

Chapter 8
Function

Outline

- Functions: Program Modules in C
- Function Definitions
- Function Prototypes
- Math Library Functions
- Random Number Generation
- Headers
- Calling Functions: Call-by-value and Call-by-reference
- Scope Rules
- Recursion

What is a Function?

- Top-down design: Divide and conquer
  - Partition a problem into several small sub-problems
  - Each sub-problem can be solved by a function (module)
- Function
  - A module with a specific function name
  - Given a set of inputs, return an output
  - Two types of functions
    - User-defined functions
    - ANSI C standard library functions, such as print(), scanf(), pow()

Function Call

- Invoke functions
  - Provide function name and arguments (inputs)
  - Return results (output)
  - Format: output = function_name(input1, input2, …)
- Example
  - ***printf("There are 50 students\n");***
    - Function name: ***printf***
    - Inputs: ***"There are 50 students\n"***
  - ***pow(5.0, 3.0)***
    - Function name: ***pow***
    - Inputs: ***5.0, 3.0***
    - Output: ***125 (5^3)***

Advantages of Functions

- Reusability
- Readability
- Ease of maintenance
- Avoid code repetition
- Use ANCI C standard library to improve portability



Function Definition

```
return-data-type function-name (data-type arg1, data-type arg2, …) {
        declarations;
        statements;
        return output;
}
```

- Function-name
  - Any valid identifier (follow the same rule with variables)
- Parameter list
  - A set of inputs separated by comma (,)
  - Specify a data type for each input argument



Function Definition (Cont.)

- Return-data-type
  - Data type of the result
  - void: Indicate that the function returns nothing
  - Need to return an output with the specified data type
    - If return-data-type is void
      return;
    - If something returned
      return output; // output needs to be the specified type
- Function body
  - Declarations and statements
  - Declarations: *variables used inside the function block, which can not be used outside the function*.
  - Functions can not be defined inside a function.

An Example of Function Definition

```
double compute_avg(int a, int b, int c) {
    return (double) (a + b + c)/3;
}
```

Starting point

```
int main() {
    double avg;
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    avg = compute_avg(a, b, c);        // call function
    return 1;
}
```

Function definition:
Name: compute_avg
Inputs: a, b, c (integer)
Output: return a double value

A function needs to be defined before function call

Function Prototype

- Declare function prototypes before function call if the function is defined after function call.
- Put semicolon (;) at the end of function prototype
- Argument names can be omitted in function prototypes

```
double compute_avg(int, int, int);                    // function prototype

int main() {
        double avg;
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        avg = compute_avg(a, b, c);                   // call function
        return 1;
}

double compute_avg(int a, int b, int c) {   // function definition
    return (double) (a + b + c)/3;
}
```

Function Prototype (Cont.)

- Parameter names are sometimes included in function prototypes for documentation purposes. The compiler ignores these names.

```
double compute_avg(int a, int b, int c);   // function prototype

int main() {
        double avg;
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        avg = compute_avg(a, b, c);              // call function
        return 1;
}

double compute_avg(int a, int b, int c) {   // function definition
    return (double) (a + b + c)/3;
}
```



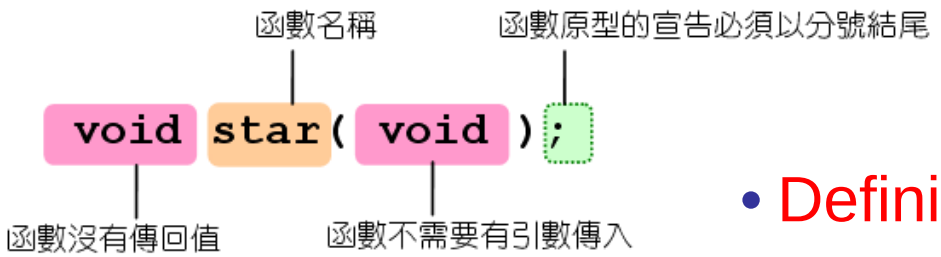
Simple Function Example

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  void star(void);      /* star()函數的原型 */
05  int main(void)
06  {
07     star();                          /* 呼叫 star 函數 */
08     printf("歡迎使用 C 語言\n");
09     star();                          /* 呼叫 star 函數 */
10     system("pause");
11     return 0;
12  }
13
14  void star(void)
15  {
16     printf("**************\n");
17     return;
18  }
```

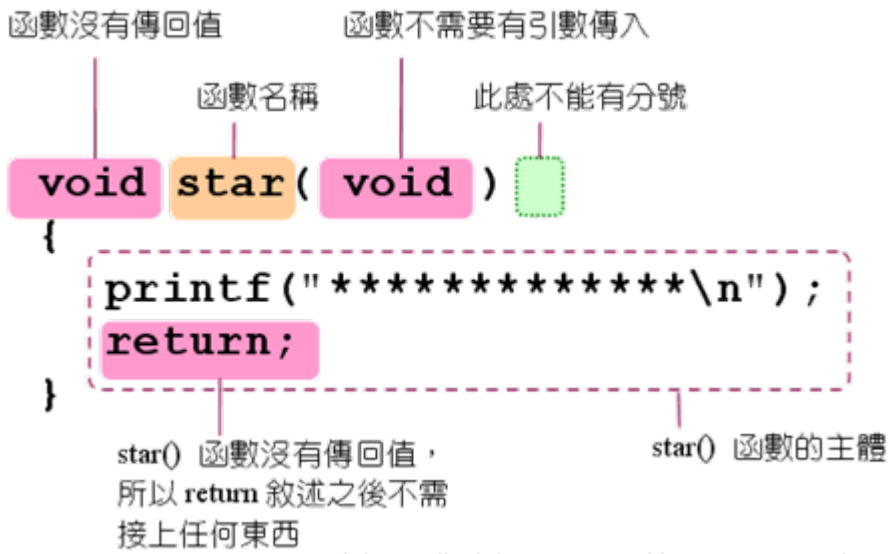
```
/* OUTPUT---

**************
歡迎使用 C 語言
**************
--------------------*/
```

Simple Function Example (Cont.)

- Declaration of star():



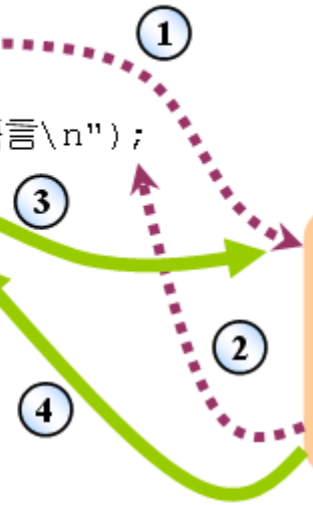
- Definition of star():



Simple Function Example (Cont.)

```
05  int main(void)
06  {
07     star();
08     printf("歡迎使用 C 語言\n");
09     star();
10     system("pause");
11     return 0;
12  }
```

```
14  void star(void)
15  {
16     printf("*************\n");
17     return;
18  }
```



1. 第 7 行呼叫 star() 函數，此時程式跳到第 14 行執行
2. star() 函數執行完畢，此時返回主程式，繼續執行第 8 行
3. 第 9 行呼叫 star() 函數，此時程式再度跳到第 14 行執行
4. star() 函數執行完畢，此時返回主程式，繼續執行第 10 行

Another Simple Example

- Call function in main()

```
/* OUTPUT---
5+3=8
--------------------*/
```

```
01  /* prog8_2, 使用 add()函數 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int add(int,int);              /* add()函數的原型 */
05  int main(void)
06  {
07     int sum, a=5, b=3;
08     sum=add(a,b);               /* 呼叫 add()函數，並把傳回值設給 sum */
09     printf("%d+%d=%d\n",a,b,sum);
10     system("pause");
11     return 0;
12  }
13  int add(int num1, int num2)    /* add()函數的定義 */
14  {
15     int a;                      /* 於 add()函數裡宣告變數 a */
16     a=num1+num2;
17     return a;                   /* 傳回 num1+num2 的值 */
18  }
```

Another Simple Example (Cont.)

- Put function definition before main()

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   int add(int num1, int num2)
05   {
06      int a;
07      a= num1+num2;
08      return a;
09   }
10   int main(void)
11   {
12      int sum, a=5, b=3;
13      sum=add(a,b);
14      printf("%d+%d=%d\n",a,b,sum);
15
16      system("pause");
17      return 0;
18   }
```

將 add() 放在 main() 函數的前面

main() 函數置於 add() 的後面

```
/* OUTPUT---
5+3=8
--------------------*/
```

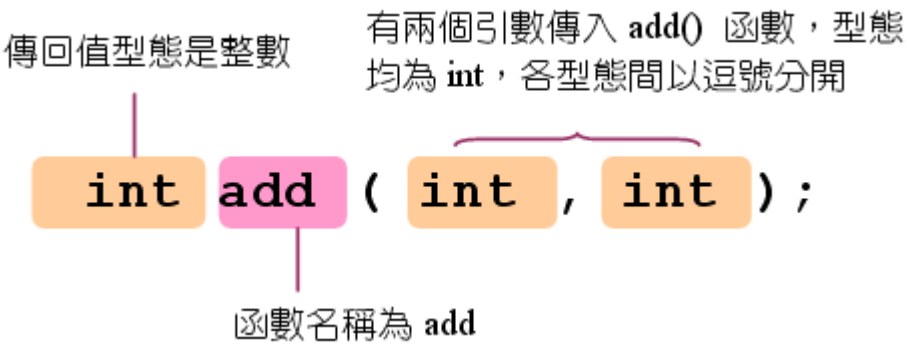


Another Simple Example (Cont.)

- **Function Declaration:**

```
return-data-type function-name(data-type1, data-type2,...);
```

- **add() is declared to accept two integers and return integer.**



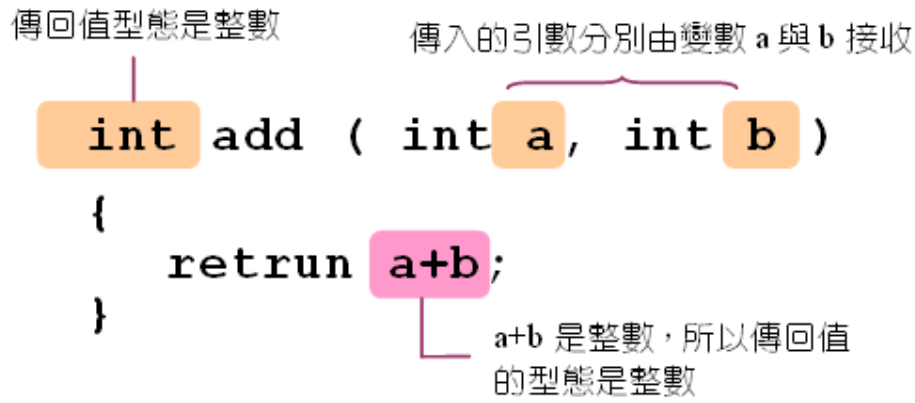


Another Simple Example (Cont.)

- **Function definition**

```
return-data-type function-name(data-type1 para1, …, data-typen paran)
{
    variable declarations;
   statements;
   return expression;    /* return the value of “expression” */
}
```

- **Definition of add():**





Function Practice – display (1/2)

- Print character with function display()

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   void display(char,int);   /* display()函數的原型 */
05   int main(void)
06   {
07      int n;
08      char ch;
09      printf("請輸入欲列印的字元:");
10      scanf("%c",&ch);
11      printf("請問要印出幾個字元:");
12      scanf("%d",&n);
13      display(ch,n);          /* 呼叫自訂的函數，印出 n 個 ch 字元 */
14
15      system("pause");
16      return 0;
17   }
```

Function Practice – display (2/2)

```
19  void display(char ch,int n)    /* 自訂的函數 display() */
20  {
21     int i;
22     for(i=1;i<=n;i++)           /* for 迴圈，可印出 n 個 ch 字元 */
23        printf("%c",ch);            /* 印出 ch 字元 */
24     printf("\n");
25     return;
26  }
```

```
/* OUTPUT--

請輸入欲列印的字元:&
請問要印出幾個字元:12
&&&&&&&&&&&&
----------------------*/
```

Function Practice – abs()

- Calculate the absolute value of the input value

$$\text{abs}(n) = \begin{cases} n; & n \geq 0 \\ -n; & n < 0 \end{cases}$$

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  int abs(int);                  /* 宣告函數 abs()的原型 */
05  int main(void)
06  {
07     int i;
08     printf("Input an integer:");          /* 輸入整數 */
09     scanf("%d",&i);
10     printf("abs(%d)=%d\n",i,abs(i));      /* 印出絕對值 */
11     system("pause");
12     return 0;
13  }
14  int abs(int n)   /* 自訂的函數 abs(),傳回絕對值 */
15  {
16     if (n<0)
17        return -n;
18     else
19        return n;
20  }
```

```
/* OUTPUT---

Input an integer:-6
abs(-6)=6
--------------------*/
```



Function Practice – power(x, n) – x^n (1/2)

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   double power(double, int);     /* 宣告函數 power()的原型 */
05   int main(void)
06   {
07      double x;       /* x 為底數 */
08      int n;          /* n 是次方 */
09      printf("請輸入底數與次方:");
10      scanf("%lf,%d",&x,&n);       /* 輸入底數與次方 */
11      printf("%lf 的%d 次方=%lf\n",x,n,power(x,n));
12      system("pause");
13      return 0;
14   }
15   double power(double base, int n)   /* power()函數的定義 */
16   {
17      int i;
18      double pow=1.0;
19      for(i=1;i<=n;i++)        /* for() 迴圈，用來將底數連乘 n 次 */
20         pow=pow*base;
21      return pow;
22   }
```

```
/* OUTPUT--------

請輸入底數與次方:5.0,3
5.000000 的 3 次方=125.000000
--------------------------*/
```



Function Practice – power(x, n) – x^n (2/2)

i	pow	pow=pow*base
1	1.0	pow=1.0*5.0=5.0
2	5.0	pow=5.0*5.0=25.0
3	25.0	pow=25.0*5.0=125.0
4		

不符合 for 迴圈的判斷條件（i<=3），
跳出 for 迴圈，傳回 pow=125.0

```
15  double power(double base, int n)
16  {
17     int i;
18     double pow=1.0;
19     for(i=1;i<=n;i++)
20        pow=pow*base;
21     return pow;
22  }
```



Function Practice – is_prime(x, n) (1/2)

Search for a prime

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  int is_prime(int);              /* 宣告函數is_prime()的原型 */
05  int main(void)
06  {
07     int i;
08     for(i=2;i<=30;i++)           /* 找出小於 30 的所有質數 */
09        if(is_prime(i))           /* 呼叫 is_prime()函數 */
10           printf("%3d",i);       /* 如果是質數,便把此數印出來 */
11     printf("\n");
12     system("pause");
13     return 0;
14  }
15  int is_prime(int num)      /* is_prime()函數,可測試 num 是否為質數 */
16  {
17     int i;
18     for(i=2;i<=num-1;i++)
19        if(num%i==0)
20           return 0;
21     return 1;
22  }
```

```
/* OUTPUT----------
  2  3  5  7 11 13 17 19 23 29
-------------------------*/
```



Function Practice – is_prime(x, n) (2/2)

num=7

i	num%i
2	7%2=1
3	7%3=1
4	7%4=3
5	7%5=2
6	7%6=1
7	

2~6 都無法整除 7，所以 7 是質數

不符合 for 迴圈的判斷條件（i<=num-1），跳出 for 迴圈，執行第 22 行，傳回 1，代表 7 是質數

num=9

i	num%i
2	9%2=1
3	9%3=0

3 可以整除 9，所以 9 不是質數

符合 20 行的判斷條件，傳回 0，代表 9 不是質數

```
15  int is_prime(int num)
16  {
17      int i;
18      for(i=2;i<=num-1;i++)
19          if(num%i==0)
20              return 0;
21      return 1;
22  }
```



Function Practice - Multiple Functions (1/2)

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   void sum(int);          /* 定義函數的原型 */
05   void fac(int);          /* 定義函數的原型 */
06   int main(void)
07   {
08      fac(5);              /* 呼叫 fac()函數 */
09      sum(5);              /* 呼叫 sum()函數 */
10
11      system("pause");
12      return 0;
13   }
```

```
/* OUTPUT---

1*2*...*5=120
1+2+...+5=15
----------------------*/
```

Function Practice - Multiple Functions (2/2)

```
14  void fac(int a)       /* 自訂函數 fac()，計算 a! */
15  {
16     int i,total=1;
17     for(i=1;i<=a;i++)
18        total*=i;
19     printf("1*2*...*%d=%d\n",a,total);   /* 印出 a!的結果 */
20  }
21
22  void sum(int a)       /* 自訂函數 sum()，計算 1+2+...+a 的結果*/
23  {
24     int i,total=0;
25     for(i=1;i<=a;i++)
26        total+=i;
27     printf("1+2+...+%d=%d\n",a,total);   /* 印出加總的結果 */
28  }
```

```
/* OUTPUT---

1*2*...*5=120
1+2+...+5=15
--------------------*/
```

Function Call Between Functions (1/2)

$$\pi = 4\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{2k-1} = 4\left(\frac{(-1)^{1-1}}{2(1)-1}+\frac{(-1)^{2-1}}{2(2)-1}+\frac{(-1)^{3-1}}{2(3)-1}+\frac{(-1)^{4-1}}{2(4)-1}+\ldots\right) = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}+\ldots\right)$$

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   double Leibniz(int);                    /* 宣告函數 Leibniz()的原型 */
05   double power(double, int);              /* 宣告函數 power()的原型 */
06   int main(void)
07   {
08      int i;
09      for(i=1;i<=10000;i++)                /* 找出前 10000 個π的估算值 */
10         printf("Leibniz(%d)=%12.10f\n",i,Leibniz(i));
11      system("pause");
12      return 0;
13   }
14
```



Function Call Between Functions (2/2)

```
15  double Leibniz(int n)
16  {
17     int k;
18     double sum=0.;
19     for(k=1;k<=n;k++)
20        sum=sum+power(-1.0,k-1)/(2*k-1);
21     return 4*sum;
22  }
23
24  double power(double base, int n)
25  {
26    int i;
27     double pow=1.0;
28     for(i=1;i<=n;i++)
29        pow=pow*base;
30     return pow;
31  }
```

$$\pi = 4\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{2k-1}$$

Recursion - Factorial Function (1/2)

- Recursion: a function calls itself.
- Use recursion to calculate factorial function (*n*!)

$$1\times 2\times\cdots\times(n-1)\times n = \underbrace{1\times 2\times\cdots\times(n-1)}_{\text{fac}(n-1)}\times n = n\times\text{fac}(n-1)$$

$$\text{fac}(n) = \begin{cases} 1\times 2\times\cdots\times n\,; & n\geq 1 \\ 1\,; & n=0 \end{cases}$$

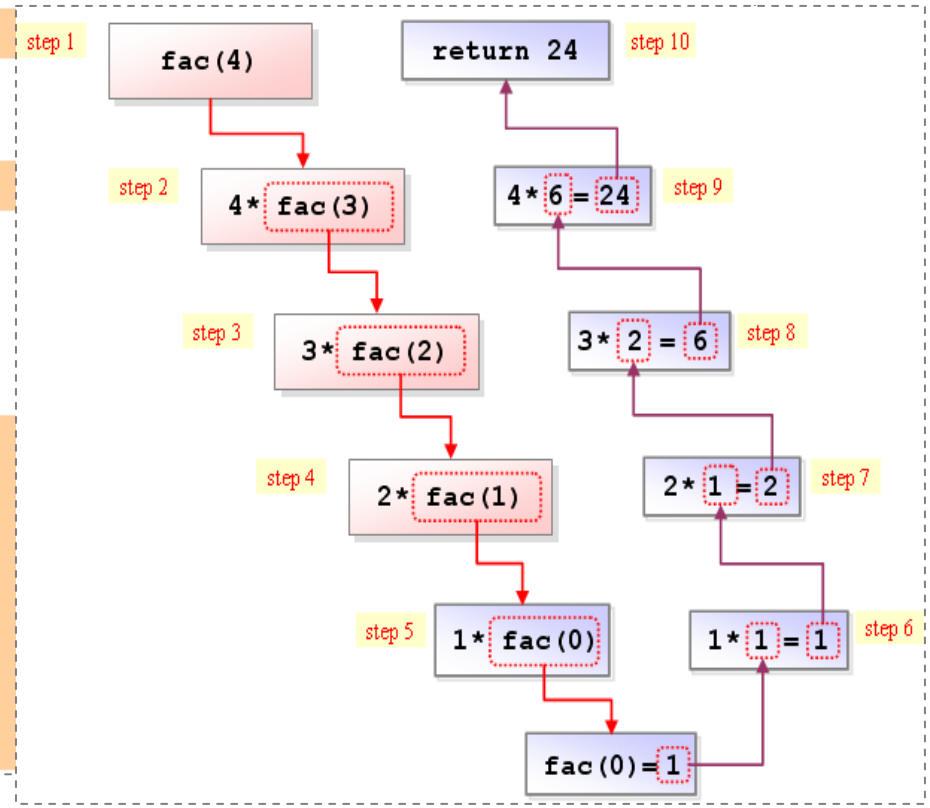
$$\text{fac}(n) = \begin{cases} n\times\text{fac}(n-1)\,; & n\geq 1 \\ 1\,; & n=0 \end{cases}$$



Recursion - Factorial Function (2/2)

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   int fac(int);
05   int main(void)
06   {
07      printf("fac(4)=%d\n",fac(4));
08
09      system("pause");
10      return 0;
11   }
12   int fac(int n)
13   {
14      if(n>0)
15         return (n*fac(n-1));
16      else
17         return 1;
18   }
```

```
/* OUTPUT--

fac(4)=24
--------------------*/
```



Recursion – Power Function (1/2)

- b^n:

$$(b \times b \times \cdots) \times b = \underbrace{(b \times b \times \cdots)}_{\text{連乘}n-1\text{次}} \times b = \text{power}(b, n-1) \times b$$

$$\text{power}(b,n) = \begin{cases} \underbrace{b \times b \times \cdots \times b}_{\text{連乘}n\text{次}}; & n \geq 1 \\ 1; & n = 0 \end{cases}$$

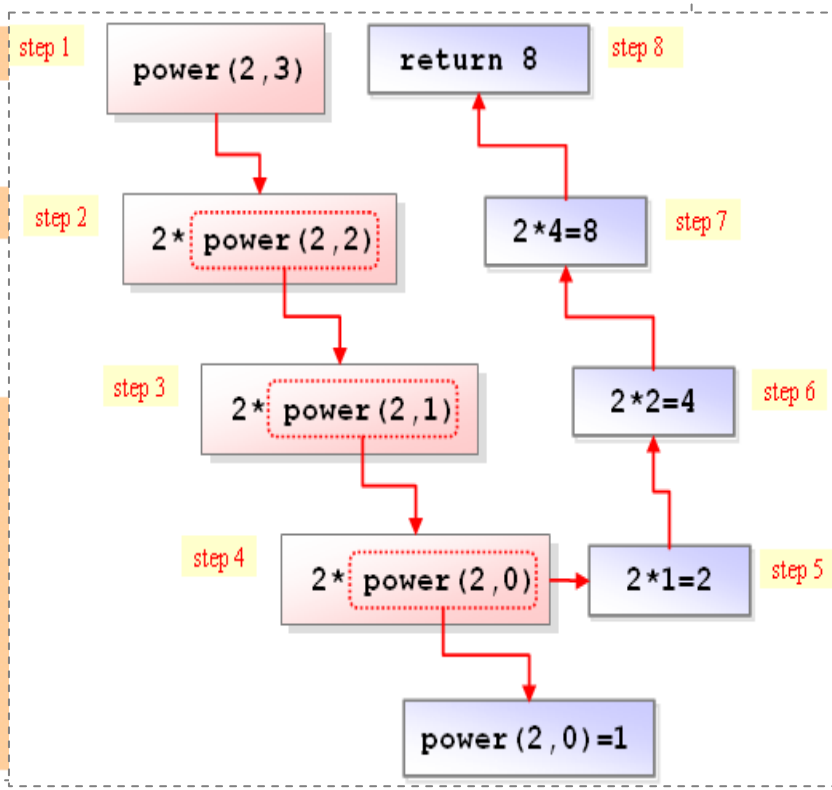
$$\text{power}(b,n) = \begin{cases} b \times \text{power}(b, n-1); & n \geq 1 \\ 1; & n = 0 \end{cases}$$



Recursion – Power Function (2/2)

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   int power(int,int);
05   int main(void)
06   {
07      printf("power(2,3)=%d\n",power(2,3))
08      system("pause");
09      return 0;
10   }
11   int power(int b,int n)
12   {
13      if(n==0)
14         return 1;
15      else
16         return (b*power(b,n-1));
17   }
```

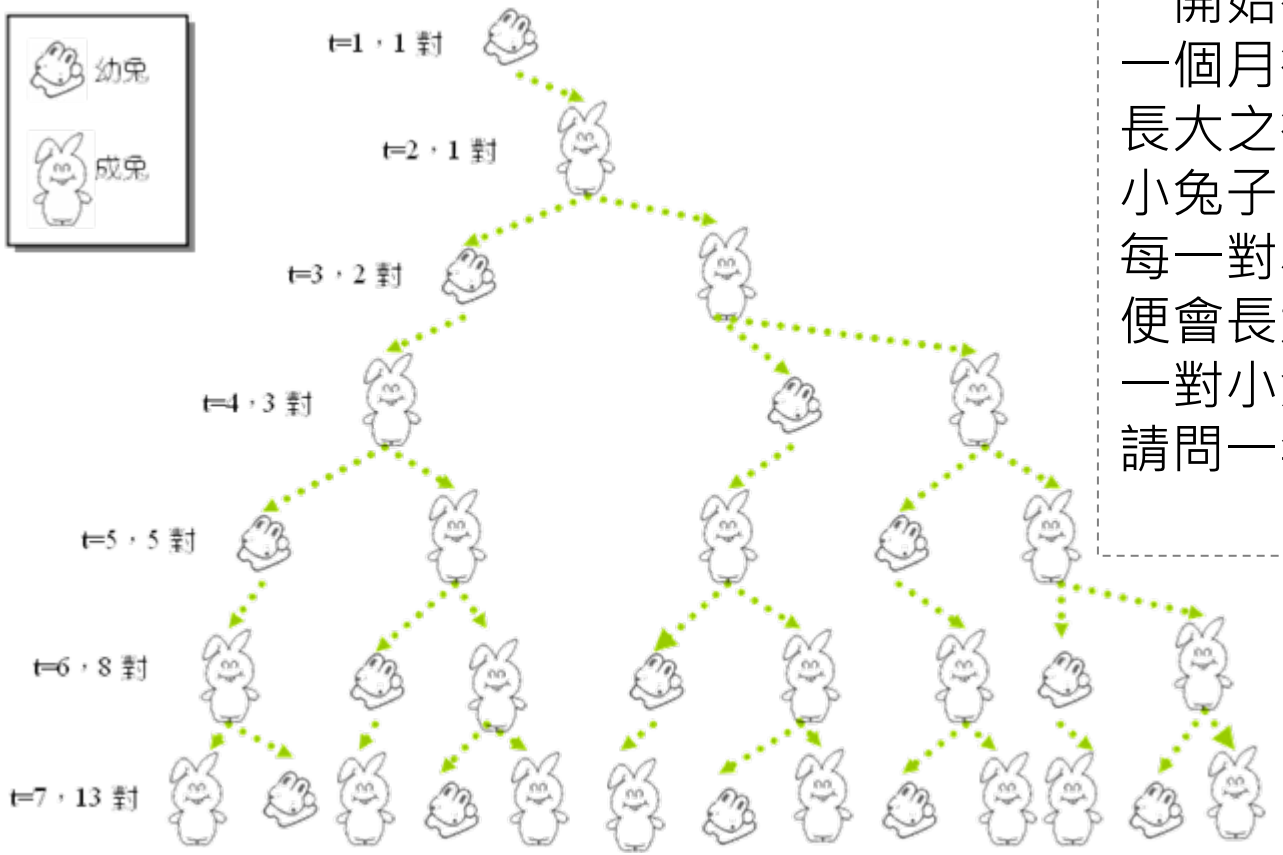
```
/*  OUTPUT---
power(2,3)=8
----------------------*/
```





Recursion – Fibonacci (1/3)

月份	1	2	3	4	5	6	7	8	9	10
兔子對數	1	1	2	3	5	8	13	21	34	55



一開始有一對小兔子
一個月後，兔子可長大為成兔。
長大之後，每個月都可再生一對小兔子
每一對小兔子在出生後滿一個月便會長大，再一個月後便可生下一對小兔
請問一年以後共有多少對兔子？



Recursion – Fibonacci (2/3)

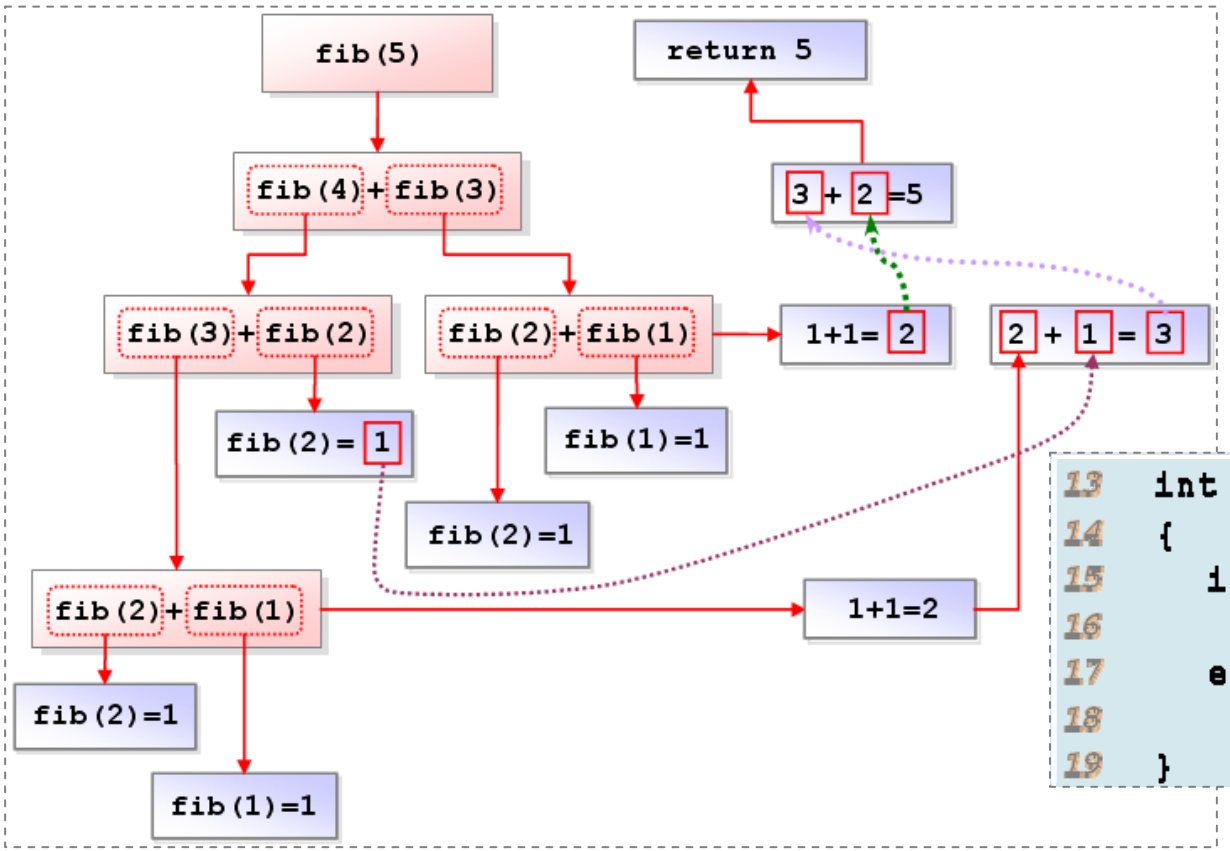
$$fib(n)=\begin{cases}1; & n=1,2\\ fib(n-1)+fib(n-2); & n\geq 3\end{cases}$$

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  int fib(int);    /* fib()函數的原型 */
05  int main(void)
06  {
07     int n;
08     for(n=1;n<=10;n++) /* 計算前 10 個費氏數列 */
09        printf("fib(%d)=%d\n",n,fib(n));
10     system("pause");
11     return 0;
12  }
13  int fib(int n)
14  {
15     if(n==1 || n==2)   /* 如果 n=1 或 n=2，則傳回 1 */
16        return 1;
17     else               /* 否則傳回 fib(n-1)+fib(n-2) */
18        return (fib(n-1)+fib(n-2));
19  }
```

```
/* OUTPUT---

fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
fib(8)=21
fib(9)=34
fib(10)=55
----------------------*/
```



Recursion – Fibonacci (3/3)




```
13  int fib(int n)
14  {
15     if(n==1 || n==2)
16        return 1;
17     else
18        return (fib(n-1)+fib(n-2));
19  }
```



Programming Error

- Variable name can not be the same with argument names

```
void foo(int tmp) {

        /* tmp is the argument name, so it can not be used as
           a variable name */
        int tmp = 1;

        return;
}

int main() {
        foo(1);                 // call function
        return 1;
}
```



Programming Error

- Must return a result in every condition

```
int foo(int tmp) {
        /* no returned value when tmp <= 0 */
        if (tmp > 0)
                return tmp + 1;
}

int main() {
        foo(-1);    // call function
        return 1;
}
```

Programming Error (Cont.)

- The returned value can only be assigned to another variable with the same type

```
double foo(int a, int b) {
        return (double) (a+b)/2
}

int main() {
        /* the returned data type of foo() is double,
           but the type of avg is int */
        int avg;
        avg = foo(2, 3);        // call function

        return 1;
}
```

Can force it to transform the data type by
avg = (int) foo(2, 3)

Programming Error (Cont.)

- A variable declared in the function can not be used outside the function

```
double foo(int a, int b) {
        int sum = a + b;
        return (double) sum/2
}

int main() {
        /* assign the results of foo() to avg */
        double avg = foo(2, 3);          // call function

        /* sum is declared in foo(), so can not be used here */
        printf("sum is %d, avg is %f\n", sum, avg);

        return 1;
}
```



Available Usage

- Function call A() can be the argument of another function call B() if the return type of A() is not *void*

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

int main() {
        int i, j;
        scanf("%d %d", &i, &j);

        /* print the output of max(i,j)
           use %d because max() returns an integer value */
        printf("max is %d\n", max(i, j));
        return 1;
}
```



Available Usage (Cont.)

- The output of function call can be used as a logical control

```
/* return 1 if a > b; otherwise, 0 */
int test_larger (int a, int b) {
        if (a > b)
                return 1;
        else
                return 0;
}

int main() {
   int i, j;
     for (i = 1; i <= 5; i++) {
       for (j = 1; j <= 5; j++) {
         if (test_larger(i, j))
           printf("(%d,%d)\n", i, j); // print if i > j
       }
     }
   return 1;
}
```

output

(2,1)
(3,1)
(3,2)
(4,1)
(4,2)
(4,3)
(5,1)
(5,2)
(5,3)
(5,4)

Available Usage (Cont.)

- Call another function in a function

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

int square_sum(int num) {
   int sum = 0;
   for(int i = 1; i <= num; i++)
            sum += square(i);
   return sum;
}

int main() {
            printf("square sum = %d\n", square_sum(10));
            return 1;
}
```

Function definition of square () must be placed before that of square_sum()

Otherwise, declare function prototypes in the beginning of the program



Math Library Functions

- Math library functions
  - perform common mathematical calculations
  - #include <math.h>
- Reference
  - http://www.cplusplus.com/reference/clibrary/cmath/

Function	Function definition	Description	Example
cos()	double cos(double x)	cosine of an angle of *x*	sin(0.0)=0.0
sin()	double sin(double x)	sine of an angle of *x*	cos(0.0)=1.0
tan()	double tan(double x)	tangent of an angle of *x*	tan(0.0)=0.0
acos()	double acos(double x)	arc cosine of *x*	
asin()	double asin(double x)	arc sine of *x*	
atan()	double atan(double x)	arc tangent of *x*	



Math Library Functions (Cont.)

Function	Function definition	Description	Example
sqrt()	double sqrt(double x)	square root of *x*	sqrt(9.0)=3.0
exp()	double exp(double x)	exponential function e^x	exp(1.0)=2.718282
log()	double log(double x)	natural logarithm of *x* *(base e)*	log(2.718282)=1.0
log10()	double log10(double x)	logarithm of *x*	log10(100.0)=2.0
fabs()	double fabs(double x)	Absolute value of *x*	abs(-2.3)=2.3
ceil()	double ceil(double x)	round *x* to the smallest integer not less than x	ceil(2.3)=3.0 ceil(-9.8)=-9.0
floor()	double floor(double x)	round x to the largest integer not greater than x	floor(2.7)=2.0 floor(-9.8)=-10
pow()	double pow(double x, double y)	x raised by power y (x^y)	pow(2.0,3)=8.0 pow(4.0,0.5)=2.0



Random Number Generation

- **rand()** function
  - Defined in <stdlib.h>
  - Return a pseudo random number between 0 and RAND_MAX (at least 32767)
    - RAND_MAX: a preprocess macro that indicates the upper bound of rand()
    - Pseudo random:
      - Preset sequence of random numbers
      - Same sequence for every function call



Random Number Generation

```
for (int i = 1; i <= 10; i++)
  printf("random number %d (RAND_MAX %d)\n", rand(), RAND_MAX);
```

```
random number 41 (RAND_MAX 32767)
random number 18467 (RAND_MAX 32767)
random number 6334 (RAND_MAX 32767)
random number 26500 (RAND_MAX 32767)
random number 19169 (RAND_MAX 32767)
random number 15724 (RAND_MAX 32767)
random number 11478 (RAND_MAX 32767)
random number 29358 (RAND_MAX 32767)
random number 26962 (RAND_MAX 32767)
random number 24464 (RAND_MAX 32767)
```

Same result for every execution



Random Number Generation (Cont.)

- Scaling
  - To get a random number between 1 and n
    - 1 + (rand() % n)
    - rand() % n: returns a number between 0 and n-1

- Example
  - Roll a six-sided die
    - int face= 1 + rand() % 6



Random Number Generation (Cont.)

- Roll a six-sided die 6000 times

```
int cnt1 = 0, cnt2 = 0, cnt3 = 0, cnt4 = 0, cnt5 = 0, cnt6 = 0;
int face, roll;
for (roll = 1; roll <= 6000; roll++) {
        face = 1 + ( rand() % 6 );
        switch (face) {
                case 1:
                        cnt1++;
                        break;
                case 2:
                        cnt2++;
                        break;
                case 3:
                        cnt3++;
                        break;
                case 4:
                        cnt4++;
                        break;
                case 5:
                        cnt5++;
                        break;
                case 6:
                        cnt6++;
                        break;
        }
}
```

Face	cnt
1	1003
2	1017
3	983
4	994
5	1004
6	999



Set Random Seed

- **srand() function**
  - <stdlib.h>
  - Takes an integer seed and jumps to that location in its "random" sequence
    - Example: srand(2);



Set Random Seed (Cont.)

```
srand(1);
for (int i = 1; i <= 10; i++)
    printf("random number %d\n", rand());
```

```
random number 41
random number 18467
random number 6334
random number 26500
random number 19169
random number 15724
random number 11478
random number 29358
random number 26962
random number 24464
```

```
srand(2);
for (int i = 1; i <= 10; i++)
    printf("random number %d\n", rand());
```

```
random number 45
random number 29216
random number 24198
random number 17795
random number 29484
random number 19650
random number 14590
random number 26431
random number 10705
random number 18316
```



Generate Real Random Numbers

- srand(time(NULL));
  - time(NULL)
    - Return the number of seconds that have passed since Jan. 1, 1970
    - #include <time.h>
  - Randomize the seed
    - Because we don’t know when the program is executed



Generate Real Random Numbers

```
srand( time(NULL) );

for (int i = 1; i <= 10; i++)
    printf("random number %d\n", rand());
```

First run	Second run	Third run
random number 25133	random number 25208	random number 25270
random number 13407	random number 31244	random number 6088
random number 3607	random number 21267	random number 238
random number 1718	random number 30886	random number 29337
random number 26487	random number 1578	random number 947
random number 4490	random number 29269	random number 5573
random number 27064	random number 347	random number 26715
random number 27560	random number 25768	random number 2918
random number 23903	random number 10454	random number 29262
random number 7385	random number 29820	random number 11310

...

The output of rand() depends on time the program is executed

Headers

- Standard library header files
  - Contain function prototypes for library functions
  - <stdlib.h> , <math.h> , etc
  - Load with #include <filename>

    #include <math.h>
- Custom header files
  - Create file with functions
  - Save as filename.h
  - Load in other files with #include "filename.h"
  - Reuse functions



Standard Library Header Files

- <assert.h>
- <ctype.h>
  - Convert lowercase letters to uppercase letters and vice versa
- <errno.h>
- <float.h>
- <limits.h>
- <locale.h>
- <math.h>
  - Math library functions
- <setjmp.h>

Standard Library Header Files (Cont.)

- <signal.h>
- <stdarg.h>
- <stddef.h>
- <stdio.h>
  - Standard input/output library functions
- <stdlib.h>
  - Convert number to text and text to number, memory allocation, random number, and other utility functions
- <string.h>
  - String-processing functions
- <time.h>
  - Time and date



Custom Header Files

.c file

```
#include “myheader.h”

int main() {
   statements;
}

void function1() {
   statements;
   return;
}

int function2(int a, int b) {
   statements;
   return output;
}
```

myheader.h file

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

void function1();
int function2(int, int);
```

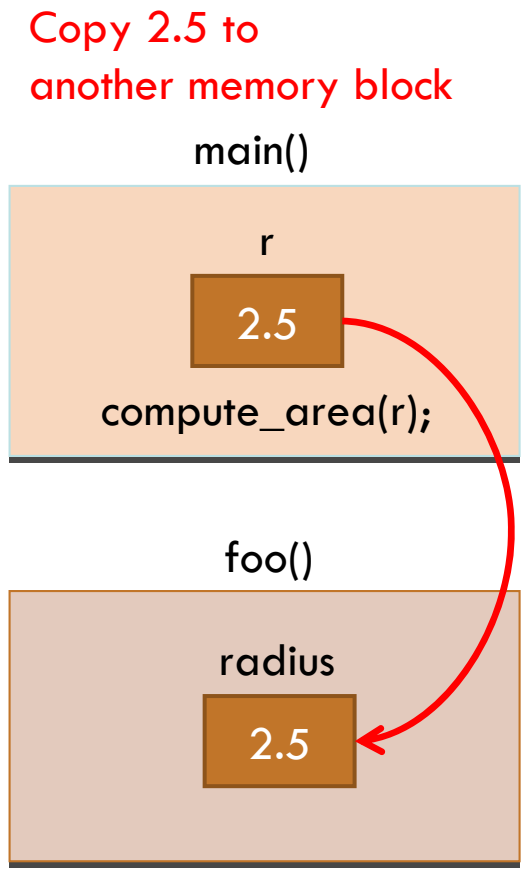
Calling Functions

- **Call by value**
  - Copy of argument passed to function
  - Changes in function do not affect original
  - Use when function does not need to modify argument
    - Avoid accidental changes
- **Call by reference**
  - Passes original argument
  - Changes in function affect original
  - Only used with trusted functions



Example: Call by Value

```
double foo (double);

int main () {
        double area, r = 2.5;
        area = foo(r);
        printf("area: %f\n", area);
        return 0;
}

double foo (double radius) {
        double a;
        a = 3.14159 * radius * radius;
        return a;
}
```



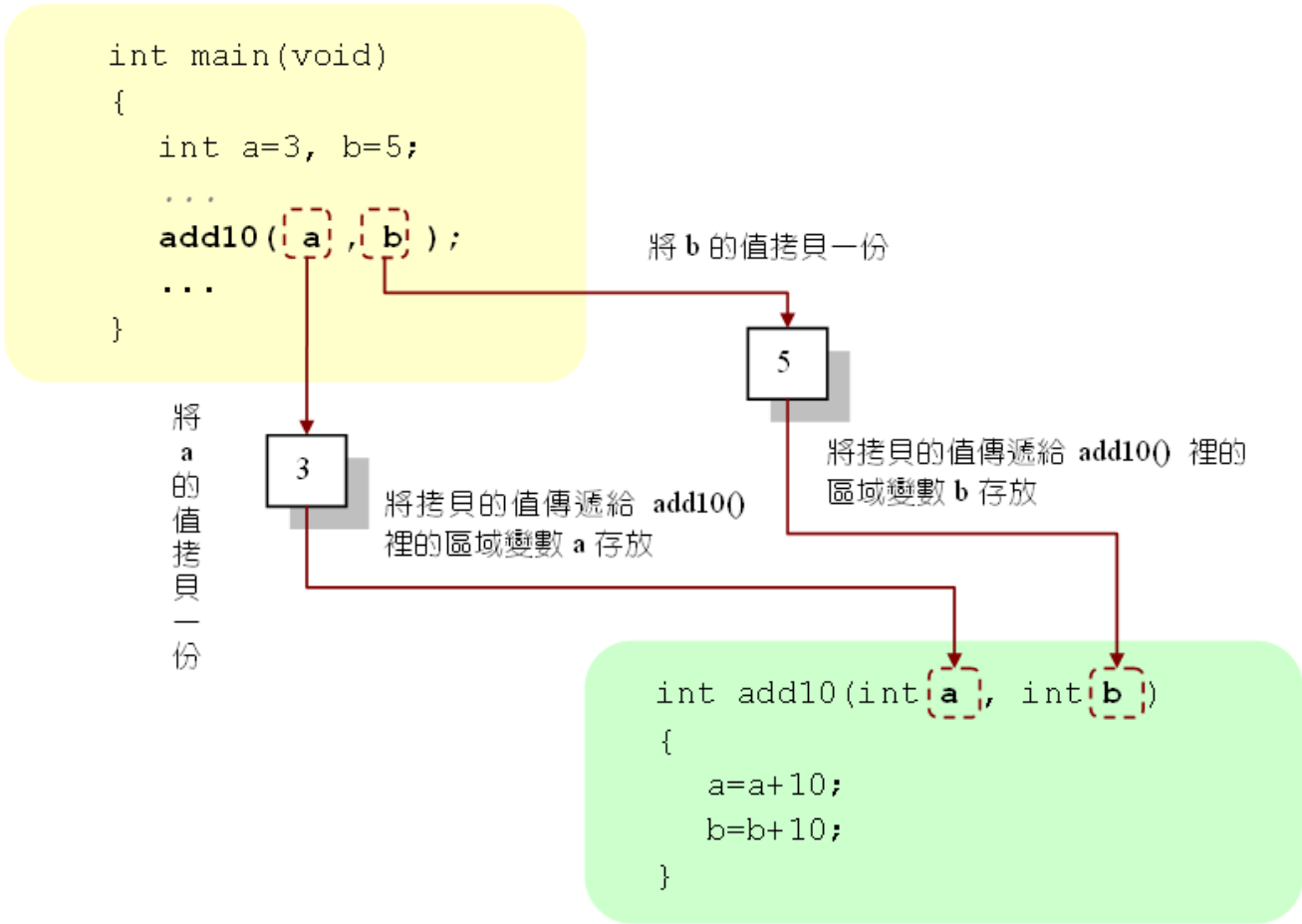


Another Example: Call by Value (1/2)

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   void add10(int,int);                    /* add10()的原型 */
05   int main(void)
06   {
07      int a=3, b=5;                        /* 宣告區域變數 a 與 b */
08      printf("呼叫函數 add10()之前: ");
09      printf("a=%d, b=%d\n",a,b);          /* 印出 a、b 的值 */
10      add10(a,b);
11      printf("呼叫函數 add10()之後: ");
12      printf("a=%d, b=%d\n",a,b);          /* 印出 a、b 的值 */
13      system("pause");
14      return 0;
15   }
16   void add10(int a,int b)
17   {
18      a=a+10;                  /* 將變數 a 的值加 10 之後，設回給 a */
19      b=b+10;                  /* 將變數 b 的值加 10 之後，設回給 b */
20   }
```

```
/* OUTPUT---------

呼叫函數 add10()之前: a=3, b=5
呼叫函數 add10()之後: a=3, b=5
-------------------------*/
```

Another Example: Call by Value (2/2)



Types of Variables

- ***Local variable***
  - Can only be referenced inside a function body
- ***Global variable***
  - Identifier defined outside function, known in all functions
  - Global variables, function definitions, function prototypes
- ***Static variable***
  - The space for static variable is determined at compiling time.
  - The static variable still exists even after the function is returned.



Example: Local Variable (1/3)

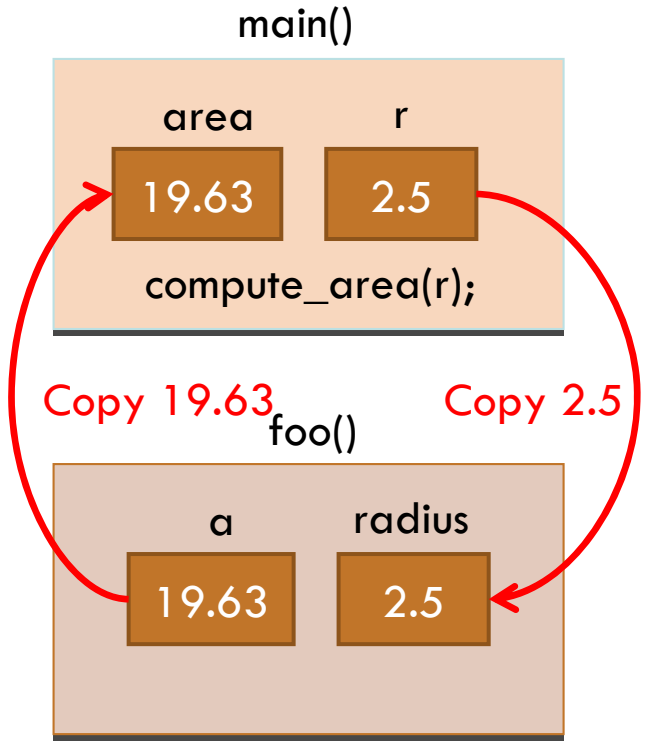
```
double foo (double);

int main () {
        double area, r = 2.5;
        area = foo(r);
        printf("area: %f\n", area);
        return 0;
}

double foo (double radius) {
        double a;
        a = 3.14159 * radius * radius;
        return a;
}
```

Scope of area and r

Scope of a and radius

Example: Local Variable (2/3)

```
double foo (double);

int main () {
        double area, r = 2.5;
        area = foo(r);
        printf("area: %f\n", area);
        return 0;
}

double foo (double radius) {
        double a;
        a = 3.14159 * radius * radius;
        return a;
}
```

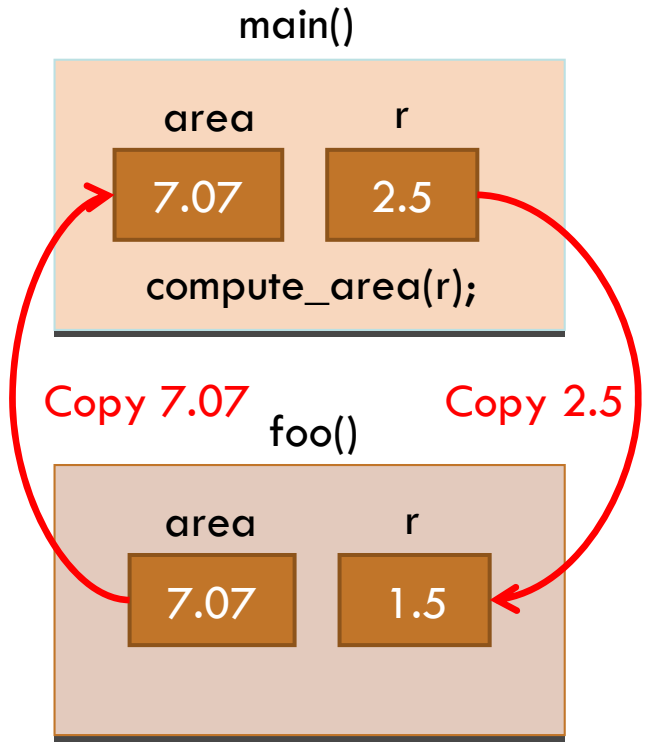




Example: Local Variable (3/3)

```
double foo (double);

int main () {
   double area, r = 2.5;
   area = foo(r);
   printf("area: %f r: %f\n", area, r);
   return 0;
}

double foo (double r) {
   double area;
   r = 1.5;
   area = 3.14159 * r * r;
   return area;
}
```

area: 7.07 r: 2.5



r in main() and foo() are two different variables occupying two memory blocks with the same name

Modifying the value of r in foo() does not affect the value of r in main()



Another Example: Local Variable (1/2)

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  int fac(int);                    /* fac()函數的原型 */
05  int main(void)
06  {
07     int ans;
08     ans=fac(5);
09     printf("fac(5)=%d\n",ans);
10     system("pause");
11     return 0;
12  }
13  int fac(int n)
14  {
15     int i, total=1;
16     for(i=1; i<=n; i++)
17        total=total*i;
18     return total;
19  }
```

```
/* OUTPUT---
fac(5)=120
----------------------*/
```

區域變數 ans 的活動範圍

區域變數 i 與 total 的活動範圍

區域變數 n 的活動範圍

Another Example: Local Variable (2/2)

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  void func(void);
05  int main(void)
06  {
07     int a=100;     /* 宣告 main()函數裡的區域變數 a */
08
09     printf("呼叫 func()之前,a=%d\n",a);    /* 印出 main()中 a 的值 */
10     func();        /* 呼叫自訂的函數 */
11     printf("呼叫 func()之後,a=%d\n",a);    /* 印出 a 的值 */
12
13     system("pause");
14     return 0;
15  }
16  void func(void)       /* 函數 func() */
17  {
18     int a=300;         /* 宣告 func()函數裡的區域變數 a */
19     printf("於 func()函數裡,a=%d\n",a);    /* 印出 func 函數中 a 的值 */
20  }
```

```
/* OUTPUT---

呼叫 func()之前,a=100
於 func()函數裡,a=300
呼叫 func()之後,a=100
----------------------*/
```



Example: Global Variable (1/4)

```
double foo ();
double r;
int main () {
        r = 2.5;
        double area;
        area = foo();
        printf("area: %f\n", area);
        return 0;
}

double foo () {
        double area;
        r = 1.5;
        area = 3.14159 * r * r;
        return area;
}
```

Scope of area in main()

Scope of r

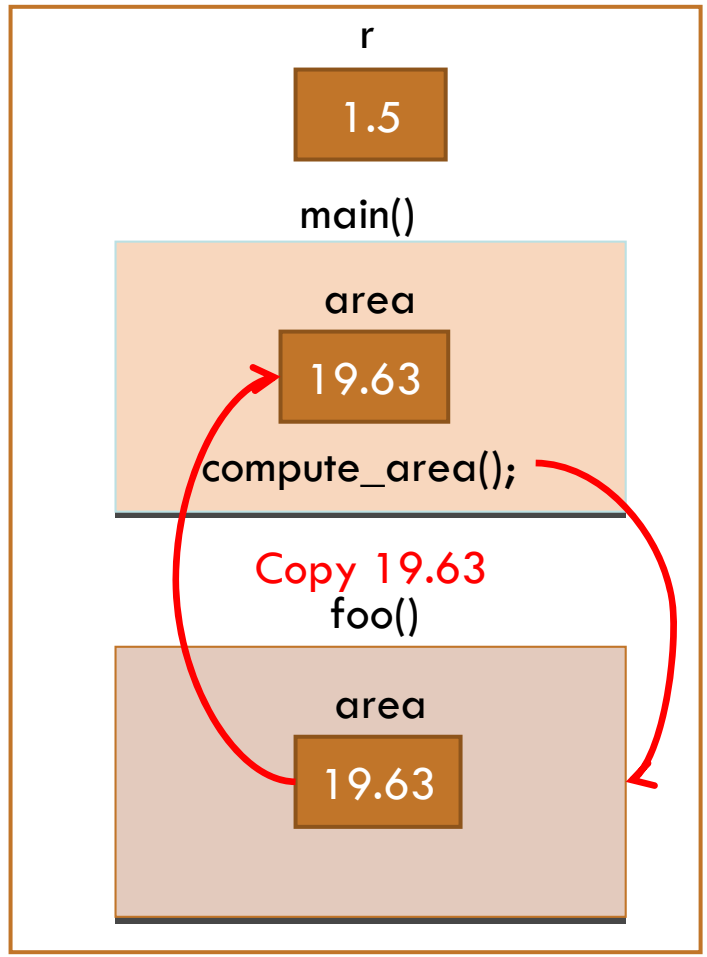
Scope of area in foo()



Example: Global Variable (2/4)

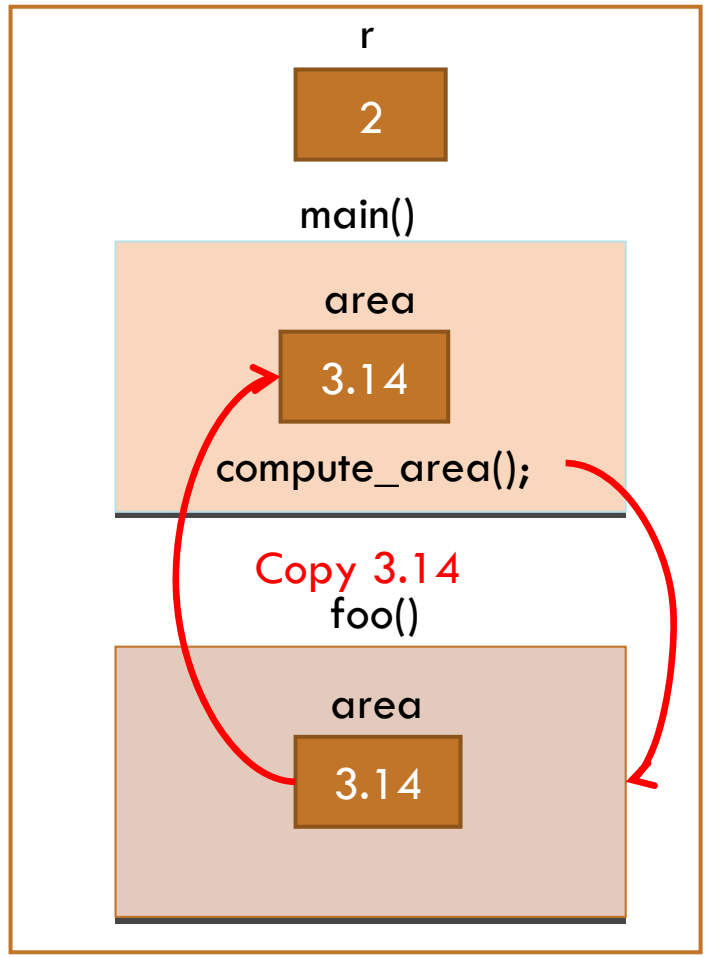
```
double foo ();
double r;
int main () {
        r = 2.5;
        double area;
        area = foo();
        printf("area: %f\n", area);
        printf("r: %f\n", r);
        return 0;
}

double foo () {
        double area;
        r = 1.5;
        area = 3.14159 * r * r;
        return area;
}
```

area: 19.63
r: 1.5



Example: Global Variable (3/4)

```
double foo ();
double r;
int main () {
   r = 1;
   double area;
   area = foo();
   printf("area: %f, r: %f\n", area, r);
   area = foo();
   printf("area: %f, r: %f\n", area, r);
   return 0;
}

double foo () {
   double area;
   area = 3.14159 * r * r;
   r++;
   return area;
}
```

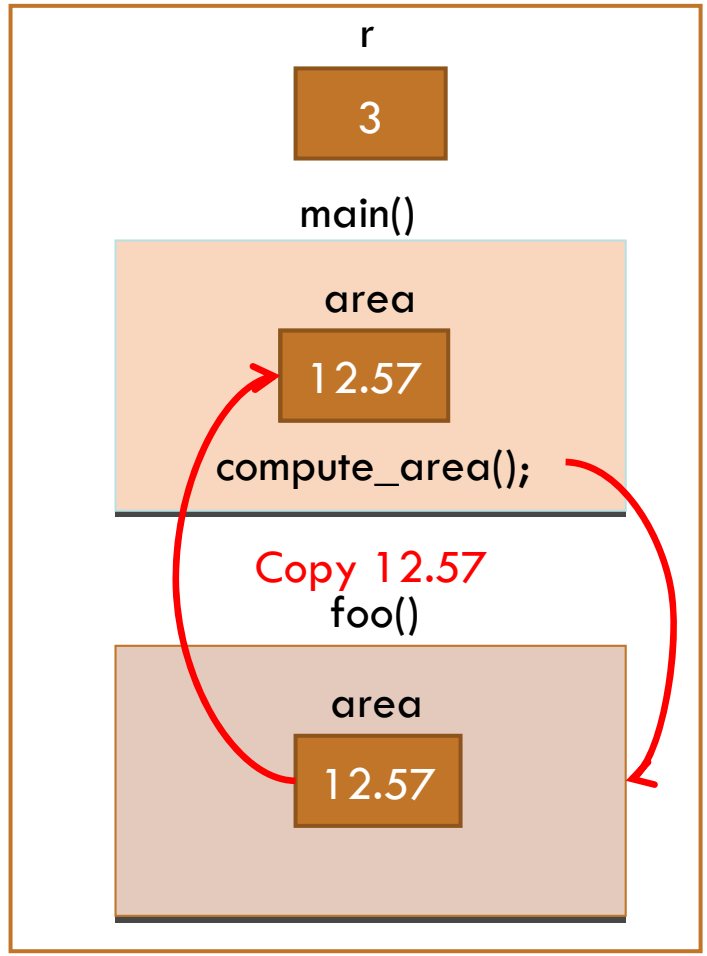




Example: Global Variable (1/4)

```
double foo ();
double r;
int main () {
   r = 1;
   double area;
   area = foo();
   printf("area: %f, r: %f\n", area, r);
   area = foo();
   printf("area: %f, r: %f\n", area, r);
   return 0;
}

double foo () {
   double area;
   area = 3.14159 * r * r;
   r++;
   return area;
}
```

```
area: 3.14, r: 2
area: 12.57, r: 3
```





Another Example: Global Variable (1/3)

```
01  /* prog8_15, 全域變數的範例(一) */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void func(void);        /* 函數 func()的原型 */
05  int a;                  /* 宣告全域變數 a */
06  int main(void)
07  {
08     a=100;               /* 設定全域變數 a 的值為 100 */
09     printf("呼叫 func()之前,a=%d\n",a);
10     func();              /* 呼叫自訂的函數 */
11     printf("呼叫 func()之後,a=%d\n",a);
12
13     system("pause");
14     return 0;
15  }
16  void func(void)         /* 自訂的函數 func() */
17  {
18     a=300;               /* 設定全域變數 a 的值為 300 */
19     printf("於 func()函數裡,a=%d\n",a);
20  }
```

```
/* OUTPUT---
呼叫 func()之前,a=100
於 func()函數裡,a=300
呼叫 func()之後,a=300
--------------------*/
```

全域變數 a 的活動範圍



Another Example: Global Variable (2/3)

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  void func(void);
05  int a=50;                          /* 定義全域變數 a */
06
07  int main(void)
08  {
09     int a=100;                      /* 定義區域變數 a */
10     printf("呼叫 func()之前,a=%d\n",a);
11     func();          /* 呼叫自訂的函數 */
12     printf("呼叫 func()之後,a=%d\n",a);
13     system("pause");
14     return 0;
15  }
16  void func(void)
17  {
18     a=a+300;                        /* 這是全域變數 a */
19     printf("於 func()函數裡,a=%d\n",a);
20  }
```

```
/* OUTPUT---
呼叫 func()之前,a=100
於 func()函數裡,a=350
呼叫 func()之後,a=100
--------------------*/
```

全域變數 a 的活動範圍

區域變數 a 的活動範圍

全域變數 a 的活動範圍



Another Example: Global Variable (3/3)

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  double pi=3.14;       /* 宣告全域變數 pi */
05  void peri(double),area(double);
06  int main(void)
07  {
08     double r=1.0;
09     printf("pi=%.2f\n",pi);     /* 於 main() 裡使用全域變數 pi*/
10     printf("radius=%.2f\n",r);
11     peri(r);       /* 呼叫自訂的函數 */
12     area(r);
13     system("pause");
14     return 0;
15  }
16  void peri(double r)  /* 自訂的函數 peri()，印出圓周 */
17  {
18     printf("圓周長=%.2f\n",2*pi*r);  /* 於 peri() 裡使用全域變數 pi */
19  }
20  void area(double r)  /* 自訂的函數 area()，印出圓面積 */
21  {
22     printf("圓面積=%.2f\n",pi*r*r);  /* 於 area() 裡使用全域變數 pi */
23  }
```

```
/* OUTPUT---

pi=3.14
radius=1.00
圓周長=6.28
圓面積=3.14
----------------------*/
```



Example: Static Variable

```
/* OUTPUT---

In func(),a=100
In func(),a=300
In func(),a=500
----------------------*/
```

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   void func(void);      /* 宣告 func()函數的原型 */
05   int main(void)
06   {
07      func();            /* 呼叫函數 func() */
08      func();            /* 呼叫函數 func() */
09      func();            /* 呼叫函數 func() */
10
11      system("pause");
12      return 0;
13   }
14   void func(void)
15   {
16      static int a=100;                 /* 宣告靜態變數 a */
17      printf("In func(),a=%d\n",a);     /* 印出 func()函數中 a 的值 */
18      a+=200;
19   }
```



Preprocessor - #define

- Preprocessor is used to replace statements with macros before the compilation.
- #define can be used to define MACRO

Format of #define

```
#define LABLE Replacement-Marker
```

→ No semicolon is needed

- Example:
  - `#define MAX 32767`
  - `#define IOU "I love you!"`



#define Example (1/3)

```
/* OUTPUT---
*
**
***
****
*****
----------------------*/
```

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define BEGIN {       /* 定義識別名稱 BEGIN 為左大括號{ */
05  #define END }         /* 定義識別名稱 END 為右大括號} */
06  int main(void)
07  BEGIN         /* 此行的 BEGIN 相當於左大括號 { */
08     int i,j;
09     for(i=1;i<=5;i++)
10     BEGIN      /* 此行的 BEGIN 相當於左大括號 { */
11        for(j=1;j<=i;j++)
12           printf("*");
13        printf("\n");
14     END        /* 此行的 END 相當於右大括號 } */
15     system("pause");
16     return 0;
17  END           /* 此行的 END 相當於右大括號 } */
```



#define Example (2/3)

- Use #define to define strings

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   #define WORD "Think of all the things \
05   we've shared and seen.\n"
06   int main(void)
07   {
08      printf(WORD);
09
10      system("pause");
11      return 0;
12   }

/* prog8_21 OUTPUT--------------------------

Think of all the things we've shared and seen.
-----------------------------------------*/
```

```
/* OUTPUT-------------------------

Think of all the things we've shared and seen.
-----------------------------------------*/
```



#define Example (3/3)

- Define π

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   #define PI 3.14              /* 定義 PI 為 3.14 */
05   double area(double);
06   int main(void)
07   {
08      printf("PI=%4.2f, area()=%6.2f\n",PI,area(2.0));
09
10      system("pause");
11      return 0;
12   }
13
14   double area(double r)
15   {
16      return PI*r*r;
17   }
```

```
/* OUTPUT---
PI=3.14, area()= 12.56
----------------------*/
```

Using the Keyword *const*

- When a variable is defined as const, it is a constant and its content can not be modified.

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  const double pi=3.14;          /* 宣告 pi 為 double 型態的常數 */
05  double area(double);
06  int main(void)
07  {
08     /* 若在此處設定 pi=3.1416，則編譯時會發生錯誤 */
09     printf("pi=%4.2f, area()=%6.2f\n",pi,area(2.0));
10
11     system("pause");
12     return 0;
13  }
14
15  double area(double r)
16  {
17     return pi*r*r;
18  }
```

```
/* OUTPUT-----
pi=3.14, area()= 12.56
-----------------------*/
```

Using #define for Function Replacement

- Without parameters

```
01
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SQUARE n*n          /* 定義巨集 SQUARE 為 n*n */
05  int main(void)
06  {
07     int n;
08     printf("Input an integer:");
09     scanf("%d",&n);
10     printf("%d*%d=%d\n",n,n,SQUARE);  /* 計算並印出 n 的平方 */
11
12     system("pause");
13     return 0;
14  }
```

```
/* OUTPUT---

Input an integer:4
4*4=16
----------------------*/
```

Using #define for Function Replacement (Cont.)

- With parameters

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   #define SQUARE(X) X*X     /* 定義巨集 SQUARE(X) 為 X*X */
05   int main(void)
06   {
07      int n;
08      printf("Input an integer:");
09      scanf("%d",&n);
10      printf("%d*%d=%d\n",n,n,SQUARE(n));  /* 計算並印出 n 的平方 */
11
12      system("pause");
13      return 0;
14   }
```

```
/* OUTPUT---

Input an integer:12
12*12=144
----------------------*/
```



Common Errors in Macro

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   #define SQUARE(X) X*X                    /* 定義巨集 SQUARE(X)為 X*X */
05   int main(void)
06   {
07      int n;
08      printf("Input an integer:");
09      scanf("%d",&n);
10      printf("%d*%d=%d\n",n+1,n+1,SQUARE(n+1));   /* 印出 n+1 的平方 */
11
12      system("pause");
13      return 0;
14   }
```

SQUARE(n+1) →n+1*n+1=12+1*12+1=25

```
/* OUTPUT---

Input an integer:12
13*13=25
----------------------*/
```

Common Errors in Macro (Cont.)

- Correct the error:

SQUARE(n+1) →(n+1)*(n+1)=(12+1)*(12+1)=169

```
01
02   #include <stdio.h>
03   #include <stdlib.h>
04   #define SQUARE(X) (X)*(X)      /* 定義巨集 SQUARE(X)為(X)*(X) */
05   int main(void)
06   {
07      int n;
08      printf("Input an integer:");
09      scanf("%d",&n);
10      printf("%d*%d=%d\n",n+1,n+1,SQUARE(n+1));  /* 印出 n+1 的平方*/
11
12      system("pause");
13      return 0;
14   }
```

```
/* OUTPUT---

Input an integer:12
13*13=169
----------------------*/
```



Using Function? Using Macro?

- Before compiling, compiler use macros to replace the original statements.
  - No function is needed because the original statement is replaced after being compiled.
  - Faster without larger program size.
- Function call is a jump statement
  - When a function call is encountered, the program jumps to the definition of the called function.
  - Slower with smaller program size.



Lab 08-1

- 試撰寫int cub(int x) 函數，可用來傳回x的3次方，並利用此函數來計算cub(2)，即計算2^3。
- 設$f(x)=3x^3+2x-1$，試寫一函數 ，用來傳回的值，並於主程式裡分別計算f(-3)、f(-2)、f(0)與f(2)。
- 撰寫一函數double my_fun (int n)，可用來計算下面的數學式，並於主程式裡計算my_fun(3)、my_fun(4)、my_fun(5)與my_fun(6) 的值:

$$\text{my\_fun}(n)=\sum_{k=1}^{n}\frac{1}{2^k}=\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\cdots+\frac{1}{2^n}$$



Lab 08-2

- 計算Fibonacci的函數int fib(n)，改以非遞迴的方式來撰寫（提示：利用for迴圈）。
- 試以遞迴的方式撰寫函數int sum(int n)，利用遞迴公式sum(n) = n+sum(n-1)， sum(1)=1用來計算的值。當在鍵盤上輸入n時，則在螢幕上輸出1+2…+n的結果。
- 試利用 #define定義一巨集函數CUBIC(X)，可用來計算X的3次方，並利用此巨集計算5^3和4.2^3。