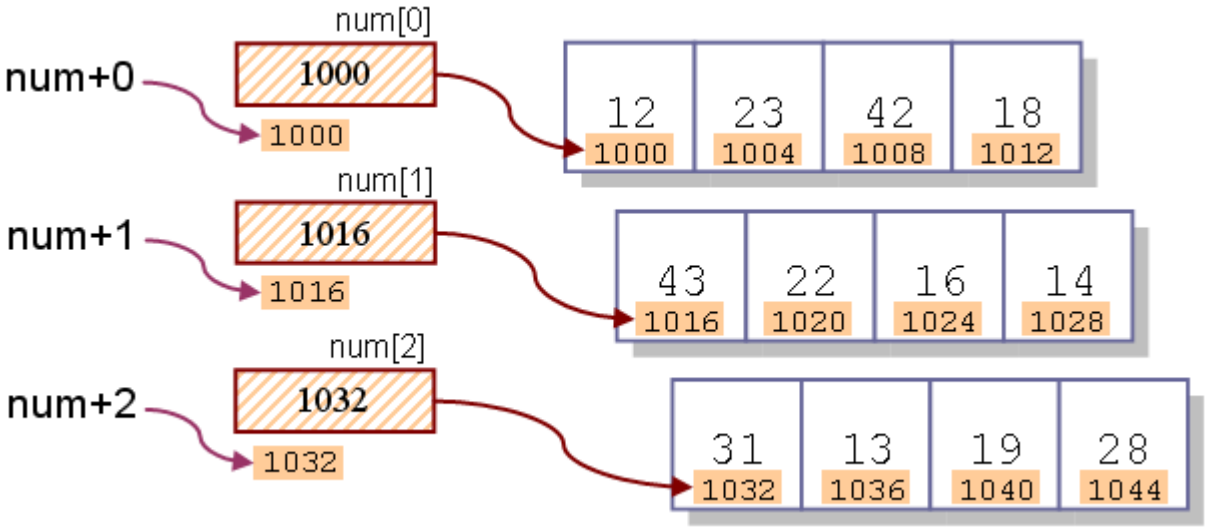
C 程式語言 6

二維陣列的指標表示方式



num+i 的值等於 num[i] 的值

第 m+1 列，第 n+1 行的位址： *(num+m)+n

第 m+1 列，第 n+1 行的元素： *(*(num+m)+n)

認識結構

- 「結構」（Structures）和聯合、列舉都屬於自訂資料型態（User-Defined Types），可以讓程式設計者自行在程式碼定義新的資料型態。

- 結構是由一或多個不同資料型態（當然也可以是相同資料型態）所組成的集合，然後使用一個新名稱來代表，新名稱就是一個新的資料型態，我們可以使用此新資料型態來宣告結構變數。

結構宣告與基本使用

- 在C程式宣告結構是使用struct關鍵字來定義新的資料型態，這是一個範本，其語法如下所示：

```
struct 結構名稱 {
    資料型態 成員1;
    資料型態 成員2;
    ……
};
```

- 上述語法定義名為結構名稱的新資料型態，在結構中宣告的變數稱為該結構的「成員」（Members）。

結構宣告與基本使用 – 宣告結構變數

- 在宣告student結構的自訂資料型態後，我們可以在程式碼使用此型態來宣告結構變數，也就是配置記憶體空間來建立「結構實例」（Structure Instance），其語法如下所示：

  struct 結構名稱 變數名稱;

- 上述宣告是使用struct關鍵字開頭加上結構名稱來宣告結構變數，例如：宣告student結構變數，如下所示：

  struct student std1;

認識結構

- 結構可將型態不同的資料合併成為新的型態
- 定義結構與宣告結構變數的格式如下：

定義結構與宣告結構變數的語法

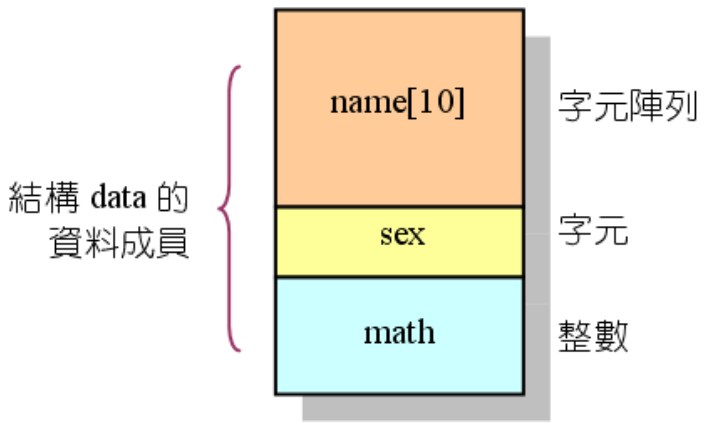
```
struct 結構名稱
{
   資料型態 成員名稱 1;
   資料型態 成員名稱 2;
                  ...
   資料型態 成員名稱 n;
};

struct 結構名稱 變數 1, 變數 2,…, 變數 n;
```

認識結構

- 結構定義的範例

```
struct data      /* 定義 data 結構*/
{
   char name[10];
   char sex;
   int math;
};
struct data mary, tom;    /* 宣告 data 型態的結構變數 */
```

定義結構的成員



定義完結構之後，立即宣告結構變數

```
struct data      /* 定義 data 結構*/
{
   char name[10];
   char sex;
   int math;
}mary,tom;    /* 宣告結構變數 mary 與 tom */
```

認識結構

- 存取結構變數的成員：

存取結構變數的成員

結構變數名稱.成員名稱;

```
mary.sex='F';                /* 設定 sex 成員為'F' */
mary.math=95;                /* 設定 math 成員為 95 */
```

使用結構的範例

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  struct data                    /* 定義結構data */
  {
    char name[10];
    int math;
  } student;                     /* 宣告data型態的結構變數student */
  printf("Please type in your name:");
  gets(student.name);
  printf("Please input a score:");
  scanf("%d",&student.math);          /* 輸入學生成績 */
  printf("Name:%s\n", student.name);
  printf("Score:%d\n", student.math);
  system("pause");
  return 0;
}
```

結構變數初值的設定

要設定結構變數的初值，可利用下面的語法：

```
struct data       /* 定義結構 data */
{
   char name[10];
   int math;
};
struct data student={"Jenny",78};    /* 設定結構變數的初值 */
```

將結構的定義與變數初值的設定合在一起：

```
struct data       /* 定義結構 data */
{
   char name[10];
   int math;
} student={"Jenny",78};    /* 宣告結構變數,並設定初值 */
```

結構變數初值的設定

- 設定結構變數初值的範例

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
  struct data              /* 定義結構data */
  {
    char name[10];
    int math;
  };
  struct data student={"Mary Wang",74};  /* 設定結構變數初值 */

  printf("Name: %s\n",student.name);
  printf("Score: %d\n",student.math);

  system("pause");
  return 0;
}
```

巢狀結構

- 結構內如有另一結構，則此結構稱為巢狀結構

巢狀結構的格式

```
struct 結構1
{
   /* 結構1的成員 */
};
struct 結構2
{
    /* 結構2的成員 */
    struct 結構1 變數名稱
};
```

結構陣列

- 下面為結構陣列的宣告格式：

結構陣列的宣告格式

struct 結構型態 結構陣列名稱[元素個數];

```
struct data s1[10];              /* 宣告結構陣列 s1 */

s1[2].math=12;                   /* 設定 s1[2].math=12 */
```

結構陣列的範例

- 利用sizeof() 計算結構陣列及其元素所佔的位元組：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  struct data                    /* 定義結構 */
  {
     char name[10];
     int math;
  }student[10];

  printf("sizeof(student[3])=%d\n",sizeof(student[3]));
  printf("sizeof(student)=%d\n",sizeof(student));
  system("pause");
  return 0;
}
```

指向結構的指標

假設於程式中定義如下的結構，並以指標ptr指向它：

```
struct data             /* 定義結構 data */
{
   char name[10];
   int math;
}student;               /* 宣告結構 data 型態之變數 student */

struct data *ptr;       /* 宣告指向結構 data 型態之指標 ptr */
ptr=&student;           /* 將指標 ptr 指向結構變數 student */
```

以指標指向的結構，必須以「->」存取其成員：

以結構為引數傳遞到函數

- 將結構傳遞到函數的格式：

將結構傳遞到函數

```
傳回值型態 函數名稱(struct 結構名稱 變數名稱)
{
  /* 函數的定義 */
}
```

傳遞到函數的範例

```
#include <stdio.h>
#include <stdlib.h>
struct data
{
  char name[10];
  int math;
};
void display(struct data);            /* 宣告函數display()的原型 */
int main(void) {
  struct data s1={"Jenny",74};        /* 設定結構變數s1的初值 */
  display(s1);                        /* 呼叫函數display()，傳入結構變數s1 */

  system("pause");
  return 0;
}

void display(struct data st)          /* 定義display()函數 */
{
  printf("Name: %s\n",st.name);
  printf("Math: %d\n",st.math);
}
```

union 聯合

- union 所定義的自訂型別稱為**共同空間型別**
  - 宣告的語法與 struct 一樣
  - 但定義在同一個共用空間型別中的所有成員，**都共用相同的記憶體空間**
- 語法與範例如下：

```
union 共同空間名稱 {
    資料型別 變數名稱;
    ......
};
```

這裡一定要有分號

宣告範例

```
union Paid {
    char CreditCard[21];
    char BankAccount[16];
    int  Cash;
} money;
```

列舉型態

- 列舉型態（enumeration）
  - 可以用某個有意義的名稱來取代較不易記憶的整數常數
- 列舉型態定義及宣告變數的格式：

列舉型態定義及宣告變數的格式

```
enum 列舉型態名稱
{
   列舉常數1,
   列舉常數2,
   ...
   列舉常數n
};
enum 列舉型態名稱 變數1, 變數2,…, 變數m;  /* 宣告變數 */
```

列舉型態的定義與變數的宣告

- 定義列舉型態與宣告變數的範例：

```
enum color   /* 定義列舉型態 color */
{
    red,
    blue,
    green
};
enum color shirt, hat; /* 宣告列舉型態 color 變數 shirt 與 hat */
```

- 定義完列舉型態後，立即宣告列舉型態的變數

```
enum color        /* 定義列舉型態 color */
{
    red,
    blue,        } 列舉常數
    green
} shirt,hat;     /* 定義列舉型態後，便立即宣告變數 shirt 與 hat */
```

-The End-