

為何需要使用陣列?

- 如果我需要寫一個程式計算班上的平均成績
鍵盤 → 變數 → 加減乘除 → 螢幕
- 變數:

```
int number_of_students=5;
int score1, score2, score3, score4, score5;
double sum, average;
```
- 鍵盤輸入:

```
scanf("%d%d%d%d%d", &score1, &score2, &score3,
&score4, &score5);
```
- 加法:

```
sum = score1 + score2 + score3 + score4 + score5;
```
- 除法:

```
average = sum / number_of_students;
```
- 輸出:

```
printf("平均成績為 %f\n", average);
```
- 如果需要用那麼多 score## 變數, 怎麼處理一整班 70 個人呢?

1

重複使用變數與迴圈

- 變數

```
int i, number_of_students=5;
int score;
double sum=0., average;
```
- 迴圈

```
for (i=0; i<number_of_students; i++)
{
```
- 鍵盤輸入

```
scanf("%d", &score);
sum = sum + score;
}
```
- 輸出:

```
average = sum / number_of_students;
printf("平均成績為 %f\n", average);
```
- 計算變異數: $\frac{\sum_{i=1}^n (s_i - \bar{s})^2}{n-1}$ 再輸入一次所有成績?

2

鍵盤 VS. 檔案 VS. 記憶體

- 變數:

```
int i, number_of_students=5;
int score;
double sum=0., average, sumsq=0., variance;
```
- ... 第一次迴圈計算 sum, average ...
- 第二次迴圈:

```
for (i=0; i<number_of_students; i++)
{
```
- 再一次鍵盤輸入:

```
scanf("%d", &score);
sumsq += (score-average)*(score-average);
}
```
- 再輸入一次所有成績! 不會吧! Copy&paste 勉強可接受

3

鍵盤 VS. 檔案 VS. 記憶體

- 變數:

```
int i, number_of_students=5, score;
FILE *infile;
double sum=0., average, sumsq=0., variance;
infile = fopen("data.txt", "r");
```
- ... 第一次迴圈讀取檔案資料, 計算 sum, average ...

```
rewind(infile); /* 重新翻回檔案起始 */
for (i=0; i<number_of_students; i++) { /* 第二次迴圈 */
```
- 檔案輸入:

```
fscanf(infile, "%d", &score);
sumsq += (score-average)*(score-average);
}
```
- 檔案只能循序讀取!
- 好一點, 可是需要藉由檔案輸入, 如果無法存取檔案!

4

鍵盤 vs. 檔案 vs. 記憶體

- 有 4GB/8GB 的記憶體, 為什麼由鍵盤讀進來以後不記住呢?
- 變數: 如何方便地定義很多變數呢?
int i, number_of_students=5;
int **score[5]**;
double sum=0., average, sumsq=0., variance;
- 第一次迴圈讀取資料, 計算 sum, 計算 average
for (i=0; i<number_of_students; i++) {
scanf("%d", &**score[i]**);
sum += **score[i]**;
}
average = sum / number_of_students;
- 第二次迴圈由記憶體直接讀取成績:
for (i=0; i<number_of_students; i++)
sumsq += (**score[i]**-average)*(**score[i]**-average);
variance = sumsq / (number_of_students-1);

$$\frac{\sum_{i=1}^n (s_i - \bar{s})^2}{n-1}$$

5

真的有差別嗎?

- 平均隨機存取速度: 1000 倍
硬碟 0.05~0.1 微秒(μ s)/Byte
記憶體 0.1~1 奈秒(ns)/Byte
- 前面這個計算變異數的程式中, 每一個資料只依序使用到兩次, 但是很多程式裡面資料需要以任意順序重複使用很多次
 - 由於媒體的物理特性, 磁碟或是磁帶上檔案存取的基本模型是順序存取 (固態硬碟的限制不一樣)
 - 記憶體可以隨機存取 (以任意順序讀寫)
 - 迴圈和陣列是好朋友

6

什麼時候需要/可以使用陣列?

- 變數 score1, score2, score3, ... 有相同的特性
 - 相同資料型態, e.g. int
 - 儲存的資料具有相似的範圍, e.g. 0~100
 - 儲存的資料具有一致的意義, e.g. 成績
 - 儲存的資料都參與同樣的運算, e.g., scanf, +, *
 - 儲存的資料需要一起傳遞, e.g. 作為函式參數
- 為了使你的程式能夠有彈性地處理不同個數的資料 (scalable), 你須要使用 **陣列** 語法
- 生活中的陣列: book pages, bus numbers, ...

7

陣列的定義與使用

- 陣列是一個以上在記憶體裡面相鄰的資料變數的集合體, 具有一個共同的名稱
- 同時定義陣列的名稱與元素的個數
double x[8];
- 如何使用陣列中個別的元素?
 - e.g. **x[0]**, **x[i+1]**, ...

8

陣列定義與使用

```
double x[8];
```

一次定義 8 個變數

Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

Statement	Explanation
<code>printf("%.1f", x[0]);</code>	Displays the value of <code>x[0]</code> , which is 16.0
<code>x[3] = 25.0;</code>	Stores the value 25.0 in <code>x[3]</code>
<code>sum = x[0] + x[1];</code>	Stores the sum of <code>x[0]</code> and <code>x[1]</code> , which is 28.0 in the variable <code>sum</code>
<code>sum += x[2];</code>	Adds <code>x[2]</code> to <code>sum</code> . The new <code>sum</code> is 34.0
<code>x[3] += 1.0;</code>	Add 1.0 to <code>x[3]</code> . The new <code>x[3]</code> is 26.0
<code>x[2] = x[0] + x[1];</code>	Stores the sum of <code>x[0]</code> and <code>x[1]</code> in <code>x[2]</code> . The new <code>x[2]</code> is 28.0

9

陣列定義語法與初始化

➤ 語法:

```
element-type array_name[size];
```

↑
C99 之前、

需要是 編譯時的常數

➤ Example:

```
#define A_SIZE 5
```

```
double a[A_SIZE];
```

➤ Example: (陣列定義以及 初始化)

```
char vowels[] = {'A', 'E', 'I', 'O', 'U'};
```

或

```
char vowels[5] = {'A', 'E', 'I', 'O', 'U'};
```

10