

7-1 結構

重點整理

- 結構(Structure)是一種**使用者自行建立的資料型態**，由一個以上的成員組成，成員可以是整數、浮點數或陣列等資料型態。
- C 不允許函式做為結構成員，但 C++允許，因此 C++的結構和類別(Class)很相似，都可以達到物件導向的功能。
- 建立結構需使用 **struct** 關鍵字，語法如下：

```
struct 結構名稱{
    成員 1;
    成員 2;
    :
};
```

```
struct body{           /* 結構名稱 : body */
    int id;           /* 成員 1 : 整數 */
    char name[10];    /* 成員 2 : 字元陣列 */
    char sex;          /* 成員 3 : 字元 */
    float height, weight; /* 成員 4,5 : 浮點數 */
};
```

- 建立結構後，即可宣告該結構型態的變數，稱為結構變數，語法如下：

```
struct 結構名稱 結構變數名稱;
```

```
struct body s1;        /* 宣告結構變數 s1，型態為 body 結構 */
```

宣告結構變數時，可以同時指定成員的初始值，

```
struct 結構名稱 結構變數名稱 {成員 1 初始值, 成員 2 初始值, ...};
```

```
struct body s2 = {111002, "王小明", 'M', 171.5, 65.8}; /* 宣告結構變數 s2，指定成員初始值 */
```

宣告結構變數時，C 語言需加上 struct 關鍵字，C++則可以省略。

- 建立結構和宣告結構變數可以在一行程式敘述句完成，如下列程式碼，同時建立 body 結構和宣告結構變數 s1、s2，

```
struct body{
    int id;
    char name[10];
    char sex;
    float height, weight;
}s1, s2={111002, "王小明", 'M', 171.5, 65.8};
```

6. 使用「.」運算子存取結構成員，

```
s1.id = 111001;
strcpy(s1.name, "王小美"); /* 使用 strcpy()函式指定字串值 */
s1.sex = 'F';
s1.height = 165.5;
s1.weight = 49.9;

cout << "學號：" << s2.id << endl;
cout << "姓名：" << s2.name << endl ;
cout << "性別：" << s2.sex << endl ;
cout << "身高：" << s2.height << endl ;
cout << "體重：" << s2.weight << endl ;
```

01 範例



保健室阿姨為了記錄新生入學時的身體健康狀況，請電腦社社長莎莎幫忙設計程式，莎莎決定使用結構來記錄每個人的基本資料(性別、身高與體重)，而除了記錄基本資料外，同時也能輸出 BMI 值，保健室阿姨提供 2 筆測試資料，供莎莎驗證程式的執行結果。

學號	姓名	性別	身高	體重
11101	王小美	F	165.5	49.9
11102	王小明	M	171.5	65.8

解

程式碼：

```
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 struct body{
7     int id;
8     char name[10];
9     char sex;
10    float height, weight;
11 };
12
13 int main() {
14     struct body s1;
15     struct body s2;
```

```

16
17     cout << "輸入第 1 位同學的基本資料：" << endl;
18     cin >> s1.id;
19     cin >> s1.name;
20     cin >> s1.sex;
21     cin >> s1.height;
22     cin >> s1.weight;
23     cout << endl;
24
25     cout << "輸入第 2 位同學的基本資料：" << endl;
26     cin >> s2.id;
27     cin >> s2.name;
28     cin >> s2.sex;
29     cin >> s2.height;
30     cin >> s2.weight;
31     cout << endl;
32
33     cout << "學號：" << s1.id << endl;
34     cout << "姓名：" << s1.name << endl;
35     cout << "性別：" << s1.sex << endl;
36     cout << "BMI：" << s1.weight/pow(s1.height/100, 2) << endl;
37     cout << "---" << endl;
38
39     cout << "學號：" << s2.id << endl;
40     cout << "姓名：" << s2.name << endl;
41     cout << "性別：" << s2.sex << endl;
42     cout << "BMI：" << s2.weight/pow(s2.height/100, 2) << endl;
43 }

```

程式碼說明：

行數	說明
6 ~ 11	建立 body 結構，具備 5 個成員。
14, 15	宣告 body 結構變數，分別為 s1 和 s2。
17 ~ 31	使用 C++ 提供的 cin 物件，由使用者輸入資料並指定給結構成員。
33 ~ 42	使用 C++ 提供的 cout 物件，輸出結構變數 s1 和 s2 的內容。 BMI 的計算公式為 $BMI = \frac{\text{體重(kg)}}{\text{身高(m)}^2}$

•立即練習

- (B) 1. C/C++語言中允許使用者自己定義資料型態，該資料型態可以包含整數、浮點數等基本型態，這種自己定義的新資料型態，稱為：
 (A)陣列 (B)結構 (C)種類 (D)集合。
- (D) 2. 在 C 語言中要建立一個結構(structure)，需使用哪一個關鍵字？
 (A)define (B)array (C)class (D)struct。
- (B) 3. 在 C/C++語言中，要存取結構的成員，要使用哪一個運算子？
 (A)「-」 (B)「.」 (C)「:」 (D)「~」。
- (A) 4. 關於 C/C++語言的結構(structure)型態，下列何者錯誤？
 (A)結構的成員不可以是陣列
 (B)是程式設計人員自行定義的資料型態
 (C)要在程式中建立結構，需使用 struct 關鍵字
 (D)要存取結構變數的成員，需使用「.」運算子。
- (C) 5. 東東使用 C 語言建立一結構 device，程式碼如下，請問宣告一個 device 結構變數，所需的記憶體空間為何？ (A)16Byte (B)4Byte (C)20Byte (D)32Byte。

```
struct device {
    int gps_x;
    int gps_y;
    int gps_z;
    double value;
};
```
- (D) 6. 承上題，要宣告一個 device 結構的結構變數 a，下列何者是正確的語法？
 (A)struct a = device; (B)a = struct(device); (C)device struct a; (D)struct device a;
- (A) 7. 承上題，要指定結構變數 a 的成員 value 為 3.14，下例何者是正確的語法？
 (A)a.value = 3.14; (B)a->value = 3.14 (C)a(value) = 3.14; (D)a::value = 3.14。

答案 打★表示有詳解

1.(B) 2.(D) 3.(B) 4.(A) ★5.(C) 6.(D) 7.(A)

解析

5. device 結構包含 3 個 int(4 Byte)成員，1 個 double(8 Byte)成員。

7-2 結構陣列

重點整理

- 陣列的元素為結構資料型態，稱為結構陣列(Structure Array)。
- 宣告結構陣列的語法如下，

```
struct 結構名稱 陣列名稱[元素個數];
```

```
struct body student[3]; /* 宣告具有 3 個元素的結構陣列 student，每個元素都是一個範例 01 的 body  
結構 */
```

- 使用陣列索引的方式存取結構的成員，

```
student[0].height = 171.5;  
student[1].height = 178.2;  
student[2].height = 166.8;  
  
for (int i=0; i<3; i++){ /* 使用 for 迴圈走訪結構陣列 */  
    cout << "身高：" << student[i].height << endl;  
}
```

02 範例

素養題

在 P7-3 範例 01 中，莎莎幫保健室阿姨設計一支程式，記錄同學的基本資料，但是要為全校每一位同學都宣告一個結構變數，是非常不智的做法，因此莎莎決定使用結構陣列來改寫程式，同時能提高程式的可讀性且使程式更易於維護。

解

程式碼：

```
1 #include <iostream>  
2 #include <cmath>  
3  
4 #define STUDENT_NUM 1000  
5  
6 using namespace std;  
7  
8 struct body{  
9     int id;  
10    char name[10];  
11    char sex;  
12    float height, weight;  
13};
```

```

14
15 int main() {
16     struct body s[STUDENT_NUM];
17
18     for (int i=0; i<STUDENT_NUM; i++){
19         cout << "輸入學號：" << endl;
20         cin >> s[i].id;
21         cin >> s[i].name;
22         cin >> s[i].sex;
23         cin >> s[i].height;
24         cin >> s[i].weight;
25         cout << endl;
26     }
27
28     for (int i=0; i<STUDENT_NUM; i++){
29         cout << "學號：" << s[i].id << endl;
30         cout << "姓名：" << s[i].name << endl;
31         cout << "性別：" << s[i].sex << endl;
32         cout << "BMI：" << s[i].weight/pow(s[i].height/100, 2) << endl;
33         cout << "---" << endl;
34     }
35 }
```

程式碼說明：

行數	說明
4	用前置處理器指令 <code>#define</code> 建立識別字 STUDENT_NUM，在程式內只要出現 STUDENT_NUM，就用 1000 取代。
8 ~ 13	建立 body 結構，具備 5 個成員。
16	宣告一個有 1000 個元素的結構陣列 s，每個元素皆為 body 結構型態。
18 ~ 26	使用 for 迴圈走訪結構陣列 s，將資料存入。
28 ~ 34	使用 for 迴圈走訪結構陣列 s，將資料輸出。 BMI 的計算公式為 $BMI = \frac{\text{體重(kg)}}{\text{身高(m)}^2}$

• 立即練習

(C) 1. C/C++的程式設計人員可以自行建立結構資料型態，並宣告結構變數來儲存資料，下列關於結構的說明，何者是錯誤的？

- (A)結構陣列是多個相同結構型態的變數的集合
- (B)結構指標指向的資料型態是結構
- (C)結構內的成員必須是相同的資料型態
- (D)結構陣列使用索引的方式存取元素。

(B) 2. 亮亮使用 C 語言的結構陣列來記錄好朋友的電話號碼，他寫了一段程式碼如下，請問要存取最後一位朋友的電話，哪一個語法才是正確的？

- 素養題**
- (A) friend[0].num; (B) friend[19].num; (C) phone[0].num; (D) phone[19].num;

```
struct phone{
    int num;
    char name[10];
};

int main() {
    struct phone friend[20];
}
```

(B) 3. 承上題，陣列 friend[]總共需要多大的記憶體空間？

- (A)14 Byte (B)280 Byte (C)140 Byte (D)200 Byte。

(D) 4. 關於下列程式碼片段的描述，何者正確？

- (A)結構 player 擁有 3 個成員
- (B) nba[]是一個結構陣列，每個元素都是一個 player 結構型態
- (C)一個 player 結構的變數會佔用記憶體 64Bbyte 的大小
- (D)以上皆是。

```
struct player{
    char name[30];
    char team[30];
    int pay;
};

int main() {
    struct player nba[100];
    strcpy(nba[0].name,"LeBron James");
    strcpy(nba[0].team,"Lakers");
    nba[0].pay = 3922;
}
```

(B) 5. 下列程式片段的輸出結果為何？ (A)10 (B)9 (C)0 (D)20。

```
struct S{
    int x;
    int y;
};

int main() {
    struct S A[10];
    for (int i=0;i<10;i++){
        A[i].x = i;
        A[i].y = A[i].x;
    }

    cout << A[0].x + A[9].y << endl;
}
```

答案 打★表示有詳解

1.(C) 2.(B) ★3.(B) 4.(D) ★5.(B)

解析

3. phone 結構所需的記憶體大小為 14 Byte，結構陣列 friend[20] 有 20 個元素，每個元素都是 phone 結構，故所需的記憶體大小為 $20 \times 14 \text{ Byte} = 280 \text{ Byte}$ 。
5. 結構陣列 A[] 的每個元素都是 S 結構，使用迴圈走訪 A[]，將每個元素的成員 x 和 y 都指定為陣列索引值，故 $A[0].x + A[9].y = 0 + 9 = 9$ 。

7-3 結構指標

重點整理

1. 如同整數指標指向整數變數，**結構指標會指向結構變數**，宣告結構指標的語法如下，

`struct 結構名稱 *指標名稱;`

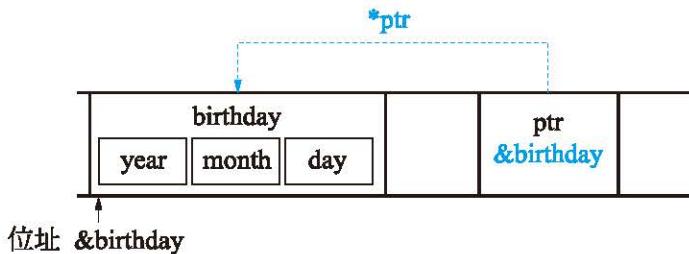
```
struct date{           /* 建立 date 結構，存放年、月、日資料 */
    int year;
    int month;
    int day;
};

struct date birthday; /* 宣告一 date 結構型態的結構變數，名稱為 birthday */
struct date *ptr;    /* 宣告一 date 結構型態的結構指標，名稱為 ptr */
```

2. 使用取址運算子「&」，將結構指標指向結構變數，

```
ptr = &birthday;      /* 結構指標 ptr 指向結構變數 birthday */
```

程式執行後，ptr 與 birthday 在記憶體的關係如下，



3. 存取結構指標指向的結構變數成員，有 2 種方式：

(1) 使用取值運算子「*」

```

(*ptr).year = 2004;
(*ptr).month = 8;
(*ptr).day = 15;

cout << "Your birthday is " << (*ptr).year << "-" << (*ptr).month << "-" << (*ptr).day << endl;
  
```

(2) 使用指標運算子「->」

```

ptr->year = 2004;
ptr->month = 8;
ptr->day = 15;

cout << "Your birthday is " << ptr->year << "-" << ptr->month << "-" << ptr->day << endl;
  
```

4. 在函式中傳遞結構型態的資料，可以採用傳值呼叫或傳址呼叫，

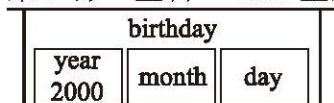
(1) 傳值呼叫，參數是結構變數，不會改變來源結構變數的內容。

```

1. void setDate(struct date);           /* 函式原型宣告，規範參數需為 date 結構型態 */
2.
3. void setDate(struct date d){         /* 函式定義(實作)，參數 d 為結構變數 */
4.     d.year = 2022;
5.     cout << d.year << endl;          /* 輸出 2022 */
6. }
7.
8. int main(){
9.     struct date birthday;
10.    birthday.year = 2000;
11.    setDate(birthday);              /* 呼叫 setDate()，引數為結構變數 birthday */
12.    cout << birthday.year << endl; /* 輸出 2000 */
13. }
  
```

程式執行後，變數在記憶體內的變化如下，

第 9 行，宣告一 date 型態的結構變數 birthday



第 11 行，呼叫函式 setDate(birthday)，採傳值呼叫，等同於執行

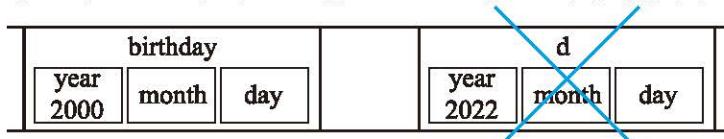
```
struct date d = birthday;
```



第 4 行，修改結構變數 d 的成員，不會改變結構變數 birthday 的內容



第 6 行，函式結束，d 消失，返回第 12 行繼續執行



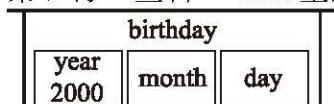
(2) 傳址呼叫，參數是結構指標，指向來源結構變數，會直接改變來源結構變數的內容。

```

1. void setDate(struct date*);           /* 函式原型宣告，規範參數需為結構指標 */
2.
3. void setDate(struct date *ptr){      /* 函式定義(實作)，參數 ptr 為結構指標 */
4.     ptr->year = 2022;
5.     cout << ptr->year << endl;        /* 輸出 2022 */
6. }
7.
8. int main(){
9.     struct date birthday;
10.    birthday.year = 2000;
11.    setDate(&birthday);             /* 呼叫 setDate()，引數為結構變數 birthday 的位址 */
12.    cout << birthday.year << endl; /* 輸出 2022 */
13. }
```

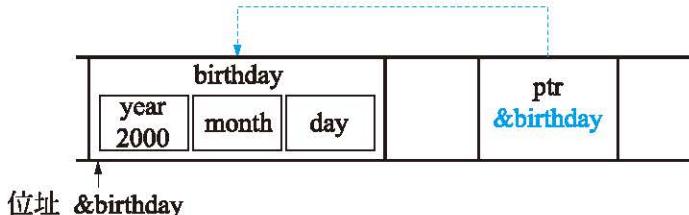
程式執行後，變數在記憶體內的變化如下，

第 9 行，宣告一 date 型態的結構變數 birthday

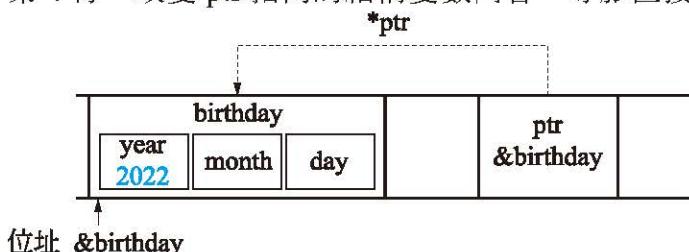


第 11 行，呼叫函式 setDate(&birthday)，採傳址呼叫，等同於執行

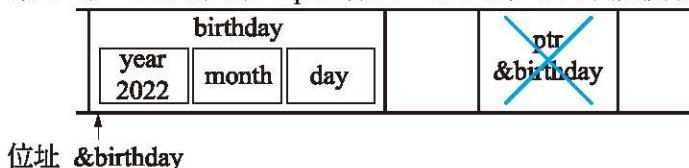
```
struct date *ptr = &birthday;
*ptr
```



第 4 行，改變 ptr 指向的結構變數內容，等於直接改變來源結構變數 birthday 的內容



第 6 行，函式結束，ptr 消失，返回第 12 行繼續執行



03 範例 家裡虎

巧虎爸爸正在學習 C++ 語言的結構，他想到可以建立一個日期結構，用來記錄巧虎的生日，同時使用 2 個函式來設定和顯示生日，巧虎媽媽也提醒爸爸，結構資料在函式中可以用傳值或傳址的方式傳遞，可以嘗試用不同的方式設計(巧虎的生日是 5 月 5 日)。

解

程式碼：

```

1 #include <iostream>
2 using namespace std;
3
4 struct date{ /* 建立 date 結構 */
5     int month;
6     int day;
7 };
8
9 void printDate(struct date);
10 void setDate(struct date*, int, int);
11
12 void printDate(struct date d){
13     cout << "Your birthday is " << d.month << "/" << d.day;

```

```

14 }
15 void setDate(struct date *ptr, int mm, int dd){
16     ptr->month = mm;
17     ptr->day = dd;
18 }
19
20 int main() {
21     struct date birthday;
22     struct date *ptr;
23
24     ptr = &birthday;
25
26     setDate(ptr, 5, 5);
27     printDate(birthday);
}

```

程式碼說明：

行數	說明
4~7	建立 date 結構，有 2 個成員。
9	宣告 printDate()函式原型，規範參數必須是 date 結構型態。
10	宣告 setDate()函式原型，規範 3 個參數分別是指向 date 結構的結構指標和 2 個整數。
12~14	實作 printDate()函式。
15~18	實作 setDate()函式，參數 ptr 是指向 date 結構的結構指標，使用「->」運算子存取結構成員，會直接改變來源結構的成員。
21	宣告一 date 結構型態的結構變數 birthday。
22	宣告一指向 date 結構型態的結構指標 ptr。
24	使結構指標 ptr 指向結構變數 birthday。
26	呼叫 setDate()函式，傳入的引數為結構指標 ptr(brithday 結構變數的位址)和 2 個整數。
27	呼叫 printDate()函式，傳入的引數為 brithday 結構變數。

•立即練習

- (B) 1. 小健宣告一個結構指標 p 並指向一自訂結構 s，要使用該指標存取結構的成員，需使用哪一個運算子？ (A) 「.」 (B) 「->」 (C) 「:」 (D) 「-」。

(D) 2. 下列有關 C 語言運算子的說明，哪一個選項不正確？

- (A) 「`&&`」是邏輯運算子
- (B) 「`.`」是結構成員運算子，用來存取結構的成員
- (C) 結構指標經由「`->`」運算子存取結構成員
- (D) 「`&`」是取值運算子。

(B) 3. 小新在 C++ 語言中宣告一個結構變數 `data` 和結構指標 `pData`，要使 `pData` 指向 `data`，需使用下列哪一個語法？

- (A) `pData = data;`
- (B) `pData = &data;`
- (C) `pData = *data`
- (D) `pData->data`。

(B) 4. 承上題，結構變數 `data` 有一個成員 `id`，下列哪一個語法，無法將 `id` 設定為 100？

- | | |
|-------------------------------------|--------------------------------------|
| (A) <code>(*pData).id = 100;</code> | (B) <code>pData.id = 100;</code> |
| (C) <code>data.id = 100;</code> | (D) <code>pData->id = 100;</code> |

(D) 5. 一程式片段如下，下列何者的值與其它 3 個不同？

- (A) `(*p).x`
- (B) `box.x`
- (C) `p->x`
- (D) `p.x`。

```
struct device{ /* 建立 device 結構 */
    int x;
    int y;
};
struct device box;
struct device *p;
p = &box;
```

(C) 6. 下列程式片段執行後的輸出為何？ (A)20 10 (B)10 10 (C)10 20 (D)20 20。

```
struct ST{
    int a;
    int b;
};

void F(struct ST m){
    int tmp;
    tmp = m.a;
    m.a = m.b;
    m.b = tmp;
}

int main() {
    struct ST m;

    m.a = 10;
    m.b = 20;

    F(m);
    printf("%d %d", m.a, m.b);
}
```

(A) 7. 下列程式片段執行後的輸出為何？ (A)20 10 (B)10 10 (C)10 20 (D)20 20。

```

struct ST{
    int a;
    int b;
};

void F(struct ST *p){
    int tmp;
    tmp = p->a;
    p->a = p->b;
    p->b = tmp;
}

int main() {
    struct ST m, *p;
    p = &m;

    m.a = 10;
    m.b = 20;

    F(p);
    printf("%d %d", m.a, m.b);
}

```

答案 打★表示有詳解

1.(B) ★2.(D) 3.(B) ★4.(B) ★5.(D) ★6.(C) 7.(A)

解析

2. 「&」是取址運算子。
4. pData 是指標，不可使用「.」運算子存取結構成員。
5. 結構指標變數 p 指向結構變數 box，選項(A)(B)(C)都會取得結構變數的成員 x 的值，選項(D)的 p 為結構指標變數，不可以使用「.」運算子存取指向的結構成員。
6. 程式執行的過程如下圖，
 (1) 定義一結構 ST。
 (2) 定義一函式 F(struct ST m)，參數 m 為結構變數，呼叫該函式時，傳入的引數必須也是結構變數。
 (3) 告 ST 結構變數 m，使用「.」運算子存取結構成員。
 (4) 呼叫 F(m)，屬於傳值呼叫，此時參數 m 可視為來源結構變數 m 的複製品(兩者皆為區域變數，允許同名)。
 (5) 將參數 m 的成員 a 和 b 互換，不會改變來源結構變數 m 的成員。
 (6) 函式結束，返回原程式位置。
 (7) 結構變數 m 的成員 a 和 b 不變，輸出 10 和 20。

```

(1) struct ST{ ...
    int a;
    int b;
};

(2) void F(struct ST m){ ...
    int tmp;
    tmp = m.a;
    m.a = m.b;
    m.b = tmp;
}

(3) int main(){
    struct ST m;
    m.a = 10;
    m.a = 20;

    (4) F(m);
    printf("%d %d",m.a, m.b);
}

```

7. 程式執行的過程如下，

```

(1) struct ST{ ...
    int a;
    int b;
};

(2) void F(struct ST *p){ ...
    int tmp;
    tmp = p->a;
    p->a = p->b;
    p->b = tmp;
}

(3) int main(){
    struct ST m, *p;
    p = &m;
    m.a = 10;
    m.a = 20;

    (4) F(p);
    printf("%d %d",m.a, m.b);
}

```

- (1) 定義一結構 ST。
- (2) 定義一函式 F(struct ST *p)，參數 p 為結構指標變數，呼叫該函式時，傳入的引數必須是結構變數的位址。
- (3) 嘗試宣告結構變數 m 和結構指標變數 p，使 p 指向 m。
- (4) 呼叫 F(p)，屬於傳址呼叫，此時參數 p 也會指向結構變數 m (參數 p 和結構指標變數 p 皆為區域變數，允許同名)。
- (5) 參數 p 使用指標運算子「->」將指向的結構變數成員 a 和 b 互換，會直接改變來源結構變數 m 的成員。
- (6) 函式結束，返回原程式位置。
- (7) 結構變數 m 的成員 a 和 b 已經被互換，輸出 20 和 10。

7-4 類別

重點整理

1. C 語言是程序導向語言，C++是物件導向語言。
2. 物件導向(Object-Oriented)語言，使用類別(Class)定義資料的屬性與操作資料的方法，物件(Object)則為類別的實作。
3. C++語言宣告類別的語法(使用 **class** 關鍵字)，

```
class 類別名稱{
    private:
        資料型態 資料成員;
        回傳值資料型態 成員函式(參數資料型態...);
    public:
        資料型態 資料成員;
        回傳值資料型態 成員函式(參數資料型態...);
};
```

類別的成員可以是資料或函式，

- 資料成員(data member)：即屬於該類別的變數，用來描述物件的屬性。
- 成員函式(member function)：即操作資料的方法，用來存取資料成員或進行運算。

而資料成員或成員函式所屬的區塊(private 或 public)，決定該成員的可存取範圍(權限)：

- **private** 區塊：成員只能在該類別內部被存取，此區塊內的成員稱為**私有成員**，可視為被封裝起來的隱藏性資料。
- **public** 區塊：成員可以在程式的任何位置被存取，此區塊內的成員稱為**公有成員**，一般做為存取資料成員的介面。

```
/* 建立 date 類別 */
class date{
    /* 私有成員區塊 */
    private:
        /* 資料成員 */
        int month;
        int day;

    /* 成員函式，只宣告函式原型 */
    int getMonth();
    int getDay();

    /* 公有成員區塊 */
    public:
```

```

/* 資料成員 */
int id;

/* 成員函式，只宣告函式原型 */
void setDate(int, int);
void printDate();

};

```

4. 上例的 date 類別，只有宣告成員函式的函式原型，因此需在類別外部實作成員函式，語法如下，

回傳值資料型態 類別名稱::成員函式(參數資料型態 參數名稱, ...){ 程式碼區塊 }

```

int date::getMonth(){
    return month;
}

int date::getDay(){
    return day;
}

void date::setDate(int m, int d){
    month = m;
    day = d;
}

void date::printDate(){
    cout << month << "/" << day << endl;
}

```

註 「::」為範圍運算子。

5. 若在類別宣告時直接實作成員函式，則可以省略函式原型的宣告，

```

/* 建立 date 類別 */
class date{
    /* 私有成員區塊 */
private:
    /* 資料成員 */
    int month;
    int day;

    /* 不宣告函式原型，直接實作成員函式 */
    int getMonth(){
        return month;
    }
}

```

```

    }

    int getDay(){
        return day;
    }

/* 公有成員區塊 */
public:
    int id;

/* 不宣告函式原型，直接實作成員函式 */
void setDate(int m, int d){
    month = m;
    day = d;
}

void printDate(){
    cout << month << "/" << day << endl;
}

};


```

6. 完成類別宣告後，即可建立該類別的物件，宣告物件的語法如下，

類別名稱 物件名稱:

```
date birthday; /* 建立 birthday 物件，為 date 類別的實作 */
```

7. 每個類別可以建立多個物件，這些物件皆為該類別的實作，稱為**實例(instances)**。
8. 物件可以直接存取公有成員，但私有成員無法直接存取，需透過公有成員函式，例如要存取 month 或 day 等私有資料成員，需透過 setDate()或 printDate()。

```

birthday.id = 1;           /* 存取公有成員 id */
birthday.setDate(10, 10);   /* 透過公有成員 setDate()，間接存取私有成員 month 和 day */
birthday.printDate();

```

9. 「**this**」關鍵字，表示指向物件本身的指標，可以透過指標運算子「->」存取自身成員，但只能在類別內部使用，

```

void date::setDate(int m, int d){
    this->month = m;
    this->day = d;
}

```

10. 類別的成員函式可以**重載(overload)**，下例為 3 個重載的成員函式：

```

void setDate(int, int, int);
void setDate(int);
void setDate(double);

```

11. **建構子(Constructor)**：或稱為建構函式，在宣告物件時，會自動呼叫建構子，建構子會對物件進行初始化。

12. 建構子的特性如下：

- (1) 建構子需與類別同名。
- (2) 建構子沒有回傳值，也不需加上 void 關鍵字。
- (3) 一個類別可以擁有多個建構子，即建構子可以重載。

13. **解構子(Destructor)**：又稱為解構函式，當物件消滅時，會自動呼叫解構子，解構子執行物件結束時要進行的動作，如釋放記憶體或關閉檔案等。

14. 解構子的特性如下：

- (1) 解構子需與類別同名，且名稱前面要加上「~」符號。
- (2) 解構子沒有回傳值，也不需加上 void 關鍵字。
- (3) 解構子只能有一個（不可重載），且不可傳入任何參數。

15. 下例中的 date 類別，有 2 個重載的建構子和 1 個解構子

```
/* 建立 date 類別 */
class date{
    /* 私有成員區塊 */
    private:
        :

    /* 公有成員區塊 */
    public:
        :
        date();           /* 建構子一的原型宣告 */
        date(int, int);  /* 建構子二的原型宣告 */
        ~date();          /* 解構子的原型宣告 */
        :
};

date::date() {           /* 建構子一的實作 */
    month = 1;
    day = 1;
}

date::date(int m, int d){ /* 建構子二的實作 */
    month = m;
    day = d;
}
```

```

date::~date() { /* 解構子的實作 */
    cout << "Goodbye" << endl;
}

date day1; /* 告知物件 day1，自動呼叫建構子一 */
date day2(12, 31); /* 告知物件 day2，自動呼叫建構子二 */

```

16. 類別的**繼承(inheritance)**：被繼承的類別稱為父類別，而繼承的類別稱為子類別，子類別會擁有父類別的公有成員，語法如下：

```

class 子類別 : public 父類別{
    :
};

```

下例中的 policeCar 類別繼承 car 類別，所以 policeCar 類別會擁有 car 類別的成員函式 power()，另外再新增一成員函式 flash()，

```

/* 告知 car 類別 */
class car{
private:
    int id;
public:
    void power(){cout << "啓動";}
};

/* 告知 policeCar 類別，繼承 car 類別 */
class policeCar : public car{
public:
    void flash(){cout << "閃燈";}
};

int main(){
    policeCar pCar; /* 建立 pCar 物件 */
    pCar.flash(); /* flash()是 policeCar 類別的成員函式 */
    pCar.power(); /* power()是 policeCar 從 car 類別繼承的成員函式 */
}

```

註 使用「:」表示子類別繼承的父類別。

17. 子類別可以繼承多個父類別，子類別擁所有父類別的公有成員，稱為多重繼承。

04 範例

素養題

巧虎終於要有妹妹了，巧虎爸爸很高興小花妹妹要出生，為此他打算改寫原來的程式，原程式使用結構來記錄巧虎的生日，參考範例 3，若改用 C++ 的物件導向功能來設計，就可以使用類別來定義資料的屬性和操作資料的方法，巧虎媽媽很支持爸爸的想法，主動幫忙規劃好 date 類別，請幫忙他們完成新版的程式吧！(巧虎的生日是 5 月 5 日，小花的生日是 4 月 4 日)

date 類別規劃：

私有成員	說明
month	月
day	日
公有成員	
setDate(int, int)	設定日期
printDate()	顯示日期

解

程式碼：

```

1 #include <iostream>
2 using namespace std;
3
4 class date{
5     private:
6         int month;
7         int day;
8
9     public:
10        void setDate(int, int);
11        void printDate();
12    };
13
14 void date::setDate(int mm, int dd){
15     month = mm;
16     day = dd;
17 }
18
19 void date::printDate(){
20     cout << "Your birthdat is " << month << "/" << day << endl;
21 }
22
23 int main() {
24     date birthday1;

```

```

25     date birthday2;
26
27     birthday1.setDate(5, 5);
28     birthday2.setDate(4, 4);
29
30     birthday1.printDate();
31     birthday2.printDate();
32 }
```

程式碼說明：

行數	說明
4 ~ 12	宣告 date 類別，成員如下： - 2 個私有資料成員 month 和 day - 2 個公有成員函式 setDate()和 printDate()
14 ~ 17	setDate()函式實作。
19 ~ 21	printDate()函式實作。
24, 25	建立物件 birthday1 和 birthday2。
27 ~ 31	透過 setDate()和 printDate()成員函式，存取私有資料成員 month 和 day。

05 範例

類別題

酷巴克咖啡店的老闆利用賣咖啡的空閒時間，自修 C++ 程式語言，在學會物件導向的觀念後，他想將店內的咖啡產品物件化，並規劃一個 cafe 類別如下，請協助他完成程式後，產生兩杯咖啡物件，驗證程式執行結果。

cafe 類別規劃：

私有成員	說明
price	價格
milk	鮮奶量(%)
suger	甜度(0 ~ 5 分)
setPrice(int)	設定價格
setMilk(int)	設定鮮奶量(0 ~ 100)
setSuger(int)	設定甜度(0 ~ 5)
int getPrice()	取得價格
int getMilk()	取得鮮奶量
int getSuger()	取得甜度

公有成員	
discount(float)	設定折扣數
show()	顯示咖啡成份(奶量、甜度、價格)
建構子	
cafe()	預設建構子
cafe(int int)	建構子重載，初始化鮮奶量和甜度

解

程式碼：

```

1 #include <iostream>
2 using namespace std;
3
4 class cafe{
5     private:
6         int price;
7         int milk;
8         int suger;
9
10    void setPrice(int price) {
11        this->price = price;
12    }
13    void setMilk(int milk){
14        this->milk = milk;
15    }
16    void setSuger(int suger){
17        this->suger = suger;
18    }
19
20    int getPrice(){
21        return this->price;
22    }
23    int getMilk(){
24        return this->milk;
25    }
26    int getSuger(){
27        return this->suger;
28    }
29
30 public:
```

```
31     cafe();                  /* 建構子一 */
32     cafe(int, int);          /* 建構子二 */
33     void discount(float);
34     void show();
35 }
36
37 cafe::cafe(){
38     setPrice(100);
39     setMilk(0);
40     setSuger(0);
41 }
42
43 cafe::cafe(int m, int s){
44     setPrice(100);
45     setMilk(m);
46     setSuger(s);
47 }
48
49 void cafe::discount(float d){ /* 依折扣重新計算價格 */
50     setPrice(getPrice()*d);
51 }
52
53 void cafe::show(){
54     cout << "奶量：" << getMilk() << "%" << endl;
55     cout << "甜度：" << getSuger() << "分糖" << endl;
56     cout << "價格：" << getPrice() << endl;
57     cout << "-----" << endl;
58 }
59
60 int main() {
61     cafe cup1;
62     cup1.show();
63
64     cafe cup2(40, 2);
65     cup2.discount(0.9);      /* 呼叫 discount 函式，打 9 折 */
66     cup2.show();
67 }
68
```

程式碼說明：

行數	說明
4~35	依據類別規劃，宣告 cafe 類別，並直接實作私有成員函式，私有資料成員(price、milk、suger)均需透過私有成員函式(setPrice()、setMilk()、setSuger())存取。
37 ~ 47	實作 2 個重載建構子。
49 ~ 58	實作公有成員函式。
61	建立物件 cup1，自動呼叫建構子 cup()。
64	建立物件 cup2，自動呼叫建構子 cup(int int)。

• 立即練習

- (B) 1. 下列哪一種程式語言，不是物件導向語言？
 (A)C++ (B)C (C)JAVA (D)Visual Basic。
- (A) 2. 關於 C++ 語言的類別建構子，下列哪一個說法是錯誤的？
 (A)建構子的名稱和類別名稱不可以相同 (B)建構子支援重載
 (C)建構子沒有回傳值 (D)一個類別可以有多個建構子。
- (B) 3. 關於 C++ 語言的類別，哪一個選項的描述是正確的？
 (A)將資料成員放在 public 區塊，可以達到物件導向的資料封裝特性
 (B)建構子可以重載
 (C)解構子可以重載
 (D)建構子的名稱需與類別名稱相同，且需加上「~」符號。
- (C) 4. 在 C++ 語言中要建立類別，需使用哪一個關鍵字？
 (A)define (B)array (C)class (D)struct。
- (B) 5. 在 C++ 語言中，要呼叫類別的成員函式，要使用哪一個運算子？
 (A)「-」 (B)「.」 (C)「:」 (D)「~」。
- (D) 6. C++ 語言允許函式重載，即函式的名稱相同，下列哪個選項內的函式，不是合法的重載函式？
- | | |
|------------------------------------|---------------------------------------|
| (A) void A(int);
void A(float); | (B) void A(int);
void A(int, int); |
| (C) int A(float);
float A(int); | (D) void A(int);
float A(int); |

- (A) 7. 志遠宣告一個 C++ 的類別(class)如下，請問哪一個成員屬於私有資料成員，只能在物件內部存取？ (A)x (B)y (C)x 和 y 皆是 (D)x 和 y 皆不是。

```
class S{
    private:
        int x;
    public:
        int y;
}
```

- (D) 8. 小麗使用 C++ 語言建立一個類別 star，下列哪一個函式不可能是該類別的建構子？

素養題 (A)star(); (B)star(int); (C)star(int, int); (D)以上皆是。

- (C) 9. 下列關於建構子之敘述，何者為非？

- (A)建構子也稱為建構函式，是物件的初始函式
- (B)建構子支援重載(overload)
- (C)建構子沒有傳回值，需加上 void
- (D)建構子與類別同名。

- (B) 10. 一程式片段如下，請問該程式執行後的輸出為何？

- (A)HippoLion (B)MonkeyLion (C)BearLion (D)LionLion。

```
class Animal{
public:
    ~Animal(){
        cout << "Lion";
    }
    Animal(){
        cout << "Bear";
    }
    Animal(int x){
        cout << "Monkey";
    }
    Animal(double x){
        cout << "Hippo";
    }
};
int main(void)
{
    Animal a(1);
}
```

答案 打★表示有詳解

1.(B) 2.(A) ★3.(B) 4.(C) 5.(B) ★6.(D) ★7.(A) 8.(D) ★9.(C) ★10.(B)

解析

3. (A)private 區塊 (C)解構子不可重載 (D)解構子。
6. 僅回傳值的型態不同，不是重載函式。
7. x 是私有成員，y 是公有成員。
9. (C)建構子沒有傳回值，也不需要加上 void。
10. 在 main()中宣告物件 a(1)時，會自動呼叫建構子 Animal(int x)，輸出"Monkey"，程式結束時會自動呼叫解構子~Animal()，輸出"Lion"。

精選試題

- (D) 1. 關於 C++ 語言的結構(struct)和類別(class)，下列哪一個敘述正確？
- (A)C++是物件導向語言，使用類別來定義資料和操作資料的方法
 (B)結構是使用者自己建立的資料型態，包含多個成員
 (C)一個類別可以有多個物件的實作
 (D)以上皆是。
- (B) 2. 有關 C++ 語言的類別描述，下列何者錯誤？
- (A)類別可以實現 C++ 物件導向的特性 (B)一個類別只能產生一個物件實例
 (C)類別的建構子可以重載 (D)類別的解構子只能有一個。
- (C) 3. 有關 C++ 語言的物件導向功能，下列何者正確？
- (A)C 語言繼承 C++ 語言，所以 C 語言也是物件導向語言
 (B)使用「::」表示類別之間的繼承關係
 (C)一個子類別可以繼承多個父類別
 (D)類別就是物件的實作。
- (C) 4. 一程式片段如下，關於程式碼的說明，何者錯誤？
- (A)I 類別繼承 J 類別 (B)I 類別會擁有 J 類別的公有成員
 (C)J 類別擁有 beep() 成員函式 (D)I 和 J 是物件導向的繼承關係。
- ```
class I : public J{
public:
 void beep();
```
- (B) 5. 小雪在 C/C++ 語言中宣告一個結構 box，程式碼如下，請問該結構會佔用多少的記憶體空間？ (A)3Byte (B)12Byte (C)24Byte (D)48Byte。
- ```
struct box{
    int height, length, width;
};
```