找出 n 個數字的最大值
與計算 n 個數字的總和

丁培毅

練習目標：

1. 簡化題目的要求
2. 漸進式地完成所有的要求
3. 掌握 for 迴圈的應用時機
4. 練習 for 迴圈的語法, 瞭解各部份執行的順序
5. 體會迴圈如何有效運用電腦的運算能力

找出 n 個數字裡的最大值

請撰寫一個程式

- 讀取下列的整數輸入 (n>0)

  n a_1 a_2 … a_n
- 計算並且印出 {a_1, a_2, …, a_n} 裡面的最大值

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

2. 程式需要讀這兩個整數進來，放到整數變數 a 與 b 裡

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

2. 程式需要讀這兩個整數進來，放到整數變數 a 與 b 裡

```
int a, b;
scanf("%d%d", &a, &b);
```

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

2. 程式需要讀這兩個整數進來，放到整數變數 a 與 b 裡

```
int a, b;
scanf("%d%d", &a, &b);
```

3. 接下來需要判斷那一個數字比較大, 列印它

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

2. 程式需要讀這兩個整數進來，放到整數變數 a 與 b 裡

```
int a, b;
scanf("%d%d", &a, &b);
```

3. 接下來需要判斷那一個數字比較大, 列印它

```
if (a>=b)
    printf("%d\n", a);
else
    printf("%d\n", b);
```

分析:

1. 簡化一下要求, 如果輸入只有兩個數字, 例如: 7 4, 希望印出最大值

2. 程式需要讀這兩個整數進來，放到整數變數 a 與 b 裡

```
int a, b;
scanf("%d%d", &a, &b);
```

3. 接下來需要判斷那一個數字比較大, 列印它

```
if (a>=b)
    printf("%d\n", a);
else
    printf("%d\n", b);
```

4. 要設計處理一般化資料的程式, 可以逐步擴增程式的能力, 如果輸入有三個數字, 例如: 7 4 9, 希望印出最大值, 很直覺地擴充上面的程式, 設計三個變數來存放資料

```
int a, b, c;
scanf("%d%d%d", &a, &b, &c);
```

5. 直接擴充時, 條件判斷式
變得比較複雜了

5. 直接擴充時, 條件判斷式變得比較複雜了

```
if ((a>=b)&&(a>=c))
    printf("%d\n", a);
else if ((b>=a)&&(b>=c)
    printf("%d\n", b);
else
    printf("%d\n", c);
```

5. 直接擴充時, 條件判斷式變得比較複雜了

6. 要處理到 n 筆資料時如果像上面一樣需要 **n** 個變數來紀錄並且比較大小, 這樣子的程式會有點困難：

```
if ((a>=b)&&(a>=c))
    printf("%d\n", a);
else if ((b>=a)&&(b>=c)
    printf("%d\n", b);
else
    printf("%d\n", c);
```

5. 直接擴充時, 條件判斷式變得比較複雜了

```
if ((a>=b)&&(a>=c))
    printf("%d\n", a);
else if ((b>=a)&&(b>=c)
    printf("%d\n", b);
else
    printf("%d\n", c);
```

6. 要處理到 n 筆資料時如果像上面一樣需要 n 個變數來紀錄並且比較大小, 這樣子的程式會有點困難：

   A. 要設計 n 個變數在程式裡還不算麻煩, 但是需要使用陣列或是動態配置的語法

   B. 真正的困難應該是條件判斷式的複雜度隨著 n 增加

5. 直接擴充時, 條件判斷式
變得比較複雜了

```
if ((a>=b)&&(a>=c))
    printf("%d\n", a);
else if ((b>=a)&&(b>=c)
    printf("%d\n", b);
else
    printf("%d\n", c);
```

6. 要處理到 n 筆資料時如果像上面一樣需要 n 個變數來紀錄並且比較大小, 這樣子的程式會有點困難：

A. 要設計 n 個變數在程式裡還不算麻煩, 但是需要使用陣列或是動態配置的語法

B. 真正的困難應該是條件判斷式的複雜度隨著 n 增加

因此需要換一個找最大值的作法, 如果你看到1000個小朋友排隊走過來, 想要找出裡面最高的, 空間不夠沒有辦法通通留下來比較高矮, 你有可能會把到目前為止最高的留下, 每一個比他矮的就讓他走過去, 比他高的就留下來取代原來的

5. 直接擴充時, 條件判斷式變得比較複雜了

```
if ((a>=b)&&(a>=c))
    printf("%d\n", a);
else if ((b>=a)&&(b>=c)
    printf("%d\n", b);
else
    printf("%d\n", c);
```

6. 要處理到 n 筆資料時如果像上面一樣需要 n 個變數來紀錄並且比較大小, 這樣子的程式會有點困難：

A. 要設計 n 個變數在程式裡還不算麻煩, 但是需要使用陣列或是動態配置的語法

B. 真正的困難應該是條件判斷式的複雜度隨著 n 增加

因此需要換一個找最大值的作法, 如果你看到1000個小朋友排隊走過來, 想要找出裡面最高的, 空間不夠沒有辦法通通留下來比較高矮, 你有可能會把到目前為止最高的留下, 每一個比他矮的就讓他走過去, 比他高的就留下來取代原來的

```
int max, a, b, c;
scanf("%d%d%d", &a, &b, &c);
max = a;
if (b > max) max = b;
if (c > max) max = c;
```

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

```
int max, a, b, c;
scanf("%d", &a);
max = a;
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &c);
if (c > max) max = c;
```

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

8. 既然可以在需要的時候再讀進變數裡, 比對完裡面的資料就可以丟棄, 所以不需要那麼多變數了, 只需要 max 和 b, 繼續擴充下去時也只要複製最後兩列程式

```
int max, a, b, c;
scanf("%d", &a);
max = a;
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &c);
if (c > max) max = c;
```

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

8. 既然可以在需要的時候再讀進變數裡, 比對完裡面的資料就可以丟棄, 所以不需要那麼多變數了, 只需要 max 和 b, 繼續擴充下去時也只要複製最後兩列程式

```
int max, a, b, c;
scanf("%d", &a);
max = a;
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &c);
if (c > max) max = c;
```

```
int max, b;
scanf("%d", &max);
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &b);
if (b > max) max = b;
```

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

```
int max, a, b, c;
scanf("%d", &a);
max = a;
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &c);
if (c > max) max = c;
```

8. 既然可以在需要的時候再讀進變數裡, 比對完裡面的資料就可以丟棄, 所以不需要那麼多變數了, 只需要 max 和 b, 繼續擴充下去時也只要複製最後兩列程式

```
int max, b;
scanf("%d", &max);
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &b);
if (b > max) max = b;
```

9. 聽到『複製與貼上』其實代表的一定是迴圈, 可以用迴圈的語法來處理輸入資料: 4 3 7 2 5

7. 問題還沒解決完, 繼續擴展這個程式時每多處理一筆資料就需要多一個變數, 比對到變數 **c** 時, 先前的變數 **a, b** 裡的資料好像就沒有再用到了, 目前每一個變數的功能在於讀到資料以後先紀錄下來, 以便逐一比對, 有必要這樣多一筆資料就多用一個變數嗎?? 有必要一次就把資料都讀到變數裡嗎?? 好像不需要!!

```
int max, a, b, c;
scanf("%d", &a);
max = a;
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &c);
if (c > max) max = c;
```

8. 既然可以在需要的時候再讀進變數裡, 比對完裡面的資料就可以丟棄, 所以不需要那麼多變數了, 只需要 max 和 b, 繼續擴充下去時也只要複製最後兩列程式

```
int max, b;
scanf("%d", &max);
scanf("%d", &b);
if (b > max) max = b;
scanf("%d", &b);
if (b > max) max = b;
```

9. 聽到『複製與貼上』其實代表的一定是迴圈, 可以用迴圈的語法來處理輸入資料: 4 3 7 2 5

```
int max, b, n, i;
scanf("%d%d", &n, &max);
for (i=1; i<n; i++) {
    scanf("%d", &b);
    if (b > max) max = b;
}
```

計算 n 個數字的總和

請撰寫一個程式

- 讀取下列的整數輸入 (n>0)

  n a_1 a_2 … a_n
- 計算並且印出 $a_1 + a_2 + \ldots + a_n$

計算 n 個數字的總和

請撰寫一個程式

- 讀取下列的整數輸入 (n>0)

  n a_1 a_2 … a_n
- 計算並且印出 $a_1 + a_2 + \ldots + a_n$

1. 理論上可以寫一個程式, 定義 n 個變數, 一次讀入所有資料, 再寫一個很長的加法敘述加總並且列印, 可是就是寫得很辛苦又很沒有彈性

計算 n 個數字的總和

請撰寫一個程式

- 讀取下列的整數輸入 (n>0)

  n a_1 a_2 … a_n
- 計算並且印出 $a_1 + a_2 + \ldots + a_n$

1. 理論上可以寫一個程式, 定義 n 個變數, 一次讀入所有資料, 再寫一個很長的加法敘述加總並且列印, 可是就是寫得很辛苦又很沒有彈性

```
int n, a, b, c, d, e, f;
scanf("%d%d%d%d%d%d%d", &n,&,a,&b,&c,&d,&e,&f);
printf("%d\n", a + b + c + d + e + f);
```

2. 實際上這個問題的簡化方法, 迴圈建構方法和前一個問題一模一樣, 雖然你也許覺得還蠻簡單的, 應該可以跳過去, 直接寫程式就好了, 但是因為這個解決問題的程序還蠻標準的, 你可能在很多問題裡都會遇見, 請快速回顧一下或是練習推演一次

2. 實際上這個問題的簡化方法, 迴圈建構方法和前一個問題一模一樣, 雖然你也許覺得還蠻簡單的, 應該可以跳過去, 直接寫程式就好了, 但是因為這個解決問題的程序還蠻標準的, 你可能在很多問題裡都會遇見, 請快速回顧一下或是練習推演一次

```
int sum, b, n, i;
scanf("%d%d", &n, &sum);
for (i=1; i<n; i++)
{
    scanf("%d", &b);
    sum += b;
}
printf("%d\n", sum);
```