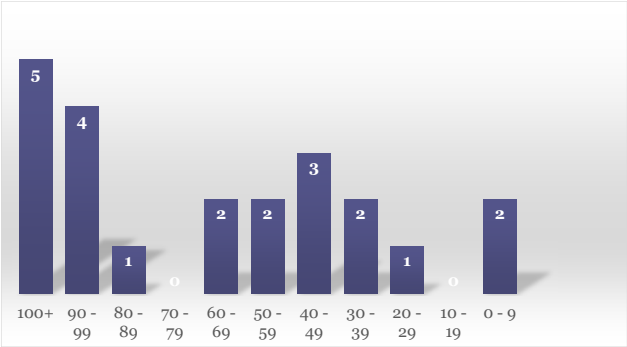
成績分佈



Lecture 8

一維陣列
陣列初始化

2

今日內容

- 一維陣列
- 陣列初始化

3

一維陣列

One-dimensional Array

4

5

一維陣列 (One-dimensional Array)

- 變數 (variable)
  - 名稱 – 用來區別不同的資料
    - 當我們需要儲存 20 個資料時需要 20 不同的變數名稱
    - 如果我們需要儲存 200 個資料的時候呢？20000 個呢？
  - 型別 – int, double, short, long, float, ...
- 當一個程式被設計需要處理不同資料量時，「通常」無法只透過簡單變數作運算，因為需要儲存的資料數量不同→這時可使用陣列來協助。
  - e.g. 成績計算程式，需要考慮班級大小
- 另外，有時候程式裡的資料存取有一定的規則性 (e.g. 前一筆資料、下一筆資料、第 k 筆資料等等)，陣列的資料儲存方式使得這規則性容易被實現！

6

陣列 (Array) 的宣告

- 陣列為多個**同一型態**變數之組合，且此組合佔據**連續的**記憶體區塊
- 陣列的宣告
  - *變數型別 陣列名稱[陣列容量];*
  - **char name [200];**
    - 宣告一陣列, 可存放 200 個字元型別的資料
      - name[0], name[1], name[2], name[3], ..., name[199]
    - 此陣列的容量為 200 個元素 (elements)，每一個元素即可視為一個獨立的變數。
  - **int cars [100];**
    - 宣告一整數陣列, 可存放 100 個整數型別的資料
    - cars[0], cars[1], cars[2], ..., cars[99]
- 透過陣列宣告，我們可以快速得到大量的變數空間供程式使用
- 注意到 C/C++ 語言的陣列是以索引 0 (zero-based) 為第一個元素。

7

陣列 (Array) 的使用

- int A[10];
  - 宣告一整數陣列, 名稱為 A，可存放 10 個整數型別的資料

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

- 使用
  - A[0] = 3;→ 將 3 存入 A 陣列中的第 0 號元素
  - A[3]++; → 將 A 陣列中的 3 號元素增加 1
  - A[2]=A[4]+a;
  - 中括弧 [] 中的數字 (0, 3, 2, 4, ...) 用來選擇使用陣列中的那一個元素，被稱為索引 (index) 或下標 (subscript)
  - 常和 **for** 迴圈配合
  - 假設一陣列有 n 個元素，則可以**正確使用**的索引值為 0, 1, 2, ... n-1

8

7-1.cpp

```
data[ 0] = 30 -  0 * 2;
data[ 1] = 30 -  1 * 2;
data[ 2] = 30 -  2 * 2;
…
data[14] = 30 - 14 * 2;

e.g. data_i = 30 - i*2
```

```
int main() {
  int data[15];

  for(int i=0;i<15;i++) {
    data[i] = 30-i*2;
  }
  for(int j=0;j<15;j++) {
    cout << j << ":" << data[j] << "\n";
  }
  for(int j=14;j>=0;j--) {
    cout << data[j] << " ";
  }
  return 0;
}
```

```
0:30
1:28
2:26
3:24
4:22
5:20
6:18
7:16
8:14
9:12
10:10
11:8
12:6
13:4
14:2
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

9

7-2.cpp

```
int a[10];

for(int i=0;i<10;i++) {
  cout << "請輸入 " << i << " 號元素: ";
  cin >> a[i];
}

for(int i=9;i>=0;i--) {
  cout << a[i] << ", ";
}
```

```
請輸入 0 號元素: 1
請輸入 1 號元素: 2
請輸入 2 號元素: 3
請輸入 3 號元素: 4
請輸入 4 號元素: 5
請輸入 5 號元素: 6
請輸入 6 號元素: 7
請輸入 7 號元素: 8
請輸入 8 號元素: 9
請輸入 9 號元素: 10
10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
```

10

7-3.cpp

```
double a[10];

for(int i=0;i<10;i++) {
  cout << "請輸入" << i << " 號元素: ";
  cin >> a[i];
}
for(int i=0;i<10;i++)
  cout << a[i] << " ";
cout << endl;
cout << a[0] << " ";
for(int i=1;i<10;i++) {
  cout << (a[i]+a[i-1]) * 0.5 << " ";
}
```

```
請輸入 0 號元素: 2
請輸入 1 號元素: 4
請輸入 2 號元素: 6
請輸入 3 號元素: 8
請輸入 4 號元素: 10
請輸入 5 號元素: 12
請輸入 6 號元素: 14
請輸入 7 號元素: 16
請輸入 8 號元素: 18
請輸入 9 號元素: 20
2 4 6 8 10 12 14 16 18 20
2 3 5 7 9 11 13 15 17 19
```

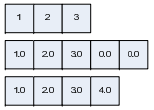
11

陣列初始化

12

陣列的初始化 (initialization)

- 變數在宣告/定義時，可給初始值。注意到此處的等號**不是**指定運算子，而是作初始化或是初始值設定。
  - `int a=5, b=3;`
  - `double c=20;`
- 陣列亦可在宣告時，給定陣列中各元素的**初始值**。
  - `int a[3] = {1, 2, 3};`
  - `double b[5] = {1, 2, 3};`
  - `float c[] = {1, 2, 3, 4};`
  - `double d[10] = {0};`
  - `char f[5] = {'H', 'E', 'L', 'L', 'O'}`



13

7-4.cpp

有了陣列以後，以此例來說寫程式和寫數學上的數列寫是不是很像呢?
(特別是和 5-11.cpp 比起來)

$$a_i = a_{i-1} + a_{i-2}$$

```
#include <iostream>
using namespace std;
int main() {
  double a[20] = {1, 1};

  for(int i=2;i<20;i++) {
      a[i] = a[i-1] + a[i-2];
  }

  for(int i=0;i<20;i++) {
      cout << a[i] << " ";
  }
  return 0;
}
```

```
a[2] =  a[1] + a[0];
a[3] =  a[2] + a[1];
a[4] =  a[3] + a[2];
.
.
.
a[19] = a[18] +a[17];
```

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

14

7-5.cpp

```
#include <iostream>
using namespace std;
int main() {
  float a[10]={1.1f, 2.2f, 3.3f, 4.4f};

  for(int i=0;i<10;i++) {
      cout << a[i] << ", ";
  }

  return 0;
}
```

```
1.1, 2.2, 3.3, 4.4, 0, 0, 0, 0, 0, 0,
```

- 只要陣列有進行初始化，則該陣列中未賦予初始值的元素會被給予 0。
- 一般而言，陣列若沒有初始值，其內的數值未知。

15

小結

- 陣列讓我們的程式有儲存/處理大量資料的能力!
- 陣列讓我們在寫程式時，容易將規則性的資料存取表現出來。
- 過去工數裡學的數列 $a_0, a_1, a_2, \ldots a_i, \ldots$ 也可以很容易對應到陣列的存取。
  - 等差級數: $a_i = a_{i-1}+d$
  - 等比級數: $a_i = r\, a_{i-1}$

隨堂練習

請撰寫一程式，請使用者輸入20個數字，輸入完後找出其中的最大值、最小值、偶數個數與奇數個數。

```
請輸入 20 個數字 1 20 2 19 3 18 4 17 5 16 6 15 7 14 8 13 9 12 10 11

最大值 : 20
最小值 : 1
偶數個數 : 10
奇數個數 : 10
```

17

作業

請撰寫一程式，讓使用者不斷輸入介於 1-6 之間的數字，輸入數字 0 代表輸入結束。輸入結束後，輸出1至6之間的每個數字出現次數，以及所輸入數字的平均值。

p.s. 若使用者輸入非0-6之間的數字，其不考慮於後續的計算。

```
請輸入一介於 1-6 的數字， 0 結束輸入。：1
請輸入一介於 1-6 的數字， 0 結束輸入。：2
請輸入一介於 1-6 的數字， 0 結束輸入。：3
請輸入一介於 1-6 的數字， 0 結束輸入。：4
請輸入一介於 1-6 的數字， 0 結束輸入。：3
請輸入一介於 1-6 的數字， 0 結束輸入。：2
請輸入一介於 1-6 的數字， 0 結束輸入。：1
請輸入一介於 1-6 的數字， 0 結束輸入。：0

1 出現 2 次
2 出現 2 次
3 出現 2 次
4 出現 1 次
5 出現 0 次
6 出現 0 次
平均值：2.28571
```

18

```
請輸入一介於 1-6 的數字， 0 結束輸入。：1
請輸入一介於 1-6 的數字， 0 結束輸入。：2
請輸入一介於 1-6 的數字， 0 結束輸入。：3
請輸入一介於 1-6 的數字， 0 結束輸入。：3
請輸入一介於 1-6 的數字， 0 結束輸入。：2
請輸入一介於 1-6 的數字， 0 結束輸入。：4
請輸入一介於 1-6 的數字， 0 結束輸入。：4
請輸入一介於 1-6 的數字， 0 結束輸入。：4
請輸入一介於 1-6 的數字， 0 結束輸入。：5
請輸入一介於 1-6 的數字， 0 結束輸入。：0

1 出現 1 次
2 出現 2 次
3 出現 2 次
4 出現 3 次
5 出現 1 次
6 出現 0 次
平均值：3.11111
```