

# 第十一章

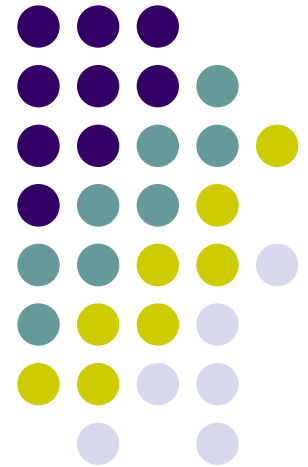
## 結構與其它資料型態

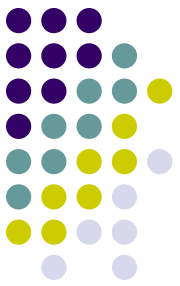
學習與使用結構

傳遞結構到函數

認識共同空間與列舉型態

使用自訂的資料型態





# 結構的宣告 (1/2)

- 結構可以同時存放不同型態的資料於同一個結構體
- 結構的定義及宣告格式如下

```
struct 結構名稱 [ ] → 不需要加分號  
{  
    資料型態 欄位名稱 1;  
    資料型態 欄位名稱 2;  
    ...  
    資料型態 欄位名稱 n;  
} [ ] → 記得要加分號  
struct 結構名稱 變數 1, 變數 2, ..., 變數 m;
```

- 下面是結構定義及宣告範例

```
struct mydata    // 定義結構 mydata  
{  
    string name;  // 各欄位的內容  
    string id;  
    int math;  
    int eng;  
};
```



## 結構的宣告 (2/2)

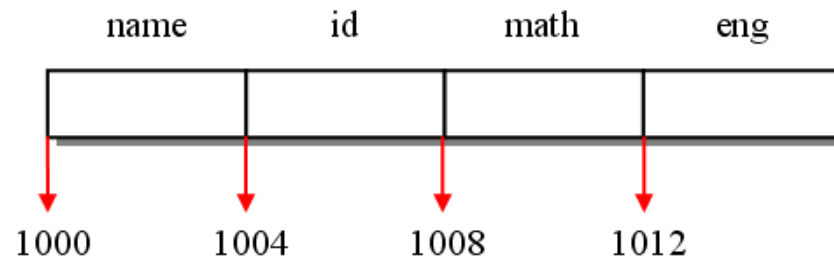
- 您也可以使用下列的格式來宣告結構

```
struct 結構名稱 {  
    資料型態 欄位名稱 1;  
    資料型態 欄位名稱 2;  
    ...  
    資料型態 欄位名稱 n;  
} 變數 1, 變數 2, ..., 變數 m;
```

不需要加分號

- 下面的結構定義及宣告範例為合法的格式：

```
struct mydata  
{  
    string name;  
    string id;  
    int math;  
    int eng;  
} student;
```





# 結構變數的使用及初值的設定

- 結構變數的使用格式

結構變數名稱 . 欄位名稱

- 結構內的成員可以利用小數點 ( . ) 來存取



# 結構變數的範例 (1/2)

- 下面的程式示範結構變數的輸入與輸出

```

01 // prog11_1, 結構變數的輸入與輸出
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct mydata
07 {
08     string name;
09     int math;
10 } student;
11 int main(void)
12 {
13     cout << "Student's name:";
14     getline(cin, student.name);
15     cout << "Math score:";
16     cin >> student.math;
17     cout << "*****Output*****" << endl;
18     cout << student.name << "'s Math score is " << student.math;
19
20     system("pause");
21     return 0;
22 }
```

// 定義並宣告結構變數

```

/* prog11_1 OUTPUT-----
Student's name:Tippi Hong
Math score:95
*****Output*****
Tippi Hong's Math score is 95
-----*/
```

// 輸出結構變數內容



## 結構變數的範例 (2/2)

- 結構所佔用的記憶體有多少呢？請看看下面的程式

```
01 // prog11_2, 結構的大小
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 struct mydata           // 定義結構
06 {
07     string name;
08     int math;
09 } student;
10 int main(void)
11 {
12     cout << "sizeof(student)=" << sizeof(student) << endl;
13                                     /* prog11_2 OUTPUT---
14     system("pause");
15     return 0;
16                                     sizeof(student)=8
17                                     -----*/
16 }
```



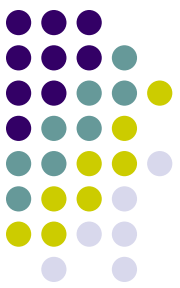
# 設定結構變數的初值 (1/3)

- 用設定運算子 ( = ) 來設定結構變數的初值
- 宣告結構變數並為其設值的範例

```
struct mygood          // 定義結構 mygood
{
    string good;        // 貨品名稱
    int cost;           // 貨品成本
};
struct mygood first={"cracker",32};
```

- 在結構定義之後，直接宣告並設定變數的初值

```
struct mygood          // 定義結構 mygood
{
    string good;        // 貨品名稱
    int cost;           // 貨品成本
} first={"cracker",32}; // 同時宣告變數 first, 並設定初值
```

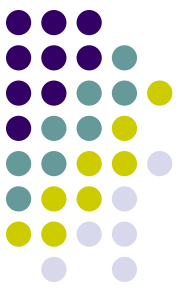


# 設定結構變數的初值 (2/3)

- 下面是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 <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct mydata // 定義並宣告結構變數
07 {
08     string name;
09     int math;
10 } student={"Mary Wang",74}; // 設定結構變數初值
11 int main(void)
12 {
13     cout << "Student's name:" << student.name; // 輸出結構變數內容
14     cout << endl << "m ath score=" << student.math << endl;
15
16     system("pause"); /* prog11_3 OUTPUT-----
17     return 0;         Student's name:Mary Wang
18 }                     Math score=74
                       -----*/
```





# 設定結構變數的初值 (3/3)

- 將結構變數x設給另一個結構變數y的練習

```
01 // prog11_4, 結構的設值
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct mydata // 定義結構
07 {
08     string name;
09     int age;
10 } x; // 宣告結構變數
11 int main(void)
12 {
13     struct mydata y={"Lily Chen",18};
14     x=y;
15     //輸出結構變數內容
16     cout << "x.name=" << x.name << ", x.age=" << x.age << endl;
17     cout << "y.name=" << y.name << ", y.age=" << y.age << endl;
18
19     system("pause");
20     return 0;
21 }
```

**/\* prog11\_4 OUTPUT-----**  
x.name=Lily Chen, x.age=18  
y.name=Lily Chen, y.age=18  
**-----\*/**



# 將整個結構傳遞到函數 (1/3)

- 下面列出將結構傳遞到函數中的格式

```
struct 結構名稱 1
{
    資料型態 欄位名稱 1;
    ...
    資料型態 欄位名稱 n;
} 變數 1, 變數 2, ..., 變數 m;

傳返回值型態 函數名稱(struct 結構名稱 1 變數名稱 1);    // 函數原型

int main(void)
{
    ...
    函數名稱(結構變數名稱);
}

傳返回值型態 函數名稱(struct 結構名稱 1 變數名稱 1)
{
    ...
}
```



## 將整個結構傳遞到函數 (2/3)

- 下面的程式，是將結構變數當成引數傳入函數中

```
01 // prog11_5, 結構與函數
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct mydata
07 {
08     string name;
09     int age;
10 };
11 void func(struct mydata);
12 int main(void)
13 {
```

```
14     struct mydata woman={"Mary Wu",5};           // 宣告結構變數
15     cout << "before process..." << endl;
16     cout << "In main(), " << woman.name;          // 印出結構變數內容
17     cout << "'s age is " << woman.age << endl;
18     cout << "after process..." << endl;
19     func(woman);                                   // 呼叫 func() 函數
20     cout << "In main(), " << woman.name;
21     cout << "'s age is " << woman.age << endl;
```

```
/* prog11_5 OUTPUT -----
before process...
In main(), Mary Wu's age is 5
after process...
In func(), Mary Wu's age is 15
In main(), Mary Wu's age is 5
-----*/
```



# 將整個結構傳遞到函數 (3/3)

```
22
23     system("pause");
24     return 0;
25 }
26
27 void func(struct mydata a)                // 自訂函數 func()
28 {
29     a.age+=10;
30     cout << "In func(), " << a.name;      // 印出結構變數內容
31     cout << "'s age is " << a.age << endl;
32     return;
33 }
```

**/\* prog11\_5 OUTPUT -----**

before process...

In main(), Mary Wu's age is 5

after process...

In func(), Mary Wu's age is 15

In main(), Mary Wu's age is 5

**-----\*/**

# 將結構欄位分別傳遞

## 11.2 以結構為引數傳遞到函數

- 程式prog11\_6僅將結構變數的部分欄位傳入函數

```
01 // prog11_6, 將結構欄位分別傳遞到函數
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct mydata // 定義結構
07 {
08     string name;
09     int math;
10     int eng;
11 };
12 float avg(int,int); // 函數原型
13 int main(void)
14 {
15     struct mydata num={"Alice",71,80}; // 宣告結構變數
16     cout << num.name << "'s Math score=" << num.math; // 印出結構變數內容
17     cout << endl << "English score=" << num.eng << endl;
18     cout << "average=" << avg(num.math,num.eng) << endl;
19
20     system("pause");
21     return 0;
22 }
23
24 float avg(int a,int b) // 自訂函數 avg()
25 {
26     return (float) (a+b)/2;
27 }
```

/\* prog11\_6 OUTPUT -----

Alice's Math score=71  
English score=80  
average=75.5

-----\*/



# 傳遞結構的位址 (1/2)

- 下面的程式利用指標的方式傳遞結構變數到函數

```

01 // prog11_7, 以指標傳遞結構到函數
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 struct data                                     // 定義結構
07 {
08     string name;
09     int a,b;
10 };
11 void change(struct data *),prnstr(struct data);    // 函數原型
12 int main(void)
13 {
14     struct data first={"David Young",9,2};        // 宣告結構變數
15     prnstr(first);
16     cout << "after process..." << endl;
17     change(&first);
18     prnstr(first);
19
20     system("pause");
21     return 0;
22 }

```

/\* prog11\_7 OUTPUT ----

name=David Young  
a=9      b=2  
after process...  
name=David Young  
a=2      b=9

-----\*/



## 傳遞結構的位址 (2/2)

```
23
24 void change(struct data *ptr)           // 自訂函數 change()
25 {
26     int temp;
27     temp=ptr->a;           // ptr->a 可取出 ptr 所指向之結構的欄位 a 之值
28     ptr->a=ptr->b;         // 取出欄位 b 的值，並設定給欄位 a 存放
29     ptr->b=temp;          // 將 temp 設定給 ptr 所指向之結構的欄位 b 存放
30     return;
31 }
32
33 void prnstr(struct data in)             // 印出結構變數內容
34 {
35     cout << "name=" << in.name << endl;
36     cout << "a=" << in.a << "\t";
37     cout << "b=" << in.b << endl;
38     return;
39 }
```

**/\* prog11\_7 OUTPUT ----**

```
name=David Young
a=9      b=2
after process...
name=David Young
a=2      b=9
```

**-----\*/**



# 共同空間的定義及宣告 (1/3)

- 共同空間又稱為聯合或同位，它是利用同一塊共用的空間來存放資料，其格式可依需要變更
- 共同空間可以讓多種不同型態的變數共用同樣的記憶體空間，以達到節省記憶體的目的





# 共同空間的定義及宣告 (2/3)

- 共同空間的定義及宣告方式之格式如下

```
union 共同空間型態 [ ] → 不需要加分號
{
    資料型態 欄位名稱 1;
    資料型態 欄位名稱 2;
    ...
    資料型態 欄位名稱 n;
}; → 記得要加分號
union 共同空間型態 變數 1, 變數 2, ..., 變數 m;
```

- 下面是共同空間定義及宣告的範例

```
union mydata // 定義共同空間 mydata
{
    char grade;
    int score;
};
union mydata student; // 宣告共同空間 mydata 型態之變數
```



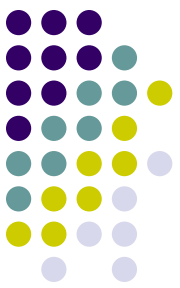
# 共同空間的定義及宣告 (3/3)

- 共同空間的定義及宣告格式的另一種方式

```
union 共同空間型態 { → 不需要加分號  
{  
    資料型態 欄位名稱 1;  
    資料型態 欄位名稱 2;  
    ...  
    資料型態 欄位名稱 n;  
} 變數 1, 變數 2, ..., 變數 m;
```

- 下面的共同空間定義及宣告範例為合法的格式

```
union mydata      // 定義共同空間 mydata  
{  
    char grade;  
    int score;  
} student;        // 直接於定義後面宣告變數 student
```



# 共同空間的使用及設值 (1/3)

- 下面的程式是共同空間的使用範例

```
01 // prog11_8, 共同空間的使用
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 union mydata // 定義共同空間
06 {
07     char grade;
08     int score;
09 } student; // 宣告共同空間變數
10 int main(void)
11 {
12     char sex;
13     do {
14         cout << "Your sex is (1)Male (2)Female:"; // 輸入性別
15         cin.get(sex);
16         cin.get(); // 吸收多餘的 enter 值
17     } while ((sex>50) || (sex<49));
18     if (sex=='1')
19     {
20         cout << "Input score:";
21         cin >> student.score;
22     }
```



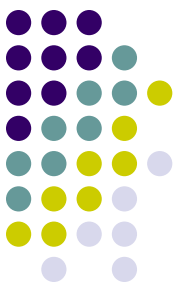
## 共同空間的使用及設值 (2/3)

```
23     else
24     {
25         cout << "Input grade:";
26         cin.get(student.grade);
27     }
28     cout << "**** Output ****" << endl;           // 輸出
29     if (sex=='1')
30         cout << "student.score=" << student.score << endl;
31     else
32         cout << "student.grade=" << student.grade << endl;
33
34     system("pause");
35     return 0;
36 }
```

**/\* prog11\_8 OUTPUT -----**

Your sex is (1)Male (2)Female:9  
Your sex is (1)Male (2)Female:1  
Input score:78  
\*\*\*\* Output \*\*\*\*  
student.score=78

**-----\*/**



# 共同空間的使用及設值 (3/3)

- 下面是設定共同空間變數初值的範例

```
01 // prog11_9, 共同空間的設值
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 union mydata          // 定義共同空間
06 {
07     int score;
08     char grade;
09 } student={65};      // 宣告共同空間變數
10 int main(void)
11 {
12     cout << "sizeof(student)=" << sizeof(student) << endl;
13     cout << "student.score=" << student.score << endl;
14                                     /* prog11_9 OUTPUT---
15     system("pause");                sizeof(student)=4
16     return 0;                       student.score=65
17 }                                   -----*/
```



# 共同空間與結構的差異 (1/2)

- 下面的程式可驗證共同空間記憶體的安排

```
01 // prog11_10, 共同空間的大小及位址
```

```
02 #include <iostream>
```

```
03 #include <cstdlib>
```

```
04 using namespace std;
```

```
05 union mydata
```

```
06 {
```

```
07     short math;
```

```
08     float avg;
```

```
09 } student;
```

```
10 int main(void)
```

```
11 {
```

```
12     cout << "sizeof(student)=" << sizeof(student) << endl;
```

```
13     cout << "address of student.math=" << &student.math << endl;
```

```
14     cout << "address of student.avg=" << &student.avg << endl;
```

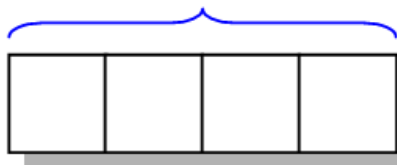
```
15
```

```
16     system("pause");
```

```
17     return 0;
```

```
18 }
```

student.avg, 4 個位元組

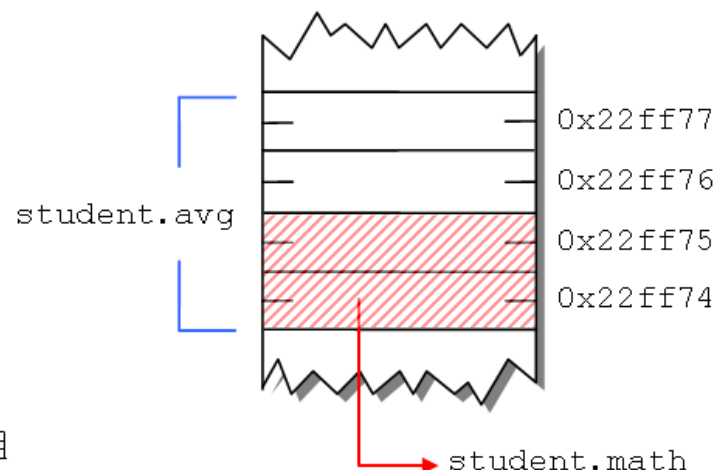


student.math, 2 個位元組



低位元組

高位元組



/\* prog11\_10 OUTPUT -----

sizeof(student)=4

address of student.math=0x443010

address of student.avg=0x443010

-----\*/



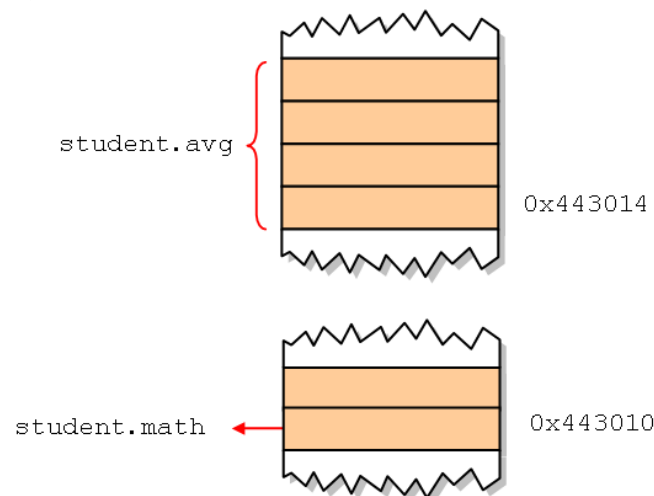
# 共同空間與結構的差異 (2/2)

- 下面的程式可驗證結構記憶體的安排

```

01 // prog11_11, 結構的大小及位址
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 struct mydata          // 定義結構
06 {
07     short math;
08     float avg;
09 } student;
10 int main(void)
11 {
12     cout << "sizeof(student)=" << sizeof(student) << endl;
13     cout << "address of student.math=" << &student.math << endl;
14     cout << "address of student.avg=" << &student.avg << endl;
15
16     system("pause");
17     return 0;
18 }

```



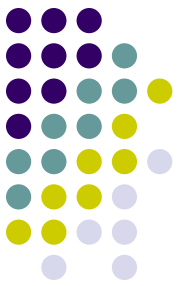
/\* prog11\_11 OUTPUT -----

```

sizeof(student)=8
address of student.math=0x443010
address of student.avg=0x443014

```

-----\*/

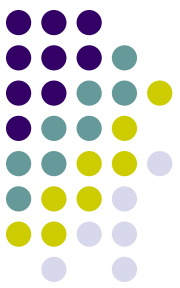


# 列舉型態的定義及宣告 (1/2)

- 列舉型態可以定義某種資料型態，並設定此資料型態內所包含的成員，以方便程式碼的撰寫
- 列舉型態的定義及宣告格式如下

```
enum 列舉型態名稱 [ ] → 不需要加分號  
{  
    列舉常數 1,  
    列舉常數 2,  
    ...  
    列舉常數 n  
} [ ; ] → 記得要加分號  
enum 列舉型態名稱 變數 1, 變數 2, ..., 變數 m;
```





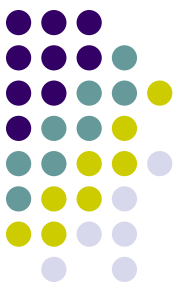
## 列舉型態的定義及宣告 (2/2)

- 下面為列舉型態定義及宣告的範例

```
enum desktop          // 定義列舉型態 desktop
{pen,pencil,eraser,book,tape};
enum desktop mine;     // 宣告列舉型態 desktop 之變數 mine
```

- 列舉型態的定義及宣告格式之另一種方式

```
enum desktop          // 定義列舉型態 desktop
{   pen,pencil,eraser,
    book,tape
} mine;               // 宣告列舉型態 desktop 之變數 mine
```



# 列舉型態的使用與設值 (1/8)

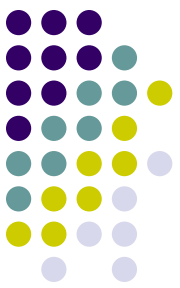
- 列舉型態會自動轉換成整數型態
- 整數型態不會自動轉換成為列舉型態
- 整數與列舉型態的轉換格式

列舉型態變數=static\_cast<列舉型態名稱>(欲轉換之內容);

- 下面列出合法與不合法的設值方式

```
May;  
six=static cast<month>(3);  
six=static cast<month>(six+2);  
six=4;  
six="May";  
six=July;
```

```
// 合法的列舉型態變數設值  
// 合法的列舉型態變數設值  
// 合法的列舉型態變數設值  
// 不合法的列舉型態變數設值  
// 不合法的列舉型態變數設值  
// 不合法的列舉型態變數設值
```

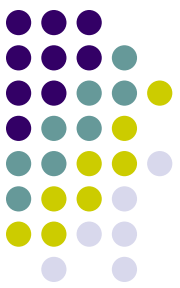


# 列舉型態的使用與設值 (2/8)

- 下面的程式示範列舉型態變數的使用方式

```
01 // prog11_12, 列舉型態的使用
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 enum month // 定義列舉型態
06 { January, February, March,
07   April, May, June } six;
08 int main(void)
09 {
10     cout << "sizeof(six)=" << sizeof(six) << endl; // 列舉型態的長度
11     cout << "January=" << January << endl;        // 印出列舉常數的值
12     cout << "February=" << February << endl;
13     cout << "March=" << March << endl;
14     cout << "April=" << April << endl;
15     cout << "May=" << May << endl;
16     cout << "June=" << June << endl;
17
18     system("pause");
19     return 0;
20 }
```

```
/* prog11_12 OUTPUT---
sizeof(six)=4
January=0
February=1
March=2
April=3
May=4
June=5
-----*/
```



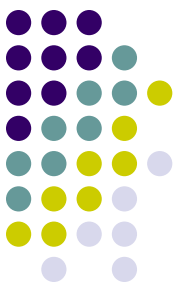
# 列舉型態的使用與設值 (3/8)

- 列舉常數值會由所設定的值開始遞增，如下面的程式

```

01 // prog11_13, 列舉常數的設值
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 enum month // 定義列舉型態
06 { January, February, March=4,           // 將 March 設值為 4
07   April, May, June } six;
08 int main(void)
09 {
10     cout << "January=" << January << endl; // 印出列舉常數的值
11     cout << "February=" << February << endl; /* prog11_13 OUTPUT-----
12     cout << "March=" << March << endl;      January=0
13     cout << "April=" << April << endl;        February=1
14     cout << "May=" << May << endl;           March=4
15     cout << "June=" << June << endl;         April=5
16                                           May=6
17                                           June=7
18                                           -----*/
19     system("pause");
20     return 0;
21 }
```

列舉常數值因設定而隨之更改

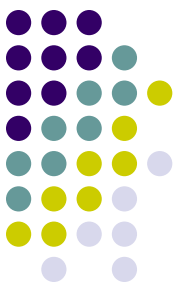


# 列舉型態的使用與設值 (4/8)

- 下面的程式是將列舉型態中的列舉常數印出

```
01 // prog11_14, 列舉型態的使用
02 #include <iostream>
03 #include <cstdlib>
04 #include <string>
05 using namespace std;
06 enum month // 定義列舉型態
07 { January, February, March,
08   April, May, June } six;
09 int main(void)
10 {
11     string a[6]={"January", "February", "March",
12                "April", "May", "June"};
13     for(six=January; six<=June; six=static cast<month>(six+1))
14         cout << "six(" << six << ")=" << a[six] << endl;
15
16     system("pause");
17     return 0;
18 }
```

```
/* prog11_14 OUTPUT-----
six(0)=January
six(1)=February
six(2)=March
six(3)=April
six(4)=May
six(5)=June
-----*/
```



# 列舉型態的使用與設值 (5/8)

- 下面的程式是利用列舉型態模擬滑鼠的三個按鈕

```
01 // prog11_15, 列舉型態的使用      /* prog11_15 OUTPUT -----
02 #include <iostream>                  Button press?(0) Left (1)Right (2)Middle: 5
03 #include <cstdlib>                   Button press?(0) Left (1)Right (2)Middle: 2
04 using namespace std;                Middle Button Pressed!
05 int main(void)                      -----*/
06 {
07     enum mykey                       // 定義列舉型態
08     {
09         left, right, middle
10     } mouse;                         // 宣告列舉型態變數
11     int key;
12     do                               // 輸入 0~2 的值
13     {
14         cout << "Button press?(0) Left (1)Right (2)Middle: ";
15         cin >> key;
16     } while((key>2) || (key<0));
17     mouse=static cast<mykey>(key);
```



# 列舉型態的使用與設值 (6/8)

```
18     switch(mouse)                                // 根據 key 的值印出字串
19     {
20         case left: cout << "Left Button Pressed!" << endl;
21                 break;
22         case right: cout << "Right Button Pressed!" << endl;
23                 break;
24         case middle: cout << "Middle Button Pressed!" << endl;
25     }
26
27     system("pause");
28     return 0;
29 }
```

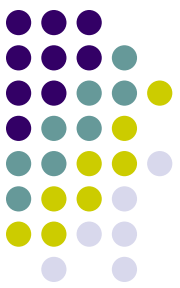
**/\* prog11\_15 OUTPUT -----**

Button press?(0) Left (1)Right (2)Middle: 5

Button press?(0) Left (1)Right (2)Middle: 2

Middle Button Pressed!

**-----\*/**



# 列舉型態的使用與設值 (7/8)

- 設定列舉變數初值的範例

```
enum sports      // 定義列舉型態 sports
{
    tennis,swimming,baseball,ski
} favorite=ski; // 設定 favorite 的初值為 ski
```

- 下面的程式是為列舉型態變數設值的範例

```
01 // prog11_16, 列舉變數的設值                                /* prog11_16 OUTPUT---
02 #include <iostream>                                           favorite=ski
03 #include <cstdlib>
04 using namespace std;                                         -----*/
05 enum sports                                                  // 定義列舉型態
06 {
07     tennis,swimming,baseball,ski
08 } favorite=ski;                                              // 宣告列舉變數並設值
09 int main(void)
10 {
```





# 列舉型態的使用與設值 (8/8)

```
11      cout << "favorite=";                // 印出列舉變數所對應的內容
12      switch(favorite)
13      {
14          case 0:cout << "tennis" << endl;
15                  break;
16          case 1:cout << "swimming" << endl;
17                  break;
18          case 2:cout << "baseball" << endl;
19                  break;
20          case 3:cout << "ski" << endl;
21      }
22
23      system("pause");
24      return 0;
25  }
```

**/\* prog11\_16 OUTPUT---**  
favorite=ski  
**-----\*/**



# typedef

- typedef是type definition的縮寫，就是定義型態之意
- typedef的使用格式如下所示

```
typedef 資料型態 識別字;
```

- 下面的型態定義及宣告範例

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



# 使用typedef

- 程式prog11\_17是利用typedef自訂資料型態的範例

```
01 // prog11_17, 自訂型態—typedef 的使用
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 int main(void)
06 {
07     typedef float temper;           // 定義自訂型態
08     temper f, c;                    // 宣告自訂型態變數
09     cout << "Input Celsius degrees:";
10     cin >> c;
11     f=(float) (9.0/5.0)*c+32;       // 轉換公式
12     cout << c << " Celsius is equal to "; // 印出轉換後的結果
13     cout << f << " Fahrenheit degrees" << endl;
14
15     system("pause");
16     return 0;
17 }
```



# #define與typedef

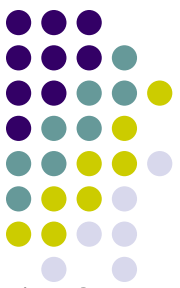
- 在某些情況下，#define可以取代typedef

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

- 在此可將 #define取代為typedef，成為如下面的敘述

```
#define CLOCK int    // 定義識別名稱 CLOCK 為 int  
CLOCK hour,second;    // 前置處理器會將 CLOCK 替換為 int
```

- 使用typedef時是由編譯器來執行
- #define是由前置處理器主導



# typedef的使用範例 (1/2)

- 下面的程式是使用typedef  
自訂新的型態之範例

```

01 // prog11_18, 自訂型態—typedef 的使用
02 #include <iostream>
03 #include <cstdlib>
04 #include <iomanip> // 將標頭檔 iomanip 含括進來
05 using namespace std;
06 typedef struct // 定義自訂型態
07 {
08     int hour;
09     int minite;
10     float second;
11 } mytime;
12 void subs(mytime t[]); // 函數原型
13 int main(void)
14 {
15     int i;
16     mytime t[3]={ {6,24,45.58f}, {3,40,17.43f} };
17     cout << setfill('0');
18     subs(t); // 呼叫subs()函數,計算t[0]+t[1]
19     for(i=0;i<3;i++) // 印出陣列內容
20     {
21         cout << "t[" << i << "]= " << setw(2) << t[i].hour << ":";
22         cout << setw(2) << t[i].minite << ":";
23         cout << setw(5) << t[i].second << endl;
24     }

```

/\* prog11\_18 OUTPUT----

```

t[0]=06:24:45.58
t[1]=03:40:17.43
t[2]=10:05:03.01
-----*/

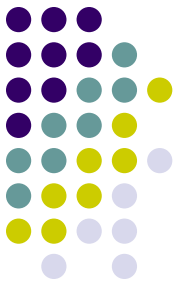
```



## typedef的使用範例 (2/2)

```
25
26     system("pause");
27     return 0;
28 }
29
30 void subs(mytime t[])           // 自訂函數 subs ()
31 {
32     int count2=0,count3=0;
33     t[2].second=t[0].second+t[1].second;           // 秒數相加
34     while(t[2].second>=60)
35     {
36         t[2].second-=60;
37         count3++;
38     }
39     t[2].minite=t[0].minite+t[1].minite+count3;     // 分數相加
40     while(t[2].minite>=60)
41     {
42         t[2].minite-=60;
43         count2++;
44     }
45     t[2].hour=t[0].hour+t[1].hour+count2;          // 時數相加
46     return;
47 }
```

/\* prog11\_18 OUTPUT-----  
t[0]=06:24:45.58  
t[1]=03:40:17.43  
t[2]=10:05:03.01  
-----\*/



The End-