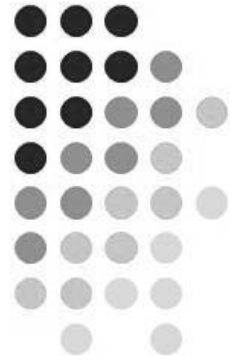


第八章 函數

認識函數與其原型
學習函數的宣告方式與定義
認識區域、全域與靜態變數
學習前置處理器的用法

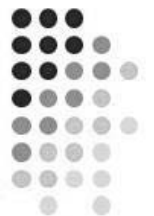


1

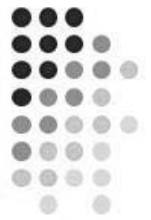
8.1 簡單的函數範例

C 語言的函數

- Top-down design 的目的是要達到逐步單純化
 - 將大問題細分成小問題
 - 將解決這些小問題的方法，撰寫成較小的程式區塊
- C 語言的函數
 - 如賦予程式區塊一個名字
 - 並且指定它的輸出與輸入
 - 則此程式區塊就是一個 C 語言的函數

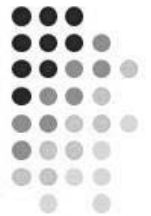


2



函數的功用

- 函數可以使程式的設計與維護更加容易
- 可提高程式的可讀性
- 函數可重複被呼叫，提高程式的再利用率
- 每一個函數有自己的任務，可按任務來規劃函數



簡單的函數範例

- `star()` 函數可印出一行星號

```

01  /* prog8_1, 簡單的函數範例 */
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

```

```

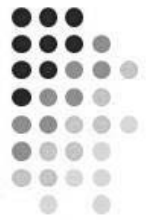
14  void star(void)
15  {
16      printf("*****\n");
17      return;
18  }

```

```

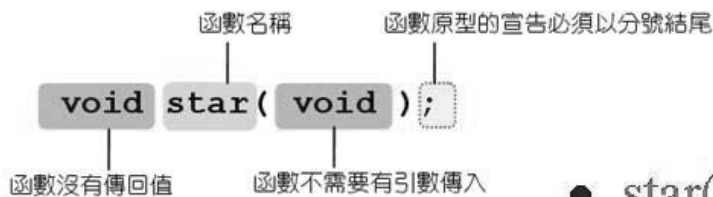
/* prog8_1 OUTPUT---
*****
歡迎使用 C 語言
*****
-----*/

```

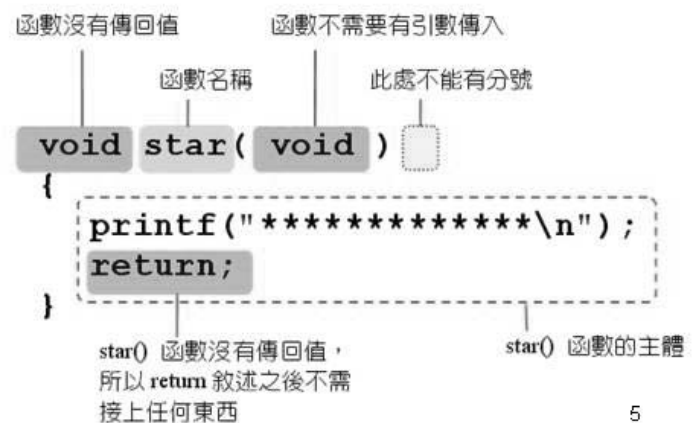


star() 函數原型的宣告與定義

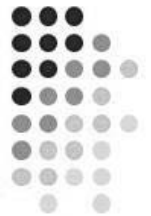
- star() 函數原型宣告的語法：



- star() 函數的定義：

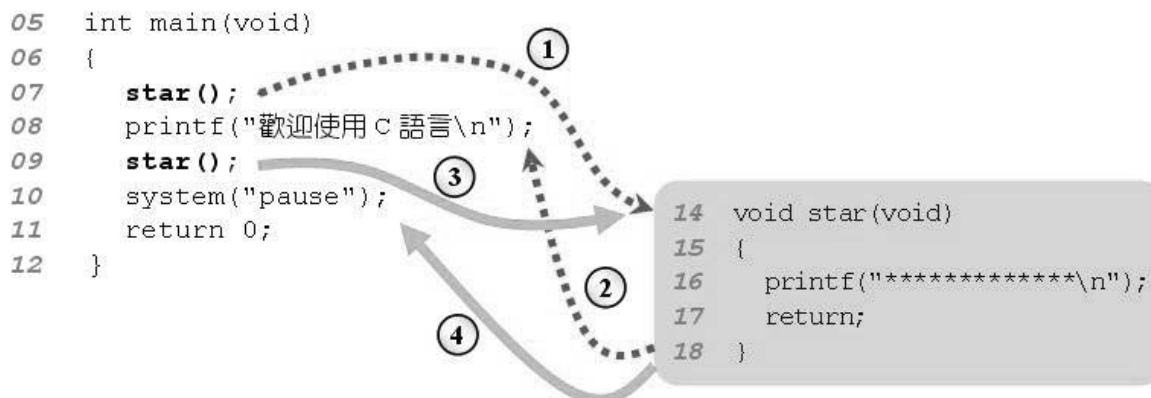


5



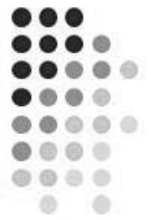
函數呼叫時，程式執行的流程

- 主函數main與star之間，程式執行的流程：



- 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

6



函數的基本架構

• 函數原型宣告

函數原型宣告的格式

傳回值型態 函數名稱 (引數型態1, 引數型態2, ...);

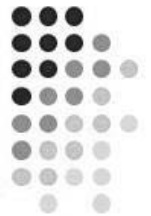
• add() 可接收二整數，傳回值為整數之和，其原型為：

傳回值型態是整數 有兩個引數傳入 add() 函數，型態均為 int，各型態間以逗號分開

```
int add ( int , int );
```

函數名稱為 add

7



函數的基本架構

• 函數的定義

函數的定義

傳回值型態 函數名稱 (型態1 引數1, ..., 型態n 引數n)

```
{
    變數宣告;
    敘述主體;
    return 運算式;    /* 傳回運算式的值 */
}
```

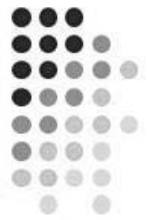
• add() 函數的定義：

傳回值型態是整數 傳入的引數分別由變數 a 與 b 接收

```
int add ( int a, int b )
{
    return a+b;
}
```

a+b 是整數，所以傳回值的型態是整數

8



於程式裡呼叫函數

● add() 函數的使用範例：

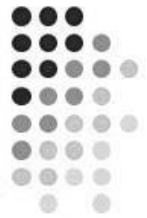
```

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  }

```

/* prog8_2 OUTPUT---
5+3=8
-----*/

9



函數擺放的位置

● 將add()函數放在main()函數的前面：

```

01  /* prog8_3, 將 add() 函數放在 main() 函數的前面 */
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  }

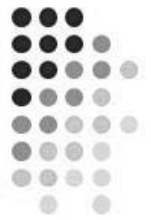
```

/* prog8_3 OUTPUT---
5+3=8
-----*/

將 add() 放在 main() 函數的前面

main() 函數置於 add() 的後面

10



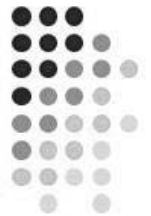
函數練習一display() (1/2)

● 字元列印函數display() :

```

01  /* prog8_4, display() 的練習 */
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; /* prog8_4 OUTPUT---
08      char ch; 請輸入欲列印的字元:&
09      printf("請輸入欲列印的字元:"); 請問要印出幾個字元:12
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  }
```

11



函數練習一display() (2/2)

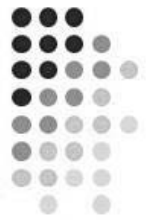
```

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  }
```

/* prog8_4 OUTPUT--

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

12



絕對值函數 abs()

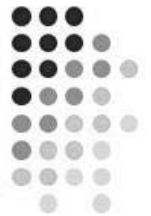
/* prog8_5 OUTPUT---

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

```
01  /* prog8_5, 求絕對值函數 abs() */
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  }
```

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

13

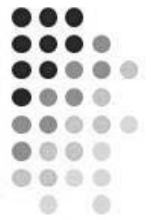


次方函數 power(x,n) — x^n (1/2)

```
01  /* prog8_6, 計算 x 的 n 次方 */
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  }
```

/* prog8_6 OUTPUT-----
請輸入底數與次方: 5.0, 3
5.000000 的 3 次方=125.000000
-----*/

14



次方函數 $\text{power}(x, n) = x^n$ (2/2)

- 下表是在 $\text{power}()$ 中變數 pow 變化的情形：

表 8.3.1 base=5.0，n=3 時， $\text{power}()$ 函數內變數 pow 的變化

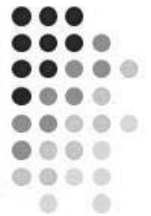
i	pow	pow=pow*base
1	1.0	pow=1.0*5.0=5.0
2	5.0	pow=5.0*5.0=25.0
3	25.0	pow=25.0*5.0=125.0
4		

不符合 for 迴圈的判斷條件 ($i \leq 3$)，
跳出 for 迴圈，傳回 $\text{pow}=125.0$

```

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 }
```

15

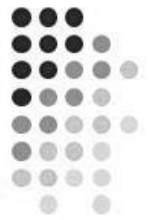


質數測試函數 $\text{is_prime}()$ (1/2)

```

01 /* prog8_7, 質數的找尋 */
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++) /* prog8_7 OUTPUT-----
19         if(num%i==0)          2 3 5 7 11 13 17 19 23 29
20             return 0;          -----*/
21     return 1;
22 }
```

16



質數測試函數 is_prime() (2/2)

- is_prime() 函數內變數變化情形：

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

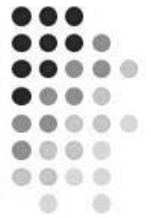
符合 20 行的判斷條件，傳回 0，代表 9 不是質數

```

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 }

```

17



同時使用多個函數 (1/2)

- 下面的程式定義了 fac(n) 與 sum(n) 函數：

```

01  /* prog8_8, 同時呼叫多個函數 */
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  }

```

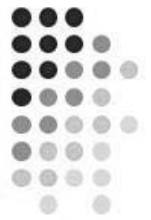
/* prog8_8 OUTPUT---

1*2*...*5=120

1+2+...*5=15

-----*/

18



同時使用多個函數 (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 }

```

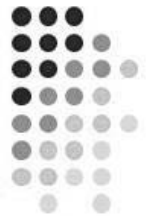
/* prog8_8 OUTPUT---

1*2*...*5=120

1+2+...+5=15

-----*/

19



函數之間的相互呼叫 (1/2)

● 用萊布尼茲方法估算 π

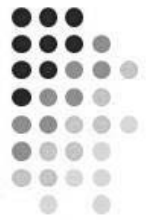
$$\pi = 4 \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{2k-1} = 4 \left(\frac{(-1)^{1-1}}{2(1)-1} + \frac{(-1)^{2-1}}{2(2)-1} + \frac{(-1)^{3-1}}{2(3)-1} + \frac{(-1)^{4-1}}{2(4)-1} + \dots \right) = 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \dots \right)$$

```

01 /* prog8_9, 用萊布尼茲方法估算π */
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

```

20



函數之間的相互呼叫 (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 }

```

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

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 }

```

/* prog8_9 OUTPUT-----

Leibniz(1)=4.0000000

Leibniz(2)=2.6666667

Leibniz(3)=3.4666667

....

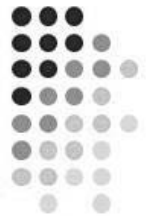
Leibniz(9999)=3.1416927

Leibniz(10000)=3.1414927

-----*/

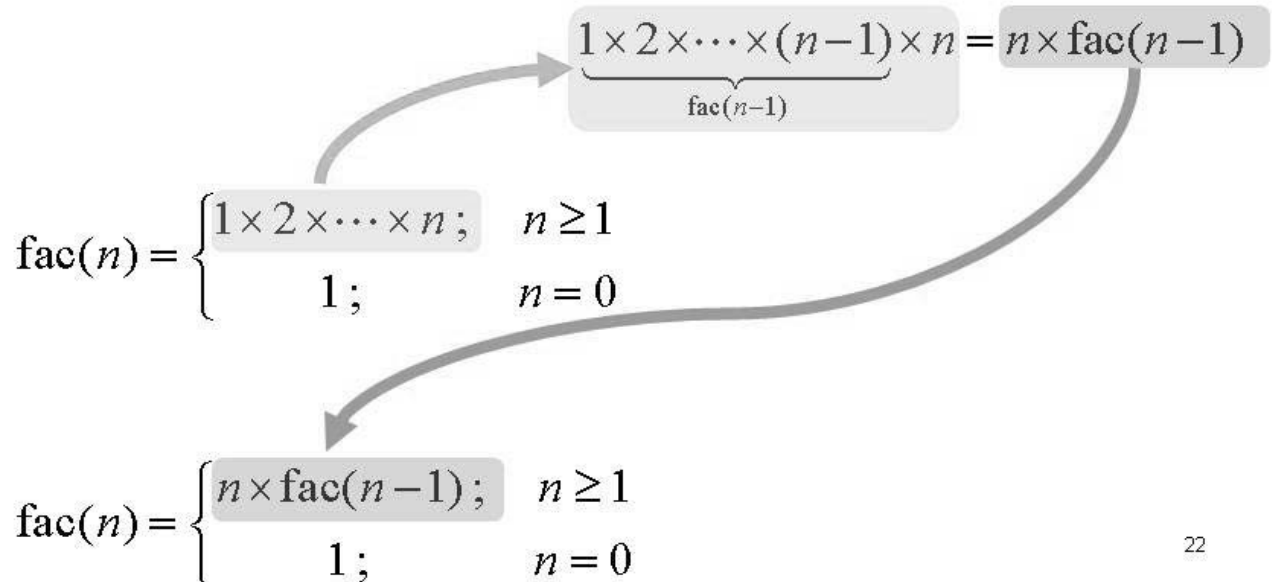
21

8.4 遞迴函數

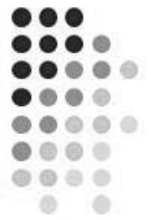


遞迴-階乘函數 (1/2)

- 遞迴 - 函數自己呼叫自己
- 階乘函數 (factorial function, $n!$) 的遞迴



22



遞迴-階乘函數 (2/2)

• 遞迴函數：

```

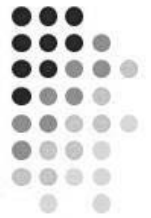
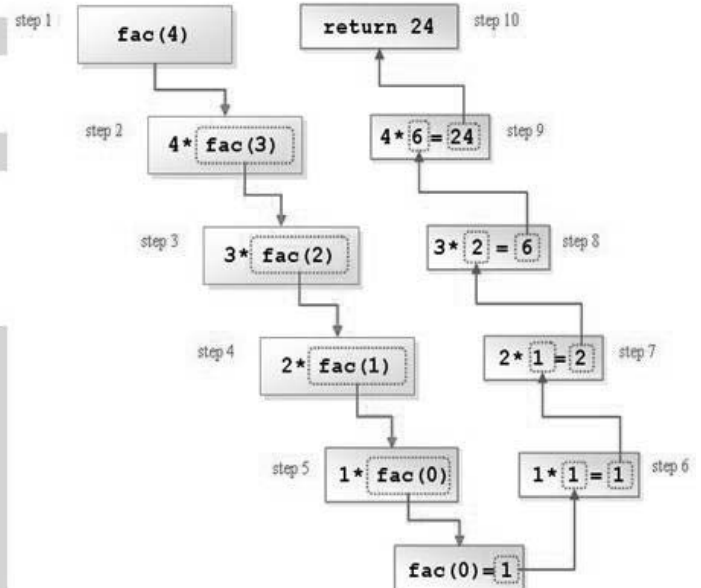
01  /* prog8_10, 遞迴函數，計算階乘 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int fac(int);
05  int main(void)
06  {
07      printf("fac(4)=%d\n", fac(4));
08
09      system("pause");
10      return 0;
11  }
12  int fac(int n)
13  {
14      if(n>0)
15          return (n*fac(n-1));
16      else
17          return 1;
18  }

```

/* prog8_10 OUTPUT--

fac(4)=24

-----*/



遞迴-次方函數 (1/2)

• 次方函數 b^n ：

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

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

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

遞迴-次方函數 (2/2)

● 計算 b^n

```

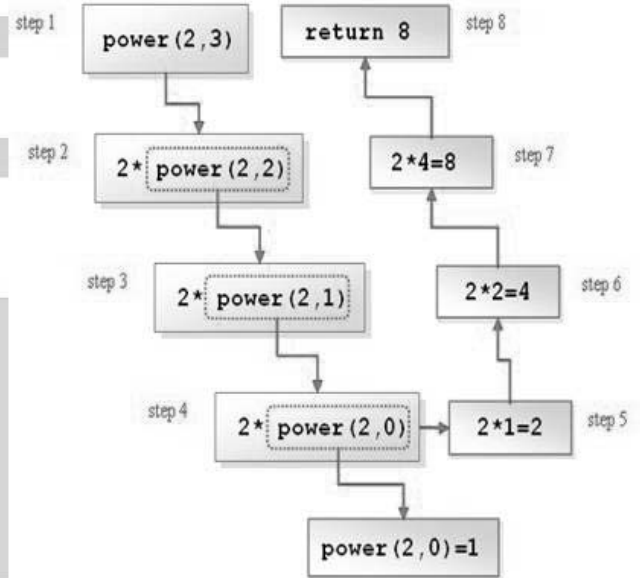
01  /* prog8_11, 遞迴計算 b 的 n 次方 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int power(int,int);
05  int main(void)
06  {
07      printf("power(2,3)=%d\n",power(2,3));
08      system("pause");
09      return 0;
10  }
11  int power(int b,int n)
12  {
13      if(n==0)
14          return 1;
15      else
16          return (b*power(b,n-1));
17  }

```

```

/* prog8_11 OUTPUT---
power(2,3)=8
-----*/

```

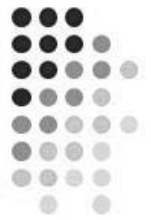


Fibonacci 數列 (1/3)

- 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？



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



Fibonacci 數列 (2/3)

```

01  /* prog8_12, 費氏數列 */
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  }

```

/* prog8_12 OUTPUT---

```

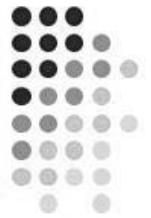
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
fib(8)=21
fib(9)=34
fib(10)=55
-----*/

```

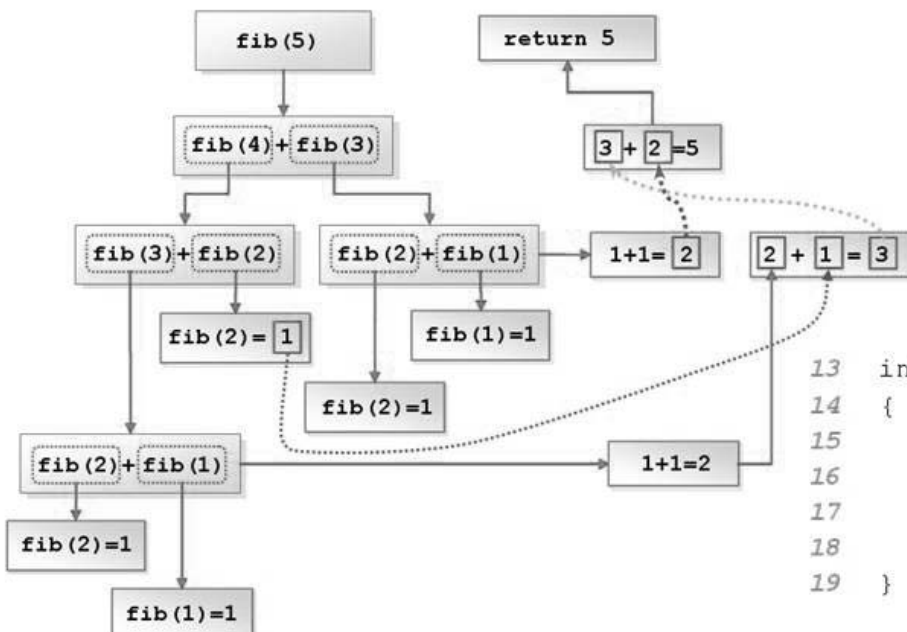
27

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

Fibonacci 數列 (3/3)



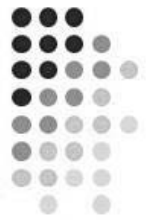
- 遞迴函數 fib(5) 的執行流程：



```

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  }

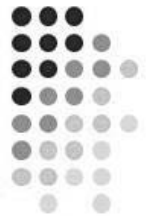
```



變數的分類

- 變數種類可概分為
 - 區域變數 (local variable)
 - 區域變數的生命週期只在函數內
 - 全域變數 (global variable)
 - 所有的函數模組皆可使用全域變數
 - 靜態變數 (static variable)
 - 函數執行完時，函數內的靜態變數之值不會消失，而會保留在函數內

29



區域變數的範例 (1/2)

```

01  /* prog8_13, 區域變數的範例 (一) */
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  }

```

區域變數 **ans** 的活動範圍

區域變數 **i** 與 **total** 的活動範圍

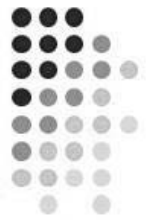
區域變數 **n** 的活動範圍

```

/* prog8_13 OUTPUT---
fac(5)=120
-----*/

```

30



區域變數的範例 (2/2)

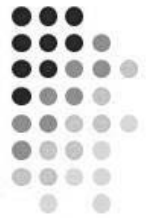
```

01  /* prog8_14, 區域變數的範例 (二) */
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  }
```

```

/* prog8_14 OUTPUT---
呼叫 func() 之前, a=100
於 func() 函數裡, a=300
呼叫 func() 之後, a=100
-----*/
```

31



全域變數 (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  }
```

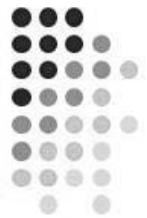
```
/* prog8_15 OUTPUT---
```

```

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

32



全域變數 (2/3)

```

01  /* prog8_16, 全域變數的範例(二) */
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  }

```

/* prog8_16 OUTPUT---

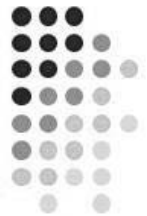
呼叫 func() 之前,a=100
 於 func() 函數裡,a=350
 呼叫 func() 之後,a=100
 -----*/

} 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

33



全域變數 (3/3)

```

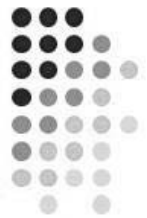
01  /* prog8_17, 全域變數的使用範例(三) */
02  #include <stdio.h>
03  #include <stdlib.h>
04  double pi=3.14;    /* 宣告全域變數 pi */
05  void peri(double), area(double);
06  int main(void)
07  {
08      double r=1.0;
09      printf("pi=%.2f\n", pi);    /* 於 main() 裡使用全域變數 pi */
10      printf("radius=%.2f\n", r);
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  }

```

/* prog8_17 OUTPUT---

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

34



靜態變數

```

01  /* prog8_18, 區域靜態變數使用的範例 */
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  }

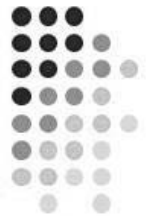
```

```

/* prog8_18 OUTPUT---
In func(), a=100
In func(), a=300
In func(), a=500
-----*/

```

35



引數傳遞的機制 (1/2)

```

01  /* prog8_19, 函數的傳值機制 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void add10(int, int);
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  }

```

```

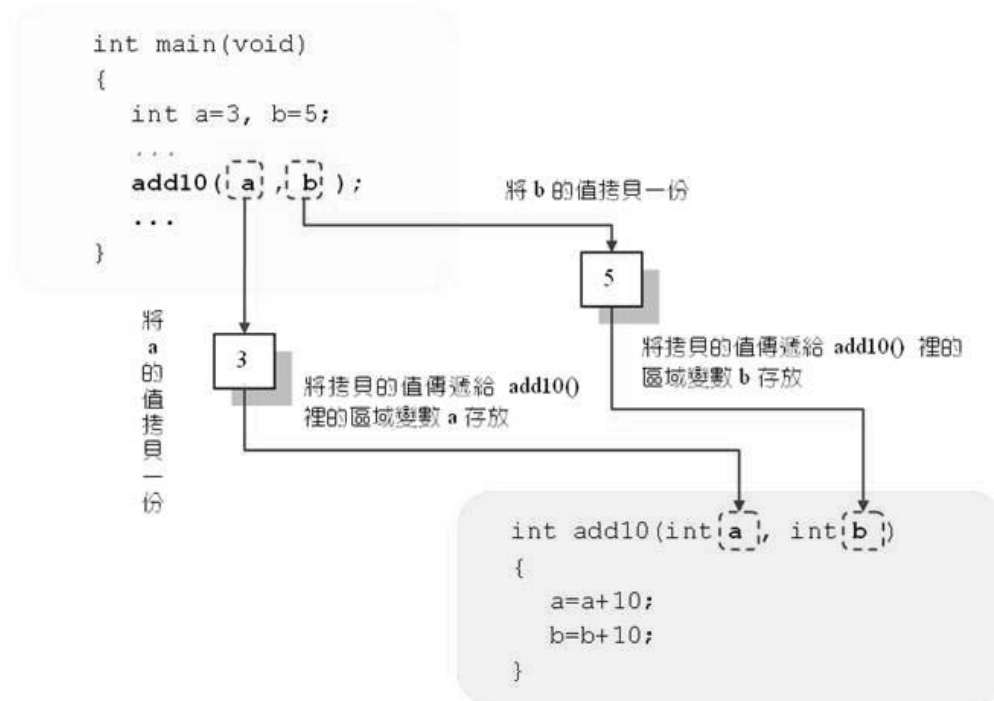
/* prog8_19 OUTPUT-----
呼叫函數 add10() 之前: a=3, b=5
呼叫函數 add10() 之後: a=3, b=5
-----*/

```

36

引數傳遞的機制 (2/2)

- 函數引數傳遞的情形：



37

前置處理器—#define

- 用來處理編譯前之動作的指令，稱為前置處理器指令
- `#define` 可用來定義巨集，也就是以一個識別字，取代一連串動作或程式敘述

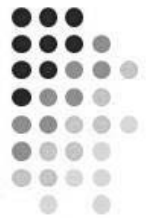
#define 前置處理器的使用格式

#define 識別名稱 代換標記 \rightarrow 這兒不可以加分號

- 下面的範例為合法的 `#define` 定義：

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

38



#define的使用範例 (1/3)

- 將 C 語言裡的左右大括號以 #define 重新定義：

```

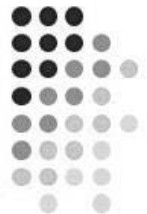
01  /* prog8_20, 使用#define */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define BEGIN {          /* 定義識別名稱 BEGIN 為左大括號{ */
05  #define END }           /* 定義識別名稱 END 為右大括號} */
06  int main(void)
07  BEGIN                  /* 此行的 BEGIN 相當於左大括號 { */
08      int i,j;
09      for(i=1;i<=5;i++)
10  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 } */

```

/* prog8_20 OUTPUT---
*
**

-----*/

39



#define的使用範例 (2/3)

- 使用 #define 定義字串：

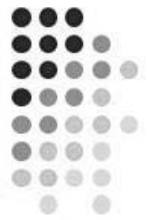
```

01  /* prog 8_21,使用#define */
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.
-----*/

40



#define 的使用範例 (3/3)

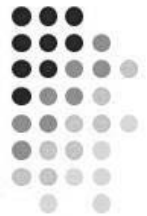
- 利用 #define 將 π 定義成常數 3.14：

```

01  /* prog8_22, 使用前置處理器來定義數學常數 */
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  }
```

/* prog8_22 OUTPUT---
PI=3.14, area()= 12.56
-----*/

41



使用 const 關鍵字

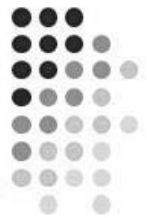
- 使用 const 定義的變數，不能再重新設值：

```

01  /* prog8_23, const 關鍵字使用的範例 */
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  }
```

/* prog8_23 OUTPUT-----
pi=3.14, area()= 12.56
-----*/

42



利用 #define 取代簡單的函數

- 利用巨集定義函數(沒有引數的版本)：

```

01  /* prog8_24, 使用巨集的範例 */
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  }

```

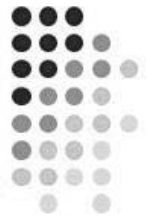
/* prog8_24 OUTPUT---

Input an integer:4

4*4=16

-----*/

43



利用 #define 取代簡單的函數

- 利用巨集定義函數(有引數的版本)：

```

01  /* prog8_25, 帶有引數的巨集 */
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  }

```

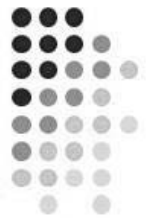
/* prog8_25 OUTPUT---

Input an integer:12

12*12=144

-----*/

44



使用巨集常見的錯誤

- 未將引數用括號括起來所造成的錯誤：

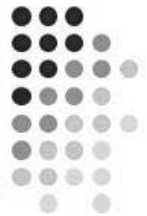
```

01  /* prog8_26, 使用巨集常見的錯誤 */
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  }

```

/* prog8_26 OUTPUT---
Input an integer:12
13*13=25
-----*/

45



更正巨集的錯誤

- 在各個運算元外加上括號，即可訂正錯誤：

```

01  /* prog8_27, 修改 prog8_26 的錯誤 */
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  }

```

/* prog8_27 OUTPUT---
Input an integer:12
13*13=169
-----*/

46

使用函數？使用巨集？

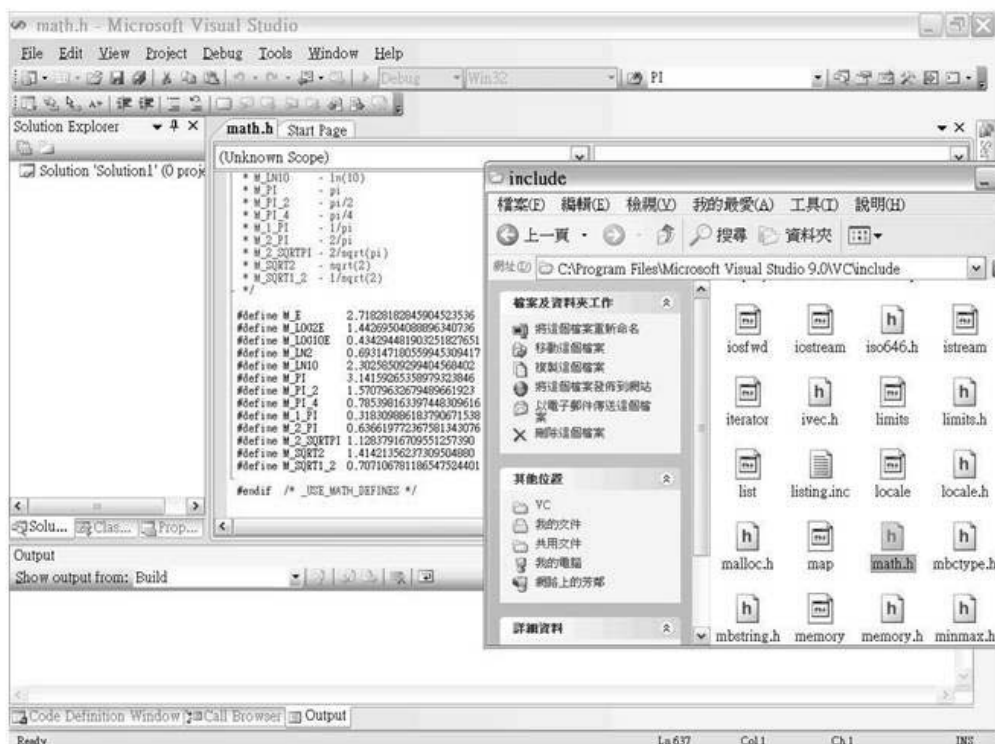
- 在編譯前，編譯器會以巨集取代原來的敘述
 - 巨集在編譯前已取代原來的敘述，程式不必跳到函數執行
 - 以巨集撰寫函數，執行速度較快，但編譯後的程式碼較大
- 函數是一個跳躍敘述
 - 在程式執行時期，碰到函數則是跳到函數的定義區去執行
 - 函數的寫法執行速度較慢，但是執行檔較小

47

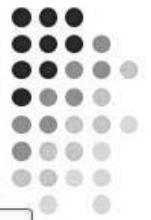
8.7 再來看看 #include前置處理器

標準的標頭檔

- 開啟math.h標頭檔的畫面：



48



撰寫標頭檔

area.h

● 使用自訂的標頭檔：

```
#define PI 3.14
#define CIRCLE(r) ((PI)*(r)*(r))
#define RECTANGLE(length,height) ((length)*(height))
#define TRIANGLE(base,height) ((base)*(height)/2.)
```

```
01  /* prog8_28, 使用自訂的標頭檔 area.h */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #include "C:\area.h" /* 載入 C:\ 路徑下的標頭檔 area.h */
05  int main(void)
06  {
07      float base, height;
08      printf("請輸入三角形的底:");
09      scanf("%f",&base);
10      printf("請輸入三角形的高:");
11      scanf("%f",&height);
12      printf("三角形面積為:%.2f\n",TRIANGLE(base,height));
13      system("pause");
14      return 0;
15  }
```

/* prog8_28 OUTPUT---

請輸入三角形的底:3

請輸入三角形的高:5

三角形面積為:7.50

-----*/