國立屏東大學 資訊工程學系 C++程式設計入門教材

9. 陣列

在許多真實的應用裡，程式主要的作用就是幫助人們進行各式各樣資料的處理，因此程式裡通常都會宣告一些變數來存放使用者所輸入的資料，並進行後續相關的資料處理。例如一個(受到學生們歡迎的)在學期末用來調整學生成績的程式，它可以宣告並使用名為score的變數來取得學生成績，並進行 score=sqrt(score)*10; 的運算。然而當我們面對大量資料處理需求時(例如有上百筆或上千筆資料要處理時)，難道也只能宣告上百個、上千個變數來處理嗎C++語言提供名為「陣列(Array)的資料結構，幫我們管理相同資料型態的多筆資料，對於有大量資料要處理的程式來說非常方便。本章將針對陣列的概念，以及其相關的操作方法分別加以說明。

9.1 何謂陣列?

陣列(Array)是程式設計時，相當簡單同時也相當重要的一種資料結構。基本上，陣列可視為一些相同型態的資料集合，同時在這個集合中每一筆資料都有一個唯一的編號，我們將這個編號稱為索引；透過索引，您就可以存取在集合中的特定資料。以下本章將就陣列的宣告及使用等細節做一說明，現在先讓我們來看看陣列在使用上需要瞭解的一些性質：

- 陣列中所存放的資料必須為同一種資料型態
- 存放在陣列中的內容被稱為元素(Element)
- 每一個在陣列中的元素都有一個唯一的索引值(Index)
- 索引值的範圍是由0開始，意即第一筆資料的索引值為0、第二筆為1、第三筆為2、餘依此類推。

現在讓我們假設有一個陣列，其中包含了五個學生的成績，請參考figure 1

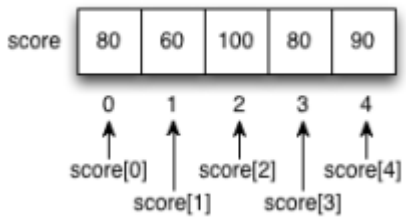

Fig. 1: 一個擁有5個元素的student陣列

出現在figure 1裡的陣列名稱為score它是由5個int整數所組成，也可以說這個陣列擁有5個元素。若要存取這5個元素，你只要透過陣列的索引(index有時亦稱為subscript)就可以完成，例如score陣列的第一個元素為score[0]第二個元素為score[1]...第五個元素為score[4]

為什麼我們需要這樣的資料結構呢？

> 五位學生的成績難道不可以使用五個變數來處理嗎？例如score1, score2, ..., score5? 😕
>
> > 假設需要處理的學生成績變成了50筆，又該如何呢?

> 那就宣告50個變數！
>
> > 變成500個？5000個？或是更多？
>
> 那…. 還是來好好地學習陣列吧！
>
> >

假設您需要記錄的是班上50位學生的成績，您當然可以宣告50個變數來儲存它們。下面的程式片段，將宣告50個變數並且設定它們的初始值：

```
//學生成績變數宣告
int score1=0;
int score2=0;
int score3=0;
…
int score49=0;
int score50=0;
```

這樣的寫法，您覺得如何呢? 是不是相當的難以使用呢? 現在讓我們來看看改用陣列後，同樣的程式會變成怎樣：

```
//學生成績變數宣告
int score[50];

//配合迴圈來設定初始值
int i;
for ( i=0;i<50;i++)
   score[i]=0;
```

上面的程式片段中，我們先宣告了一個陣列名為score[]然後再使用一個迴圈來設定每一個學生的成績。

現在我們要存取學生成績時，不用再透過變數score1[]score2[]…[]score50來存取；而是改由score[0][]score[1][]…[]score[49]來存取與操作學生的成績。其中最主要的差異在於，陣列是一組聚合式的資料，對其中某一筆資料操作時是透過陣列名稱再加上一個索引來完成的。索引值如果配合迴圈變數來操作，陣列的存取就變得更為容易了。以下為您詳細說明陣列的宣告、初始化、使用等主題。

9.2 一維陣列

在數學裡[]$\mathcal{a} = < a_0, a_1, \cdots, a_{n-1}>$[]其中$\mathcal{a}$表示一個數列，具有$a_0, a_1, \cdots, a_{n-1}$共n個相同值域(Domain)的元素。像這樣的數列就是C語言中的一維陣列，我們可以宣告一個陣列a[]並設定其具有n個相同資料型態的元素。

9.2.1 宣告

一維陣列的宣告語法如下：

一維陣列宣告語法

型態 陣列名稱[陣列大小];

請注意在陣列宣告語法中所出現的中括號[]，並不是代表可選擇性的語法構件，此處的中括號是必要的。

其中「型態(Type)」是指定存放在陣列中的資料為何種資料型態 — 一般變數可使用的資料型態，陣列就可以使用；「陣列大小(size)」則為陣列所要存放的元素個數。以下是一些不同資料型態的陣列宣告：

```
int score[50];
float data[450];
double dist[50];
char c[10];
```

為了程式日後易於維護與修改，有時我們會配合#define這個前置處理器指令來設定陣列的大小：

```
#define N 500

...
int main()
{
   int score[N];
   ...
   for(i=0;i<N;i++)
   {
      score[i]=0;
   }
}
```

9.2.2 初始化

陣列的初始化的方式有兩種，您可以在宣告時就設定預設的值，當然也可以在宣告後才另行設定其值。宣告陣列時設定初始值的語法如下：

一維陣列宣告初始化語法

型態 陣列名稱[陣列大小] = { 數值組 };

型態 陣列名稱[] = { 數值組 };

其中「數值組」的部份即為欲設定的資料數值，若(通常都是)超過一筆以上的資料，請在任兩筆資料間都加上一個逗號','分隔。請參考下面的例子：

```
int score[5]={80, 60, 100, 80, 90};

int score[5]={80, 60, 100}; //所給定的資料數值少於所宣告的元素個數，以0填補。
          //等同於int score[5]={80, 60, 100, 0, 0};

int score[5]={0};   // 設定所有元素皆為0

int score[5]= {};   // 這行是錯誤的，不允許{}中一筆數值都不給定

int score[] = {80, 60, 100, 80, 90}  // 陣列的大小被省略，由初始值的個數決定

char data[3] = { 'a', 'b', 'c' }; //宣告並設定char陣列
```

從上面的例子可以發現，若在宣告陣列的同時給定其初始值，編譯器會幫自動我們計算陣列的大小。

9.2.3 存取元素

陣列的宣告與初始化您都學會之後，接下來就是如何使用它。要存取陣列內的元素，只要以陣列名稱加上一組中括號，並於其中指定索引值即可，其語法如下：

存取一維陣列元素的語法

陣列名稱[索引值運算式]

其中的索引值運算式是用以索引存取陣列特定元素的運算式，其運算結果必須為整數且應該要小於陣列大小。以score陣列為例，下列都是正確的使用方式：

```
x=score[3];
```

```
score[2]=x;

score[x]=5;

score[x]=score[y]+2;

x=score[i>j?i:0];
```

9.2.4 應用

請看下面的例子：

```
sum2score = score[3] + score [4]; //將陣列的第4個元素和第5個元素相加

//搭配迴圈, 讓使用者輸入陣列的值
for(i=0;i<5;i++)
{
    cout << "Please input score #" << i+1;
    cin >> a[i];
}

//全班同學一律加10分
for(i=0;i<5;i++)
   score[i]+=10;
//求全班平均成績
for(i=0;i<N;i++)
{
   sum+=score[i];
}
average = sum /(float)N;

//求全班最高分
max=score[0];
for(i=1;i<N;i++)
{
   if( max < score[i] )
      max = score[i];
}
printf("max=%d\n", max);
```

```
//設定一付撲克牌
```

```
// 0-12 spade
// 13-25 heart
// 26-38 diamond
// 39-51 club
// 0 for A, 1 for 2, ..., 8 for 9, 9 for 10, 10 for J, 11 for Q and 12 for K

#include <iostream>
#include <cstdlib>
#include <ctime>


int main()
{
   int cards[52];
   char suits[]= {'S', 'H', 'D', 'C'};
   int i, pivot, point, suit, temp;

   for(i=0;i<52;i++)
   {
      cards[i]=i;
   }

   srand(time(NULL)); // 呼叫定義在C語言的stdlib.h裡的srand()函式, 以設定隨機數的種子數
         // 並呼叫定義在C語言的time.h裡的time()函式, 做為隨機數的種子數

   for(i=0;i<52;i++)
   {
      pivot = rand()%52;
      temp = cards[i];
      cards[i] = cards[pivot];
      cards[pivot] = temp;
   }

   for(i=0;i<52;i++)
   {
      suit = cards[i]/13;
      point = cards[i]%13+1;
      // char points[] = {'A', '2', '3', '4', ...., 'T', 'J', 'Q', 'K' };
      cout << suits[suit];

      //cout << points[point-1];

      switch(point)
      {
         case 2: case 3: case 4: case 5: case 6: case 7: case 8: case 9:
            cout << point;
            break;
         case 1:
            cout << 'A';
            break;
```

```
        case 10:
           cout << 'T';
           break;
        case 11:
           cout << 'J';
           break;
        case 12:
           cout << 'Q';
           break;
        case 13:
           cout << 'K';
           break;
     }
     if( (i+1)%13 ==0)
        cout << endl;
     else
        cout << ", ";
   }
}
```

9.3 多維陣列

除了一維陣列外□C++語言也支援多維度的陣列(Multidimensional Array)□

9.3.1 宣告與初始化

以二維陣列為例，其宣告語法如下：

二維陣列宣告初始化語法

型態 陣列名稱[size1][size2];

|共size2個|
型態 陣列名稱[size1][size2]={ { 數值組 }, {數值組}, ⋯, {數值組} };

|————— 共計size1個 ————–|

其中size1與size2分別為第一個維度與第二個維度的大小，我們也可以在宣告後直接給定初始值，但size2不可省略。

現在考慮前面介紹過的範例，宣告一個陣列用以記載5個學生的成績，其中每個學生有國文與英文兩個科目的成績，有兩種思考的方式：

1. int score[2][5]; *兩個科目，每個科目有五個學生的成績* - *int score[5][2];* 五個學生，每個學生有兩個科目的成績

兩種方式都是正確地，要使用哪一種取決於您慣用的思考方式。我們以第一種方式為例，在宣告的同時，順便給定初始值：

```
int score[2][5] = { { 80, 60, 100, 80, 90 }, {100, 60, 90, 80, 75} };
```

要特別注意下面的寫法是不正確的!

```
int score[][] = { { 80, 60, 100, 80, 90 }, {100, 60, 90, 80,
75} };
```

在程式執行時，會得到如figure 2的陣列：

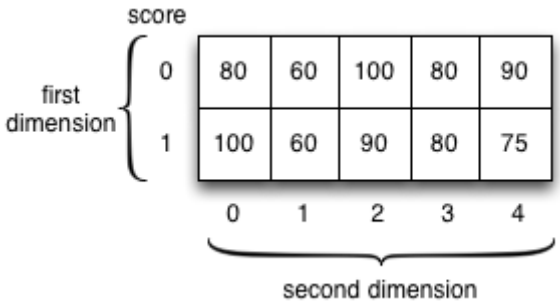

Fig. 2: 二維陣列範例(兩個科目，每個科目有五個學生的成績)

figure 2是我們所認為的二維陣列的長相，其實在記憶體中並不是這樣的。請參考下面的程式碼，把陣列所有元素的記憶體位址印出：

```
cout << "
cout << "&score = " << showbase << hex << &score << endl;
cout << "&score[0] = " << showbase << hex << &score[0] << endl;
cout << "&score[1] = " << showbase << hex << &score[1] << endl;

for(i=0;i<2;i++)
   for(j=0;j<5;j++)
      cout << "&score[" << i << "][" << j << "]=" << showbase << hex <<
&score[i][j] << endl;
```

以下是可能的輸出結果(注意，我們假設一個整數為4個bytes，亦即32bits)：

```
&score=0x7fff50f59970
&score[0]=0x7fff50f59970
&score[1]=0x7fff50f59984
&score[0][0]=0x7fff50f59970
&score[0][1]=0x7fff50f59974
&score[0][2]=0x7fff50f59978
&score[0][3]=0x7fff50f5997c
&score[0][4]=0x7fff50f59980
&score[1][0]=0x7fff50f59984
&score[1][1]=0x7fff50f59988
&score[1][2]=0x7fff50f5998c
&score[1][3]=0x7fff50f59990
&score[1][4]=0x7fff50f59994
```

從上述的輸出結果可得知，score陣列被配置到0x7fff50f59970位址，也就是score[0][0]所在的位址，當然score[0]這一列(row)也是從這個位址開始。再仔細看一下結果，因為陣列元素的型態為int(大小為4bytes)，所以score[0][0]用以存放一個整數的記憶體空間是位於0x7fff50f59970 - 0x7fff50f59973，緊接的下一個位址0x7fff50f59974正好就是score[0][1]的位址，再加4之後，0x7fff50f59978就是score[0][2]的位置。依此類推，可得知：

```
&score[0][j] =  &score[0] + j*4
```

請檢查一下輸出結果與上述公式是否一致，score[0][4]是score[0]這一列的最後一個元素，其所在位址為0x7fff50f59980，觀察輸出的結果，下一列score[1]的元素是存放於0x7fff50f59984這個位址，也就是說其實它是連續的空間配置，請參考figure 3。

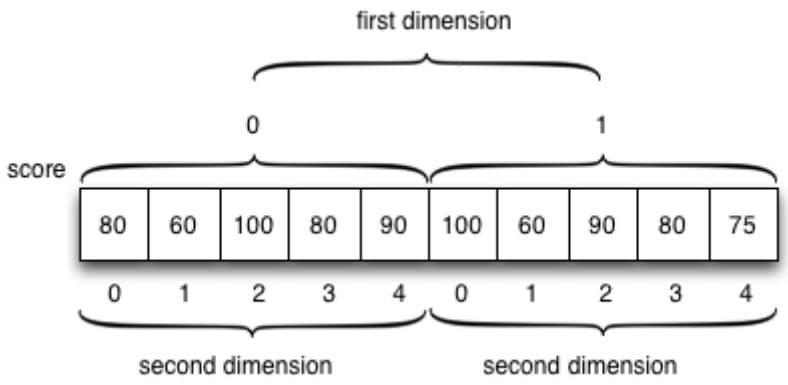

Fig. 3: 二維陣列的記憶體空間是連續配置的

所以score[i][j]的位址可以由以下公式計算：

```
&score[i][j] = &score + (i*size2*4) + j*4;

或

&score[i][j] = &score + (i*size2+j)*sizeof(score[0][0]);
```

雖然我們現在已瞭解了陣列在記憶體中實際的配置為連續空間配置，但平常在使用陣列時不需要使用如figure 3這種與我們日常生活中的概念不一致的方式思考。建議還是以figure 2的方式構思即可。

其它更多維度的陣列也是類似的觀念，下面我再舉一個三維陣列的例子：宣告一個陣列用以記載5個學生的成績，其中每個學生有國文與英文兩個科目的成績，每個成績又可分成平時成績與考試成績，參考下面的程式宣告：

```
int score[2][5][2] = { { {80, 90}, {60,50} , {100,95}, {80,90}, {90,85} },
                       { {100,95}, {60,100}, {90,100}, {80,90}, {75,65} }
};
```

figure 4與figure 5顯示了這個例子的概念圖。

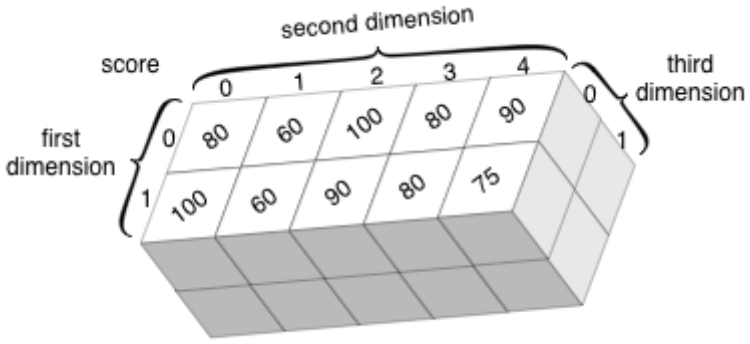

Fig. 4: 三維陣列範例(兩個科目，每個科目有五個學生的成績，每個學生又有平時成績與考試成績)

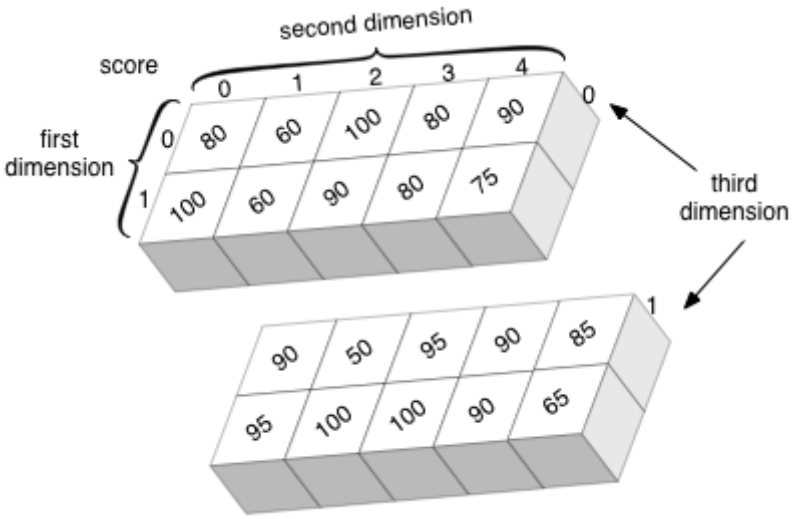

Fig. 5: 三維陣列範例(依第三維度分層))

考慮一個三維陣列宣告為 int a[10][5][2]，您能夠導出計算a[i][j][k]所在的記憶體位址的公式嗎？

9.3.2 應用

多維陣列的應用很廣泛，以下是一些例子：

- 成績處理(多個學生、多個科目、多個成績項目等)
- 棋奕遊戲(利用二維陣列來定義棋盤)
  - 井字遊戲(3×3陣列)
  - 五子棋(19×19陣列)
  - ...
- 影像處理
  - 以640×480解析度
  - 每個像素須包含R, G, B數值
  - 每個數值可能為0-255
- 還有更多…

9.3.3 排序應用

```
#include <iostream>
using namespace std;

int main()
{
  int score[10]={ 10, 33, 13, 60, 65, 25, 100, 34, 99, 0};
```

```
  int i, j;
  int max, maxIndex;

  for(i=0;i< 10;i++)
  {
    maxIndex=0;
    for(j=1;j<10;j++)
    {
      if(score[maxIndex] < score[j])
      {
        maxIndex = j;
      }
    }
    cout << score[maxIndex];
    if(i!=9)
        cout << " > ";
    score[maxIndex]=(-1);
  }
  cout << endl;
  for(i=0;i<10;i++)
  {
    cout << score[i] << " ";
  }
  cout << endl;
}
```

```
#include <iostream>
using namespace std;

int main()
{
  int score[10]={ 10, 33, 13, 60, 65, 25, 100, 34, 99, 0};
  int flag[10] ={0};
  int i, j;
  int max, maxIndex;

  for(i=0;i< 10;i++)
  {
    maxIndex=0;
    for(j=0;j<10;j++)
    {
      if(flag[j]!=1)
      {
        maxIndex=j;
        break;
      }
    }
```

```
    for(j=1;j<10;j++)
    {
      if((score[maxIndex] < score[j])&&(flag[j]!=1))
      {
        maxIndex = j;
      }
    }
    cout << score[maxIndex];
    if(i!=9)
      cout << " > ";
    flag[maxIndex]= 1;
  }
  cout << endl;
  for(i=0;i<10;i++)
  {
    cout << score[i] << " ";
  }
  cout << endl;
}
```

```
#include <iostream>
using namespace std;

#define N 10

int main()
{
  int score[N]={ 10, 33, 13, 60, 65, 25, 100, 34, 99, 0};
  int i, j, temp;
  int max, maxIndex;

  for(i=0;i<N-1;i++)
  {
    maxIndex=i;
    for(j=i+1;j<N;j++)
    {
      if(score[maxIndex] < score[j])
      {
        maxIndex = j;
      }
    }
    cout << score[maxIndex];
    if(i!=N-1)
       cout << " > ";
    temp=score[i];
    score[i]=score[maxIndex];
    score[maxIndex]=temp;
```

```
  }
  cout << score[N-1] << endl;

  for(i=0;i<10;i++)
  {
    cout << score[i] << " ";
  }
  cout << endl;
}
```

From:
https://junwu.nptu.edu.tw/dokuwiki/ - **Jun Wu的教學網頁**
國立屏東大學資訊工程學系
CSIE, NPTU
Total: 195503

Permanent link:
https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cppbook:ch-array

Last update: **2024/03/21 05:13**