    local_temp_C = (5.0/9.0) *
(fahrenheit - 32);
    return local_temp_C;
}
int main(){
    double temp_f;
    double temp_c;
    temp_f = 60;

    temp_c = convertFtoC(temp_f);
    printf("%.2f degree F is degree
%.2f C\n", temp_f, temp_c);
    return 0;
}
```

- After main()

```c
#include <stdio.h>
#include <stdlib.h>
double convertFtoC(double fahrenheit);
int main(){
    double temp_f;
    double temp_c;
    temp_f = 60;

    temp_c = conver
    printf("%.2f degree F is d    e
%.2f C\n", temp_f, temp_c);
    return 0;
}
double convertFtoC(double fahrenheit){
    double local_temp_C;
    local_temp_C = (5.0/9.0) *
(fahrenheit - 32);
    return local_temp_C;
}
```


Function Types

	Sample Function Prototype

ไม่มีการรับค่า และไม่มีการคืนค่า
No Argument – No Return
	`void functionName()`
	OR
	`void functionName(void)`

มีการรับค่า ไม่มีการคืนค่า With Argument – No Return	`void functionName(int a)`

ไม่มีการรับค่า มีการคืนค่า
No Argument – Return
	`int functionName()`
	OR
	`int functionName(void)`

มีการรับค่า มีการคืนค่า With Argument – Return	`int functionName(int a)`

A First Book of ANSI C, Fourth Edition

Calling a Function

- พารามิเตอร์ Parameter ตัวแปรที่ถูกส่งไปกับการเรียกฟังก์ชัน
- อาร์กิวเมนต์ Argument ตัวแปรที่รับค่า จากพารามิเตอร์ที่ส่งมาตามลำดับ

```c
void findMax(int a, float b){
...
}
main(){
    int a=3;
    float average = 5.5;
    findMax(a,average);
}
```

Calling a Function (2)

- **Pass by value:** เป็นการส่งค่าของข้อมูลเข้าไปในฟังก์ชัน เมื่อฟังก์ชันรับค่าข้อมูลจะเก็บค่าข้อมูลไว้ในตัวแปรที่ประกาศไว้ในส่วนของ **arguments** ตามลำดับของข้อมูลที่ส่งเข้ามา
- **Pass by reference** เป็นการส่งค่า ตำแหน่งในแอดเดรสของตัวแปรเข้าไปในฟังก์ชัน **(Later in 204112)**

Calling a Function (3)

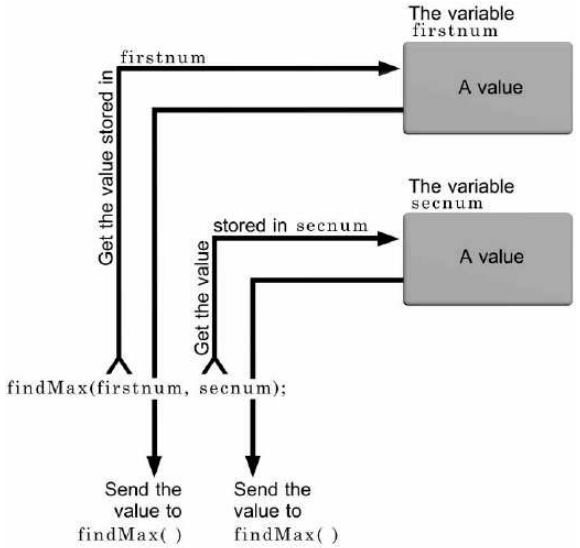


Figure 6.3 `findMax()` receives values

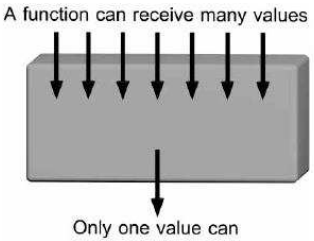
Function Stubs

- **Stubs** คือ ฟังก์ชันฉบับร่าง โดยในการเขียนโปรแกรม เมื่อวางแผนการเรียกใช้งานฟังก์ชันแล้ว สามารถเขียนส่วน **Function Body** ที่แสดงผลค่าตัวแปรไว้ตรวจสอบ และทำให้โปรแกรมทำงานอย่างสมบูรณ์ ก่อนทำการเขียนส่วนการคำนวณจริง **(temporary substitute for yet-to-be-developed code)**

```
float findMax(float x, float y)
{
  printf("In findMax()\n");
  printf("The value of x is %f\n", x);
  printf("The value of x is %f\n ", y);
  return 1.0;
}
```


Returning a Value `return`

- สำหรับฟังก์ชันที่มีการ **return** คือฟังก์ชันที่กำหนด **return type** เป็น **datatype** ต่าง ๆ **(ไม่ใช่ void)**
  - ค่าที่ส่งกลับจะต้องมี **Data type** ตรงกับที่ประกาศไว้ใน **Function Definition** และ **Function Prototype**
  - คำสั่ง `return` จะต้องปรากฏในส่วนของ **Function Body**
  - คำสั่ง `return` จะเป็นคำสั่งสุดท้ายที่ทำงานในฟังก์ชัน
  - ฟังก์ชันสามารถส่งค่ากลับได้เพียง 1 ค่าเท่านั้น



Returning a Value (2)

- เราสามารถส่งค่ากลับในรูปของ **expression** ได้ ดังคำสั่ง

```
return (expression); //or
    return expression;
```

- ค่าส่งกลับที่อยู่ในรูปของ **expression** จะถูกแปลงอัตโนมัติให้อยู่ในรูปแบบเดียวกับชนิดข้อมูลที่กำหนดไว้ใน **Function Header Line** ก่อนที่จะส่งค่ากลับไปยังผู้ที่เรียกใช้ฟังก์ชัน
- สิ่งที่จะต้องระมัดระวังคือ หลังจากแปลงค่าข้อมูลใน **expression** แล้วค่าผลลัพธ์เป็นไปตามที่คาดหวังหรือไม่ และเป็นไปตามชนิดข้อมูลที่กำหนดไว้หรือไม่

Example of Function `findMax()`

```
01 #include <stdio.h>
02 int findMax(int a, int b);  //function prototype
03 int main(){
04     int x, y;
05     printf("Enter x:");scanf("%d",&x);
06     printf("Enter y:");scanf("%d",&y);
07     printf("Max value is: %d",findMax(x,y)); //function call
08     return 0;
09 }
10 int findMax(int a, int b){ //function header
11     if(a>b)
12         return a;
13     else                    //function body
14         return b;
15 }
```

```
Enter x:43
Enter y:23
Max value is: 43
```


Example of Function (2) `convertFtoC`

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 void convertFtoC(double fahrenheit);
04 int main(){
05     float temp_f =0;
06
07     while (temp_f !=-1){
08         printf("Input degree in Fahrenheit :");
09         scanf("%f", &temp_f);
10         if(temp_f!=-1)
11             convertFtoC(temp_f);
12     }
13     return 0;
14 }
15 void convertFtoC(double fahrenheit){
16     double local_temp_C;
17     local_temp_C = (5.0/9.0) * (fahrenheit - 32);
18     printf("%.2f degree F is degree %.2f C\n", fahrenheit, local_temp_C);
19 }
```

```
Input degree in Fahrenheit :34
34.00 degree F is degree 1.11 C
Input degree in Fahrenheit :54
54.00 degree F is degree 12.22 C
Input degree in Fahrenheit :66
66.00 degree F is degree 18.89 C
Input degree in Fahrenheit :-1
```


Common Programming Errors

- ส่งค่าพารามิเตอร์ผิด **Data type** (Passing incorrect data types)
- ไม่ได้ประกาศ**Prototype** ก่อนการเรียกใช้งาน (Omitting a called function's prototype)
- ใส่ ; ที่ **Function Header** (Terminating a function's header line with a semicolon)
- ลืมประกาศ **Datatype** ของอากิวเมนต์ ในส่วน **Function Header** (Forgetting to include a data type for each parameter listed in a function's header line)
- **Return** ค่าผิด **data type** จากที่ประกาศไว้ (Returning a different data type from a function than the data type specified in the function's header line)

Common Compiler Errors

Error	Typical Unix-based Compiler Error Message	Typical Windows-based Compiler Error Message
Terminating a function's header line with a semicolon.	(S) Syntax error	error: missing function header (old-style formal list?)
Passing the incorrect number of parameters into a function.	(E) Missing argument(s)	error C2660: function does not take … arguments
Not having a function prototype.	(S) Syntax error: possible missing ';' or ',' Note that each parameter of the function will generate the following error: (S) Undeclared identifier	error: identifier not found, even with argument-dependent lookup error C2365: redefinition; previous definition was a 'formerly unknown identifier'
Changing the data types of the parameters between the prototype and the function header.	(S) Redeclaration of … differs from previous declaration (I) The data type of parameter differs from the previous type	error: unresolved external symbol referenced in function fatal error: unresolved externals

Common Compiler Errors (2)

Error	Typical Unix-based Compiler Error Message	Typical Windows-based Compiler Error Message
Forgetting to separate the parameters in the function header with a comma.	(S) Syntax error: possible missing ')' or ',' ? Redeclaration of doit differs from previous declaration on line 3 of "Filename.c" (I) Redeclaration of doit has a different number of fixed parameters than the previous declaration (S) Undeclared identifier b	error: syntax error : argument should be preceded by ','
Using a function name that is a reserved syntax word.	(S) Redeclaration of function differs from previous declaration	error: identifier not found, even with argument-dependent lookup

Summary

- **Predefined function**
  - ฟังก์ชันในการทำงานเฉพาะทางบางอย่าง เช่นการคำนวณทางคณิตศาสตร์ การรับค่า การแปลงค่า สามารถหาได้จาก **Standard Library (ขึ้นกับ C-Compiler แต่ละตัว)**
  - โดยการเลือกใช้ฟังก์ชันที่เป็น **Standard Library** นั้น ต้องทำการ **include header file**
  - ในการเลือกใช้งานต้องเลือกส่ง-รับค่าตัวแปร ให้ตรงกับที่มีการประกาศไว้ที่ **Function Prototype**

Summary (2)

- **User Defined Function**
  - ใช้เพื่อแบ่งปัญหาออกเป็นส่วน ๆ โดยในการกำหนดฟังก์ชัน ต้องมีการระบุรูปแบบตัวแปรที่รับเข้า และส่งออก **(Argument and Return Type)**
  - ในการเขียนฟังก์ชัน ต้องมีการประกาศฟังก์ชันก่อนการเรียกใช้
    - เขียนฟังก์ชันย่อยไว้ก่อนถึง **main()**
    - ประกาศ **Function Prototype** ไว้ก่อน **main()** และเขียนฟังก์ชันไว้ต่อท้าย **main()**