slido:~$

# 陣列 & 記憶體簡介

# 巢狀迴圈 :~/ 程式碼翻譯 / 範例 $

以下的程式碼可以被翻譯成重複 n 次，並且每一次都印出 i 的值

```
for (int i=0 ; i<n ; i++){
    cout << "i: " << i << "\n"; // 印出i
}
```

# 巢狀迴圈 :~/ 程式碼翻譯 / 例題 /p1$

請試著用同樣概念翻譯以下程式碼（slido 回答）

```
for (int i=0 ; i<n ; i++){
    for (int j=0 ; j<m ; j++){
        cout << "i: " << i << "j: " << j << "\n"; // 印出i和j
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /p1$

請計算以下程式碼經過 //  code 的次數

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<7 ; j++){
        // code
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /a1$

第一層迴圈會重複執行 3 次，並進入第二層迴圈再重複執行 7 次
總共 21 次

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<7 ; j++){
        // code
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /p2$

請計算以下程式碼經過 //  code 的次數

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<2 ; j++){
        for (int k=0 ; k<5 ; k++){
            // code
        }
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /a2$

第一層迴圈會重複執行 3 次，並進入第二層迴圈再重複執行 2 次
最後進入第三層迴圈重複執行 5 次
總共 30 次

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<2 ; j++){
        for (int k=0 ; k<5 ; k++){
            // code
        }
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /p3$

請計算以下程式碼經過 //  code 的次數

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<2 ; j++){
        // nothing
    }
    for (int j=0 ; j<5 ; j++){
        // code
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /a3$

第一層迴圈會重複執行 3 次，並進入**第二個**第二層迴圈重複執行 5 次
總共 15 次（第一個第二層迴圈不予理會，因為不會影響答案）

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<2 ; j++){
        // nothing
    }
    for (int j=0 ; j<5 ; j++){
        // code
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /p4$

請計算以下程式碼經過 //  code 的次數

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<5 ; j++){
        if (j%2==0){ // 如果j 可以被2 整除就執行下面行為
            // code
        }
    }
}
```

# 巢狀迴圈 :~/ 重複次數計算 / 例題 /a4$

第一層迴圈會重複執行 3 次，並進入第二層迴圈
此時 j 為 {0, 1, 2, 3, 4}，但是只有 {0, 2, 4} 會繼續執行
因此總共 9 次

```
for (int i=0 ; i<3 ; i++){
    for (int j=0 ; j<5 ; j++){
        if (j%2==0){ // 如果j 可以被2 整除就執行下面行為
            // code
        }
    }
}
```

# 巢狀迴圈 :~/ 結論 $

以上的試題不是用來考數學　xd
而是用來讓大家熟悉巢狀迴圈的計數規則以及各種情況
以上的試題可以得到以下結論

1. 總次數為右界的積
2. 不同大括號內的程式不會互相影響
3. 巢狀迴圈仍可以搭配 `if-else` 使用

# 多維陣列 :~/ 宣告陣列 $

```
int arr[5][5];
```

```
int arr1[1000000];
int arr2[1000][1000];
int arr3[100][100];
int main(){
    // 對於太大的陣列，請放在main()外面
}
```

# 多維陣列 :~/ 取值 & 賦值 $

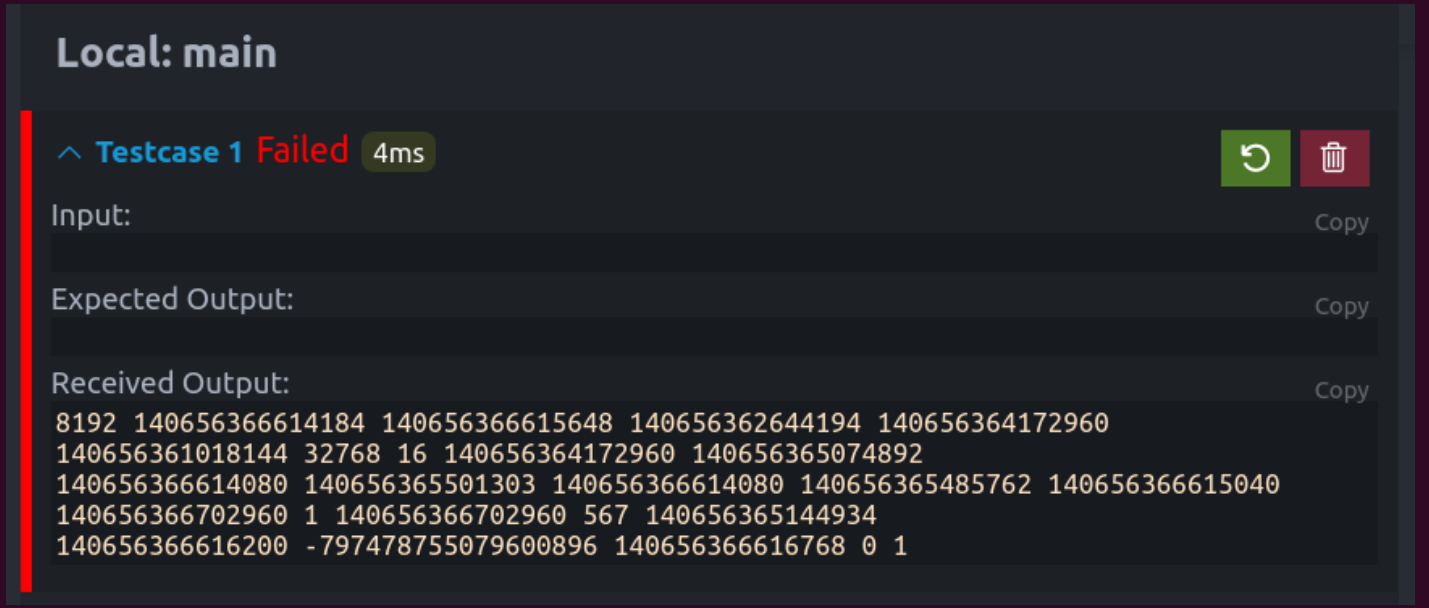
```cpp
int arr1[5][5];
int arr2[5][5][5];

arr1[3][4]=3;
arr2[2][3][4]=3;
cout << arr1[3][4] << "\n";
cout << arr2[2][3][4] << "\n";
```

# 多維陣列 :~/ 注意事項 $

**請記得初始化**，否則會像下圖一樣

```
int arr[5][5]={0};
```



# 多維陣列 :~/ 注意事項 $

多維陣列請務必熟練，屬於非常常見的語法

以 APCS 來說，二維陣列可以說是 pB 的必考題
通常配合複雜的 `if-else`、 `for`、 `while` 綜合使用

（如果需要更多題目練習可以找我拿）

# 多維陣列 :~/ 例題 $

根據題目，我們需要**重複 n 次，每次再重複 m 次**
這種架構可以用 for 迴圈處理

```cpp
int main(){
    // init
    int n, m;
    int arr[105][105];

    // input
    cin >> n >> m;
    for (int i=0 ; i<n ; i++){
        for (int j=0 ; j<m ; j++){
            cin >> arr[i][j];
        }
    }
    return 0;
}
```

# 多維陣列 :~/ 例題 $

接下來把每一個數字都乘上 k

```
// 對於陣列裡的每一個數字乘上k
    int k;
    cin >> k;
    for (int i=0 ; i<n ; i++){
        for (int j=0 ; j<m ; j++){
            arr[i][j]=arr[i][j]*k;
            // arr[i][j]*=k;
        }
    }
```

# 多維陣列 :~/ 例題 $

最後把陣列裡的所有數值輸出

```
// output
for (int i=0 ; i<n ; i++){
    for (int j=0 ; j<m ; j++){
        cout << arr[i][j] << " ";
    }
    // 別忘了每一行都要換行
    cout << "\n";
}
```

## 記憶體簡介 :~ 前言 $

以下是上上禮拜的內容，不過似乎少了一個東西
我們可以知道 arr[i] 的值，不過 arr 是什麼呢?

```
int arr[5]={4, 8, 7, 6, 3};
// arr[0] 為 4;
// arr[1] 為 8;
// arr[2] 為 7;
// arr[3] 為 6;
// arr[4] 為 3;
```

## 記憶體簡介 :~ 前言 $

以下是我輸出 arr 的結果，獲得了 `0x7ffe2c436a60` 的輸出
這就是一個**記憶體位置**，也就是 arr 是一段記憶體位置

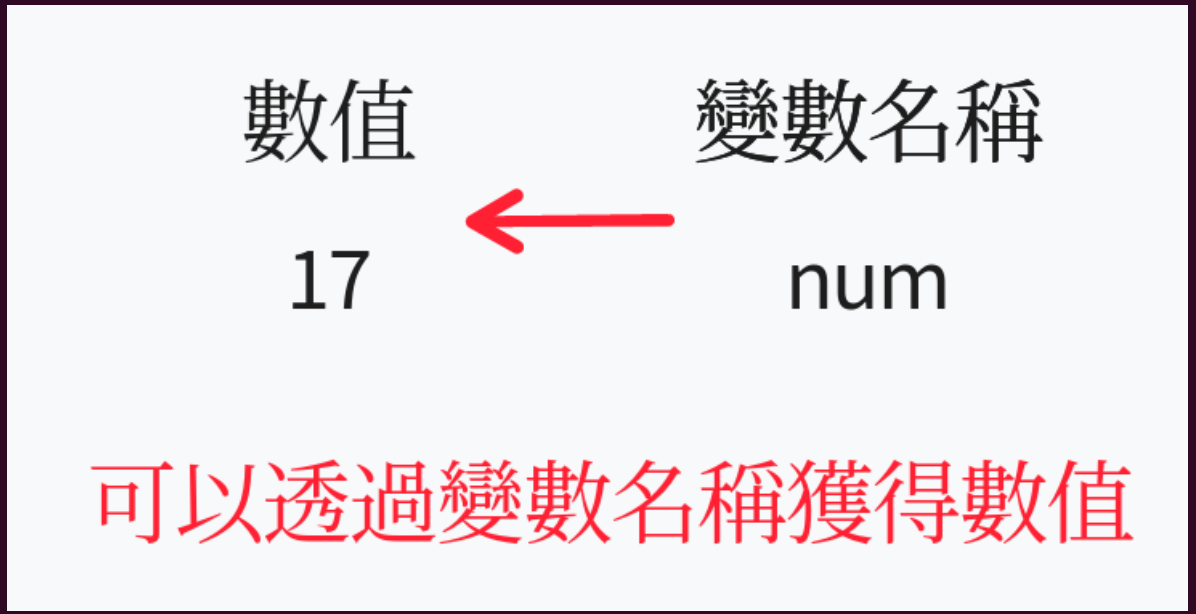
（更準確的來講，其實是 `arr[0]` 的記憶體位置）

```cpp
int arr[5]={4, 8, 7, 6, 3};
cout << arr << "\n";

// output: 0x7ffe2c436a60
// (每個人的輸出不會一樣)
```

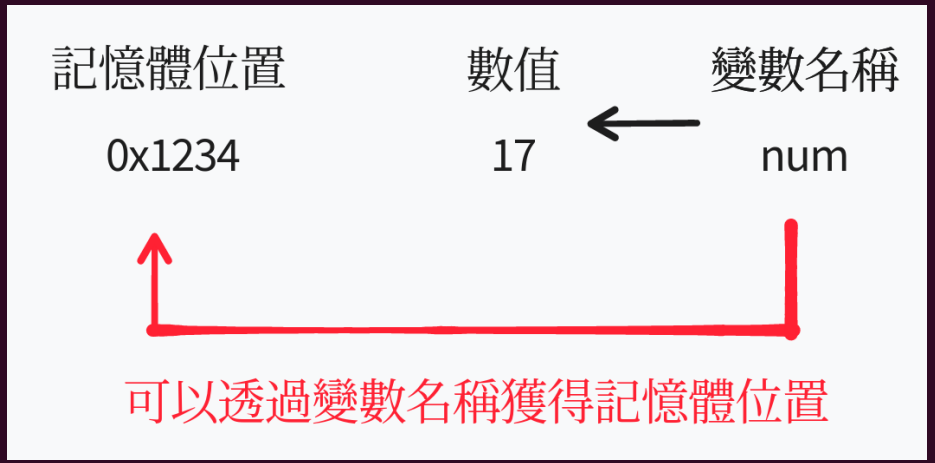
# 記憶體簡介 :~ 基礎重導 $

對於糟糕的字體窩很抱歉　QQ



# 記憶體簡介 :~ 基礎重導 / 取址符號 $

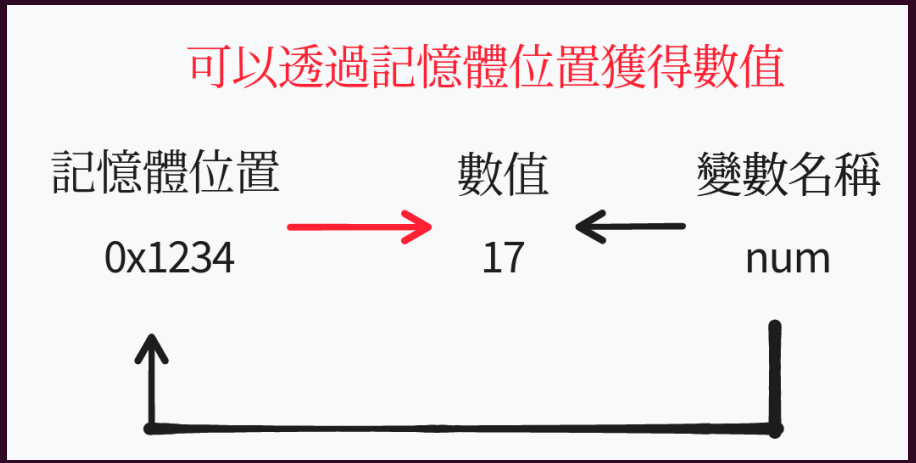
可以透過在變數名稱前加上 &，就可以獲得此變數的記憶體位置




```cpp
int num=17;
cout << &num << "\n";
```

# 記憶體簡介 :~ 基礎重導 / 間接符號 $

可以透過在記憶體位置前加上 * ， 就可以獲得此記憶體位置的值




```
int arr[5]={4, 8, 7, 6, 3};
cout << *arr << "\n";
```

## 記憶體簡介 :~ 自主練習 $

以下提供了 6 個任務，可以自己實做看看

1. 宣告一個變數（num），並且找到地址
2. 宣告一個陣列（arr），並且找到地址
3. 找到陣列裡的每個值（arr[i]）地址
4. 同 (3.)，但是把陣列得型別改成 bool, char
5. 將陣列（arr） +1、 +2… 並且找到地址
6. 將陣列（arr） +1、 +2… 並且找到值

# 記憶體簡介 :~ 自主練習 / 範例程式碼 $

```cpp
int num=17;
int arr[5]={4, 8, 7, 6, 3};

// p1
cout << &num << "\n";

// p2
cout << &arr << "\n";
```

```cpp
// p3, p4
for (int i=0 ; i<5 ;  i++){
    cout << &arr[i] << "\n";
}

// p5
for (int i=0 ; i<5 ;  i++){
    cout << arr+i << "\n";
}

// p6
for (int i=0 ; i<5 ; i++){
    cout << *(arr+i) << "\n";
}
```