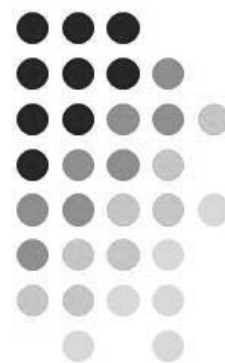


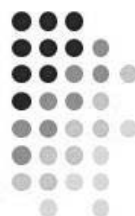
第十一章 結構與其它資料型態

認識結構與巢狀結構
學習結構陣列的各種使用方法
學習列舉的使用
學習使用自訂的型態—typedef



1

11.1 認識結構



認識結構

- 結構可將型態不同的資料合併成為新的型態
- 定義結構與宣告結構變數的格式如下：

定義結構與宣告結構變數的語法

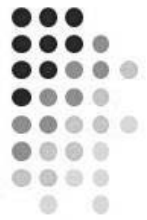
```
struct 結構名稱  
{  
    資料型態 成員名稱 1;  
    資料型態 成員名稱 2;  
    ...  
    資料型態 成員名稱 n;  
};
```

結構的成員

定義結構

```
struct 結構名稱 變數 1, 變數 2, ..., 變數 n; —— 宣告結構變數
```

2



認識結構

● 結構定義的範例

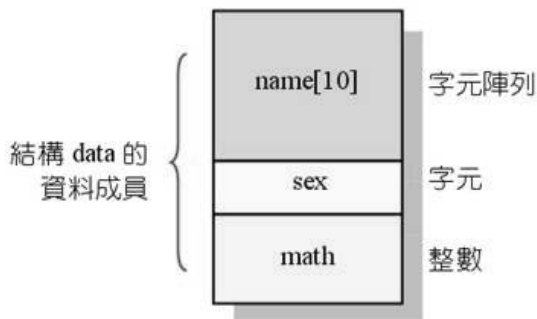
```
struct data    /* 定義 data 結構*/
{
    char name[10];
    char sex;
    int math;
};
```

} 定義結構的成員

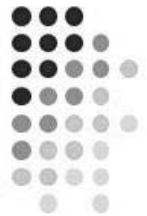
```
struct data mary, tom; /* 宣告 data 型態的結構變數 */
```



定義完結構之後，立即宣告結構變數



```
struct data    /* 定義 data 結構*/
{
    char name[10];
    char sex;
    int math;
}mary, tom;    /* 宣告結構變數 mary 與 tom */
```



認識結構

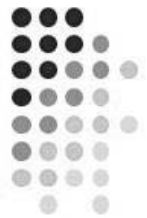
● 存取結構變數的成員：

存取結構變數的成員

結構變數名稱.成員名稱;

結構成員存取運算子

```
strcpy(mary.name, "Mary"); /* 設定 mary 的 name 成員為 "Mary" */
mary.sex='F';              /* 設定 sex 成員為 'F' */
mary.math=95;              /* 設定 math 成員為 95 */
```



使用結構的範例

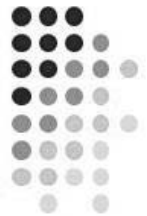
```

01  /* prog11_1, 結構變數的輸入與輸出 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data      /* 定義結構 data */
07      {
08          char name[10];
09          int math;
10      } student;      /* 宣告 data 型態的結構變數 student */
11      printf("請輸入姓名: ");
12      gets(student.name);      /* 輸入學生姓名 */
13      printf("請輸入成績:");
14      scanf("%d",&student.math);      /* 輸入學生成績 */
15      printf("姓名:%s\n", student.name);
16      printf("成績:%d\n", student.math);
17      system("pause");
18      return 0;
19  }

```

/* prog11_1 OUTPUT---
 請輸入姓名: Tom Lee
 請輸入成績: 89
 姓名:Tom Lee
 成績:89
 -----*/

5



結構變數所佔的記憶空間

- 利用sizeof() 求出結構所佔用的記憶體空間：

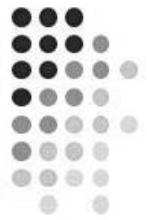
```

01  /* prog11_2, 結構的大小 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data      /* 定義結構 */
07      {
08          char name[10];
09          int math;
10      } student;
11      printf("sizeof(student)=%d\n", sizeof(student));
12
13      system("pause");
14      return 0;
15  }

```

/* prog11_2 OUTPUT--
 sizeof(student)=16
 -----*/

6



結構變數初值的設定

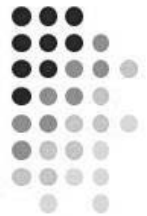
- 要設定結構變數的初值，可利用下面的語法：

```
struct data      /* 定義結構 data */
{
    char name[10];
    int math;
};
struct data student={"Jenny",78}; /* 設定結構變數的初值 */
```

- 將結構的定義與變數初值的設定合在一起：

```
struct data      /* 定義結構 data */
{
    char name[10];
    int math;
} student={"Jenny",78}; /* 宣告結構變數,並設定初值 */
```

7



結構變數初值的設定

- 設定結構變數初值的範例

```
01  /* prog11_3, 結構變數的初值設定 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data /* 定義結構 data */
07      {
08          char name[10];
09          int math;
10      };
11      struct data student={"Mary Wang",74}; /* 設定結構變數初值 */
12      printf("學生姓名: %s\n",student.name);
13      printf("數學成績: %d\n",student.math);
14
15      system("pause");
16      return 0;
17  }
```

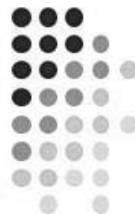
/* prog11_3 OUTPUT--

學生姓名: Mary Wang

數學成績: 74

-----*/

8



結構變數的設定

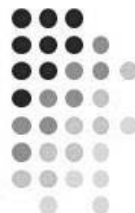
- 把結構變數的值設給另一個結構變數：

```

01  /* progl1_4, 結構的設值 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data
07      {
08          char name[10];
09          int math;
10      } s1={"Lily Chen",83};    /* 宣告結構變數 s1，並設定初值 */
11      struct data s2;          /* 宣告結構變數 s2 */
12      s2=s1;                   /* 把結構變數 s1 的值設定給結構變數 s2 */
13      printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
14      printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);
15      system("pause");
16      return 0;
17  }

```

9



巢狀結構

- 結構內如有另一結構，則此結構稱為巢狀結構

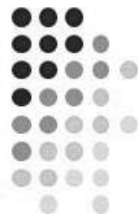
巢狀結構的格式

```

struct 結構1
{
    /* 結構1的成員 */
};
struct 結構2
{
    /* 結構2的成員 */
    struct 結構1 變數名稱
};

```

結構 2 內包含有結構 1



巢狀結構的範例

```

01  /* prog11_5, 巢狀結構的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct date      /* 定義結構 date */
07      {
08          int month;
09          int day;
10      };
11      struct data      /* 定義巢狀結構 data */
12      {
13          char name[10];
14          int math;
15          struct date birthday;
16      } s1={"Mary Wang",74,{10,2}}; /* 設定結構變數 s1 的初值 */
17      printf("學生姓名: %s\n",s1.name);
18      printf("生日: %d月%d日\n",s1.birthday.month,s1.birthday.day);
19      printf("數學成績: %d\n",s1.math);
20
21      system("pause");
22      return 0;
23  }

```

/* prog11_5 OUTPUT---

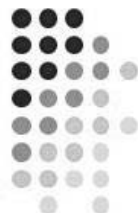
學生姓名: Mary Wang

生日: 10 月 2 日

數學成績: 74

-----*/

11



結構陣列

- 下面為結構陣列的宣告格式：

結構陣列的宣告格式

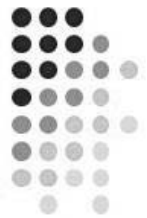
struct 結構型態 結構陣列名稱 [元素個數];

```

struct data s1[10];          /* 宣告結構陣列 s1 */
s1[2].math=12;               /* 設定 s1[2].math=12 */
strcpy(s1[2].name,"Peggy");  /* 設定 s1[2].name 的值為 "Peggy" */

```

12



結構陣列的範例

- 利用sizeof() 計算結構陣列及其元素所佔的位元組：

```

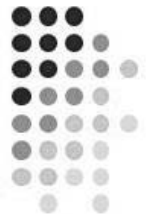
01  /* prog11_6, 結構陣列的大小 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data      /* 定義結構 */
07      {
08          char name[10];
09          int math;
10      }student[10];
11
12      printf("sizeof(student[3])=%d\n",sizeof(student[3]));
13      printf("sizeof(student)=%d\n",sizeof(student));
14      system("pause");
15      return 0;
16  }

```

/* prog11_6 OUTPUT----

sizeof(student[3])=16
sizeof(student)=160
-----*/

13



結構陣列的範例

```

01  /* prog11_7, 結構陣列的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 2
05  int main(void)
06  {
07      int i;
08      struct data
09      {
10          char name[10];
11          int math;
12      } student[MAX];      /* 宣告結構陣列 student */
13      for(i=0;i<MAX;i++)
14      {
15          printf("學生姓名: ");
16          gets(student[i].name);
17          printf("數學成績: ");
18          scanf("%d",&student[i].math);
19          fflush(stdin);
20      }
21      for(i=0;i<MAX;i++)
22          printf("%s的數學成績=%d\n",student[i].name,student[i].math);
23      system("pause");
24      return 0;
25  }

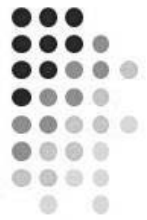
```

/* prog11_7 OUTPUT--

學生姓名: Jenny
數學成績: 65
學生姓名: Teresa
數學成績: 88
Jenny 的數學成績=65
Teresa 的數學成績=88
-----*/

/* 輸入學生姓名 */
/* 輸入學生數學成績 */
/* 清空緩衝區內的資料 */
/* 輸出結構陣列的內容 */

14



指向結構的指標

- 假設於程式中定義如下的結構，並以指標ptr指向它：

```
struct data          /* 定義結構 data */
{
    char name[10];
    int math;
}student;            /* 宣告結構 data 型態之變數 student */

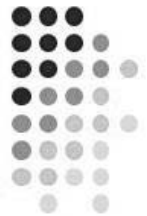
struct data *ptr;     /* 宣告指向結構 data 型態之指標 ptr */
ptr=&student;         /* 將指標 ptr 指向結構變數 student */
```

- 以指標指向的結構，必須以「->」存取其成員：

```
strcpy(ptr->name, "Mary"); /* 設定 ptr 所指向之結構的 name 成員為 "Mary" */
ptr->math=95;              /* 設定 ptr 所指向之結構的 math 成員等於 95 */
```

以「->」存取成員

15

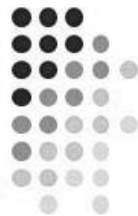


指向結構的指標

```
01  /* prog11_8, 使用指向結構的指標 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      struct data /* 定義結構 */
07      {
08          char name[10];
09          int math;
10          int eng;
11      } student, *ptr; /* 宣告結構變數 student 及指向結構的指標 ptr */
12      ptr=&student;    /* 將 ptr 指向結構變數 student 的位址 */
13      printf("學生姓名: ");
14      gets(ptr->name); /* 輸入字串給 student 的 name 成員存放 */
15      printf("數學成績: ");
16      scanf("%d", &ptr->math); /* 輸入整數給 student 的 math 成員存放 */
17      printf("英文成績: ");
18      scanf("%d", &ptr->eng); /* 輸入整數給 student 的 eng 成員存放 */
19      printf("數學成績=%d, ", ptr->math);
20      printf("英文成績=%d, ", ptr->eng);
21      printf("平均分數=%.2f\n", (ptr->math + ptr->eng)/2.0);
22      system("pause");
23      return 0;
24  }
```

```
/* prog11_8 OUTPUT-----
學生姓名: Jenny
數學成績: 78
英文成績: 89
數學成績=78, 英文成績=89, 平均分數=83.50
-----*/
```

16



以指標的方式表示結構陣列 (1/2)

- 以指標表示結構陣列的語法：

以指標的方式表示結構陣列

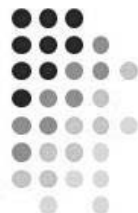
(結構陣列名稱+i) -> 結構成員;

```

01  /* prog11_9, 以指標來表示結構陣列 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 3
05  int main(void)
06  {
07      int i,m,index=0;
08      struct data          /* 定義結構 data */
09      {
10          char name[10];
11          int math;
12      } student[MAX]={ {"Mary",87}, {"Flora",93}, {"Jenny",74}};
13

```

17



以指標的方式表示結構陣列 (2/2)

```

14      m=student->math;          /* 將 m 設值為 student[0].math */
15      for(i=1;i<MAX;i++)        /* 輸出結構陣列的內容 */
16      {
17          if((student+i)->math > m)
18          {
19              m=(student+i)->math;
20              index=i;
21          }
22      }
23      printf("%s 的成績最高, ", (student+index)->name);
24      printf("分數為%d 分\n", (student+index)->math);
25      system("pause");
26      return 0;
27  }

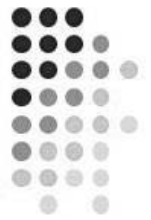
```

/* prog11_9 OUTPUT -----

Flora 的成績最高, 分數為 93 分

-----*/

18



以結構為引數傳遞到函數

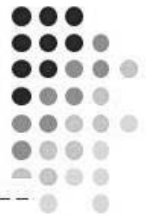
- 將結構傳遞到函數的格式：

將結構傳遞到函數

傳返回值型態 函數名稱 (**struct** 結構名稱 變數名稱)

```
{
    /* 函數的定義 */
}
```

19



傳遞結構到函數的範例

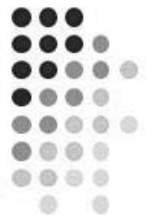
/* prog11_10 OUTPUT ---

```
01  /* prog11_10, 傳遞結構到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  struct data
05  {
06      char name[10];
07      int math;
08  };
09  void display(struct data); /* 宣告函數 display() 的原型 */
10  int main(void)
11  {
12      struct data s1={"Jenny",74}; /* 設定結構變數 s1 的初值 */
13      display(s1); /* 呼叫函數 display(), 傳入結構變數 s1 */
14      system("pause");
15      return 0;
16  }
17  void display(struct data st) /* 定義 display() 函數 */
18  {
19      printf("學生姓名: %s\n",st.name);
20      printf("數學成績: %d\n",st.math);
21  }
```

學生姓名: Jenny
數學成績: 74
-----*/

將結構 data 定義在 main() 的外部，這個結構就成了全域的結構

20



傳遞結構的位址 (1/2)

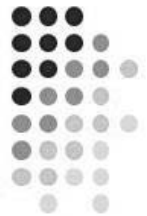
● 傳遞結構位址的範例：

```

01  /* prog11_11, 傳遞結構的位址到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04
05  struct data      /* 定義全域的結構 data */
06  {
07      char name[10];
08      int math;
09  };
10  void swap(struct data *,struct data *); /* swap() 的原型 */
11
12  int main(void)
13  {
14      struct data s1={"Jenny",74}; /* 宣告結構變數 s1，並設定初值 */
15      struct data s2={"Teresa",88}; /* 宣告結構變數 s2，並設定初值 */
16

```

21



傳遞結構的位址 (2/2)

```

17      swap(&s1,&s2); /* 呼叫 swap() 函數 */
18      printf("呼叫 swap() 函數後:\n");
19      printf("s1.name=%s, s1.math=%d\n",s1.name,s1.math);
20      printf("s2.name=%s, s2.math=%d\n",s2.name,s2.math);
21
22      system("pause");
23      return 0;
24  }
25  void swap(struct data *p1,struct data *p2)
26  {
27      struct data tmp;
28      tmp=*p1;
29      *p1=*p2;
30      *p2=tmp;
31  }

```

/* prog11_11 OUTPUT -----

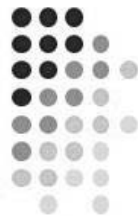
呼叫 swap() 函數後：

s1.name=Teresa, s1.math=88

s2.name=Jenny, s2.math=74

-----*/

22



傳遞結構陣列 (1/2)

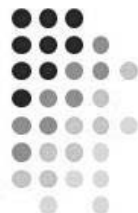
- 傳遞結構陣列到函數裡的範例：

```

01  /* prog11_12, 傳遞結構陣列 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 3
05
06  struct data          /* 定義全域的結構 data */
07  {
08      char name[10];
09      int math;
10  };
11  int maximum(struct data arr[]); /* 宣告函數 maximum() 的原型 */
12  int main(void)
13  {
14      int idx;
15      struct data s1[MAX]={{"Mary",87},{"Flora",93},{"Jenny",74}};
16

```

23



傳遞結構陣列 (2/2)

```

17      idx=maximum(s1); /* 呼叫 maximum() 函數 */
18      printf("%s 的成績最高, ", (s1+idx)->name); /* 印出最高分的姓名 */
19      printf("分數為%d分\n", (s1+idx)->math); /* 印出最高分的成績 */
20
21      system("pause");
22      return 0;
23  }
24  int maximum(struct data arr[]) /* maximum() 函數的定義 */
25  {
26      int m,i,index;
27      m=arr->math; /* 將 m 設值為 arr[0].math */
28      for(i=0;i<MAX;i++)
29          if((arr+i)->math>m)
30          {
31              m=(arr+i)->math;
32              index=i;
33          }
34      return index; /* 傳回陣列的索引值 */
35  }

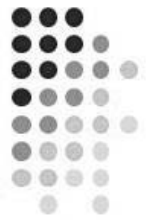
```

```

/* prog11_12 OUTPUT -----
Flora 的成績最高, 分數為 93 分
-----*/

```

24



列舉型態

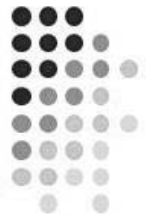
- 列舉型態 (enumeration)
 - 可以用某個有意義的名稱來取代較不易記憶的整數常數
- 列舉型態定義及宣告變數的格式：

列舉型態定義及宣告變數的格式

```
enum 列舉型態名稱
{
    列舉常數1,
    列舉常數2,
    ...
    列舉常數n
};

enum 列舉型態名稱 變數1, 變數2, ..., 變數m; /* 宣告變數 */
```

25



列舉型態的定義與變數的宣告

- 定義列舉型態與宣告變數的範例：

```
enum color /* 定義列舉型態 color */
{
    red,
    blue,
    green
};

enum color shirt, hat; /* 宣告列舉型態 color 變數 shirt 與 hat */
```

預設值為0
預設值為1
預設值為2

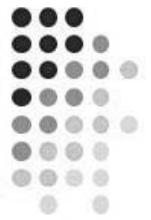
shirt與hat的值只能是0, 1 與 2 其中之一

- 定義完列舉型態後，立即宣告列舉型態的變數

```
enum color /* 定義列舉型態 color */
{
    red,
    blue,
    green
} shirt, hat; /* 定義列舉型態後，便立即宣告變數 shirt 與 hat */
```

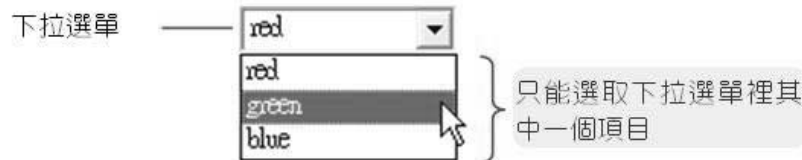
列舉常數

26



下拉選單與列舉型態的關係

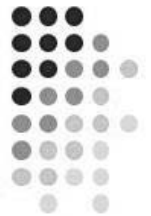
- 下拉選單的設計非常類似於列舉型態：



```
enum color
{
    red,
    blue,
    green
} shirt, hat;
```

shirt與hat的值只可以是red, green 與 blue 其中之一，不能為其它的值

27

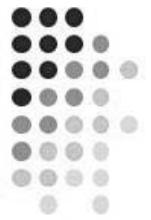


列舉型態的使用範例(一)

```
01  /* prog11_13, 列舉型態的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      enum color /* 定義列舉型態 color */
07      {
08          red,
09          green,
10          blue
11      };
12      enum color shirt; /* 宣告列舉型態的變數 shirt */
13      printf("sizeof(shirt)=%d\n", sizeof(shirt));
14      printf("red=%d\n", red);
15      printf("green=%d\n", green);
16      printf("blue=%d\n", blue);
17      shirt=green; /* 將 shirt 的值設為 green */
18      if (shirt==green)
19          printf("您選擇了綠色的衣服\n");
20      else
21          printf("您選擇了非綠色的衣服\n");
22      system("pause");
23      return 0;
24  }
```

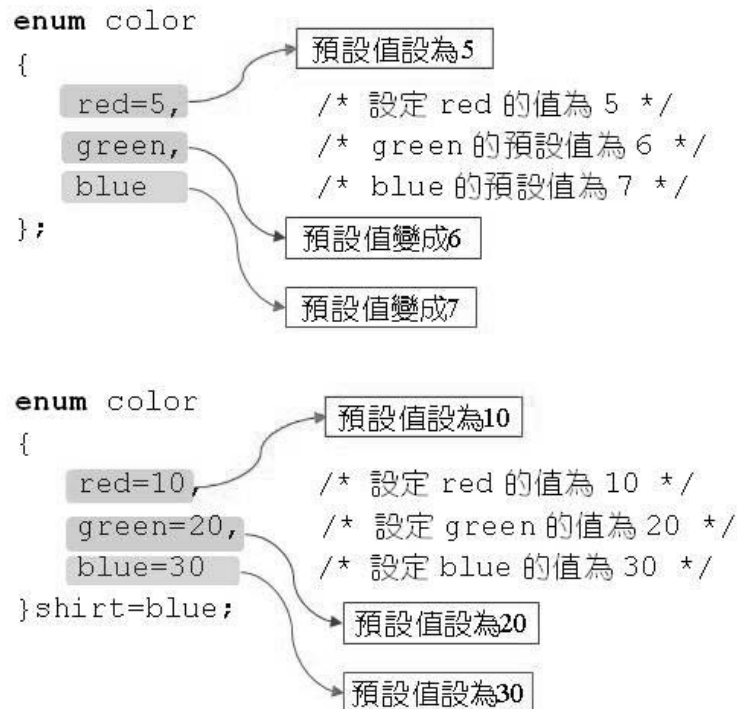
```
/* prog11_13 OUTPUT ---
sizeof(shirt)=4
red=0
green=1
blue=2
您選擇了綠色的衣服
-----*/
```

28

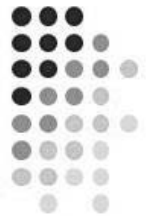


列舉常數的值

- 列舉常數的值可從其它整數開始：



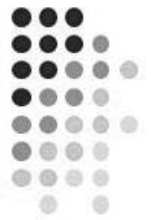
29



列舉型態的使用範例(二)

```
01  /* prog11_14, 列舉型態的使用範例 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char key; /* 用來儲存按鍵的資訊 */
07      enum color /* 定義列舉型態 color */
08      {
09          red=114, /* 將列舉常數 red 設定為 114, 即字母 r 的 ASCII 碼 */
10          green=103, /* 將列舉常數 green 設定為 103 (g 的 ASCII 碼) */
11          blue=98 /* 將列舉常數 blue 設定為 98 (b 的 ASCII 碼) */
12      } shirt; /* 宣告列舉型態的變數 shirt */
13
14      do
15      {
16          printf("請輸入 r, g 或 b: ");
17          scanf("%c", &key); /* 讀入一個字元 */
18          fflush(stdin); /* 清空緩衝區內的資料 */
19      } while ((key!=red) && (key!=green) && (key!=blue));
20
```

30



列舉型態的使用範例(二)

```

21  shirt=key;          /* 將 key 的值指定給 shirt 變數存放 */
22
23  switch(shirt)        /* 根據 shirt 的值印出字串 */
24  {
25      case red:
26          printf("您選擇了紅色\n");
27          break;
28      case green:
29          printf("您選擇了綠色\n");
30          break;
31      case blue:
32          printf("您選擇了藍色\n");
33          break;
34  }
35  system("pause");
36  return 0;
37  }

```

/* prog11_14 OUTPUT --

請輸入 r,g 或 b: *h*

請輸入 r,g 或 b: *k*

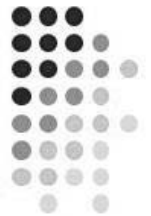
請輸入 r,g 或 b: *b*

您選擇了藍色

-----*/

31

11.7 使用自訂的型態—typedef



自訂型態—typedef

- typedef 可將原有的資料型態重新命名
 - 目的是為了使重新命名更易於閱讀和理解

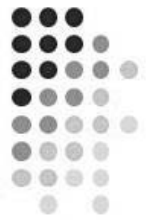
typedef 的使用格式

typedef 資料型態 識別字;

```

typedef int clock;          /* 定義clock為整數型態 */
clock hour,second;         /* 宣告hour,second為clock型態 */

```

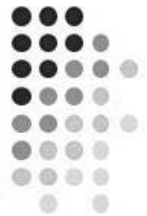
typedef 的使用範例

```

01  /* prog11_15, 利用 typedef 來定義資料型態 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  struct data
05  {
06      char name[10];
07      int math;
08  };
09  typedef struct data SCORE; /* 把 struct data 定義成新的型態 */
10  void display(SCORE);      /* 宣告函數 display() 的原型 */
11  int main(void)
12  {
13      SCORE s1={"Jenny",74}; /* 設定結構變數 s1 的初值 */
14      display(s1);           /* 呼叫 display(), 傳入結構變數 s1 */
15      system("pause");
16      return 0;
17  }
18  void display(SCORE st)     /* 定義函數 display() */
19  {
20      printf("學生姓名: %s\n",st.name);
21      printf("數學成績: %d\n",st.math);
22  }

```

33



簡化typedef的定義

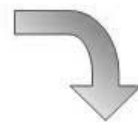
- 將prog11_15的定義簡化成一個步驟：

```

struct data
{
    char name[10];
    int math;
};
typedef struct data SCORE;

```

需要兩個步驟



由兩個步驟簡化成
一個步驟

只需要一個步驟 —

```

typedef struct
{
    char name[10];
    int math;
} SCORE;

```

34