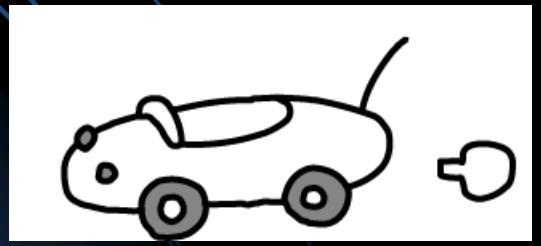
各式各樣的資料型態

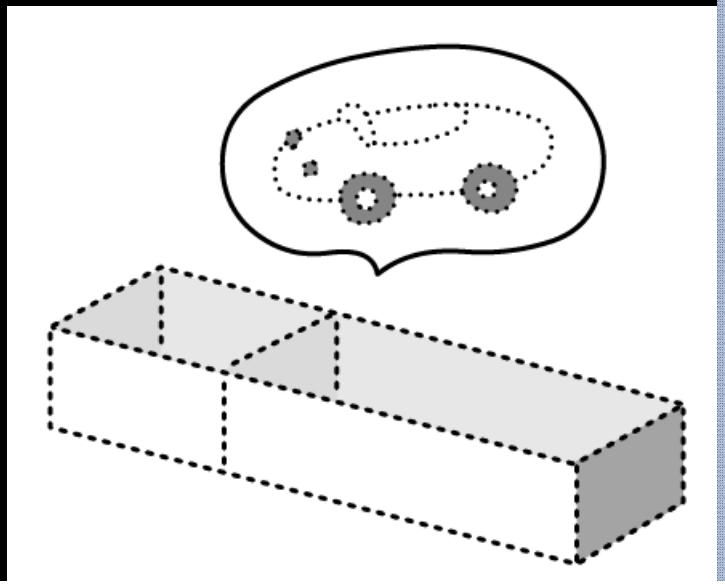
11-1 結構的基礎知識

- 決定新的型態
- 關於結構
  - 結構資料型態可以將不同資料型態的值整合成新的型態。
  - 結構型態的宣告語法：

```
struct  結構型態 {
   資料型態  識別字;
   資料型態  識別字;
   …
};
```

加上**struct**進行宣告



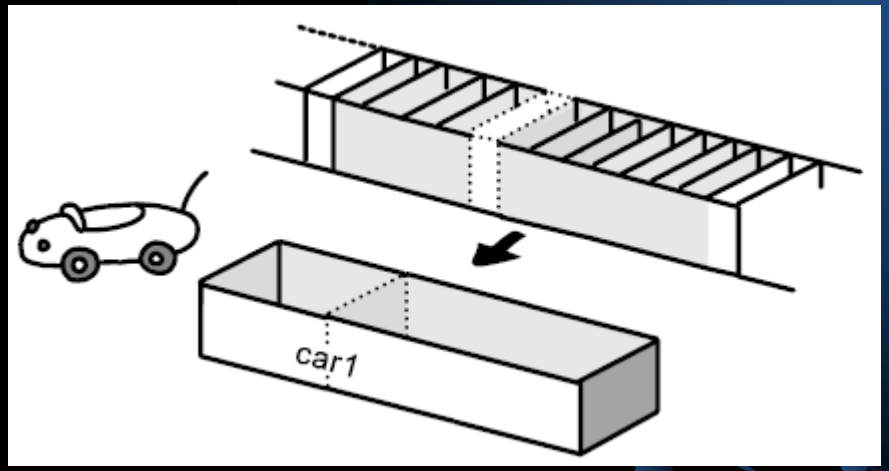


- 宣告結構變數
  - 語法：

    結構型態 結構變數名稱;

  - 範例：

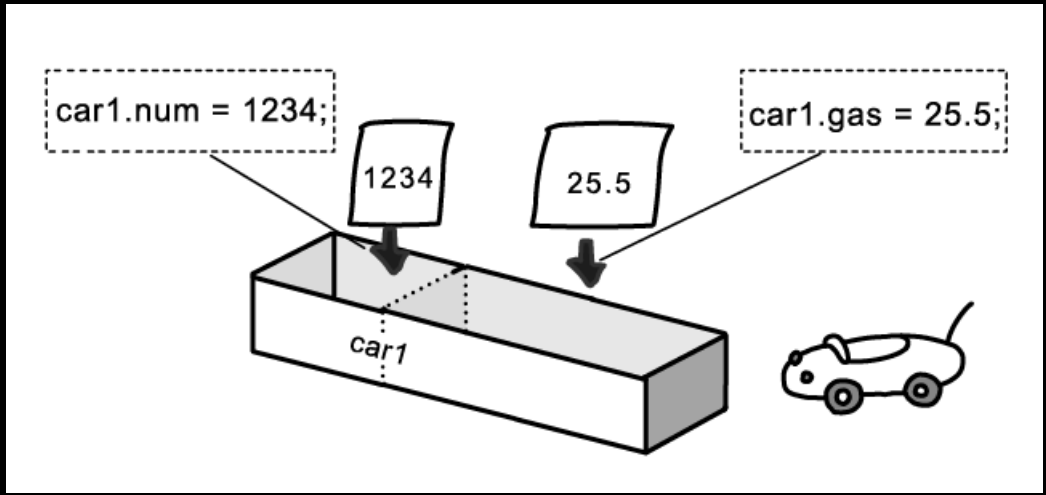
    struct Car car1;



這是儲存**struct Car**型態
的值的變數**car1**

- 對成員進行存取
  - 使用結構型態的成員時，必須使用**成員選擇運算子(.)**。
  - 語法：

    結構變數名稱.成員



- 存取結構型態的成員之範例：

```
#include <stdio.h>
/* 結構資料型態struct Car的宣告 */
struct Car{
    int num;
    double gas;
};

int main(void)
{
    struct Car car1;

    car1.num = 1234;
    car1.gas = 25.5;
    printf("車牌號碼是%d；汽車容量是%f。\n", car1.num, car1.gas);
    return 0;
}
```

宣告結構型態

宣告結構變數

把值指定給成員

輸出成員的值

- 從鍵盤輸入成員的值之範例：

```
#include <stdio.h>
/* 結構資料型態struct Car的宣告 */
struct Car{
    int num;
    double gas;
};
int main(void)
{
    struct Car car1;
    printf("請輸入車牌號碼。\n");
    scanf("%d", &car1.num);
    printf("請輸入汽油容量。\n");
    scanf("%lf", &car1.gas);
    printf("車牌號碼是%d；汽油容量是%f。\n", car1.num, car1.gas);
    return 0;
}
```

在前面加上&

從鍵盤輸入成員的值

11-2 結構的寫法

- 利用typedef來分割名稱
  - typedef的語法：

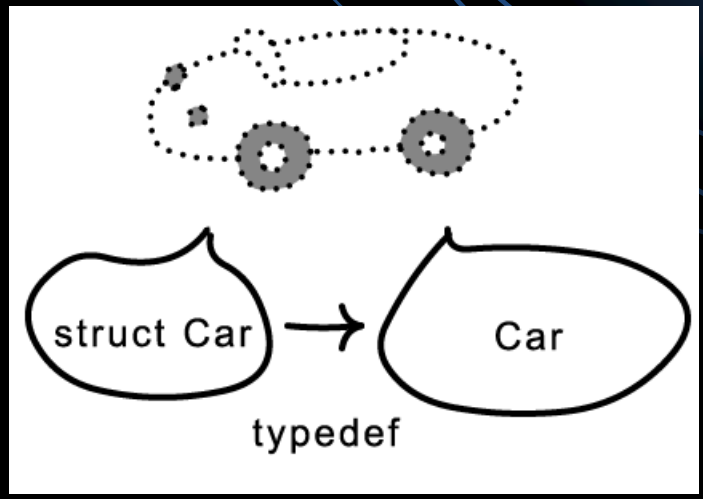
    typedef 資料型態 識別字;

  - 範例：

```
/* 結構型態struct Car的宣告 */
typedef struct Car{
  int num;
  double gas;
} Car;
```

把**typedef**加在「**struct Car**」型態的前面

命名爲「**Car**」型態



Sample3.c ▶ 使用 typedef

```
#include <stdio.h>

/* 結構資料型態 struct Car 的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;

int main(void)
{
    Car car1;

    car1.num = 1234;
    car1.gas = 25.5;

    printf("車牌號碼是 %d；汽油容量是 %f 。\n", car1.num,
           car1.gas);

    return 0;
}
```

使用 typedef 來縮短名稱

與宣告 struct Car 型態的變數相同

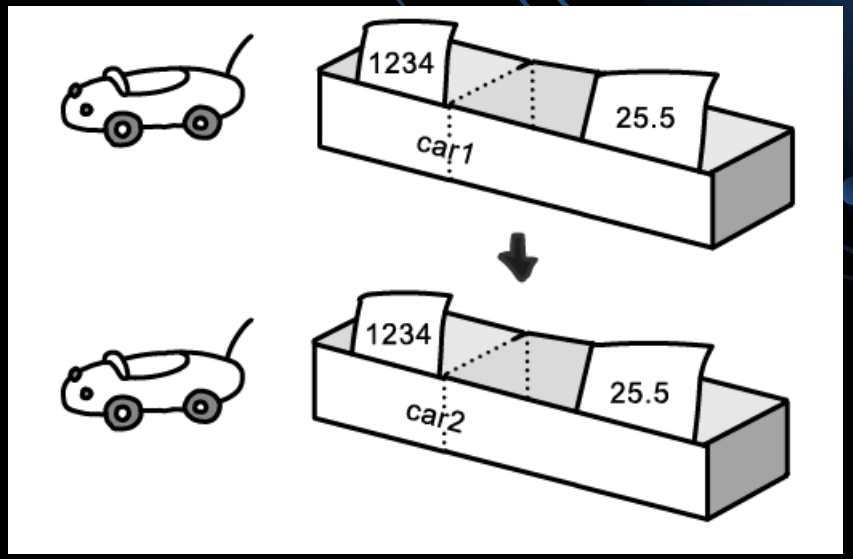
- 結構的初始化
  - 語法：

    結構型態 結構變數名稱 = {值, 值, …};

  - 範例：

```
#include <stdio.h>
/* 結構型態struct Car的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;
int main(void)
{
    Car car1 = {1234,25.5};
    printf("車牌號碼是%d；汽車容量是%f。\n", car1.num, car1.gas);
    return 0;
}
```

- 把值指定給結構
  - 對結構進行指定後，就會逐一複製其成員的值，並儲存至所指定的目的地。
  - 範例：

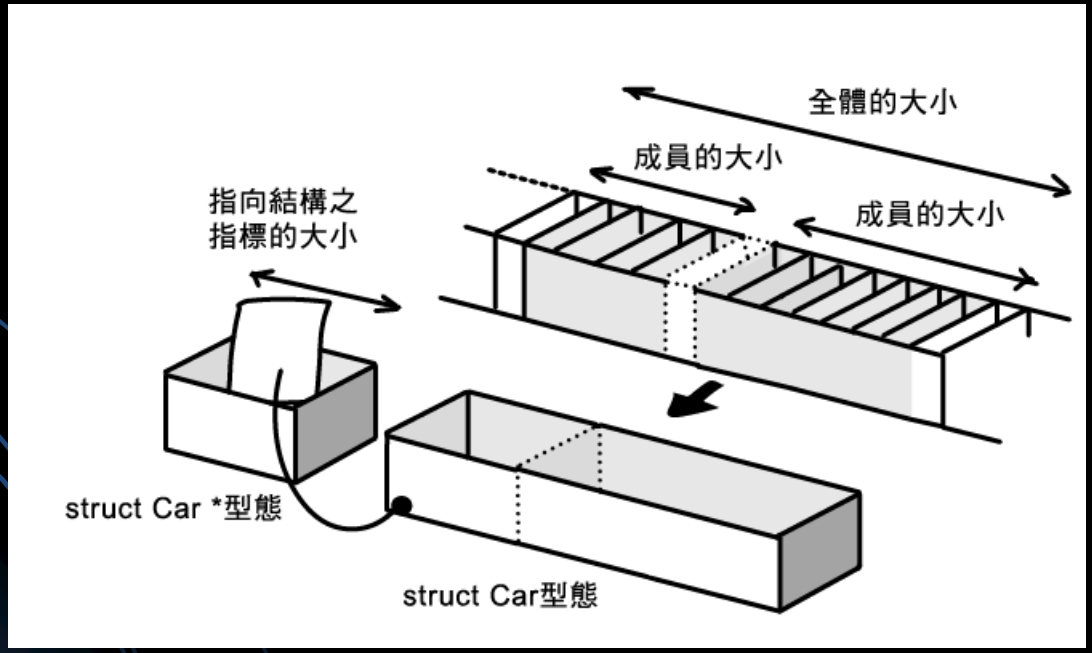



```
#include <stdio.h>
/* 結構型態struct Car的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;
int main(void)
{
    Car car1 = {1234, 25.5};
    Car car2 = {4567, 52.2};
    printf("car1的車牌號碼是%d、汽油容量是%f。\n", car1.num, car1.gas);
    printf("car2的車牌號碼是%d、汽油容量是%f。\n", car2.num, car2.gas);
    car2 = car1;
    printf("把car1指定給car2。\n");
    printf("car2的車牌號碼變成%d、汽油容量變成%f。\n", car2.num, car2.gas);
    return 0;
}
```

以其他的結構來指定

11-3 結構的大小

- 理解結構型態的大小
  - 結構的大小不一定是成員大小的加總。同時，指向大型結構的指標，通常會小於該資料型態的大小



- 查詢結構型態的大小之範例：

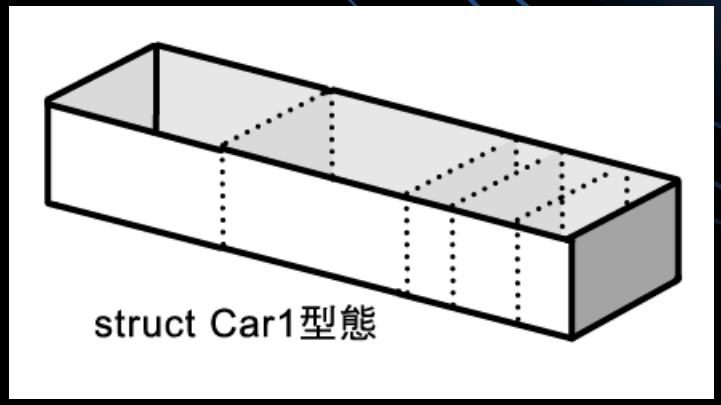
```
#include<stdio.h>

/* 結構型態struct Car的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;

int main(void)
{
    printf("int型態的大小爲%dbyte。\n", sizeof(int));
    printf("double型態的大小爲%dbyte。\n", sizeof(double));
    printf("結構structCar型態的大小爲%dbyte。\n", sizeof(Car));
    printf("指向結構struct Car型態的指標大小爲%dbyte。\n", sizeof(Car *));
    return 0;
}
```

查詢結構型態的大小

查詢指向結構型態的指標大小

- 使用位元欄（bit field）
  - 位元欄是會影響結構大小的成員。
  - 位元欄的語法：

    struct 結構型態名稱{
    資料型態 識別字 ： 位元數;
    資料型態 識別字 ： 位元數;
    …
    };



  - 範例：

```
typedef struct Car1{
    int num;
    double gas;
    unsigned int tire : 3;
    unsigned int roof : 1;
    unsigned int color : 4;
}Car1;
```

設爲**3bit**的成員

設爲**1bit**的成員

設爲**4bit**的成員

Sample7.c ▶ 使用位元欄

```
#include<stdio.h>

/* 結構型態 struct Car1 的宣告 */
typedef struct Car1{
    int num;
    double gas;
    unsigned int tire : 3;
    unsigned int roof : 1;
    unsigned int color : 4;
}Car1;

/* 結構型態 struct Car2 的宣告 */
typedef struct Car2{
    int num;
    double gas;
    unsigned int tire;
    unsigned int roof;
    unsigned int color;
}Car2;

int main(void)
{
    printf("使用了位元欄的結構，其大小為 %dbyte。\n",
sizeof(Car1));
    printf("未使用位元欄的結構，其大小為 %dbyte。\n",
sizeof(Car2));
}
    return 0;
```

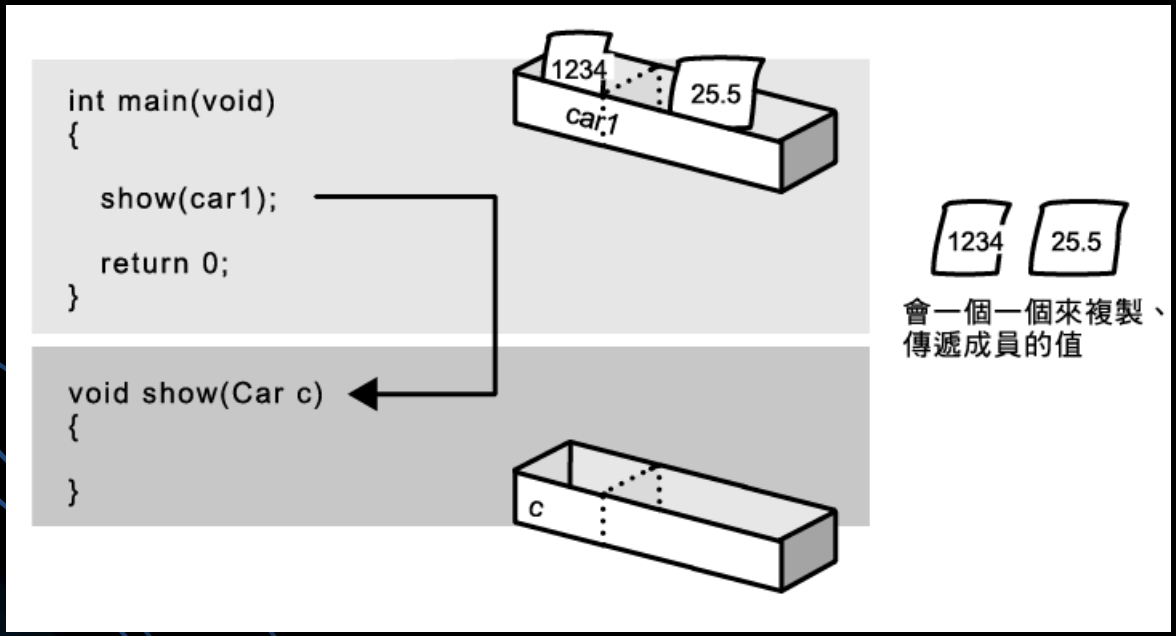
這是使用了位元欄的結構

這是未使用位元欄的結構

查詢結構的大小

11-4 結構的應用

- 把結構當做引數來用
  - 把結構當作函數的引數來用，各成員便會被複製並傳送。
  - 範例：



Sample8.c ▶ 把結構當作引數來用

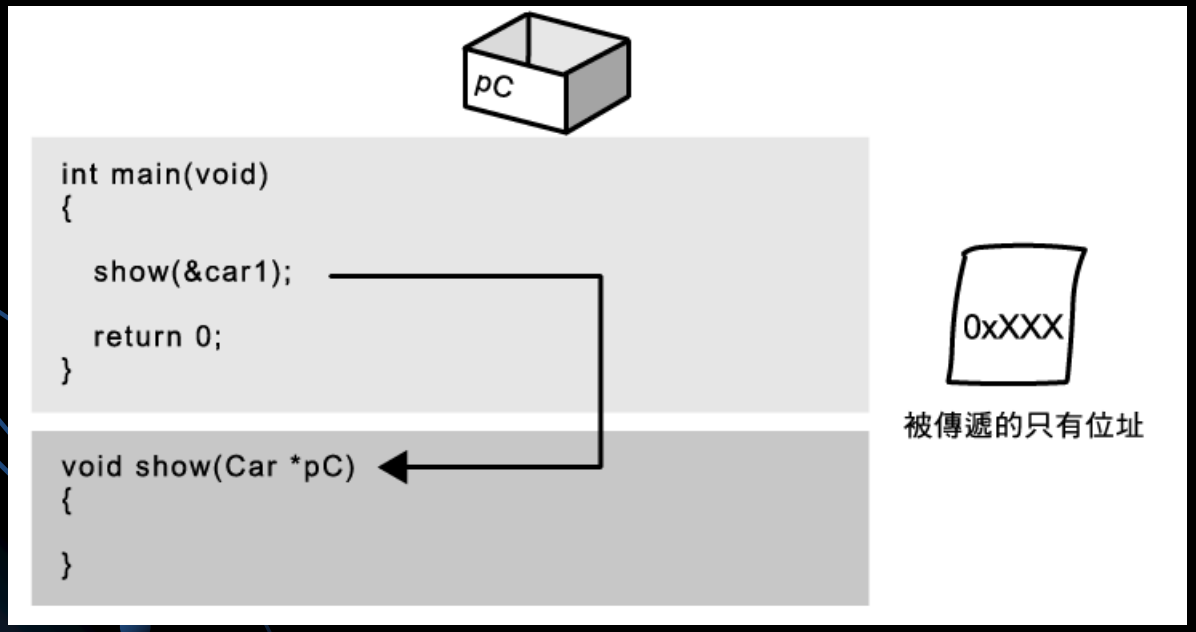
```
#include <stdio.h>

/* 結構型態 struct Car 的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;

/* show 函數的宣告 */
void show(Car c);

int main(void)
{
    Car car1 = {0, 0.0};

    printf("請輸入車牌號碼。\n");
    scanf("%d", &car1.num);

    printf("請輸入汽油容量。\n");
    scanf("%lf", &car1.gas);

    show(car1);

    return 0;
}
```

這個函數是以結構來作為引數

傳送結構 car1 的值

- 把指向結構的指標當作引數來用
  - 把指向結構的指標拿來當作函數的引數，所傳送的就是位址。
  - 從指向結構的指標存取成員時，使用成員選擇運算子(->)會很方便。
  - 從指向結構的指標來存取成員的語法：

    指向結構的指標 -> 結構的成員
  - 範例：



Sample9.c ▶ 把指向結構的指標用作函數的引數

```
#include <stdio.h>

/* 結構資料型態 struct Car 的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;

/* show 函數的宣告 */
void show(Car *pC);

int main(void)
{
    Car car1 = {0, 0.0};

    printf("請輸入車牌號碼。\n");
    scanf("%d", &car1.num);

    printf("請輸入汽油容量。\n");
    scanf("%lf", &car1.gas);

    show(&car1);

    return 0;
}
```

這個函數是把指向結構的指標當作引數

傳送結構型態 car1 的位址

```
/* show 函數的定義 */
void show(Car *pC)
{
    printf("車牌號碼是 %d 、汽油容量是 %f 。\n", pC->num,
           pC->gas);
}
```

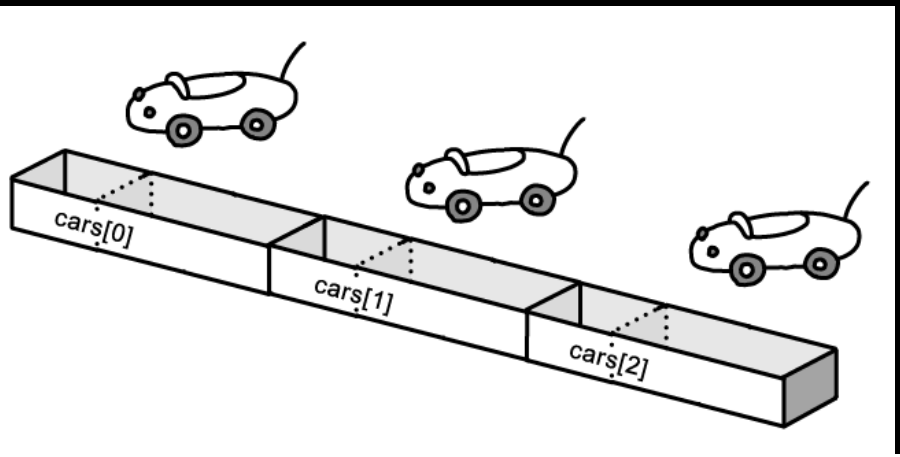
從指標來存取成員

- 製作結構的陣列之範例：

```
#include <stdio.h>
/* 結構型態struct Car的宣告 */
typedef struct Car{
    int num;
    double gas;
}Car;
int main(void)
{
    Car cars[3];
    int i;
    cars[0].num = 1234; cars[0].gas = 25.5;
    cars[1].num = 4567; cars[1].gas = 52.2;
    cars[2].num = 7890; cars[2].gas = 20.5;
    for(i=0; i<3; i++){
      printf("車牌號碼是%d、汽油容量是%f。\n", cars[i].num, cars[i].gas);
    }
   return 0;
}
```

把值儲存在三個結構的各個元素

宣告結構的陣列



- 以結構建立清單之範例：

```
#include <stdio.h>
/* 結構型態struct Car的宣告 */
typedef struct Car{
   int num;
   double gas;
   struct Car *next;
}Car;
int main(void)
{
   Car car0;
   Car car1;
   Car car2;
   Car *pcar;
   car0.num = 1234; car0.gas = 25.5;
   car1.num = 4567; car1.gas = 52.2;
   car2.num = 7890; car2.gas = 20.5;
   car0.next = &car1;
   car1.next = &car2;
   car2.next = NULL;
   for(pcar = &car0; pcar!=NULL; pcar = pcar->next){
      printf("車牌號碼是%d、汽油容量是%f。\n", pcar->num, pcar->gas);
   }
return 0;
}
```

包含指向**struct Car**型態的指標

car0之後是**ctr1**

car1之後是**ctr2**

以**car2**作爲結尾

依照排好的順序輸出結構

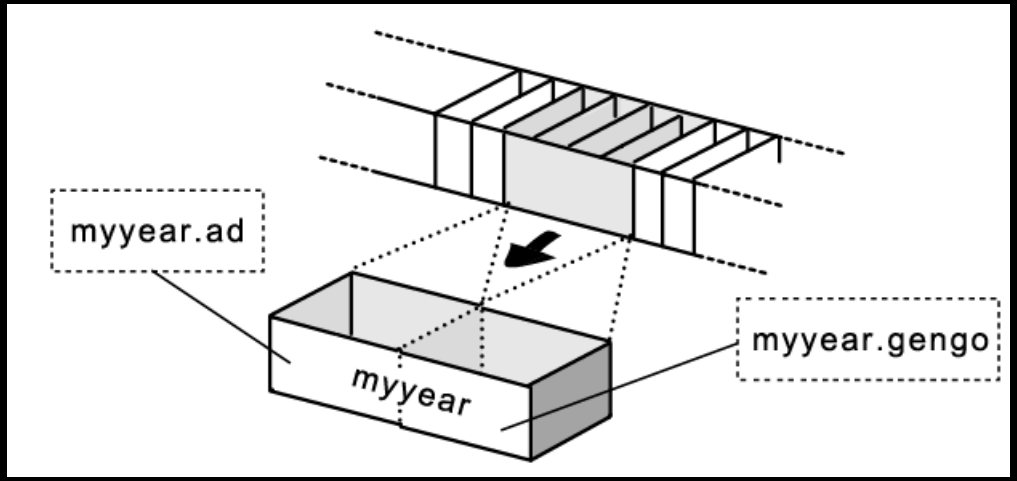
11-5 共同空間

- 關於共同空間
  - 共同空間型態的各成員無法同時記憶值，全體成員只能記憶住一個值。

  - 宣告共同空間型態的語法：

```
union  共同空間型態名稱{
   資料型態  識別字;
   資料型態  識別字;
   …
};
```

宣告時要加上union



- 使用共同空間型態之範例：

```
#include <stdio.h>
/* 共同空間型態union Year的宣告 */
typedef union Year{
    int ad;
    int gengo;
}Year;

int main(void)
{
    Year myyear;
    int a, g;
    printf("請輸入西元年份。\n");
    scanf("%d", &a);
    myyear.ad  = a;
    printf("西元爲%d年。\n", myyear.ad);
    printf("民國爲%d年。\n", myyear.gengo);

    printf("請輸入民國年份。\n");
    scanf("%d",&g);
    myyear.gengo = g;
    printf("民國爲%d年。\n", myyear.gengo);
    printf("西元爲%d年。\n", myyear.ad);
    return 0;
}
```

宣告共同空間型態

命名

宣告共同空間型態的變數

把值儲存至**myyear**的成員**ad**後…

成員**gengo**也變成相同的值

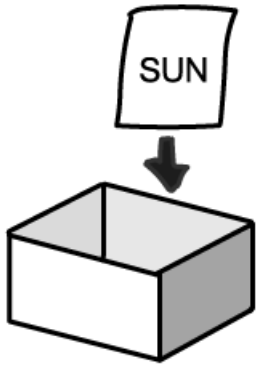
11-6 列舉型態

- 關於列舉型態
  - 宣告列舉型態的語法：

    enum 列舉型態名稱 {識別字1, 識別字2, 識別字3, …}
  - 列舉型態是一種可以把識別字當做值來儲存的型態，例如：

    enum Week{SUN, MON, TUE, WED, THU, FRI, SAT} Week;

- 指定列舉型態的數值
  - 列舉型態之所以能夠儲存識別字的值，是因爲其內部具有分開處理從0開始逐次加一之整數值的機制。



- 使用列舉型態之範例：

```
#include <stdio.h>
/* 列舉型態enum Week的宣告 */
typedef enum Week{SUN, MON, TUE, WED, THU, FRI, SAT} Week;

int main(void)
{
  Week w;
  w = SUN;

  switch(w){
    case SUN: printf("是星期天。\n"); break;
    case MON: printf("是星期一。\n"); break;
    case TUE: printf("是星期二。\n"); break;
    case WED: printf("是星期三。\n"); break;
    case THU: printf("是星期四。\n"); break;
    case FRI: printf("是星期五。\n"); break;
    case SAT: printf("是星期六。\n"); break;
    default: printf("不知道是星期幾。\n"); break;
  }
return 0;
}
```

宣告列舉型態

命名爲**Week**

宣告列舉型態的變數

可以儲存識別字的值

利用識別字的功能可使程式簡明易懂