C++程式設計：陣列

**目錄**

[C++程式設計：陣列 1](#_Toc204274919)

[1. 陣列的基本觀念 1](#_Toc204274920)

[1.1 初始化(initialization) 1](#_Toc204274921)

[1.2 範例：固定大小的一維陣列宣告 2](#_Toc204274922)

[2. 在迴圈中使用陣列 2](#_Toc204274923)

[2.1 範例：印出陣列中的值 2](#_Toc204274924)

[2.2 範例：陣列建立與輸出平均 3](#_Toc204274925)

[2.3 範例：陣列輸入與最小值搜尋 3](#_Toc204274926)

[2.4 範例：一維陣列中尋找特定值 4](#_Toc204274927)

[2.5 範例：質數判定 4](#_Toc204274928)

[3. 一維記憶體配置與位址 5](#_Toc204274929)

[3.1 範例：使用迴圈印出陣列地址 5](#_Toc204274930)

[3.2 範例：C 語言：查看 arr 的記憶體空間與長度 6](#_Toc204274931)

[3.3 範例：C 語言：查看 arr 的記憶體空間與長度 6](#_Toc204274932)

[3.4 範例：超出邊界 7](#_Toc204274933)

[4. 函數庫assert 7](#_Toc204274934)

[4.1 範例：只能輸入比 5 小的數，否則報錯 7](#_Toc204274935)

[5. 泡沫排序法 8](#_Toc204274936)

[5.1 範例：實作泡沫排序法 8](#_Toc204274937)

[6. 二維陣列的記憶體配置 9](#_Toc204274938)

[6.1 範例：二維陣列的記憶體配置 9](#_Toc204274939)

# 陣列的基本觀念

陣列的基本核心是占有一塊連續的記憶體空間。在需要同時儲存多個同型態變數的情境下，可以使用陣列一次宣告多個變數，減少重複撰寫。

## 初始化(initialization)

// 宣告空間

int score[5];

// 在宣告空間的時候，同時需要宣告值

int arr[3] = {5,7,10}; // array of int

float arr[3] = {5.1,7.1,10.1}; // array of float

char arr[3] = {'a', 'b', 'c'}; // array of charts

int arr[5] = {0}; //相當於{0,0,0,0,0}

int arr[5] = {1,2}; //相當於{1,2,0,0,0}

## 範例：固定大小的一維陣列宣告

#define ARRAYSIZE 10 // macro

int a[ARRAYSIZE];

# 在迴圈中使用陣列

## 範例：印出陣列中的值

// 寫法01

#include <iostream>

using namespace std;

#define ARRAYSIZE 10

main(){

    // int a[ARRAYSIZE];

    // for (int i = 0; i < ARRAYSIZE; i++)

    //     cin >> a[i];

    int a[ARRAYSIZE] = {3, 6, 1, 8, 4, 9, 10, 4, 7, 6};

    int sum {0};

    for (int i = 0; i < ARRAYSIZE; i++)

        sum += a[i];

    cout << sum << endl;

}

// 寫法02

#include <iostream>

using namespace std;

#define ARRAYSIZE 10

main(){

    // int a[ARRAYSIZE];

    // for (int i = 0; i < ARRAYSIZE; i++)

    //     cin >> a[i];

    int a[ARRAYSIZE] = {3, 6, 1, 8, 4, 9, 10, 4, 7, 6};

    int sum = 0;

    for (int i: a)

        sum += i;

    cout << sum << endl;

}

## 範例：陣列建立與輸出平均

#include <iostream>

#include <iomanip>  // for std::setprecision

using namespace std;

int main() {

    int score[5] = {87, 66, 90, 65, 70};  // 成績陣列

    float total = 0;                      // 加總用變數

    // 輸出每位同學的分數

    for (int i = 0; i < 5; i++) {

        cout << "The score of student " << (i + 1) << " is: " << score[i] << endl;

        total += score[i];  // 累加

    }

    // 輸出總分與平均分，保留1位小數

    cout << fixed << setprecision(1);

    cout << "Total: " << total << "   Average: " << total / 5 << endl;

    return 0;

}

## 範例：陣列輸入與最小值搜尋

#include <iostream>

using namespace std;

int main() {

    int grades[5] = {135, 4, 7, 8, 3};  // 可改為使用者輸入

    int low\_grade = grades[0];  // 先假設第一個為最小值

    for (int i = 0; i < 5; i++) {

        if (grades[i] < low\_grade) {

            low\_grade = grades[i];  // 更新最小值

        }

    }

    cout << "Lowest grade: " << low\_grade << endl; // Lowest grade: 3

    return 0;

}

## 範例：一維陣列中尋找特定值

#include <iostream>

using namespace std;

#define ARRAYSIZE 6

main(){

    int keys[] = {3, 4, 6, 2, 8, 10};

    int key; // 要找的值

    while (cin >> key) {

            bool found = false;

            for (int i = 0; i < ARRAYSIZE && !found; i++)

                if (key == keys[i]) {

                    cout << i << endl;

                    found = true;

                    }

            if (!found)

            cout << "-1" << endl;

    }

}

## 範例：質數判定

#include <iostream>

#include <cassert>

using namespace std;

int main() {

    int n;

    cin >> n;  // 目標：列出前 n 個質數，例如 20

    int count = 0;

    int number = 2;

    while (count < n) {

        bool isPrime = true;

        for (int i = 2; i \* i <= number; ++i) {

            if (number % i == 0) {

                isPrime = false;

                break;

            }

        }

        if (isPrime) {

            cout << number << " ";

            count++;

        }

        number++;

    }

    cout << endl;

    return 0;

}

# 一維記憶體配置與位址

使用 &變數可以取得該變數在記憶體中的位址，C++ 預設就是印出十六進位。

cout << &a[i] << endl; // 印出地址

## 範例：使用迴圈印出陣列地址

#include <iostream>

using namespace std;

int main() {

    int arr[] = {5, 4, 3, 2, 1};  // 初始陣列

    for(int i=0;i<5;i++){

        cout << &(arr[i]) << endl;

    }

    return 0;

}

// 0x7ffffcc00

// 0x7ffffcc04

// 0x7ffffcc08

// 0x7ffffcc0c

// 0x7ffffcc10

## 範例：C 語言：查看 arr 的記憶體空間與長度

#include <stdio.h>

#include <stdlib.h>

int main(){

    int arr1[1];

    float arr2[1];

    char arr3[1];

    printf("%d\n", sizeof(arr1));

    printf("%d\n", sizeof(arr2));

    printf("%d\n", sizeof(arr3));

    return 0;

}

// 4

// 4

// 1

## 範例：C 語言：查看 arr 的記憶體空間與長度

// TODO: 印出記憶體位置：要使用%p

#include <stdio.h>

#include <stdlib.h>

#define ARRAYSIZE 5

int main(){

    int a[ARRAYSIZE] ={1,2,3,4,5};

    printf("%ld\n", a[0]); // 輸出第0個值 => 1

    printf("%ld\n", sizeof(a[0])); // array 第0個位置的長度 => 4 byte

    printf("%ld\n", sizeof(a)); // array 總byte數 => 4 byte \*5 = 20bytes

    // 需要知道array長度 => sizeof(a)/sizeof(a[0])

    for (int i = 0; i < ARRAYSIZE; i++){

        printf("%p\n", &(a[i]));

    }

    return 0;

}

// 4

// 20

// 0061FF08

// 0061FF0C

// 0061FF10

// 0061FF14

// 0061FF18

## 範例：超出邊界

#include <iostream>

using namespace std;

int main() {

    int arr[3] = {1, 2, 3};  // 宣告大小為 3 的整數陣列

    for (int i = 0; i < 5; i++) {  // ⚠️ 超出範圍訪問 arr[3] 和 arr[4]

        cout << "arr num: " << arr[i] << endl;

    }

    return 0;

}

// arr num: 1

// arr num: 2

// arr num: 3

// arr num: 3 (無法預測)

// arr num: -13008 (無法預測)

# 函數庫assert

是用來「偵錯用的斷言檢查工具」，可幫助程式確認某條件在執行時是否成立。

// 語法格式

#include <cassert> // 必須引入 <cassert> 標頭檔。

assert(condition); // 當 condition為 false，會顯示錯誤訊息並終止程式

## 範例：只能輸入比 5 小的數，否則報錯

#include <iostream>

#include <cassert>

using namespace std;

int main() {

    int j = 0;

    while (cin >> j) {

        assert(j >= 0 && j < 5);  // 檢查 j 合法範圍

        cout << "OK" << endl;

    }

}

# 泡沫排序法

反覆比較相鄰的元素，並將較大的元素逐步「冒泡」到數列的末尾。會有雙重迴圈：

* 外部：回合數
* 內部：負責兩兩交換

未排序

64, 34, 25, 12, 22, 11, 90

第一輪： 最大值 90 冒泡到最後一位

64, 34, 25, 12, 22, 11, 90

第二輪： 次大值 64 冒泡到倒數第二位

34, 25, 12, 22, 11, 64, 90

第三輪： 次次大值 34 冒泡到倒數第三位

25, 12, 22, 11, 34, 64, 90

## 範例：實作泡沫排序法

// C++ 實作泡沫排序法

#include <iostream>

#include <cassert>

using namespace std;

int main() {

    int n[] = {5, 4, 3, 2, 1};  // 初始陣列

    int m = sizeof(n) / sizeof(n[0]);  // 陣列長度

    // Bubble Sort 排序

    for (int i = 0; i < m - 1; i++) {

        for (int j = 0; j < m - 1 - i; j++) {

            if (n[j] > n[j + 1]) {

                swap(n[j], n[j + 1]);

            }

        }

    }

    // 輸出排序後的結果

    for (int i = 0; i < m; i++)

        cout << n[i] << ' ';

    cout << endl;

    return 0;

}

# 二維陣列的記憶體配置

## 範例：二維陣列的記憶體配置

#include <iostream>

using namespace std;

int main() {

    int a[3][4] = {

        {1,  2,  3,  4},

        {5,  6,  7,  8},

        {9, 10, 11, 12}

    };

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 4; j++) {

            cout << &a[i][j] << "\t";

        }

        cout << endl;

    }

    return 0;

}

// 0x7ffffcbc0     0x7ffffcbc4     0x7ffffcbc8     0x7ffffcbcc

// 0x7ffffcbd0     0x7ffffcbd4     0x7ffffcbd8     0x7ffffcbdc

// 0x7ffffcbe0     0x7ffffcbe4     0x7ffffcbe8     0x7ffffcbec