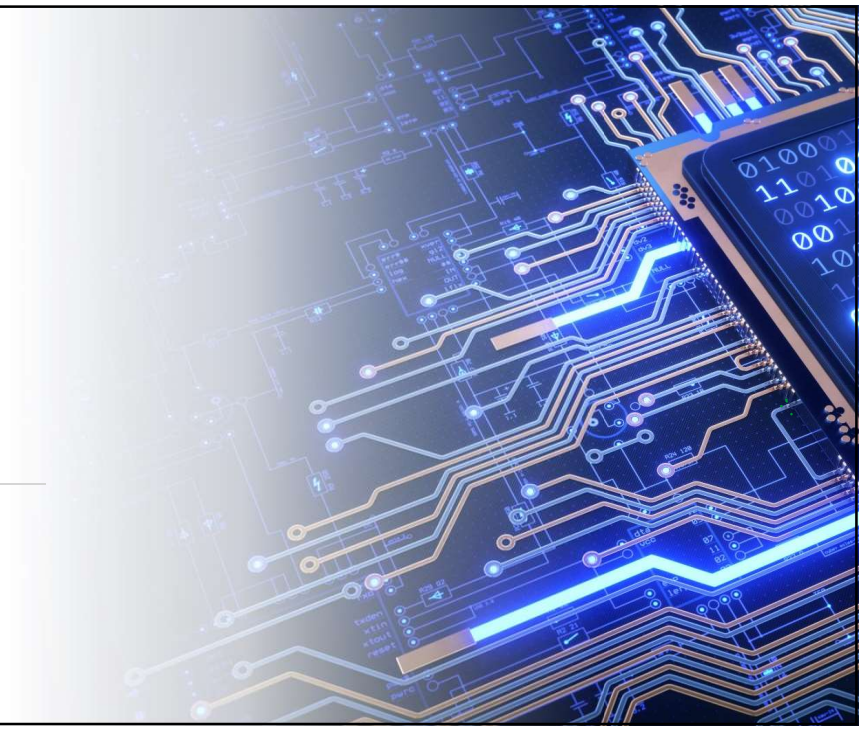


Chapter 7: Function

06016409 – Physical Computing
Kitsuchart Pasupa, PhD
School of Information Technology
King Mongkut's Institute of Technology Ladkrabang

1

Outline

- Introduction to function
- User defined function
  - Function declaration
  - Function design
  - Parameter passing
- Standard function

2

What is function?

It is a block of code that has a name, and it has a property that it is reusable

It can be executed from as many different points in a C Program as required

We pass information to the function called arguments specified when the function is called.

The function either returns some value to the point it was called from or returns nothing.

3

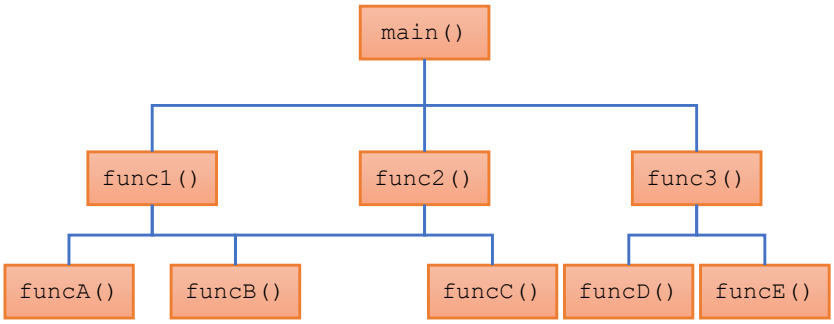
Functions in C

User defined function — Write your own function

Standard function — In standard library

4

Function in C



5

User defined function

Header

```
return_type function_name (parameter_list)
```

Body

```
{   // Local Declarations
    ...
    // Statements
    ...
    // Return
}
```

6

Void Type

- To signify that a function returns no value

```
int FirstEx()
{
  int x;
  ...
  return x;
}
```

```
void SecondEx()
{

  ...

}
```

7

Function Declaration

```
#include<file.h>

type func_name(type);

type variable

int main()
{
   type variable;
   statement.1;
   ...
   statement.n;
   var = func_name(value);
   return 0;
}
```

```
type func_name(type variable)
{
   statement.1;
   ...
   statement.n;
   return(var);
}
```

8

Can we write functions before main program?

9

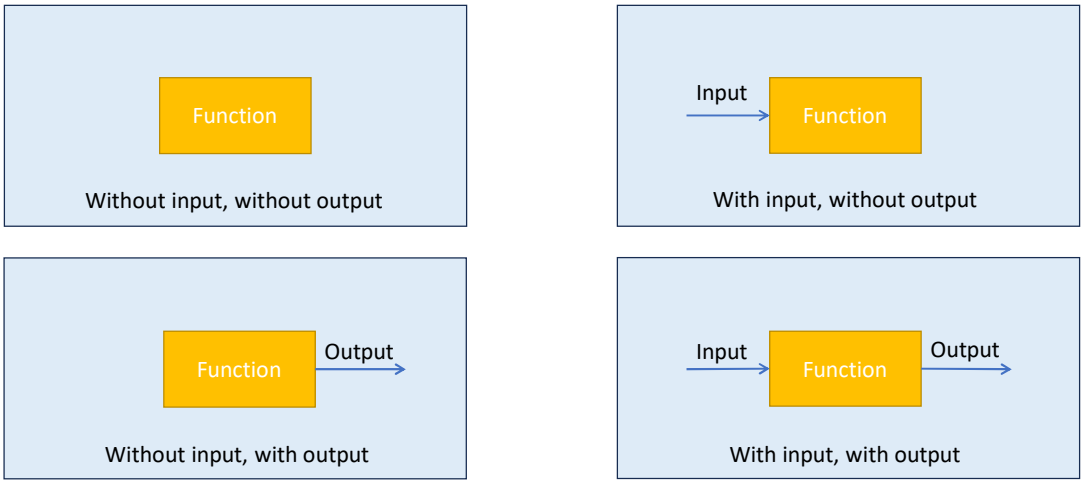
Function Declaration

```
#include<file.h>

type variable

type func_name(type variable)
{
   statement.1;
   ...
   statement.n;
   return(var);
}
```

```
int main()
{
   type variable;
   statement.1;
   ...
   statement.n;
   var = func_name(value);
   return 0;
}
```

10

Function Design



11

Function: Without Input/Output

```
#include <stdio.h>

void hello();

int main()
{
   printf("Function main 1\n");
   hello();
   printf("Function main 2\n");
   return 0;
}

void hello()
{
   printf("Hello World!!\n");
}
```

12

Function: With Input, Without Output

```
#include <stdio.h>

void showValue(int x);

int main()
{
   int i=20;
   showValue(i);
   return 0;
}

void showValue(int x)
{
   printf("%d\n",x);
}
```

i=20

x=20

```
20
```

13

Function: Without Input, With Output

```
#include <stdio.h>

int getInput();

int main()
{
   int input;
   input = getInput();
   printf("input = %d", input);
   return 0;
}

int getInput()
{
   int num;
   printf("Input number: ");
   scanf("%d", &num);
   return num;
}
```

input = 50

num = 50

```
Input number: 50
input = 50
```

14

Function: With Input, With Output

```
#include <stdio.h>

float cubeVol(float x);

int main()
{
   float side = 10.5;
   float volume;
   volume = cubeVol(side);
   printf("Volume of cube = %f", volume);
   return 0;
}

float cubeVol(float x)
{
   return x*x*x;
}
```

side = 10.5

Volume =

x = 10.5

Volume of cube = 1157.625

15

Parameter Passing

```
#include <stdio.h>

void printValue(int x, float y)

int main()
{
   int a=10;
   ...
   printValue(a, 20.5);
   ...
   return 0;
}

void printValue(int x, float y)
{
   printf("%d %f\n", x, y);
}
```

#argument = #parameter

Type of argument ≈ Type of parameter

16

Parameter Passing

Pass by value

Pass by reference

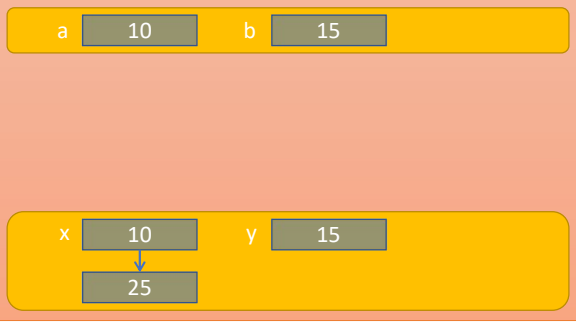
Array

Pointer – *Will be in Pointer Topic*

17

Pass by Value

```
#include <stdio.h>

void add(int x, int y)

int main()
{
   int a = 10;
   int b = 15;
   printf("a = %d\n", a);
   add(a,b);
   printf("a = %d\n", a);
   return 0;
}

void add(int x, int y)
{
   x = x + y;
   printf("x = %d\n", x);
}
```




```
a = 10
x = 25
a = 10
```

18

Passing Array Argument to Functions

```
#include <stdio.h>
#define MAX_NUM 10

int findMax (int list[MAX_NUM]);
int findMin (int list[MAX_NUM]);

int main()
{
   int list[MAX_NUM];
   int i;
   printf("Enter 10 numbers: ");
   for (i = 0; i < MAX_NUM; i++)
      scanf("%d", &list[i]);
   printf("Maximum number = %d\n", findMax(list));
   printf("Minimum number = %d\n", findMin(list));
   return 0;
}
```

```
Enter 10 numbers: 10 11 5 6 4 2 24 16 32 14
Maximum number = 32
Minimum number = 2
```

19

Passing Array Argument to Functions

```
int findMax (int list[MAX_NUM])
{
   int max;
   int i;
   max = list[0];
   for (i = 0; i < MAX_NUM; i++)
      if (list[i] > max)
         max = list[i];
   return max;
}
int findMin (int list[MAX_NUM])
{
   int min;
   int i;
   min = list[0];
   for (i = 0; i < MAX_NUM; i++)
      if (list[i] < min)
         min = list[i];
   return min;
}
```

```
Enter 10 numbers: 10 11 5 6 4 2 24 16 32 14
Maximum number = 32
Minimum number = 2
```

20

Scope of Visibility

- The scope of variables can be global or local.
  - A global variable's scope includes all the statements in a program.
  - The scope of a local variable includes only statements inside the function in which it is declared.
- The same identifier can be reused inside different functions to name different variables.
- A name is local if it is declared in the current scope, and it is global if declared in an outer scope.

21

Example 1

```
#include <stdio.h>

int a;                    // Global Scope
void functionA();

int main()
{
   a = 20;
   printf("main (start) -> a = %d\n", a);
   functionA();
   printf("main (after) -> a = %d\n", a);
   return 0;
}
void functionA()
{
   int a = 10;            // Local Scope
   a = a+5;
   printf("in functionA -> a = %d\n", a);
}
```

```
main (start) -> a = 20
in functionA -> a = 15
main (after) -> a = 20
```

22

Example 2 (1/2)

```
#include <stdio.h>

int x;                    // Global Scope
void functionA();
void functionB(int x);

int main()
{
   x = 10;
   printf("main (start) -> x = %d\n", x);
   functionA();
   printf("main (after funcA) -> x = %d\n", x);
   functionB(x);
   printf("main (after funcB) -> x = %d\n", x);
   return 0;
}
```

```
main (start) -> x = 10
in functionA -> x = 20
main (after funcA) -> x = 20
in functionB -> x = 30
main (after funcB) -> x = 20
```

23

Example 2 (2/2)

```
void functionA()
{
   x = x + 10;            // Global Scope
   printf("in functionA -> x = %d\n", x);
}

void functionB(int x)
{
   x = x + 10;            // Local Scope
   printf("in functionB -> x = %d\n", x);
}
```

```
main (start) -> x = 10
in functionA -> x = 20
main (after funcA) -> x = 20
in functionB -> x = 30
main (after funcB) -> x = 20
```

24

Standard Functions

```
#include <header_file.h>
```

- Standardized collection of header files and library routines used to implement common operations

25

Standard Functions

Mathematics	•math.h
String	•string.h
Character	•ctype.h

26

math.h (1/2)

Functions	Descriptions
double sin(double x)	Sine of x, where x is in radian
double cos(double x)	Cosine of x, where x is in radian
double tan(double x)	Tan of x, where x is in radian
double exp(double x)	e^x where x≈2.718282
double log(double x)	National logarithm of x, where x>0
double log10(double x)	Logarithm of x to base 10, where x>0
double pow(double x, double y)	x^y, where x>0
double sqrt(double x)	√x

27

math.h (2/2)

Functions	Descriptions
double ceil(double x)	Round up to integral value
float ceilf(float x)	
long double ceill(long double x)	
double floor(double x)	Round down to integral value
float floorf(float x)	
long double floorl(long double x)	
int abs(int x)	Round up to integral value
long labs(long x)	
double fabs(double x)	

28

Example

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
   double degree;
   double radian;
   degree = 60.0;
   radian = degree * PI/180;
   printf("degree = %f\n", degree);
   printf("radian = %f\n", radian);
   printf("cos    = %f\n", cos(radian));
   printf("sin    = %f\n", sin(radian));
   printf("tan    = %f\n", tan(radian));
   return 0;
}
```

```
degree = 60.000000
radian = 1.047198
cos    = 0.500000
sin    = 0.866025
tan    = 1.732051
```

29

string.h

Functions	Descriptions
strcpy(str1, str2)	Copy string
strcat(dest1, dest2)	Concatenate strings
strcmp(str1, str2)	Compare strings
strcmpi(str1, str2)	Compare strings (not case sensitive)
strlen(str);	Determine string length

30

ctype.h

Functions	Descriptions
tolower(ch);	Convert character case to lower case
toupper(ch);	Convert character case to upper case

31

Chapter 7: Function - Completed

06016409 – Physical Computing
Kitsuchart Pasupa, PhD
School of Information Technology
King Mongkut's Institute of Technology Ladkrabang

32