C++程式設計：常數

**目錄**

[C++程式設計：常數 1](#_Toc205474581)

[1. 基本概念 1](#_Toc205474582)

[1.1 範例：const 參數防止修改 1](#_Toc205474583)

[2. 常數陣列(const array) 2](#_Toc205474584)

[2.1 範例：const 陣列參數防止內容被更動 2](#_Toc205474585)

[3. 常數指標(const pointer) 3](#_Toc205474586)

[3.1 種類 01：普通指標：int\* ptrToInt 3](#_Toc205474587)

[3.2 種類 02：指向常數的指標：const int\* ptrToConstInt 3](#_Toc205474588)

[3.3 種類 03：常數指標：int\* const constPtrToInt 4](#_Toc205474589)

[3.4 種類 04：常數指標 + 指向常數：const int\* const constPtrToConstInt 4](#_Toc205474590)

[3.5 範例：實作練習範例 5](#_Toc205474591)

# 基本概念

常數(const) 就是要讓我們的參數不要被更動到。

## 範例：const 參數防止修改

#include <iostream>

using namespace std;

// 接收常數參數，不能被修改

int square(const int i) {

    i++; // (要這注解)❌ 編譯錯誤：const 參數不可修改

    return i \* i;

}

int main() {

    int x;

    cout << "Enter a number: ";

    cin >> x;

    cout << "Square of " << x << " is " << square(x) << endl;

    return 0;

}

// .\test.cpp: 在函式 「int square(int)」 中:

// .\test.cpp:6:5: 錯誤：increment of read-only parameter 「i」

//     6 |     i++; // ❌ 編譯錯誤：const 參數不可修改

//       |     ^

# 常數陣列(const array)

## 範例：const 陣列參數防止內容被更動

#include <iostream>

using namespace std;

// 接收 const int 陣列與大小 n，保證不修改陣列內容

int sum(const int a[], const int n) {

    int total = 0;

    for (int i = 0; i < n; ++i) {

        a[i]++; // (要註解)❌ 錯誤：不能修改 const 陣列內容

        total += a[i];

    }

    return total;

}

const int ARRAYSIZE = 5;

int main() {

    int array[ARRAYSIZE];

    cout << "Enter " << ARRAYSIZE << " integers: ";

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

        cin >> array[i];

    }

    cout << "Sum = " << sum(array, ARRAYSIZE) << endl;

    return 0;

}

// .\test.cpp: 在函式 「int sum(const int\*, int)」 中:

// .\test.cpp:8:12: 錯誤：increment of read-only location 「\*(a + ((sizetype)(((long unsigned int)i) \* 4)))」

//     8 |         a[i]++; // (要註解)❌ 錯誤：不能修改 const 陣列內容

//       |

# 常數指標(const pointer)

我們常會遇到兩種需求：

* 保護內容：指標指向的值不能改。
* 保護指標：指標本身不能換位置（不能指向別人）。

// 總攬

int **\***ptrToInt

const int **\***ptrToConstInt

int **\***const constPtrToInt

const int **\***const constPtrToConstInt

接下來，我們這樣可以分為下列子種類：

## 種類 01：普通指標：int\* ptrToInt

什麼都能改，就是一般指標。

#include <iostream>

using namespace std;

int main() {

    int a = 10, b = 20;

    int\* ptrToInt = &a;

    \*ptrToInt = 100;   // ✅ 可以改值

    cout << \*ptrToInt << endl; // 100

    ptrToInt = &b;     // ✅ 可以改指向

    cout << \*ptrToInt << endl; // 20

    return 0;

}

## 種類 02：指向常數的指標：const int\* ptrToConstInt

不能改內容，能改指向。

#include <iostream>

using namespace std;

int main() {

    int a = 10, b = 20;

    const int\* ptrToInt = &a;

    // \*ptrToInt = 100;   // ❌ 錯誤：不能改值

    cout << \*ptrToInt << endl; // 100

    ptrToInt = &b;     // ✅ 可以改指向

    cout << \*ptrToInt << endl; // 20

    return 0;

}

## 種類 03：常數指標：int\* const constPtrToInt

#include <iostream>

using namespace std;

int main() {

    int a = 10, b = 20;

    int\* const ptrToInt = &a;

    \*ptrToInt = 100;   // ✅ 可以改值

    cout << \*ptrToInt << endl; // 100

    // ptrToInt = &b;     // ❌ 錯誤：不能改指向

    cout << \*ptrToInt << endl; // 100

    return 0;

}

## 種類 04：常數指標 + 指向常數：const int\* const constPtrToConstInt

#include <iostream>

using namespace std;

int main() {

    int a = 10, b = 20;

    const int\* const ptrToInt = &a;

    // \*ptrToInt = 100;   // ❌ 錯誤：不能改值

    cout << \*ptrToInt << endl; // 10

    // ptrToInt = &b;     // ❌ 錯誤：不能改指向

    cout << \*ptrToInt << endl; // 10

    return 0;

}

## 範例：實作練習範例

#include <iostream>

using namespace std;

int main() {

    int a = 10, b = 20;

    // 普通指標

    int\* p1 = &a;

    \*p1 = 100;   // ✅ ok

    p1 = &b;     // ✅ ok

    // 指向 const int 的指標

    const int\* p2 = &a;

    // \*p2 = 100; // ❌ error: 不能改值

    p2 = &b;     // ✅ ok

    // const 指標（不能改指向）

    int\* const p3 = &a;

    \*p3 = 100;   // ✅ ok

    // p3 = &b;   // ❌ error: 不能改指向

    // const 指標 + 指向 const

    const int\* const p4 = &a;

    // \*p4 = 100; // ❌ error

    // p4 = &b;   // ❌ error

    return 0;

}

# GPT給的常數練習題目

## Read-Only Squaring

|  |  |
| --- | --- |
| 題目描述  請寫一個函式 int square(const int x)，回傳 x 的平方。注意：不能修改 x 的值，若嘗試修改將導致編譯錯誤。 | |
| 輸入格式  一個整數 x (1 ≤ x ≤ 1000) | 輸出格式  x 的平方 |
| 範例輸入  5 | 範例輸出  25 |
| 程式碼  #include <iostream>  using namespace std;  int square(const int x) {      return x \* x;  }  int main() {      int x;      cin >> x;      cout << square(x) << endl;      return 0;  } | |

## Sum of Constant Array

|  |  |
| --- | --- |
| 題目描述  給你一個整數陣列，請寫一個 int sum(const int a[], const int n) 函式回傳總和，且不能修改 a 中的任何值。 | |
| 輸入格式  第一行為一個整數 n (1 ≤ n ≤ 100) 第二行為 n 個整數 | 輸出格式  這些整數的總和 |
| 範例輸入  5 1 2 3 4 5 | 範例輸出  15 |
| 程式碼  #include <iostream>  using namespace std;  int sum(const int a[], const int n) {      int total = 0;      for (int i = 0; i < n; ++i)          total += a[i];      return total;  }  int main() {      int n;      cin >> n;      int a[n];      for (int i = 0; i < n; ++i)          cin >> a[i];      cout << sum(a, n) << endl;      return 0;  } | |