第三週

王子銜, 陳毅軒, 吳尚龍

電機通訊程式設計

March 4, 2024

Outline

Judge 判斷方式

在沒有開嚴格比對的情況下，Judge 會從 Team 端讀一個字串、Judge 端 (答案) 讀一個字串，直到檔案尾端 (EOF)，全部都相等就 correct，舉例:

```
// Team:
1 2 3
// Judge:
1 2 3
```

Correct

```
// Team:
1
2

3
// Judge:
1 2 3
```

Correct

Judge 判斷方式

```
1 // Team:
2 1 23
3 // Judge:
4 1 2 3
```

Wrong Answer

提前終止輸入時機

當測資不只一筆時，注意不要在當次輸入結束前就 break 或 return，因爲當你在讀下一筆測資時會讀到不對的東西

```
// input
2
3 1
1 2 3
下一筆...
```

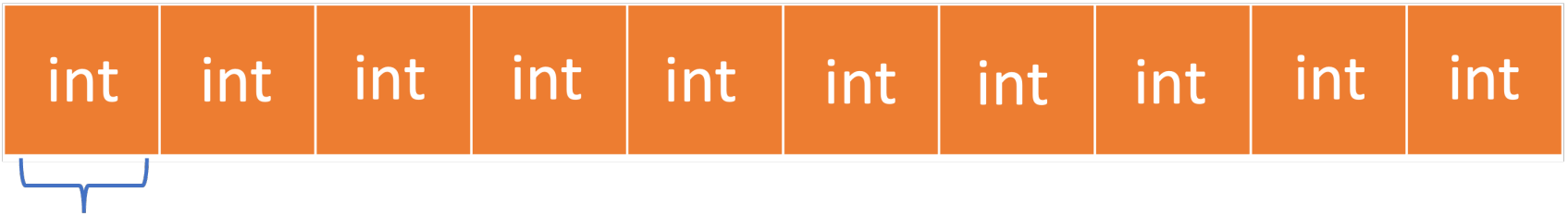
Wrong Answer

例如上周的芙莉蓮，不要看到 $k = 1$ 就直接很開心的輸出 0 然後 break，因爲下次會讀到 1 2 3 的 1，就錯了。

宣告陣列

宣告一個大小爲 10，名爲 array 的陣列，最前面跟變數一樣，必須宣告要儲存的 type

```
1  int array[10];
```




```
1  float array[10];
```

整個陣列裡的變數型態皆相同

陣列宣告

不同方式的宣告

```
1  int arr[10];
2  int arr[] = {1, 2, 3};  // 會根據資料數量決定大小
3  int arr[3] = {1, 2, 3};
4  int arr[10] = {1, 2, 3}; // 沒有指定到的會用 0 填滿
```

1. 建議區域陣列都還是要初始化比較好，至少全部初始化爲 0
2. 所以 int arr[10] = {1}，並不是所有值初始化爲 1，只有第一個數爲 1，其他都爲 0

陣列的 index

陣列的 index 從 0 開始!

陣列的 index 從 0 開始!

陣列的 index 從 0 開始!

0	1	2	3	4	5	6	7	8	9

範例

陣列名稱 [index] 可以取得儲存的值

```c
#include <stdio.h>

int main() {
  int arr[5] = {4, 8, 7, 6, 3};
  printf("%d ", arr[0]);
  printf("%d ", arr[2]);
  printf("%d ", arr[4]);
}
```

```
// output
4 7 3
```

切記 index 不要超過陣列的範圍，例如在這裡只能存取 0 到 4

全域陣列

```c
#include <stdio.h>
int arr[5];
int main(){
  for(int i = 0; i < 5; i++){
    printf("%d ", arr[i]);
  }
}
```

```
// output
0 0 0 0 0
```

開在全域的陣列，會自動初始化爲 0
區域的話資料會不可測，要小心

陣列 Maxsize

陣列大小取決於記憶體大小。

通常 Judge 的 int 陣列限制

- 全域最多開到 10^8
- 區域最多開到 10^6

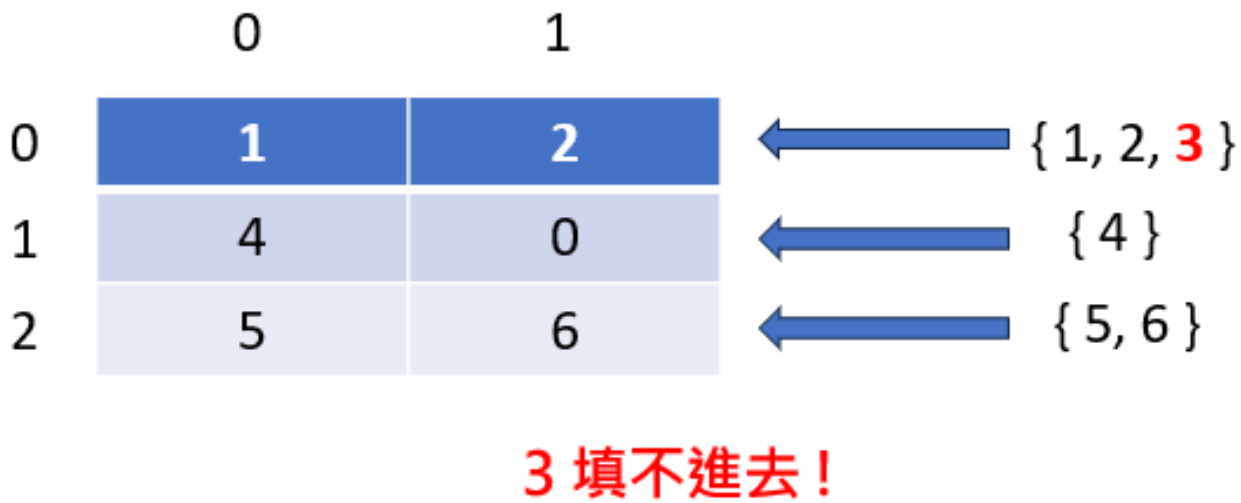
開再更大會導致 RE

二維陣列的初始化

不同方式的初始化

```
1  int arr[10][20];
2  int arr[2][2] = {{1, 2}, {3, 4}};
3  int arr[][2] = {{1, 2}, {3, 4}, {5, 6}};
4  // int arr[][2] = {{1, 2, 3}, {4}, {5, 6}};  // Compiler Error
```

1. 沒有指定到的部分也會預設爲 0
2. 也要注意後面一維參數的數量不要超過前面的大小
3. 只有最前面的中括號可以不放數字 (不管幾維陣列都是)



二維陣列輸出

使用雙層迴圈輸出，記得注意 i、j 的範圍，不要讓程式 RE

```
int arr[][2] = {{1,2}, {2,3}, {4,5}};
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 2; j++)
    printf("%d ", arr[i][j]);
  printf("\n");
}
```

```
// output
1 2
2 3
4 5
```

多維陣列

有二維當然就可以拓展到多維範例:

```
int arr[2][2][2] = {{{1, 2}, {3, 4}}, {{5, 6}, {7, 8}}};
int sum = 0;
for (int i = 0; i < 2; i++) {
  for (int j = 0; j < 2; j++) {
    for (int k = 0; k < 2; k++) {
      sum += arr[i][j][k];
    }
  }
}
printf("%d", sum);
```

```
// output
36
```

補充 - 其他初始化 - 一維轉二維 (或多維)

```
1  int arr[2][3] = {1, 2, 3, 4}
```

```
1  // 儲存方式
2  1 2 3
3  4 0 0
```

1. 會以低層優先填滿
2. 一樣沒有指定到的會預設爲 0
3. 基本上沒人會這樣用，知道不會報錯即可
4. 跟 C 陣列初始化的參數配對有關，有興趣可以玩玩看就會懂了

補充 - 早期 C 的陣列宣告

早期的 C 版本陣列宣告時，括號內不能填入變數，只能是常數
範例:

```c
int a = 1;
// int arr[a];  // compile error
const int b = 1;
int arr[b]; // 合法
```

本週加分

因爲這週的題目普遍很長，可能會比較躁億點。
爲了鼓勵大家要寫作業，我們與老師討論後決定，今天上課全部 AC 且 demo 完，總成績 $+1$ 分。
或是在 deadline 前全部 AC 總成績則 $+0.5$ 分。