มหาวิทยาลัยมหิดล
Mahidol University
Wisdom of the Land

Chapter 8

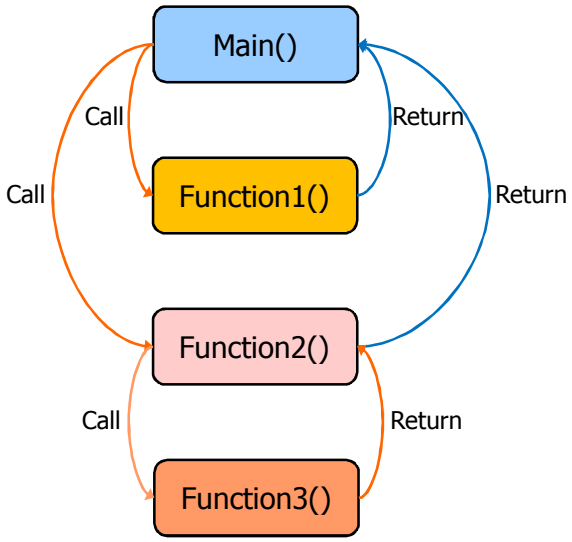
Functions(part1)

Narit Hnoohom

Definition

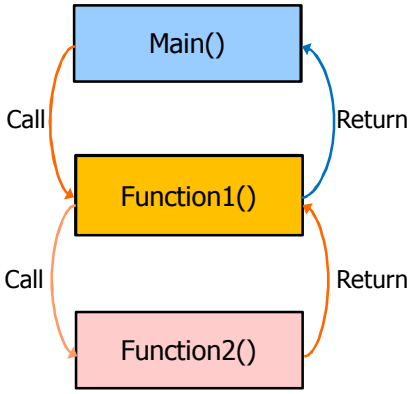
Function is a portion of code within a larger program, which performs a specific task and is relatively independent of the remaining code.

Concept



Structure of Functions

```
1   #include <stdio.h>
2   int sum(int ,int);
3   void PrintOut(int);
4   void main() {
5   int a, b, result;
6   printf("\nEnter numbers(a b): ");
7   scanf("%d" "%d", &a, &b);
8   result = sum(a,b);
9   }
10  int sum(int x, int y) {
11  int temp;
12  temp = x+y;
13  PrintOut(temp);
14  return temp;
15  }
16  void PrintOut(int x) {
17  printf("\nResult of sum is : %d\n", x);
18  }
```



Design methodology

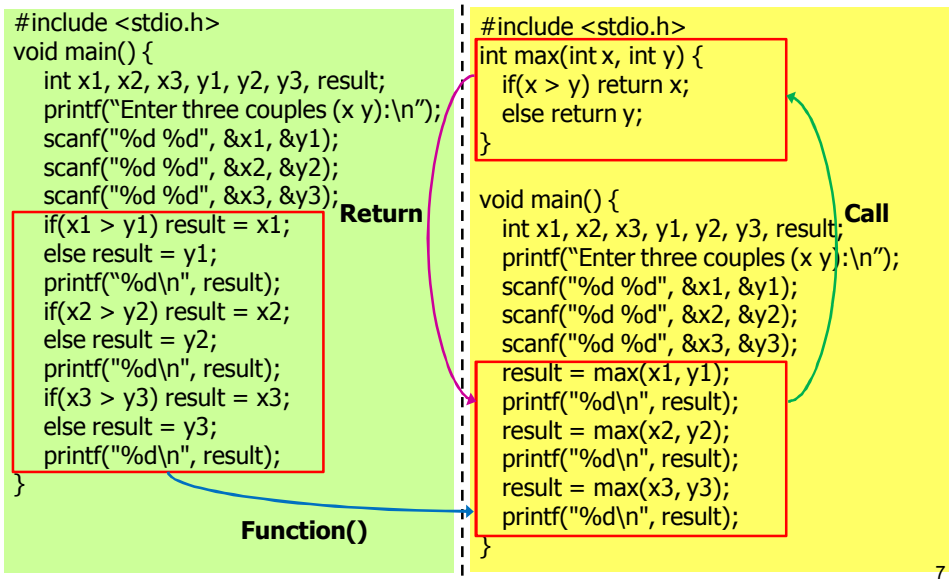
- Step 1: Specify the task.
- Step 2: Break task into smaller pieces.
- Step 3: Any piece that is not small enough then break it further.
- Step 4: Implement each piece using function where appropriate.

Example of Function

- **Write a program to get the integer number three couples in the following order: x1 y1, x2 y2 and x3 y3. Then, print the number that is most out of each couple.**
- Example of result:

```
Enter three couples(x y):
6  9
0  -3
-1  7
9
0
7
```

Non-Function VS. Function

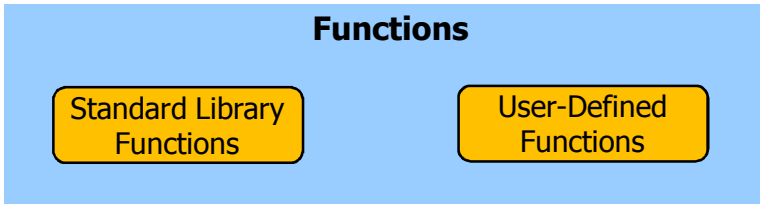


Benefits

There are many advantages to breaking a program up into functions, including:

- reducing the duplication of code in a program, i.e. functions can be called many times.
- enabling reuse of code across multiple programs.
- decomposing complex problems into simpler pieces (this improves maintainability and ease of extension).
- improving readability of a program.
- hiding or regulating part of the program.

Types of Functions



Standard Library Functions

string.h

```
int strlen(char*);
void strcpy(char*,char*);
int strcmp(char*,char*);
void strcat(char*,char*);
```

math.h

```
double sqrt(double);
double pow(double,double);
double ceil(double);
double floor(double);
double fabs(double);
```

stdlib.h

```
double atof(char);
int atoi(char);
int rand();
```

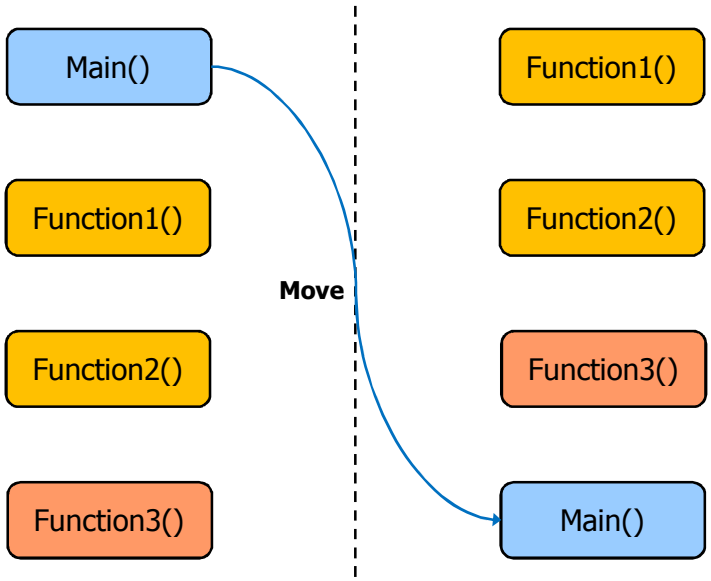
ctype.h

```
char toupper(char);
char tolower(char);
```

User-Defined Functions

- Function prototype
- Function call
- Function implementation: the components of a function include:
  - a body of code to be executed when the function is called.
  - parameters that are passed to the function from the point where it is called.
  - a value that is returned to the point where the call occurs.

Format of Placement User-Defined Functions



Structure of User-Defined Functions

```
1   #include <stdio.h>
2   int square(int);
3   void main()
4   {
5     int a, result;
6     printf("\nEnter a number: ");
7     scanf("%d" , &a);
8     result = square(a);
9     printf("\nResult of square is : %d\n", result);
10  }
11
12  int square(int x)
13  {
14  int temp;
15  temp = x*x;
16  return temp;
17  }
```

Parameter

Argument

Function Prototype

Function Call

Function Implementation

Structure of User-Defined Functions

```
1   #include <stdio.h>
2   int sum(int ,int);
3   void PrintOut(int);
4   void main() {
5   int a, b, result;
6   printf("\nEnter numbers(a b): ");
7   scanf("%d" "%d", &a, &b);
8   result = sum(a,b);
9   }
10  int sum(int x, int y) {
11  int temp;
12  temp = x+y;
13  PrintOut(temp);
14  return temp;
15  }
16  void PrintOut(int x) {
17  printf("\nResult of sum is : %d\n", x);
18  }
```

Parameter

Argument

Function Prototype

Function Call

Function Implementation

Structure of User-Defined Functions

```
1   #include <stdio.h>
2   int square (int x)
3   {
4   int temp;
5   temp = x*x;
6   return temp;
7   }
8
9   void main()
10  {
11    int a, result;
12    printf("\nEnter a number: ");
13    scanf("%d" , &a);
14    result = square(a);
15    printf("\nResult of square is : %d\n", result);
16  }
```

Argument

Parameter

Function Implementation

Function Call

Function Prototype

- Function prototype declaration is necessary in order to provide information to the C compiler about function.
- Function prototype can be declared as follows:
  1) int sum(int a, int b);
  2) int sum(int, int);
  3) int sum(int x, int y);

```
int sum(int x, int y)
{
  int temp;
  temp = x+y;
  return (temp);
}
```

Types of User-Defined Functions

User Defined Functions

- No argument, No return value
- No argument, Return a value
- Arguments, No return value
- Arguments, Return a value

Types of User-Defined Functions

- **No argument, No return value**

```
void function_name (void)
{
    statement;
}
```

- **No argument, Return a value**

```
return_type function_name (void)
{
    variable_type variable_name;       //declaration variable in function
    statement;
    return (value or variable_name);   //function can return only one
}
```

Types of User-Defined Functions

- **Arguments, No return value**

```
void function_name (argument_type, argument_name, …)
{
    statement;
}
```

- **Arguments, Return a value**

```
return_type function_name (argument_type, argument_name, …)
{
    variable_type variable_name;       //declaration variable in function
    statement;
    return (value or variable_name);   //function can return only one
}
```


No argument, No return value

- **Example1**

```
1   #include <stdio.h>
2   void myprint (void); //function prototype
3
4   void main()
5   {
6       myprint(); //function call
7   }
8
9   void myprint (void)
10  {
11      printf("Mahidol University");
12  }
```

- **No argument, No return value**

```
void function_name (void)
{
    statement;
}
```

- **Result:**

  Mahidol University

No argument, No return value

- **Example2**

```
1  #include <stdio.h>
2  void header (void); //function prototype
3  void main()
4  {
5  int a,b;
6  header(); //function call
7  printf("Inputs(a b): ");
8  scanf("%d %d", &a, &b);
9  printf("Output(a+b): %d", a+b);
10 printf("\n************************\n");
11 getch();
12 }
13 void header (void)
14 {
15 printf("************************\n");
16 printf("* My Summation Program *\n");
17 printf("************************\n");
18 }
```


No argument, No Return a value

- **Example2**

```
#include <stdio.h>
void header (void); //function prototype
void main()
{
int a,b;
header(); //function call
printf("Inputs(a b): ");
scanf("%d %d", &a, &b);
printf("Output(a+b): %d", a+b);
printf("\n************************\n");
getch();
}
void header (void)
{
printf("************************\n");
printf("* My Summation Program *\n");
printf("************************\n");
}
```

Call **Return**

No argument, No return value

- **Example2**

```
1   #include <stdio.h>
2   void header (void); //function prototype
3   void main()
4   {
5   int a,b;
6   header(); //function call
7   printf("Inputs(a b): ");
8   scanf("%d %d", &a, &b);
9   printf("Output(a+b): %d", a+b);
10  printf("\n**********************\n");
11  getch();
12  }
13  void header (void)
14  {
15  printf("**********************\n");
16  printf("* My Summation Program *\n");
17  printf("**********************\n");
18  }
```

- **Result:**

```
**********************
* My Summation Program *
**********************
Inputs(a b): 3 5
Output(a+b): 8
**********************
```


Exercise1

- **Write a program to display a rectangle, using the asterisk (*).**
- Example of result:

  Rectangle size: 5x4

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

No argument, Return a value

- **Example1**

```
1   #include <stdio.h>
2   float inputvalue (void); //function prot
3
4   void main()
5   {
6     printf("%4.2f", inputvalue()); //func
7   }
8
9   float inputvalue (void)
10  {
11    float x;
12    scanf("%f",&x);
13    return (x);  // return x;
14  }
```

- **No argument, Return a value**

```
return_type function_name (void)
{
    variable_type variable_name;
    statement;
    return (value or variable_name);
}
```

- **Result:**

  2.5

  2.50

No argument, Return a value

- **Example2**

```
1   #include <stdio.h>
2   int calculate(void); //function prototype
3   void main()
4   {
5   int value=1;
6   printf("(before calling a function) The number is %d", value);
7   value = calculate(); //function call
8   printf("\n(after calling a function) The number is %d", value);
9   getch();
10  }
11  int calculate(void) //function implementation
12  {
13  int val=2, i;
14  for (i=0; i<4; i++)
15  val = (val*i)+i;
16  return val;
17  }
```

No argument, Return a value

- **Example2**

```
#include <stdio.h>
int calculate(void); //function prototype
void main()
{
int value=1;
printf("(before calling a function) The number is %d", value);
value = calculate(); //function call
printf("\n(after calling a function!!!) The number is %d", value);
getch();
}
int calculate(void) //function implementation
{
int val=2, i;
for (i=0; i<4; i++)
val = (val*i)+i;
return val;
}
```

Call **Return**

No argument, Return a value

- **Example2**

```
1  #include <stdio.h>
2  int calculate(void); //function pr
3  void main()
4  {
5  int value=1;
6  printf("(before calling a function) The number is %d", value);
7  value = calculate(); //function call
8  printf("\n(after calling a function) The number is %d", value);
9  getch();
10 }
11 int calculate(void) //function implementation
12 {
13 int val=2, i;
14 for (i=0; i<4; i++)
15 val = (val*i)+i;
16 return val;
17 }
```

- **Result:**

(Before calling a function) The number is 1
(After calling a function) The number is 15

- **Result:**

2*0+0=0
0*1+1=1
1*2+2=4
4*3+3=15

Arguments, No return value

- **Example1**

```
1  #include <stdio.h>
2  void myprint (char x, int y); /
3
4  void main() {
5     myprint('b', 2); //function c
6  }
7
8  void myprint (char x, int y)
9  {
10    while (y > 0)
11    {
12       printf("%c", x);
13       y--;
14    }
15 }
```

- **Arguments, No return value**

```
void function_name (parameter_type,
parameter_name, ...)
{
    variable_type variable_name;
    statement;
}
```

- **Result:**

  b b

Arguments, No return value

```
#include <stdio.h>
void increment(int, float);
//function prototype
void main()
{
int i; float x;
printf("\nEnter an integer number:");
scanf("%d", &i);
printf("\nEnter a floating point
    number:");
scanf("%f", &x);
printf("\n\n(Before calling a function)
The numbers are %5d %10.4f", i, x);
increment(i, x); //function call
printf("\n\n(After calling a function)
The numbers are %5d %10.4f", i, x);
getch();
}
```

```
/*function implementation
first parameter passed is renamed m
second parameter passed is renamed n*/
void increment(int m, float n)
{
//this i is local to increment()
//it is not the same i defined in main()
int i = 2;
m = m + i;
//first passed parameter is modified
n = n + i;
//second passed parameter is modified
//(float = float + int)
printf("\n\n(Print within a function)
The numbers are %5d %10.4f", m, n);
}
```

Arguments, No return value

```
#include <stdio.h>
void increment(int, float);
//function prototype
void main()
{
int i; float x;
printf("\nEnter an integer number:");
scanf("%d", &i);
printf("\nEnter a floating point
    number:");
scanf("%f", &x);
printf("\n\n(Before calling a function)
The numbers are %5d %10.4f", i, x);
increment(i, x); //function call
printf("\n\n(After calling a function)
The numbers are %5d %10.4f", i, x);
getch();
}
```

Call

```
/*function implementation
first parameter passed is renamed m
second parameter passed is renamed n*/
void increment(int m, float n)
{
//this i is local to increment()
//it is not the same i defined in main()
int i = 2;
m = m + i;
//first passed parameter is modified
n = n + i;
//second passed parameter is modified
//(float = float + int)
printf("\n\n(Print within a function)
The numbers are %5d %10.4f", m, n);
}
```

Return

Arguments, No ret

- **Result:**

```
Enter an integer number: 2
Enter a floating point number: 4
(Before calling a function) The numbers are  2   4.0000
(Print within a function) The numbers are    4   6.0000
(After calling a function) The numbers are   2   4.0000
```

```
#include <stdio.h>
void increment(int, float);
//function prototype
void main()
{
int i; float x;
printf("\nEnter an integer number:");
scanf("%d", &i);
printf("\nEnter a floating point
    number:");
scanf("%f", &x);
printf("\n\n(Before calling a function)
The numbers are %5d %10.4f", i, x);
increment(i, x); //function call
printf("\n\n(After calling a function)
The numbers are %5d %10.4f", i, x);
getch();
}
```

```
void increment(int m, float n)
{
//this i is local to increment()
//it is not the same i defined in main()
int i = 2;
m = m + i;
//first passed parameter is modified
n = n + i;
//second passed parameter is modified
//(float = float + int)
printf("\n\n(Print within a function)
The numbers are %5d %10.4f", m, n);
}
```

Exercise2

- **Write a program to get an integer number.**
- Check that the number is even or odd.

- Example of result:

  Enter an integer number: 5
  The number is even = 0
  The number is odd = 1

Arguments, Return a value

```
1  #include <stdio.h>
2  char myprint (int y); //function prototype
3  void main() {
4     char result;
5     result = myprint(2); //function call
6     printf("%c", result);
7  }
8
9  char myprint (int y) {
10    char ch;
11    scanf("%c", &ch);
12    while (y > 0) {
13       printf("%c", ch);
14       y--;
15    }
16    return ch;
17 }
```

- **Arguments, Return a value**

```
return_type function_name (parameter_type, parameter_name, ...)
{
    variable_type variable_name;
    statement;
    return (value or variable_name);
}
```

- **Result:**

  b b b

Arguments, Return a value

```
#include <stdio.h>
float sum(float, float);
//function prototype
void main()
{
float x, y, result;
printf("\nEnter 1st floating point
number: ");
scanf("%f",&x);
printf("\nEnter 2nd floating point
number: ");
scanf("%f",&y);
printf("\n\n(Before calling a function)
The numbers are %10.4f %10.4f", x,y);
result = sum(x,y); //call function
printf("\n\n(After calling a function) The
summation result is %10.4f", result);
getch();
}
```

```
/*function implementation
first passed parameter is renamed m
second passed parameter is renamed n*/
float sum(float m, float n)
{
float temp; //used to hold result
temp = m + n;
return(temp);
}
```


Arguments, Return a value

```
#include <stdio.h>
float sum(float, float);
//function prototype
void main()
{
float x, y, result;
printf("\nEnter 1st floating point
number: ");
scanf("%f",&x);
printf("\nEnter 2nd floating point
number: ");
scanf("%f",&y);
printf("\n\n(Before calling a function)
The numbers are %10.4f %10.4f", x,y);
result = sum(x,y); //call function
printf("\n\n(After calling a function) The
summation result is %10.4f", result);
getch();
}
```

Call

```
/*function implementation
first passed parameter is renamed m
second passed parameter is renamed n*/
float sum(float m, float n)
{
float temp; //used to hold result
temp = m + n;
return(temp);
}
```

Return

Arguments, Retu

- **Result:**

Enter 1st floating point number: 2
Enter 2nd floating point number: 3
(Before calling a function) The numbers are 2.0000 3.0000
(After calling a function) The summation result is 5.0000

```
#include <stdio.h>
float sum(float, float);
//function prototype
void main()
{
float x, y, result;
printf("\nEnter 1st floating point
number: ");
scanf("%f",&x);
printf("\nEnter 2nd floating point
number: ");
scanf("%f",&y);
printf("\n\n(Before calling a function)
The numbers are %10.4f %10.4f", x,y);
result = sum(x,y); //call function
printf("\n\n(After calling a function) The
summation result is %10.4f", result);
getch();
}
```

```
second passed parameter is renamed n*/
float sum(float m, float n)
{
float temp; //used to hold result
temp = m + n;
return(temp);
}
```


Exercise3

- **Write a program to get two integer numbers that are in range 5 and 50.**
- **If the entered numbers are out of range, showing prompt to enter numbers again.**

- Example of result:

```
Enter x: 3
Enter y: 30
Enter number in range >= 5 and <= 50
Enter x: 5
Enter y: 50

Multiply of 5 and 50 is 250
```

Thanks for your attention