C++程式設計：先備知識&資料型態

**目錄**

[C++程式設計：先備知識&資料型態 1](#_Toc204174454)

[1. 程式(Program) 2](#_Toc204174455)

[2. 程式語言基本區分 2](#_Toc204174456)

[2.1 C 語言從設計、到最後執行會依據下列步驟進行： 2](#_Toc204174457)

[2.2 機器語言(Machine Language)： 2](#_Toc204174458)

[3. 編譯器(Compiler) 3](#_Toc204174459)

[4. 第一支 C 程式 3](#_Toc204174460)

[5. 程式基本架構 3](#_Toc204174461)

[5.1 名稱空間(Namespace) 4](#_Toc204174462)

[6. 變數 4](#_Toc204174463)

[7. 指定運算子(就是=) 4](#_Toc204174464)

[7.1 範例：交換變數 4](#_Toc204174465)

[7.2 範例：計算長方形周長與面積 5](#_Toc204174466)

[7.3 範例：換算零錢 5](#_Toc204174467)

[8. 邏輯判斷 6](#_Toc204174468)

[8.1 範例：閏年判斷 6](#_Toc204174469)

[8.2 範例：判斷三角形邊嘗是否合理 6](#_Toc204174470)

[9. 整數int 6](#_Toc204174471)

[10. sizeof() 函數：取得變數佔用記憶體大小 7](#_Toc204174472)

[11. 溢位(overflow) 7](#_Toc204174473)

[12. 浮點數(Floating Point) 7](#_Toc204174474)

[12.1 C語言：範例：浮點數資料宣告 8](#_Toc204174475)

[12.2 C語言：範例：浮點數資料宣告 8](#_Toc204174476)

[13. 字元(char) 9](#_Toc204174477)

[13.1 跳脫字符 9](#_Toc204174478)

[13.2 操控子(Manipulator) 9](#_Toc204174479)

[13.3 C語言：範例：字元計算大小 9](#_Toc204174480)

[14. ASCII 編碼 10](#_Toc204174481)

[14.1 範例：字元轉換為 ASCII 整數 10](#_Toc204174482)

[15. 字串(string) 10](#_Toc204174483)

[16. 宣告修飾 11](#_Toc204174484)

[16.1 範例：輸出 short int 至 long int 11](#_Toc204174485)

[16.2 範例：for 迴圈測試 signed/unsigned 行為 11](#_Toc204174486)

[17. 強型別&弱型別 12](#_Toc204174487)

[17.1 範例：隱式型別轉換（int → double） 13](#_Toc204174488)

[17.2 範例：顯式型別轉換（double → int） 13](#_Toc204174489)

[18. auto 關鍵字 14](#_Toc204174490)

[18.1 範例：搭配容器與迭代器 14](#_Toc204174491)

[19. C 語言：C99 14](#_Toc204174492)

[19.1 範例：C99 有號整數 15](#_Toc204174493)

[19.2 範例：C99 無號整數 15](#_Toc204174494)

# 程式(Program)

簡單來說就是「劇本」，只是這個劇本是寫給電腦(computer)看的。由一條條指令(command)所組成，每一條指令是讓電腦執行特定動作的命令。

# 程式語言基本區分

程式語言基本可以分為高階語言(high-level languages)與低階語言(low-level languages)

|  |  |
| --- | --- |
| 高階語言 | * 貼近人類的表達方式，易學易懂 * 缺點是無法有效控制低階的硬體，執行效率也相對較差。 |
| 低階語言 | * 擁有良好的執行效率，並提供強大的硬體控制 * 很難學習和理解。編寫、讀取、除錯和維護相對較具挑戰性。 |

## C 語言從設計、到最後執行會依據下列步驟進行：

1. Create a source file.
2. Use an editor to write the original source code.
3. Compile and link the program. At this stage, the system will generate an executable module.
4. Execute the program.

## 機器語言(Machine Language)：

原始的電腦語言：由二進位指令組成，難以學習、不易理解。

假設要計算三個數字的總和，使用機器語言可能需要以下步驟：

* 將一個數字從記憶體載入暫存器 A
* 將另一個數字載入暫存器 B
* 將 A、B 相加，結果放入暫存器 C
* 將 C 儲存到記憶體

而高階語言只需要i = j + k;，大幅簡化了撰寫難度與步驟

# 編譯器(Compiler)

電腦必須先將高階語言轉換成機器語言(machine code)才能被電腦理解與執行。編譯器(Compiler)負責將整個高階語言程式轉換為等效的機器語言程式。

這邊使用的編譯器為g++(GNU C++ Compiler)

電腦單位

|  |  |
| --- | --- |
| 位元 (bit) | 由1與0表示，是電腦中最小的數據單位。 |
| 位元組 (byte) | * 是由 8 個位元 (bits) 組成的數據單位 * 一個 Byte 通常代表一個字符(如 ASCII 編碼中的英文字母 A 或 B)。 * 每個位元都有兩種可能性(0 或 1)，所以 8 個位元可以有 256 種組合。 |
| 字(Word) | * 一個 Word 是計算機 CPU 處理數據的基本單位，長度依 CPU 架構而定。 * 根據 CPU 不同 * 16-bit CPU 👉 1 Word = 16-bit = 2 Byte * 32-bit CPU 👉 1 Word = 32-bit = 4 Byte * 64-bit CPU 👉 1 Word = 64-bit = 8 Byte |

# 第一支 C 程式

#include <iostream>

using namespace std;

int main() {

int no = 2;

cout << "Hello World!!" << endl;

cout << "There are " << no << " pandas in Taipei" << endl;

return 0;

}

# 編譯&執行方式

gcc test.c -o test *// 將.C檔編譯成執行檔*

.\test *// 執行*

# 程式基本架構

#include <iostream>

using namespace std;

main(){

int i;

cin >> i;

cout << i << endl;

}

## 名稱空間(Namespace)

* 名稱空間(Namespace) 是 C++ 中用來組織程式碼的一種機制，目的是避免名稱衝突
* 標準庫的所有內容(如 cout、vector、string 等)都定義在 std 名稱空間中
* 如果沒有使用 using namespace std;，每次使用標準庫時都需要寫 std::

*// 沒使用using namespace std*

main(){

std::cout << "Hello, World!" << std::endl;

}

*// 有使用using namespace std*

using namespace std;

main(){

cout << "Hello, World!" << endl;

}

# 變數

宣告變數等同於告訴電腦要為變數分配多少記憶體空間。也因為變數是強型別(strongly typed)語言的一部分，這意味著在宣告變數時，必須明確指定其資料型態。

變數命名規則：<https://reurl.cc/A2zqNQ>

# 指定運算子(就是=)

## 範例：交換變數

#include <iostream>

using namespace std;

main(){

int i, j;

cin >> i >> j;

int temp = i;

i = j;

j = temp;

cout << i << endl;

cout << j << endl;

}

## 範例：計算長方形周長與面積

#include <iostream>

using namespace std;

main(){

int a, b, c, d;

cin >> a >> b >> c >> d;

int area {(c - a) \* (d - b)};

int perriphral {2 \* ((c - a) + (d - b))};

cout << area << endl;

cout << perriphral << endl;

}

## 範例：換算零錢

#include <iostream>

using namespace std;

main(){

int change;

cin >> change;

int fifty {change / 50};

change = change % 50;

int ten {change / 10};

change = change % 10;

int five {change / 5};

int one {change % 5};

cout << fifty << endl;

cout << ten << endl;

cout << five << endl;

cout << one << endl;

}

# 邏輯判斷

## 範例：閏年判斷

#include <iostream>

using namespace std;

main(){

if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))

*// 是閏年*

else

*// 不是閏年*

}

## 範例：判斷三角形邊嘗是否合理

檢查：

* 三邊皆為正數
* 兩邊和要大於第三邊

#include <iostream>

using namespace std;

main(){

int a = 8, b = 4, c = 3;

bool valid = {a > 0 && b > 0 && c > 0 && a + b > c && b + c > a && a + c > b};

cout << valid << endl;

}

# 整數int

在 C++ 中，int 是最常用的整數資料型別。在多數作業系統與編譯器設定下，int 佔用 4 bytes(位元組)，即 32 bits(位元)。而最高位元是符號位(sign bit)用來區分正負號。

每個 bit 可儲存 0 或 1，32 bit 可表示的整數範圍為：

* 最小值：-2³¹ = -2,147,483,648
* 最大值：2³¹ - 1 = 2,147,483,647

# sizeof() 函數：取得變數佔用記憶體大小

sizeof(變數名稱) *//回傳該變數佔用的位元組(byte)數*

# 溢位(overflow)

2147483647是int的最大值，加一後應為 2147483648。但因為超出範圍，電腦記憶體只有 32 bit，導致數值循環回最小值 -2147483648。

#include <iostream>

using namespace std;

main(){

int i;

cout << sizeof(i) << endl; *// 顯示 int 佔用幾個位元組*

i = 2147483647;

cout << i << endl; *// 顯示原本的數值*

i++; *// 進行加一運算*

cout << i << endl; *// 顯示加一後的結果*

}

*// 輸出*

*// 4*

*// 2147483647*

*// -2147483648*

避免 overflow，使用更大的資料型別，像是：

* long long(64 位元，可儲存更大範圍的整數)
* unsigned int(不儲存負數，正整數範圍更大)

# 浮點數(Floating Point)

float f; *// 單精度浮點數 佔4byte*

double df; *// 雙精度浮點數 佔8byte*

也可以使用 sizeof 得到 float 與 double 占用的記憶體空間。

#include <iostream>

using namespace std;

int main() {

float f;

double df;

cout << sizeof(f) << endl; *// 4*

cout << sizeof(df) << endl; *// 8*

return 0;

}

## C語言：範例：浮點數資料宣告

#include <stdio.h>

#include <stdlib.h>

int main(){

float f1 = 123.45f; *// 宣告單精度浮點數*

double d1 = 123456789.123456789123; *// 宣告雙精度浮點數*

printf("f1=%f\n", f1); *// f1=123.456337*

printf("d1=%f\n", d1); *// d1=123456789.123457*

return 0;

}

## C語言：範例：浮點數資料宣告

#include <stdio.h>

#include <stdlib.h>

#define ARRAYSIZE 5

int main() {

float f; double df;

scanf("%f", &f);

scanf("%lf", &df);

printf("%f\n", f);

printf("%f\n", df);

return 0;

}

*// 輸入*

*// 3.14 3.15*

*// 輸出*

*// 3.140000*

*// 3.150000*

# 字元(char)

佔用 1 個 byte。範圍為-128到127或0到255(取決於是否使用 signed 或 unsigned)。使用 ASCII 編碼來表示字符的數值。

## 跳脫字符

在C語言中，字串其實是一個字符陣列，並以空字元'\0' 作為結束標誌，這叫做跳脫字符。

## 操控子(Manipulator)

操控子是 C++ 中用來控制輸入輸出格式的工具。操控子可以設置輸出的格式，例如數字進制、欄寬、對齊方式等。

需要引入#include <iomanip>

|  |  |
| --- | --- |
| **操控子** | **功能描述** |
| dec | 將數字以十進位格式輸出。 |
| oct | 將數字以八進位格式輸出。 |
| hex | 將數字以十六進位格式輸出。 |
| setw(n) | 設置下一個輸出欄位的寬度為 n 個字符。 |
| setprecision(n) | 設置浮點數的有效位數為 n。 |
| setfill(c) | 設置填充字符為 c，通常與 setw 一起使用。 |
| left | 將輸出內容靠左對齊。 |
| right | 將輸出內容靠右對齊(預設行為)。 |
| endl | 插入換行符並刷新輸出緩衝區。 |

## C語言：範例：字元計算大小

#include <stdio.h>

#include <stdlib.h>

int main() {

char c = 'r';

char str[] = "Hello, World!"; *// \0 結尾*

printf("%c\n", c);

printf("%s\n", str);

*// 印出每個型別的實際大小*

printf("sizeof： %ld bytes\n", sizeof(c)); *// 1byte*

printf("sizeof： %ld bytes\n", sizeof(str)); *// 14byte*

return 0;

}

# ASCII 編碼

ASCII(American Standard Code for Information Interchange，美國資訊交換標準碼)將英文大小寫字母、數字、標點符號等對應到 0~127 的數值。每個字元在電腦中實際上以 整數編碼儲存

## 範例：字元轉換為 ASCII 整數

#include <iostream>

#include <iomanip> *// 為了使用 hex 輸出*

using namespace std;

int main() {

char c = 'a';

cout << c << endl; *// a*

cout << static\_cast<int>(c) << endl; *// 97*

cout << hex << static\_cast<int>(c) << endl; *// 0x61*

return 0;

}

# 字串(string)

雖然 C 字串使用 char[]，但 C++ 建議使用 std::string，更安全也更方便！

#include <iostream>

#include <string> *// 導入 string 類別*

using namespace std;

int main() {

char c = 'r';

string str = "Hello, World!"; *// 使用 string 而非 char[]*

cout << c << endl;

cout << str << endl;

cout << "sizeof(char): " << sizeof(c) << " bytes" << endl;

cout << "str.length(): " << str.length() << " characters" << endl;

return 0;

}

# 宣告修飾

|  |  |  |
| --- | --- | --- |
| **型別** | **一般大小** | **範圍(大約)** |
| short int | 2 bytes | -32,768 ~ 32,767 |
| int | 4 bytes | -2,147,483,648 ~ 2,147,483,647 |
| long int | 8 bytes | 約 ±9 × 10¹⁸ |
| long long int | 8 bytes | 約 ±9 × 10¹⁸ |

## 範例：輸出 short int 至 long int

#include <iostream>

using namespace std;

int main() {

cout << "sizeof(short int) = " << sizeof(short int) << " bytes" << endl; *// 2*

cout << "sizeof(int) = " << sizeof(int) << " bytes" << endl; *// 4*

cout << "sizeof(long) = " << sizeof(long) << " bytes" << endl; *// 8*

cout << "sizeof(long int) = " << sizeof(long int) << " bytes" << endl; *// 8*

cout << "sizeof(long long int) = " << sizeof(long long int) << " bytes" << endl; *// 8*

return 0;

}

## 範例：for 迴圈測試 signed/unsigned 行為

#include <iostream>

using namespace std;

int main() {

*// 從0跑到200，就會結束*

cout << "--- unsigned char 測試 ---" << endl;

for (unsigned char uc = 0; uc < 200; uc++) {

cout << "uc = " << static\_cast<int>(uc) << endl; *// 需轉型以顯示數字*

}

*// 從0跑到127，再跳到-128，然後一直重複 => 無限輪迴*

cout << "--- signed char 測試 ---" << endl;

for (signed char sc = 0; sc < 200; sc++) {

cout << "sc = " << static\_cast<int>(sc) << endl; *// 無限迴圈！因為會跳成負數*

}

return 0;

}

# 強型別&弱型別

強型別 (Strongly Typed)類型檢查嚴格，不允許變數做隱式類型轉換。E.g., C、C++、pythob、java。

# 範例：以 python 作為舉例

a = "10"

b = 5

print(a + b) # ❌ TypeError: can only concatenate str (not "int") to str

# 需要顯式轉換：

print(int(a) + b) # ✅ 正確輸出 15

而弱型別(Weakly Typed)可以根據上下文自動轉換（自動類型提升）。允許隱式轉換（例如：字串變數可以與數字變數相加）比較靈活，但也容易產生隱式錯誤。

let a = "10";

let b = 5;

console.log(a + b); *// ✅ "105" (JavaScript 自動把 5 轉成字符串)*

console.log(a - b); *// ✅ 5 (JavaScript 自動把 "10" 轉成數字)*

console.log(a \* b); *// ✅ 50 (JavaScript 自動把 "10" 轉成數字)*

那C語言是強型別還是弱型別？C語言允許某些隱式轉換 (Implicit Conversion) 。像是允許數值型別的隱式轉(int→float→double)。

*// 範例：以 C++ 作為舉例*

int a = 10;

char b = 'A';

a = b; *// ✅ 允許，因為 'A' 會被轉換為 ASCII 值 65*

b = a; *// ⚠️ 可能導致數據丟失，因為 int 轉 char 會丟失高位數據*

## 範例：隱式型別轉換（int → double）

#include <iostream>

using namespace std;

int main() {

int a = 5;

double b = a; *// ✅ 隱式轉換：int 轉為 double*

cout << "b = " << b << endl; *// 輸出：5*

return 0;

}

## 範例：顯式型別轉換（double → int）

// 以C-style去實現

#include <iostream>

using namespace std;

int main() {

int a = 5;

double b = a; *// ✅ 隱式轉換：int 轉為 double*

cout << "b = " << b << endl; *// 輸出：5*

return 0;

}

*// 以 C++ 語法實現(推薦)*

#include <iostream>

using namespace std;

int main() {

double x = 3.14;

int y = static\_cast<int>(x); *// ✅ C++ 推薦使用的顯式轉型方式*

cout << "y = " << y << endl; *// 輸出：3*

return 0;

}

# auto 關鍵字

auto 會根據變數右邊賦值的型別自動推導變數的資料型別。這是個非常實用的語法糖，讓你不用自己手動寫出變數型別，讓編譯器自動推導。

auto x = 10; *// 編譯器自動推導為 int*

auto y = 3.14; *// 推導為 double*

auto s = std::string("Hello"); *// 推導為 std::string*

*// auto 會 複製值*

for (auto n : nums) {

n += 1; *// 不會改變原本的 nums 裡的數值*

}

*// 想要改變原本的內容，要加上參考 &*

for (auto& n : nums) {

n += 1; *// 會真正改變原本的資料*

}

## 範例：搭配容器與迭代器

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> nums = {1, 2, 3, 4, 5};

for (auto it = nums.begin(); it != nums.end(); ++it) {

cout << \*it << " ";

}

}

# C 語言：C99

為了處理 int、long 長度會因不同平台會變，C99 開出了統一型別

## 範例：C99 有號整數

#include<stdio.h>

#include<stdint.h>

int main(void){

printf("sizeof(int8\_t) = %ld\n", sizeof(int8\_t));

printf("sizeof(int16\_t) = %ld\n", sizeof(int16\_t));

printf("sizeof(int32\_t) = %ld\n", sizeof(int32\_t));

printf("sizeof(int64\_t) = %ld\n", sizeof(int64\_t));

*// 測試不同型別之最小值與最大值*

printf("INT8\_MIN = %d INT8\_MAX = %d\n", INT8\_MIN, INT8\_MAX);

printf("INT16\_MIN = %d INT16\_MAX = %d\n", INT16\_MIN, INT16\_MAX);

printf("INT32\_MIN = %d INT32\_MAX = %d\n", INT32\_MIN, INT32\_MAX);

printf("INT64\_MIN = %ld\nINT64\_MAX = %ld\n", INT64\_MIN, INT64\_MAX);

return 0;

}

*// sizeof(int8\_t) = 1*

*// sizeof(int16\_t) = 2*

*// sizeof(int32\_t) = 4*

*// sizeof(int64\_t) = 8*

*// INT8\_MIN = -128 INT8\_MAX = 127*

*// INT16\_MIN = -32768 INT16\_MAX = 32767*

*// INT32\_MIN = -2147483648 INT32\_MAX = 2147483647*

*// INT64\_MIN = -9223372036854775808*

*// INT64\_MAX = 9223372036854775807*

## 範例：C99 無號整數

#include<stdio.h>

#include<stdint.h>

int main(void){

printf("sizeof(uint8\_t) = %ld\n",sizeof(uint8\_t));

printf("sizeof(uint16\_t) = %ld\n",sizeof(uint16\_t));

printf("sizeof(uint32\_t) = %ld\n",sizeof(uint32\_t));

printf("sizeof(uint64\_t) = %ld\n", sizeof(uint64\_t));

printf("UINT8\_MAX = %u\n", UINT8\_MAX);

printf("UINT16\_MAX = %u\n", UINT16\_MAX);

printf("UINT32\_MAX = %u\n", UINT32\_MAX);

printf("UINT64\_MAX = %lu\n", UINT64\_MAX);

return 0;

}

*// sizeof(uint8\_t) = 1*

*// sizeof(uint16\_t) = 2*

*// sizeof(uint32\_t) = 4*

*// sizeof(uint64\_t) = 8*

*// UINT8\_MAX = 255*

*// UINT16\_MAX = 65535*

*// UINT32\_MAX = 4294967295*