

程式設計 (一)

CH10. 結構與位元處理

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Last Semester, 2021

本章目錄

1. struct (結構)
2. typedef
3. union
4. 位元運算子
5. 位元欄位
6. enum (列舉型別常數)

struct (結構)

struct (結構)

簡介

結構 (structure) 會將一些彼此相關的變數結合成相同的名稱，
可以含有許多不同資料型別的變數。

結構的主要用途

- 使有關聯的資料結合在一起，方便管理與傳值。
- 定義儲存在二進制檔案裡的紀錄 (record)。
- 指標和結構可以構成複雜的資料結構，例如鏈結串列、佇列、堆疊、樹…等。
(會在第 15 章介紹，並且為二年級上學期〈資料結構〉的重要課題。)

定義 struct

struct(結構, structure) 是一種衍生的資料型別 (derived data type), 它是以其他型別的物件來建構的。

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
}
```

上方為定義一個 struct 型別的範例。
注意在大括號的後方有一個分號 (;)。

struct (結構)

關於 struct student 定義的一些名稱：

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
}
```

- 結構標籤 (structure tag): student
- 結構型別 (structure type): struct student
- 成員 (member): name、sex、grade、GPA

宣告 struct 型別的變數

定義 struct 與宣告 struct 變數可以寫在一起，也可以分開寫。

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
} classmates[100], studentA;
```

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
};  
struct student classmates[100], studentA;
```

初始值設定 (1/2)

struct 可以像陣列一樣使用初始值串列，
初始值會照定義 struct 時的成員順序來設定。

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
};  
struct student studentA = {"Jason", 'M', 3, {3.8, 3.6, 4.0, 4.0}};
```


初始值設定 (2/2)

struct 也可以指定成員來設定初始值，
注意初始值串列中的成員名稱前面要加上點 (.)。

```
struct student{  
    char name[40];  
    char sex;  
    int grade;  
    float GPA[8];  
};  
struct student studentA = { .name = "Jason",  
                             .sex = 'M',  
                             .GPA = {3.8, 3.6, 4.0, 4.0} };
```

struct 型別變數的運算

- struct 型別變數只可以做
 - 同樣 struct 型別的變數互相指派 (賦值)。
 - 取得結構變數的位址 (&).
 - 存取結構變數成員。
 - 使用 sizeof 運算子來計算結構變數的大小。
- struct 型別變數常見的錯誤
 - 未取得成員，直接做加減乘除、取餘數。
 - 未取得成員，直接使用關係運算子比較。

存取成員

存取 struct 變數的成員需要使用結構成員運算子 (.)。

```
1 // struct
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct student {
6     char name[40];
7     char sex;
8 };
9
10 int main() {
11     struct student studentA;
12     studentA.sex = 'M';
13     strcpy(studentA.name, "Jason");
14     printf("%s sex: %c\n", studentA.name, studentA.sex);
15 }
```

D:\codeblocks\struct.exe

Jason sex: M

使用 struct 作為函式參數與回傳值

使用 struct 變數作為函式參數屬於傳值呼叫

```
1 | // struct & function
2 | #include <stdio.h>
3 | #include <string.h>
4 |
5 | struct student {
6 |     char name[40];
7 |     char sex;
8 |     int grade;
9 | };
10 |
11 | struct student setStudent(char name[], char sex, int grade);
12 | struct student addGrade(struct student s, int n);
13 | //下一頁繼續
```

struct (結構)

```
15 int main() {  
16     struct student classmate[5];  
17     char names[5][40] = {"Jason", "Kevin", "Jim", "Lisa", "Klaudia"};  
18     char sexes[5] = {'M', 'M', 'M', 'F', 'F'};  
19     int grades[5] = {1, 2, 1, 3, 3};  
20     for(int i = 0; i < 5; i++) //將3個陣列的資料填到陣列classmate中  
21         classmate[i] = setStudent(names[i], sexes[i], grades[i]);  
22     for(int i = 0; i < 5; i++) //將陣列classmate中所有元素的成員grade加1  
23         classmate[i] = addGrade(classmate[i], 1);  
24     for(int i = 0; i < 5; i++) //將陣列classmate的資料印出  
25         printf("%s\tsex: %c\tgrade: %d\n",  
26             classmate[i].name, classmate[i].sex, classmate[i].grade);  
27 }  
28 //下一頁繼續
```

"D:\codeblocks\struct & function.exe"

```
Jason sex: M grade: 2  
Kevin sex: M grade: 3  
Jim sex: M grade: 2  
Lisa sex: F grade: 4  
Klaudia sex: F grade: 4
```

struct (結構)

```
30 struct student setStudent(char name[], char sex, int grade) {  
31     struct student set = {.sex = sex, .grade = grade};  
32     strcpy(set.name, name);  
33     return set;  
34 }  
35  
36 struct student addGrade(struct student s, int n ) {  
37     s.grade += n;  
38     return s;  
39 }
```

程式碼第 31 行，初始化 struct 變數時，若成員為陣列，則不可以直接將其他陣列的名稱當作初始化的數值。

typedef

關鍵字 typedef 可以為資料型態建立別名。
在關鍵字 typedef 之後加上宣告敘述句，該宣告敘述句中，
原本是變數名稱的識別字 會被定義為該變數型態的別名。
之後就可以用此別名來宣告該型態的變數。

例 1：建立 LL 作為 long long int 的別名

```
typedef long long int LL;
```

例 2：建立 Int8 作為大小為 8 的 int 陣列的別名

```
typedef int Int8[8];
```


定義 struct 與建立別名可以寫在一起，也可以分開寫。
在建立完別名之後，就可以使用較短的类型名稱來做宣告。

```
struct student {  
    char name[40];  
    char sex;  
    int grade;  
};  
typedef struct student Student;  
  
Student studentA = {"Jason", 'M', 1};
```

```
typedef struct student {  
    char name[40];  
    char sex;  
    int grade;  
} Student;  
  
Student studentA = {"Jason", 'M', 1};
```

甚至，使用 typedef 定義 struct 型別可以連同結構標籤都省略。

```
typedef struct {  
    char name[40];  
    char sex;  
    int grade;  
} Student;  
  
Student studentA = {"Jason", 'M', 1};
```

將 typedef 的名稱的第一個字母大寫，表示這個名稱
是自行建立的型別名稱。

union

union 和結構很類似，是一種衍生的資料型別，但它的成員會共用相同的儲存空間。

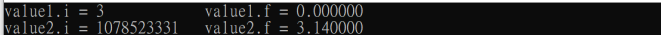
在一些情況下，有些變數可能彼此有關聯，卻不會同時使用到，我們可以把這些有關聯的變數建立成一個 union，以免非使用中的變數浪費了儲存空間。

union 的宣告具有和 struct 宣告相同的格式，
也都可以使用 typedef。

```
union number {  
    int i;  
    float f;  
};  
  
typedef union {  
    int i;  
    float f;  
} Number;
```

在宣告時，union 的初始值預設初始化給第一個成員。

```
1 // union
2 #include <stdio.h>
3
4 typedef union {
5     int i;
6     float f;
7 } Number;
8
9 int main() {
10     Number value1 = {3.14};
11     Number value2 = {.f = 3.14};
12     printf("value1.i = %-10d\tvalue1.f = %f\n", value1.i, value1.f);
13     printf("value2.i = %-10d\tvalue2.f = %f\n", value2.i, value2.f);
14 }
15
```



```
value1.i = 3          value1.f = 0.000000
value2.i = 1078523331 value2.f = 3.140000
```

union 內的成員，使用的記憶體空間是共通的。

```
1 // union
2 #include <stdio.h>
3
4 typedef union {
5     int i;
6     float f;
7 } Number;
8
9 int main() {
10     Number value;
11     value.i = 1000000000;
12     printf("value.i = %d\tvalue.f = %f\n", value.i, value.f);
13     value.f = 1000000000;
14     printf("value.i = %d\tvalue.f = %f\n", value.i, value.f);
15 }
16
```

D:\codeblocks\union.exe

value.i = 1000000000	value.f = 0.004724
value.i = 1315859240	value.f = 1000000000.000000

位元運算子

位元運算子說明 (1/2)

- 位元 AND (&)
若兩運算元同個位置的位元 (bit) 都為 1，則運算結果的同位置位元為 1，否則為 0。
- 位元 OR (|)
若兩運算元同個位置的位元至少有一位元為 1，則運算結果的同位置位元為 1，否則為 0。
- 位元 XOR (^)
若兩運算元同個位置的位元只有一為 1，則運算結果的同位置位元為 1，否則為 0。

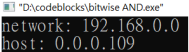
位元運算子說明 (2/2)

- 左移 (\ll), 例如 $x \ll n$
將運算元 x 往右平移 n 個位元 (bit), 右邊以 0 填滿。
- 右移 (\gg), 例如 $x \gg n$
將運算元 x 往左平移 n 個位元 (bit), 左邊填滿方式取決於 x 為有號數 (算數位移) 或無號數 (邏輯位移)。
- 1 補數 (\sim), 例如 $\sim x$
所有為 0 的位元數都設定為 1, 所有為 1 的位元數都設定為 0。

位元運算子

使用位元 AND 與位元遮罩，取得想要的一段位元

```
1 // bitwise AND
2 #include <stdio.h>
3
4 typedef union {
5     unsigned char bytes[4];
6     unsigned int i;
7 } IPv4;
8
9 int main() {
10     IPv4 myAddress = {{192, 168, 0, 109}};
11     IPv4 mask = {{255, 255, 255, 0}};
12     IPv4 network, host;
13     network.i = myAddress.i & mask.i;
14     host.i = myAddress.i & ~mask.i;
15     printf("network: ");
16     for (int i = 0; i < 4; i++)
17         printf("%hhu%c", network.bytes[i], i < 3 ? '.' : '\n');
18     printf("host: ");
19     for (int i = 0; i < 4; i++)
20         printf("%hhu%c", host.bytes[i], i < 3 ? '.' : '\n');
21 }
```



Terminal output for "D:\codeblocks\bitwise AND.exe":
network: 192.168.0.0
host: 0.0.0.109

平移範例

```
1 // shift
2 #include <stdio.h>
3
4 typedef union {
5     unsigned int i;
6     unsigned char bytes[4];
7 } Number;
8
9 void printBytes(char text[], Number n) {
10     puts(text);
11     for (int i = 3; i >= 0; i--)
12         printf("%02hhX%s",
13             n.bytes[i],
14             i > 0 ? " " : "\n\n");
15 }
16
```

```
17 int main() {
18     Number value = {0X00FC00FC};
19     printBytes("Original", value);
20     value.i = value.i << 4;
21     printBytes("Left shift 4 bits", value);
22     value.i = value.i >> 8;
23     printBytes("Right shift 8 bits", value);
24 }
```

D:\codeblocks\shift.exe

Original
00 FC 00 FC

Left shift 4 bits
0F C0 0F C0

Right shift 8 bits
00 0F C0 0F

位元欄位 (bit field)

位元欄位

我們可以在 struct 或 union 的 unsigned int 或 int 成員裡指定儲存的位元數量。這種功能稱為位元欄位 (bit field)。
位元欄位可以將資料存放在最少的位元裡，提高記憶體的使用率。

位元欄位

在 unsigned int 或 int 成員名稱 (member name) 之後加上冒號 (:) 以及一個欄位寬度 (width, 單位為 bit) 的整數常數。

```
typedef struct {  
    unsigned int face : 4; //Ace = 1, ..., King = 13  
    unsigned int suit : 2; // ♦ = 0, ♥ = 1, ♣ = 2, ♠ = 3  
    unsigned int coloe : 1; //紅色 = 0, 黑色 = 1  
} Card;
```

位元欄位的成員類別不可以是 unsigned int 或 int 以外的型態，包括 int 陣列也是不被允許的。

位元欄位

我們可以使用不具名稱的位元欄位作為填補欄位
(沒有資料會被放在此欄位上)。

```
1 typedef struct {  
2     unsigned int face : 4;  
3     unsigned int suit : 2;  
4     unsigned int : 25;  
5     unsigned int color : 1; //於這個單元的末端  
6 } Card;
```

D:\codeblocks\struct.exe

size of Card = 4

```
1 typedef struct {  
2     unsigned int face : 4;  
3     unsigned int suit : 2;  
4     unsigned int : 26;  
5     unsigned int color : 1; //於下個單元的邊界  
6 } Card;
```

D:\codeblocks\struct.exe

size of Card = 8

※ 位元欄位以 4 個位元組 (bytes) 為一個單元。

位元欄位

或是使用不具名稱的 0 位元欄位表示將下一個儲存單元調整到下一個單元的邊界上。

```
1 typedef struct {  
2     unsigned int face : 4;  
3     unsigned int suit : 2;  
4     unsigned int : 0; //將下一個成員調整到新的儲存單元的邊界  
5     unsigned int color : 1; //於下個單元的邊界  
6 } Card;
```

D:\codeblocks\struct.exe

size of Card = 8

位元欄位

關於位元欄位

- 位元欄位成員的存取方式跟一般結構成員的存取方式相同。
- 位元欄位成員的值域是由它的欄位寬度決定，例如欄位寬度為 4 的 unsigned int 成員，其值域為 0~15。
- 位元欄位成員沒有記憶體位址，故無法使用 & 運算子。

enum (列舉型別常數)

enum (列舉型別常數)

enum (列舉型別常數)

列舉由關鍵字 `enum` 定義，它是一組由識別字所代表的整數列舉常數 (enumeration constant)。

除非特別指定，否則 `enum` 內的值都由 0 開始，然後逐漸遞增 1。

enum (列舉型別常數)

下面是 enum 的定義範例，其內部的識別字分別會設定為整數 0~6。

```
enum days {  
    SUN, MON, TUE, WED, THU, FRI, SET  
};
```

enum 中的識別字如同 const int 一樣，是無法修改其值的常數，故使用大寫英文來命名。

enum (列舉型別常數)

跟 struct 一樣，enum 也能使用 typedef。

```
typedef enum {  
    SUN, MON, TUE, WED, THU, FRI, SET  
} Days;
```

enum (列舉型別常數)

我們可將 enum 中的識別字指定數值，而接下來的數值就會逐次遞增 1。

```
typedef enum {  
    JAN = 1, FEB, MAR, APR, MAY, JUN,  
    JUL, AUG, SEP, OCT, NOV, DEC  
} Months;
```


enum (列舉型別常數)

enum 的識別字常被當作符號使用，有時甚至不會在意其內部數值。

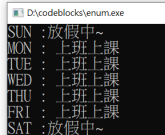
```
//自定義boolean
typedef enum {
    FALSE, TRUE
} Boolean;

//設計遊戲時也許會用到的flag
typedef enum {
    CONTINUE, WIN, LOSE
} Status;
```

enum (列舉型別常數)

使用 enum 的範例

```
1 // enum
2 #include <stdio.h>
3
4 typedef enum {
5     SUN, MON, TUE, WED, THU, FRI, SAT
6 } Days;
7
8 int main() {
9     char week[][4] = {"SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT"};
10    for (Days today = SUN; today <= SAT; today++) {
11        if (today == SUN || today == SAT)
12            printf("%s :放假中~\n", week[today]);
13        else
14            printf("%s : 上班上課\n", week[today]);
15    }
16 }
```



D:\codeblocks\enum.exe

SUN :放假中~
MON : 上班上課
TUE : 上班上課
WED : 上班上課
THU : 上班上課
FRI : 上班上課
SAT :放假中~

參考資料： Deitel, H. M., & Deitel, P. J. (2015). C: How to program.
Upper Saddle River, N.J: Prentice Hall.