

計算機程式設計

C語言 Struct

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

結構 (structure)

□ 結構

- 使用者自定的資料型別
- 由基本資料型別構成的複合資料型別。
- 結構與陣列都屬複合資料型別，但陣列是相同型別資料集合，結構體可為不同型別資料集合

□ 結構型別 (person) 定義與結構型別變數 (Tom)宣告

```
struct 結構名稱 {  
    資料型別 變數名稱;  
    ....  
};
```

```
struct person {  
    char name[30];  
    int age;  
};  
  
int main() {  
    struct person student;  
    return 0;  
}
```

```
int main() {  
    struct person {  
        char name[30];  
        int age;  
    } student;  
    return 0;  
}
```

結構 (structure)

❑ 使用typedef定義結構型別(person_t)

○ _t 資料型別的命名原則

```
typedef struct 結構名稱 {  
    資料型別 變數名稱;  
    ....  
}結構型別名稱;
```

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
int main() {  
    person_t student;  
}
```

struct person == person_t

計算結構大小

□ 使用typedef定義結構型別(person_t)

```
#include <stdio.h>
//預設以最大的元素byte對齊
//此處為 int 4 byte 對齊，
//每一個結構元素不能跨過4byte。
typedef struct person {
    char name[30];
    int age;
} person_t;
int main() {
    person_t student;
    printf("%d\n", sizeof(student)); // 36
    printf("%d\n", sizeof(person_t)); // 36
    return 0;
}
```

name[30]	age
32 bytes (4的乘方)	4 bytes

```
#include <stdio.h>
#pragma pack(1) //以一個 byte對齊
typedef struct person {
    char name[30];
    int age;
} person_t;
int main() {
    person_t student, teacher[20] ;
    printf("%d\n", sizeof(student)); // 34
    printf("%d\n", sizeof(person_t)); // 34
    printf("%d\n", sizeof(teacher)); // 34*20
    return 0;
}
```

name[30]	age
30 bytes	4 bytes

結構體初始值設定

□ 初始值以大括號設定

○ 陣列多一個大括號

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
int main() {  
    person_t teacher = {"Mary", 36};  
    person_t student[2] = {  
        {"Tom", 18},  
        {"John", 19}  
    };  
}
```

存取結構體資料

- ❑ 結構體資料的存取法一：'.' 運算子(operator)
 - 一般變數使用直接運算子'

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
int main() {  
    person_t teacher;  
    scanf("%s", teacher.name); //陣列名稱是記憶體位址  
    scanf("%d", &teacher.age); //一般變數要加&  
    printf("%s, %d", teacher.name, teacher.age);  
}
```

結構體指標

- ❑ 結構體資料的存取法二：'->' 運算子(operator)
 - 指標變數使用間接運算子'->'

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
int main() {  
    person_t teacher[20];  
    person_t *p = teacher;  
    int i=0;  
    for (i=0; i<5; i++, p++) {    // p++ 指到下一個元素  
        scanf("%s", p->name);    //陣列名稱是記憶體位址  
        scanf("%d", &p->age);    //一般變數要加&  
        printf("%s, %d\n", p->name, p->age);  
        printf("%s, %d\n", (*p).name, (*p).age); //使用直接運算子  
        printf("%d - %p\n", i, p); //記憶體位址相差32byte  
    }
```

Exercise

- 請寫一段CODE，輸入兩個人的資料，印出年紀比較大的資料。

將結構體值設給另一個結構體

❑ 直接=設定

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
int main() {  
    person_t teacher1 = {"Lin", 38};  
    person_t teacher2;  
    teacher2 = teacher1;  
    printf("%s, %d\n", teacher2.name, teacher2.age);  
}
```

結構體的結構體

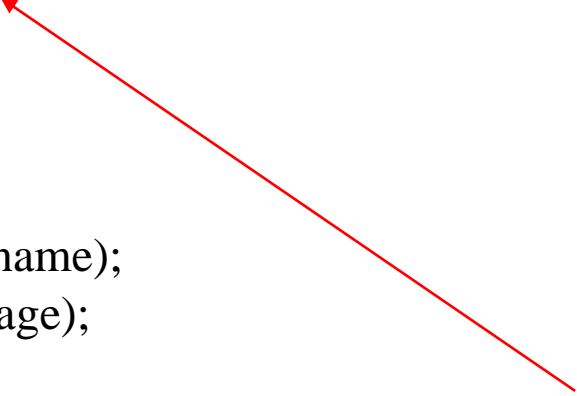
- ❑ 結構內的元素，其資料型別可以是另一個結構型別

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
typedef struct class_s {  
    person_t student[100];  
    int num;  
} class_t;  
int main() {  
    class_t bee;  
    scanf("%d", &bee.num);           //班上幾個人  
    scanf("%s", bee.student[0].name); //第一位同學的名字  
    scanf("%d", &bee.student[0].age); //第一位同學的年齡  
    printf("%s, %d\n", bee.student[0].name, bee.student[0].age);  
}
```

結構體與函式

□ 結構指標傳入函式

```
typedef struct person {  
    char name[30];  
    int age;  
} person_t;  
void show(person_t s) { //傳一般變數, 傳值  
    printf("%s, %d\n", s.name, s.age);  
}  
void change(person_t *s, int newAge) { //傳指標變數, 值的改變會影響主程式  
    s->age = newAge;  
}  
int main() {  
    person_t student;  
    scanf("%s", student.name);  
    scanf("%d", student.age);  
    show(student);  
    change(&student, 12); // 傳記憶體位址, 指標變數  
    show(student);  
}
```



Exercise

- ❑ 請增加學生struct資料，新增一個程式設計pd資料。
- ❑ 請寫一個function，輸入一個班級no位學生的資料，no是班級學生數。
 - `void input(person_t s[], int no);`
- ❑ 請寫一個function，輸入一個班級一位學生的資料。
`void input(person_t *s);`
- ❑ 請寫一個function，比較班級多位學生資料，回傳成績高的姓名 (參數 `char name[]`)。
 - `void getHigh(person_t s[], int no, char name[]);`
- ❑ 請寫一個function，計算回傳班級多位學生平均成績(使用 `return` 回傳)。
 - `int getAverage(person_t s[], int no);`

Exercise

- ❑ 請增加學生struct資料，新增一個程式設計pd資料。

```
int main() {  
    person_t s[2];  
    char name[30];  
    input1(s, 2);  
    input2(&s[0]);  
    getHigh(s, 2, name);  
    printf("%s\n", name);  
    printf("%d\n", getAverage(s, no));  
}
```

Exercise

□ 105APCS Q4

- 簡化版模擬讀入棒球隊每位球員打擊結果，計算得分。假設球員打擊結果只有以下情況：
 - 安打：以 1B, 2B, 3B 和 HR 代表一壘打、二三和全(四)壘打。
 - 出局：以 FO, SO, GO 表示。
 - 飛球出局(Fly Outs)、三振(Strike Out)、滾地球出局(Ground outs)
- 簡化版的規則如下：
 - 球場上有四個壘包，稱為本壘、一、二和三壘。
 - 在本壘握球棒打的稱「擊球員」，站在另外三個壘包稱「跑員」。
 - 擊球員打擊結果為「安打」時，場上擊球員與跑壘員可移動；「出局」時，跑壘員不動，擊球員離場換下一位。
 - 球隊共九位球員，依序排列。比賽開始由第 1 位打擊，當第 i 位球員打擊完，由第 $(i+1)$ 位球員打擊。當第九位球員完後，輪回第 1 位球員。

Exercise

□ 105APCS Q4

- 當打出 K 壘打時，場上擊球員和跑壘員前進 K 個壘包。從本壘到一壘，接著二、三壘，最後回到本壘。
- 每位球員回到本壘時可得 1 分。
- 每達三出局時，一、二、三壘會清空，跑壘員離開，重新開始。

○ 輸入格式

- 每組測試資料固定十行。
- 第一到九行，依照球員順序，每一行代表每位球員打擊資訊。每一行開始有一個正整數 a ($1 \leq a \leq 5$)，代表球員總共打 a 次。接下來有 a 個字串(均為兩個字元)，依序代表每次打擊結果。資料間均以一個空白字元隔開。球員打擊資訊不會錯、缺漏。
- 第十行有一個正整數 b ($1 \leq b \leq 27$)，表示要計算當總出局數累計到 b 時，該球隊的得分。輸入的打擊資訊中至少包含 b 個出局。

Exercise

○練習：請設計球員的struct資料型別

○輸出：計算第 b 個出局數發生時總得分，將此得分輸出於一行。

輸入範例一

```
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO FO FO
3
```

正確輸出

0

輸入範例二

```
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO FO FO
6
```

正確輸出

5

Exercise

□ 105APCS Q4

```
#include <stdio.h>
typedef struct member_s{
    int no;                //球員上場打幾次
    int data[10];          //球員打擊資訊，幾壘安打，0是出局
} member_t;
void input(member_t m[9], int *goal) {
    char temp[10];
    for (int i=0; i<9; i++) {
        scanf("%d", &m[i].no);
        for (int j=0; j<m[i].no; j++) {
            scanf("%s", temp);
            if (temp[1]=='O') m[i].data[j]=0;    //出局
            else if (temp[0]=='H') m[i].data[j]=4; //全壘打
            else m[i].data[j]=(temp[0]-'0');    //1或2或3安打
        }
    }
    scanf("%d", goal);    //計算總出局數累計到b局時，該球隊的得分
}
```

Exercise

□ 105APCS Q4

```
void f() {
    member_t m[9];                // 9位球員
    int goal=0, out=0, index=0, score=0; // index 球員上場的累計人次，out是出局數
    int state=0, r=0, c=0;         // state =000表示三個壘包上均無人
    input(m, &goal);
    while (out<goal) {
        r = index%9; c = index/9; // r 是九人中的第幾位打者，c是球員第幾打次
        if(c>=m[who].no) break;   //out of index range
        if (m[r].data[c]==0) {     // 該球員該次打擊出局
            out++;
            if (out%3==0) state=0; //每三個出局數 跑壘員都得離開，壘包上無人
        }
        else {
            state = (state<<m[r].data[c]) | (1<<(m[r].data[c]-1)); //跑壘狀況
            for (int i=0; i<4; i++) score = score + ((state>>(3+i))&1); //跑回本壘加一分
            state = state&7;       // ex 0001000 0010000 0100000 1000000 超過7代表某選手跑回本壘
        }
        index++;
    }
    printf("score=%d, mem=%d", score, index);
}

int main() {    f();    return 0; }
```

位元欄位 (bit field)

❑ 宣告

```
struct  
    type-specifier declarator : constant-expression
```

- constant-expression 指定欄位的位元寬度，非負整數。若該值是零，宣告沒有 declarator。
- 標準 ANSI C，type-specifier 須是 **unsigned int**、**signed int** 或 **int**。
- Microsoft 的 ANSI C 標準延伸模組允許 char, long, unsigned。
- 不允許位元欄位陣列、位元欄位指標與傳回位元欄位的函式。
- address-of 運算子 (&) 無法套用至位元欄位元件。
- 位元欄位可以有名字、或沒有命名。
- 未命名位元欄位內容在執行期無法預測，一般作為「虛設」欄位，用於**補空間到**對齊位元。
- 寬度為 0，規範 struct-declaration-list 在其後的成員儲存區以下一個 int 界限開始。

位元欄位 (bit field)

- 位元欄位只能宣告為結構的成員。

```
short a:17; /* Illegal! 不能用於一般變數*/  
int long y:33; /* Illegal! */
```

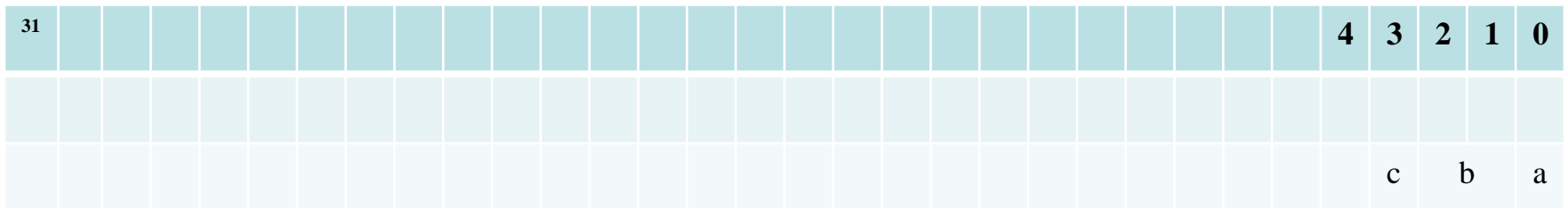
- 將資料以位元形式儲存，節省空間。
- 可存取整數值的部分內容，以簡化程式碼。
- 移植性不高 - 不同作業系統與硬體平台可能有不同結果。
- 一個位元欄位成員不允許跨越兩個 unsigned int 邊界，若成員總位數超過一個 unsigned int 大小，編譯器會自動移位成員，使其對齊 unsigned int 的邊界。
- field1+field2 = 34 Bits，超出 32 Bits。
- field2移位至下一个 unsigned int。
- stuff佔64 bit。

```
struct stuff {  
    unsigned int field1: 30;  
    unsigned int field2: 4;  
};
```

位元欄位 (bit field)

○ 28 bit未使用 ○

```
struct data {
    unsigned int a: 1;
    unsigned int b: 2;
    unsigned int c: 1;
};
```



```
struct data {
    unsigned int a: 4;
    unsigned int : 0;    //空的位元欄位
    unsigned int b: 1;    // 從下一個單元開始存放
};
```

位元欄位 (bit field)

31																							9	8	7	6	5	4	3	2	1	0
																							0	0	1	1	1	1	1	0	1	0
																								status				b4	b3	b2	b1	b0

```
#include <stdio.h>
typedef struct flag_s{
    unsigned b0 : 1;
    unsigned b1 : 1;
    unsigned b2 : 1;
    unsigned b3 : 1;
    unsigned b4 : 1;
    unsigned status : 4;
} flag_t;
int main() {
    flag_t flag;
    scanf("%x", &flag); // 輸入 16進位 fa = 1111 1010
    printf("%d\n", sizeof(unsigned), sizeof(flag_t)); // 4 byte
    printf("%d %d %d\n", flag.b2, flag.b1, flag.b0); // 0 1 0 byte
    printf("%d %d %d\n", flag.status, flag.b4, flag.b3); // 7 1 1 byte
    return 0;
}
```

位元欄位 (bit field)

○ 共占用byte數。

```
#include <stdio.h>
struct on_off {
    unsigned light : 1;
    unsigned toaster : 1;
    int count; /* 4 bytes */
    unsigned ac : 4;
    unsigned : 4;
    unsigned clock : 1;
    unsigned : 0;
    unsigned flag : 1;
} kitchen;
struct flag{
    unsigned short F1 : 1;
    unsigned short F2 : 1;
    unsigned short F3 : 1;
    unsigned short F4 : 1;
    unsigned short F5 : 1;
} room;
```

```
struct house_s{
    unsigned int F1 : 1;
    unsigned int F2 : 1;
    unsigned int F3 : 1;
} house;
int main() {
    printf("kitchen=%d\n", sizeof(kitchen)); // 16 byte
    printf("%d\n", sizeof(room));           // 2 byte
    printf("%d\n", sizeof(unsigned short)); // 2 byte
    printf("%d\n", sizeof(house));          // 4 byte
    return 0;
}
```

bit 位元欄位

- ❑ #pragma 是編譯器功能選擇參數，在不同編譯器可能有差異。
- ❑ GCC 和 MS 對C標準的 storage unit 解釋不同，#pack(1)會產生不同答案。

```
#include <stdio.h>
#include <stdint.h>
#pragma pack(1)
struct _s {
    uint8_t f0;
    uint32_t f1 : 17;
    uint32_t f2;
} storage;
#pragma pack()
int main() {
    printf("%ld\n", sizeof(storage)); // (GCC-8) (MS-9)
    printf("%ld\n", sizeof(uint8_t)); // 1
    printf("%ld\n", sizeof(uint32_t)); // 4
    return 0;
}
```

GCC
用 1 個 byte
用 3 個 byte (24 bit > 17bit)
用 4 個 byte

MS 微軟作業系統
用 1 個 byte
用 4 個 byte (24 bit)
用 4 個 byte

bit 位元欄位: MS

```
#include <stdio.h>
#include <stdint.h>
#pragma pack(push)
#pragma pack(1)
struct on_off1 {
    unsigned char light: 1; // 1 byte
    int count;           // 4 bytes
}kitchen1;
struct on_off2 {
    uint8_t light: 1;      // 1 byte
    int count;             // 4 bytes
    unsigned int: 3;       // 4 bytes
}kitchen2;
struct on_off3 {
    uint8_t light: 1;      // 1 bytes
    unsigned int: 0;
    unsigned int toaster: 1; // 4 bytes
    int count;              // 4 bytes
    unsigned int ac: 3;     // 4 bytes
}kitchen3;
```

```
struct on_off4 {
    int count;             // 4 bytes
    unsigned short ac: 3;  // 2 bytes
}kitchen4;
struct on_off5 {
    unsigned char light: 1;
    unsigned char toaster: 1; // 1 bytes
    int count;              // 4 bytes
    unsigned int ac: 3;
    unsigned int: 4;
    unsigned int clock: 1;  // 4 bytes
    unsigned: 0;
    unsigned int flag: 1;   // 4 bytes
}kitchen5;
#pragma pack(pop)
int main() {
    printf("kitchen=%ld\n", sizeof(kitchen1)); // 5 byte
    printf("kitchen=%ld\n", sizeof(kitchen2)); // 9 byte
    printf("kitchen=%ld\n", sizeof(kitchen3)); // 13 byte
    printf("kitchen=%ld\n", sizeof(kitchen4)); // 6 byte
    printf("kitchen=%ld\n", sizeof(kitchen5)); // 13 byte
    return 0;
}
```

bit 位元欄位: MS

```
#include <stdio.h>
#include <stdint.h>
#pragma pack(push)
//#pragma pack(1)
struct on_off1 {
    unsigned char light: 1; // 1 byte
    int count;           // 4 bytes
}kitchen1;
struct on_off2 {
    uint8_t light: 1;      // 1 byte
    int count;             // 4 bytes
    unsigned int: 3;       // 4 bytes
}kitchen2;
struct on_off3 {
    uint8_t light: 1;      // 1 bytes
    unsigned int: 0;
    unsigned int toaster: 1; // 4 bytes (or 1)
    int count;              // 4 bytes
    unsigned int ac: 3;     // 4 bytes (or 1)
}kitchen3;
```

```
struct on_off4 {
    int count;             // 4 bytes
    unsigned short ac: 3;  // 2 bytes (or 1)
}kitchen4;
struct on_off5 {
    unsigned char light: 1;
    unsigned char toaster: 1; // 1 bytes
    int count;              // 4 bytes
    unsigned int ac: 3;
    unsigned int: 4;
    unsigned int clock: 1;  // 4 bytes (or 1)
    unsigned: 0;
    unsigned int flag: 1;  // 4 bytes (or 1)
}kitchen5;
#pragma pack(pop)
int main() {
    printf("kitchen=%d\n", sizeof(kitchen1)); // 8 byte
    printf("kitchen=%d\n", sizeof(kitchen2)); // 12 byte
    printf("kitchen=%d\n", sizeof(kitchen3)); // 16 byte
    printf("kitchen=%d\n", sizeof(kitchen4)); // 8 byte
    printf("kitchen=%d\n", sizeof(kitchen5)); // 16 byte
    return 0;
}
```

bit 位元欄位: GCC (Linux)

```
#include <stdio.h>
#include <stdint.h>
#pragma pack(1)
struct on_off1{
    unsigned char light : 1;
    unsigned short toaster : 1;
    int count;
    unsigned int ac : 3;
    unsigned int : 4;
    unsigned int clock : 1;
    unsigned : 0;
    //unsigned int flag : 1;
} kitchen1;
int main() {
    printf("%ld\n", sizeof(kitchen1));    // 9 byte
    return 0;
}
```

MS 微軟作業系統
用 1 個 byte
用 4 個 byte (24 bit)
用 1 個 byte
用 2 個 byte 對齊 4 byte
用 1 個 byte

Exercise bit 位元欄位

```
#include "stdio.h"
union {
    int id;
    struct {
        unsigned int x : 1;
        unsigned int y : 2;
        unsigned int z : 2;
    } bits;
} number;
void main(){
    for (number.id=0;number.id<16; number.id++) {
        printf("id=%3d,bits=%3d%3d%3d\n",number.id,
            number.bits.z, number.bits.y, number.bits.x);
    }
}
```

函式指標-talk

- 結構內的元素，是函式指標。

```
#include <stdio.h>
#include <string.h>
typedef struct cat {
    char name[10];
    void (*talk)(int money);
} cat_t;
void talkLarge(int money) {
    printf("LARGE %d\n", money);
}
void talkSmall(int money) {
    printf("SMALL %d\n", money);
}
void CatTalk(cat_t catX, int money) {
    catX.talk(money);
}
```

函式指標-talk

- 結構內的元素，是函式指標。

```
void test01() {  
    cat_t cat001;  
    strcpy(cat001.name, "Tom");  
    cat001.talk = talkLarge;  
    CatTalk(cat001, 1000);  
    cat001.talk = talkSmall;  
    CatTalk(cat001, 10);  
}  
void talkEnglish(char name[]) {  
    printf("Hello! %s, ", name);  
}  
void talkDeutsch(char name[]) {  
    printf("Hallo! %s, ", name);  
}
```

函式指標-talk

- 結構內的元素，是函式指標。

```
void run02(void talk(char *), char name[]) {  
    void (*f)(char *);  
    talk(name);  
    talk = talkEnglish;  
    talk(name);  
    f = talk;  
    f(name);  
}  
void test02() {  
    run02(talkEnglish, "John");  
    run02(talkDeutsch, "John");  
}  
int main() {  
    test01();  
    test02();  
    return 0;  
}
```

函式指標-Shape

```
#define ShapeText(TYPE) \  
    char name[10];\  
    float (*perimeter)(struct TYPE*);  
typedef struct _Shape { // Shape 物件無欄位  
    ShapeText(_Shape);  
} Shape;  
typedef struct _Circle {  
    ShapeText(_Circle);  
    float radius;  
} Circle;  
float ShapeArea(Shape *obj) { return 0; }  
float ShapePerimeter(Shape *obj) { return 0; }  
void ShapeNew(Shape *obj) {  
    strcpy(obj->name,"shape");  
    obj->perimeter = ShapePerimeter;  
}  
float CircleArea(Circle *obj)  
    { return 3.14 * obj->radius * obj->radius; }
```

```
float CirclePerimeter(Circle *obj)  
    { return 2*3.14 * obj->radius; }  
void CircleNew(Circle *obj) {  
    strcpy(obj->name,"circle");  
    obj->perimeter = CirclePerimeter;  
}  
int main() {  
    int i;  
    Shape s;  
    Circle c;  
    ShapeNew(&s);  
    CircleNew(&c);  
    c.radius=3.0;  
    Shape *list[] = { &s, (Shape*) &c};  
    for (i=0; i<2; i++) {  
        Shape *o = list[i];  
        printf("%s.perimeter()=%G\n", o->name,  
            o->perimeter(o));  
    }  
}
```


函式指標-Shape draw

- 結構內的元素，是函式指標。不定長度參數，造出不同物件。

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
typedef enum {point, circle} shape_type;
typedef struct _shape{
    shape_type type;
    void (*destroy)();
    void (*draw)();
}shape_t;
typedef struct {
    shape_t common;
    int x,y;
}point_t;
```

函式指標-Shape draw

□ 定義各種運算函式

```
typedef struct {
    shape_t common;
    point_t *center;
    int radius;
}circle_t;
void destroyPoint(point_t *this){
    int x=this->x,y=this->y;
    free(this);
    printf("P(%d,%d) destroyed!\n",x,y);
}
void drawPoint(point_t *this){
    printf("P(%d,%d)",this->x,this->y);
}
```

函式指標-Shape draw

```
point_t * createPoint(va_list *ap){
    point_t *p_point;
    if ((p_point=(point_t*)malloc(sizeof (point_t)))==NULL)
        return NULL;
    p_point->common.type=point;
    p_point->common.destroy=destroyPoint;
    p_point->common.draw=drawPoint;
    p_point->x=va_arg(*ap,int);
    p_point->y=va_arg(*ap,int);
    return p_point;
}

void destroyCircle(circle_t *this){
    int x=this->center->x;
    int y=this->center->y;
    intr =this->radius;
    this->center->common.destroy(this->center);
    free(this);
    printf("C(P(%d,%d),%d) destroyed!\n",x,y,r);
}
```

函式指標-Shape draw

```
void drawCircle(circle_t *this){
    printf("C(");
    this->center->common.draw(this->center);
    printf(",%d)",this->radius);
}

circle_t* createCircle(va_list* ap){
    circle_t *p_circle;
    if ((p_circle=(circle_t*)malloc(sizeof(circle_t)))==NULL)
        return NULL;
    p_circle->common.type=circle;
    p_circle->common.destroy=destroyCircle;
    p_circle->common.draw=drawCircle;
    p_circle->center=createPoint(ap);
    p_circle->radius=va_arg(*ap,int);
    return p_circle;
}
```

函式指標-Shape draw

```
shape_t* createShape(shape_type st,...){
    va_list ap;
    shape_t* p_shape=NULL;
    va_start(ap,st);
    if (st==point) p_shape=(shape_t*)createPoint(&ap);
    if (st==circle) p_shape=(shape_t*)createCircle(&ap);
    va_end(ap);
    return p_shape;
}
int main(){
    int i;
    shape_t *shapes[2];
    shapes[0]=createShape(point,2,3);
    shapes[1]=createShape(circle,20,40,10);
    for (i=0;i<2;i++) {
        shapes[i]->draw(shapes[i]);
        printf("\n");
    }
    for (i=1;i>=0;i--)    shapes[i]->destroy(shapes[i]);
    return 0;}
```

Exercise 函式指標-電路圖

□ 定義各種運算函式

```
#include <stdio.h>
#include <stdlib.h>
#define GATEVALUE(TYPE) int(*GateValue)(void)
typedef struct _Gate{
    GATEVALUE();
}Gate;
int GateGetValue(){return 0;}
typedef struct _GateAnd{
    GATEVALUE();
}GateAnd;
int GateAndValue(){return 1;}
void CreateGate(Gate *obj){
    obj->GateValue = GateGetValue;
}
```

Exercise 函式指標-電路圖

□ 定義各種運算函式

```
void CreateGateAND(GateAnd *obj){
    obj->GateValue = GateAndValue;
}
int main(int argc, char *argv[]){
    Gate gate;
    CreateGate(&gate);
    GateAnd and;
    CreateGateAND(&and);
    printf("Gate = %d, GateAND = %d\n", gate.GateValue(),
and.GateValue());
    return 0;
}
```

Homework

- 以下邏輯電路圖，輸入為 X1, X2, X3，輸出為 Y1, Y2, Y3。
 - X1 --> P_Gate----->Y1
 - X2 ----->Q_Gate----->Y2
 - X3 -----R_Gate--->Y3
 - P_Gate 邏輯閘可設定為 NOT 或空，輸入為 X1，輸出為 Y1 和 Q_Gate 邏輯閘的輸入。
 - Q_Gate 邏輯閘可設定為 AND 或 OR，輸入為 X2 和 P_Gate 邏輯閘的輸出，輸出為 Y2 和 R_Gate 邏輯閘的輸入。
 - R_Gate 邏輯閘可設定為 AND 或 OR，輸入為 X3 和 Q_Gate 邏輯閘的輸出，輸出為 Y3。
 - 輸入 X1、X2、X3，以及設定 P、Q、R、三個邏輯閘的種類。

Homework

○輸入說明:

- 第一行依次輸入X1、X2、X3 為 0 或 1，中間以逗號間隔。
- 第二行輸入 P、Q、R 邏輯閘的設定，A 代表 AND 邏輯閘，O 代表 OR 邏輯閘，N 代表 NOT 邏輯閘，E 代表空的邏輯閘，中間以逗號間隔。

○輸出說明:

- 輸出 Y1、Y2、Y3為 0 或 1，中間以逗號間隔。

○範例:

➤ Sample Input:

– 0,1,0

– N,A,O

➤ Sample Output:

– 1,1,1

Homework

□ 利用結構 struct 定義

- Shape (圖形), Circle (圓), Rectangle (矩形), Square (正方形), Triangle (三角形) 。
- 圓有半徑，矩形有長和寬，正方形有邊長，三角形有三個邊。
- 計算各個圖形的周長，以及所有圖形的周長加總。
- 此題須使用以下 struct 及 function pointer 實作，否則不予計分。

```
#define ShapeText(TYPE) char name[10];  
    double (*perimeter)(struct TYPE*);  
    double (*area)(struct TYPE*);  
typedef struct shape_s {  
    ShapeText(shape_s);  
} shape_t;  
typedef struct circle_s {  
    ShapeText(circle_s);  
    double radius;  
} circle_t;
```

Homework

□ 利用結構 struct 定義

- Shape (圖形), Circle (圓), Rectangle (矩形), Square (正方形), Triangle (三角形)。
- 圓有半徑，矩形有長和寬，正方形有邊長，三角形有三個邊。
- 計算各個圖形的周長，以及所有圖形的周長加總。
- 此題須使用以下 struct 及 function pointer 實作，否則不予計分。

輸入說明	輸出說明
第一行輸入圖形個數N。第二行到第N+1行輸入圖形種類、及該圖形所需整數資料，以空白間隔。圖形種類以字元表示，C代表圓、R代表矩形、S代表正方形、T代表三角形。輸入C跟隨一個數值為半徑，輸入R跟隨兩個數值長和寬，輸入S跟隨一個數值代表邊長，輸入T跟隨三個數值代表三個邊。	PI設為4。輸出N+1行，前N行，輸出第N個圖形的周長。第N+1行，輸出N個圖形的周長總和。

Sample Input	Sample Output
5	12
T 3 4 5	4
S 1	10
R 2 3	8
C 1	13
T 3 4 6	47

Homework 工作排程

□ 問題描述

- 有M個工作要在N台機器上加工，每個工作i包含若干個工序 o_{ij} ，這些工序須依序加工，也就是前一道工序 $o_{i(j-1)}$ 完成後才可開始下一道工序 o_{ij} 。每道工序 o_{ij} 可用一個有序對 (k_{ij}, t_{ij}) 表示它需在機器 k_{ij} 上面花費 t_{ij} 小時完成。每台機器一次只能處理一道工序。
- 所謂一道工序 o_{ij} 的「最早完成時間的 c_{ij}^* 」是指考慮目前排程中機器 k_{ij} 之可用性以及前一道工序 $o_{i(j-1)}$ (若該工序存在)之完成時間後可得的最早完成時間。工廠經理安排所有工廠經理安排所有工序的排程規則如下：
 - 針對每一個工作的第一個尚未排程的工序，計算出此工序的「最早完成時間」，然後挑選出最早完成時間最小的工序納入排程，如果有多個完成時間都是最小，則挑選其中工作編號最小之工序。一個工序一旦納入排程就不會再更改，重複以上步驟直到所有工序皆納入排程。

Exercise 工作排程

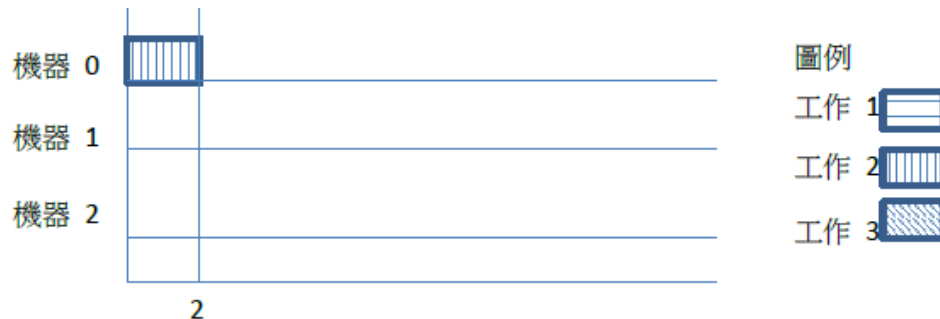
- 總是從時間0開始排程，每個工作的完成時間為其最後一個工序的完成時間，序的完成時間，本題的目標是計算出每個工作的完成時間並輸出其總和。
- 以下例子說明，此例中有三個工作要在三台機器上排程，各工作的資料如下。

	工序	說明
工作1	$o11 = (2, 4)$ $o12 = (1, 1)$	此工作有兩道工序，第一道需要在機器2執行4小時，第二道需要在機器1執行1小時。
工作2	$o21 = (0, 2)$ $o22 = (2, 2)$ $o23 = (0, 1)$	有三道工序，第一道需要在機器0執行2小時，餘類推。
工作3	$o31 = (0, 7)$	有一道工序需要在機器0執行7小時。

Exercise 工作排程

○排程過說明如下：

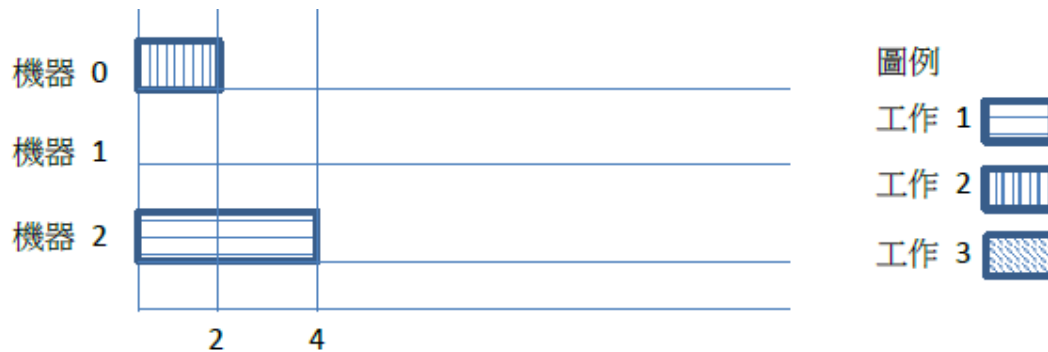
- 1. 在開始時，每個工作都是考慮第一道工序，三個工作第 1 道工序需要的時間分別是 $t_{11} = 4$ 、 $t_{21} = 2$ 、 $t_{31} = 7$ ，這也是它們的最早完成時間，也就是 $c_{11} = 4$ 、 $c_{21} = 2$ 、 $c_{31} = 7$ ，因此會先排 o21。



- 2. 接下來，三個工作要考慮的順序分別是第 1、2、1 個工序，即 o11、o22 和 o31。
 - (1) o11 需要機器 2 執行 4 小時，而機器 2 可以開始加工的時間點是 0；o11 沒有前一道工序。因此，這工序可以開始的時間是 $\max(0, 0) = 0$ 。是故，其最早完成時間 $c_{11} = \max(0, 0) + 4 = 4$ 。
 - (2) o22 需機器 2 執行 2 小時，而機器 2 可開始加工時間點是 0；o22 前一道工序 o21 完成時間是 2。因此，這工序可以開始的時間是 $\max(0, 2) = 2$ 。最早完成時間 $c_{22} = \max(0, 2) + 2 = 4$ 。

Exercise 工作排程

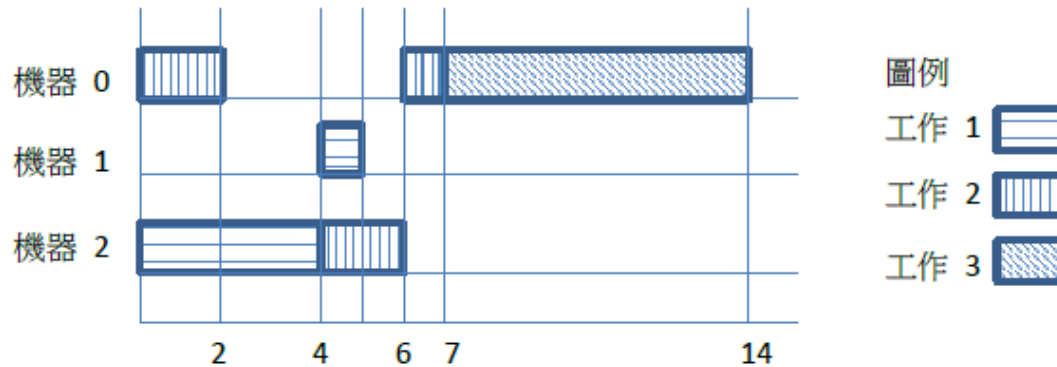
- (3) o31 需機器 0 執行 7 小時，而機器 0 可開始加工的時間點是 2；o31 沒有前一道工序。因此，這工序可開始的時間是 $\max(2, 0) = 2$ 。其最早完成時間 $c31^* = \max(2, 0) + 7 = 9$ 。
- 因此，由於 $c11^*$ 和 $c22^*$ 都是最小，根據規則，工作編號小的先排，所以會排 o11。



- 3. 三個工作目前要考慮的順序分別第 2、2、1 個工序。依照類似推論，可得到 $c12 = 5$ ， $c22 = 6$ ， $c31 = 9$ ，因此排 o12。工作 1 的工序均已排完，所以它的完成時間是 5。
- 4. 剩下工作 2 與 3。 $c22 = 6$ ， $c31 = 9$ ，因此先排 o22。
- 5. $c23 = 7$ 而 $c31 = 9$ ，因此排 o23，工作 2 的工序已排完，所以它的完成時間是 7。

Exercise 工作排程

- 6. 剩下工作 3，因為機器 0 的下一個可以開始時間是 7，工作 3 的完成時間是 $7+7=14$ 。



- 三個工作完成時間分別是 5、7、14，輸出答案 $5+7+14=26$ 。
- 輸入格式

— 第一行有兩個整數 N 與 M ，代表 N 台機器與 M 個工作，接下來有 M 個工作資訊，輸入順序即是工作編號順序。每個工作資訊包含兩行，第一是整數 P ，代表到工序數量；第二行是 $2 \times P$ 個整數，每兩個一組依序代表一道工序的機器編號與需求時間。機器的編號由 0 開始。參數 N 、 M 、 P 以及每個工序的需求時間都是不超過 100 的正整數。

Exercise 工作排程

➤ 輸出格式

– 輸出每個工作的完成時間的總和。

範例一：輸入

3 3

2

2 4 1 1

3

0 2 2 2 0 1

1

0 7

正確輸出

26

範例二：輸入

2 3

1

0 4

1

1 5

1

1 3

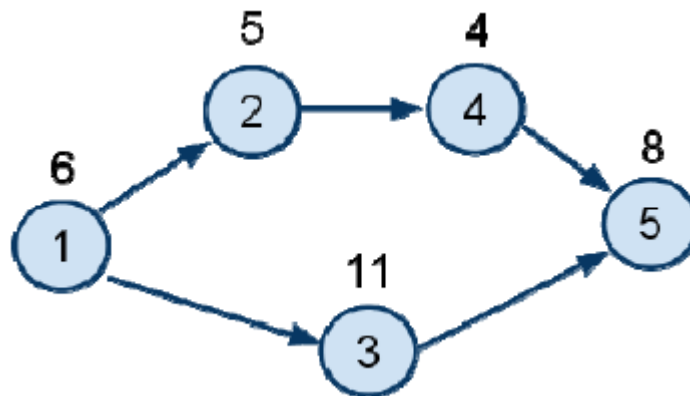
正確輸出

15

Homework

□ 題目

- 開發專案時，專案會被分割為許多項目，分配給多組程式設計師開發。但這些項目是有順序關係，只有當順序在前的項目完成，才能開始開發順序在後的項目。用一個有向圖，表示這些開發順序。每個節點代表一個項目，節點內的數字為節點編號，上方數字代表開發這個項目所需天數；邊表示開發順序。
- 以圖為例，只有在節點 2 完成後，才能開始節點 4 的開發。
- 一軟體公司的專案準備開始開發，你是公司專案經理，根據開發流程圖，老闆想知道專案能否在限制時間內完工。



Exercise

□ 輸入與輸出

- 輸入第一行是整數，代表後續測試資料組數。每組測試資料代表專案有向圖。每組測試資料的第一行是正整數 N ，代表專案共有 N 個工作事項(節點)， $N \leq 1000$ 。
- 接下來 N 行測試資料，每一行依序代表一個項目節點(從 1 開始)，第一個正整數表示完成這個項目所需天數，第二個正整數 K 表示這個節點有 K 條指向其他節點的邊，接下來 K 個正整數表示所指向的項目節點編號。

範例輸入 #1

```
2
2
8 1 2
2 0
5
6 2 2 3
5 1 4
11 1 5
4 1 5
8 0
```

範例輸出 #1

```
10
25
```

```
2    共有兩組專案測試資料
2    第一組專案有兩個工作項目 (節點)
8 1 2 第一個工作項目需8天完成，有一個工作項目(第二個工作項目)需等第一個工作項目
      完成後才能進行。
2 0   第二個工作項目需要2天才能完成
5     第二組專案有五個工作項目 (節點)
6 2 2 3 第一個工作項目需6天才能完成，有兩個工作項目 (第二、三個工作項目) 需等第
      一個工作項目完成後才能進行。
5 1 4  第二個工作項目需要5天才能完成，有一個工作項目 (第四個工作項目) 需等第二
      個工作項目完成後才能進行。
11 1 5 第三個工作項目需要11天才能完成，有一個工作項目 (第五個工作項目) 需等第三
      個工作項目完成後才能進行。
4 1 5  第四個工作項目需要4天才能完成，有一個工作項目 (第五個工作項目) 需等第四
      個工作項目完成後才能進行。
8 0   第五個工作項目需要8天才能完成
```

Exercise

```
#include <stdio.h>
typedef struct task_s{
    int pTask[10];
    int pNo;
    int day;
    int tDay;
} task_t;
int isCanCompute(task_t tasks[], task_t t) {
    int flag=1, id=0;
    for (int i=0; i<t.pNo; i++) {

    }
    return flag;
}
int isYetCompute(task_t t) {
    if (t.tDay==-1) return 1;
    else return 0;
}
```

```
int computeDay(task_t tasks[], task_t t) {
    int maxDay=0, id=0;
    for (int i=0; i<t.pNo; i++) {

    }
    return (maxDay+t.day);
}
void print(int n, task_t data[]) {
    for (int i=1; i<=n; i++) {
        printf("%d, %d, %d, %d\n", i,
data[i].pNo, data[i].day, data[i].tDay);
        for (int j=0; j<data[i].pNo; j++) {
            printf("%d ", data[i].pTask[j]);
        }
        printf("\n");
    }
}
```

Exercise

```
void input(int n, task_t data[]) {
    int c=0, id=0, index=0;
    for (int i=1; i<=n; i++) {
        data[i].pNo=0;
        data[i].tDay = -1;
    }
    for (int i=1; i<=n; i++) {
        scanf("%d", &data[i].day);
        scanf("%d",&c);
        for (int j=0; j<c;j++) {

        }
    }
}
```

```
void printAnswer(int n, task_t data[]) {
    int maxDay=0;

    printf("%d", maxDay);
}

int getId(int n, task_t data[]) {
    for (int i=1; i<=n; i++) {
        if (isYetCompute(data[i])&&isCanCompute(data,
data[i]))
            return i;
    }
    return -1;
}
```

Exercise

```
void f() {
    int n=0, count=0, id=0;
    task_t data[10];
    scanf("%d", &n);
    input(n, data);
    print(n, data);

    while (count<5) {
        id = getId(n, data);
        if (id>=1) {
            data[id].tDay = computeDay(data, data[id]);
            count++;
        }
        else break;
    }
    print(n, data);
    printAnswer(n, data);
}
```

```
int main() {
    int no =0;
    scanf("%d", &no);
    for (int i=0; i<no; i++)
        f();

    return 0;
}
```