C++程式設計：字元與字串

**目錄**

[C++程式設計：字元與字串 1](#_Toc205646541)

[1. 字元 1](#_Toc205646542)

[1.1 範例：一個 char 所佔的 byte 數 1](#_Toc205646543)

[2. 字元轉值ASCII 2](#_Toc205646544)

[2.1 範例：體會數值→字元的映射。 2](#_Toc205646545)

[3. cin, cout 2](#_Toc205646546)

[3.1 範例：輸入一個字元，分別以 char 和 int 輸出 2](#_Toc205646547)

[3.2 範例：由 cin.get()判斷是否還有資料 3](#_Toc205646548)

[4. 字元分類函式 3](#_Toc205646549)

[4.1 範例：字元分類應用程式 3](#_Toc205646550)

[5. 字元轉換函式 4](#_Toc205646551)

[5.1 範例：用 toupper() 轉換所有字元 4](#_Toc205646552)

[6. EOF(end of file)(參雜char與string) 5](#_Toc205646553)

[6.1 範例：重複獨 char 直到 end of file 5](#_Toc205646554)

[6.2 範例：重複獨字串直到 end of file 5](#_Toc205646555)

[7. 字串 5](#_Toc205646556)

[7.1 字串：getline 6](#_Toc205646557)

[7.2 範例：一次讀一行 6](#_Toc205646558)

[7.3 範例：一次讀兩行 6](#_Toc205646559)

[7.4 字串：empty() 與 clear() 6](#_Toc205646560)

[7.5 字串：length() 7](#_Toc205646561)

[7.6 字串：copy() 7](#_Toc205646562)

[7.7 字串：find () 8](#_Toc205646563)

[7.8 字串：substr () 8](#_Toc205646564)

[7.9 字串：reverse () 9](#_Toc205646565)

[7.10 字串：Sort () 9](#_Toc205646566)

# 字元

## 範例：一個 char 所佔的 byte 數

#include <iostream>

using namespace std;

int main(void){

    char c;

    cout << sizeof(c) << endl;

    return 0;

}

# 字元轉值ASCII

## 範例：體會數值→字元的映射。

用高位 i與低位j生成 i\*16 + j，轉成 char 輸出。你會看到實際的字元表（空白、符號、數字、英文字母）。

#include <iostream>

using namespace std;

int main(void){

    cout << "0123456789abcdef\n";

    for (int i = 2; i <= 7; i++) {

        for (int j = 0; j <= 15; j++) {

        char c = i \* 16 + j;

        cout << c;

        }

        cout << endl;

    }

    return 0;

}

// 0123456789abcdef

//  !"#$%&'()\*+,-./

// 0123456789:;<=>?

// @ABCDEFGHIJKLMNO

// PQRSTUVWXYZ[\]^\_

// `abcdefghijklmno

// pqrstuvwxyz{|}~

# cin, cout

## 範例：輸入一個字元，分別以 char 和 int 輸出

#include <iostream>

using namespace std;

int main(void){

    char c;

    cin.get(c);

    cout << c << endl;

    int i = c;

    cout << i << endl;

    return 0;

}

## 範例：由 cin.get()判斷是否還有資料

#include <iostream>

using namespace std;

int main(void){

    char c;

    while (cin.get(c)){

        if (c == 'q') break; // 輸入 q 就結束

        cout.put(c);

    }

  return 0;

}

# 字元分類函式

|  |  |
| --- | --- |
| 函式名稱 | 說明 |
| isalnum | 英文字母或數字 |
| isalpha | 英文字母 |
| islower | 小寫字母 |
| isupper | 大寫字母 |
| isdigit | 數字 |
| isxdigit | 16 進位數字 |
| isprint | 可顯示字元（含空白） |
| isgraph | 可顯示字元（不含空白） |
| isspace | 空白字元（空格、\t、\n） |
| ispunct | 標點符號 |
| iscntrl | 控制字元 |

## 範例：字元分類應用程式

#include <iostream>

using namespace std;

void check(char c) {

    if (isalnum(c)) cout << "isalnum\n";

    if (isalpha(c)) cout << "isalpha\n";

    if (islower(c)) cout << "islower\n";

    if (isupper(c)) cout << "isupper\n";

    if (isdigit(c)) cout << "isdigit\n";

    if (isxdigit(c)) cout << "isxdigit\n";

    if (isprint(c)) cout << "isprint\n";

    if (isgraph(c)) cout << "isgraph\n";

    if (isspace(c)) cout << "isspace\n";

    if (ispunct(c)) cout << "ispunct\n";

    if (iscntrl(c)) cout << "iscntrl\n";

}

int main(){

    char c = 'a';

    check(c);

    return 0;

}

# 字元轉換函式

|  |  |
| --- | --- |
| 函式名稱 | 說明 |
| toupper(c) | 將字元轉換為大寫（若非字母則無影響） |
| tolower(c) | 將字元轉換為小寫（若非字母則無影響） |

## 範例：用 toupper() 轉換所有字元

#include <iostream>

using namespace std;

int main() {

    char c;

    c = toupper('m'); cout << c;

    c = toupper('4'); cout << c;

    c = toupper('i'); cout << c;

    c = toupper('('); cout << c;

    c = toupper('\n'); cout << c;

    return 0;

}

// M4I(

# EOF(end of file)(參雜char與string)

cin 有一個 eof 函數，可以判斷是否還有資料。若當前狀態為 EOF，回傳 1；反之回傳 0

## 範例：重複獨 char 直到 end of file

#include <iostream>

using namespace std;

int main(void){

    char c;

    cin.get(c);

    while (!cin.eof()){

        cout.put(c);

        cin.get(c);

    }

    return 0;

}

## 範例：重複獨字串直到 end of file

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s;

    cin >> s;

    while (!cin.eof()) {

        cout << '(' << s << ')' << endl;

        cin >> s;

    }

  return 0;

}

# 字串

String s;

cin >> s;

cout << s;

## 字串：getline

希望一次讀一行，遇到空白等字元也不會斷開可以用。

## 範例：一次讀一行

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s;

    getline(cin, s);

    while (!cin.eof()) {

        cout << '(' << s << ')' << endl;

        getline(cin, s);

    }

    return 0;

}

## 範例：一次讀兩行

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s1, s2;

    getline(getline(cin, s1), s2);

    cout << '(' << s1 << ')' << endl;

    cout << '(' << s2 << ')' << endl;

    return 0;

}

## 字串：empty() 與 clear()

* empty()：回傳字串是否為空（true 表示空字串）。
* clear()：清空字串內容。

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s1, s2 = "main()\n\n\n";

    if (s1.empty()) cout << "s1 is empty\n";

    if (s2.empty()) cout << "s2 is empty\n";

    cout << "s1 = " << s1 << endl;

    cout << "s2 = " << s2;

    s2.clear(); // 字串清除

    if (s2.empty()) cout << "s2 is now empty\n";

    return 0;

}

## 字串：length()

回傳字串長度（字符數，不含結尾的 \0）

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s;

    cin >> s;

    cout << "The length of " << s << " is " << s.length();

    return 0;

}

## 字串：copy()

string s1, s2;

s2 = s1;  // 將 s1 複製給 s2

#include <iostream>

#include <string>

using namespace std;

int main(void){

    string s1, s2;

    getline(cin, s1);

    getline(cin, s2);

    cout << "before assignment s2 = " << s2 << endl;

    s2 = s1;

    cout << "after assignment s2 = " << s2 << endl;

    return 0;

}

## 字串：find ()

回傳子字串 s 出現在字串中的起始位置。

**額外補充：size\_t**

Size\_t不是一種函數，是一種表示C++表示「**大小、長度、索引位置**」的整數型別。

⚠ 因為是無號整數，size\_t 永遠大於或等於 0

#include <iostream>

using namespace std;

int main() {

    size\_t length = 5; // 無號整數

    cout << "Length: " << length << endl;

}

**額外補充：string::npos**

npos 是 std::string 定義的一個常數（static const size\_t）。

它的值是 size\_t 可表示的最大值（例如 64 位系統上是 18446744073709551615）。

* 在 find() 找不到目標時會回傳這個值。

#include <iostream>

#include <string>

using namespace std;

int main() {

    string s = "hello";

    size\_t pos = s.find("abc"); // 找不到

    if (pos == string::npos) {

        cout << "Not found" << endl;

    }

}

雖然 npos 的值是 size\_t 的最大值，但因為 size\_t 是無號型別，直接用 -1 會自動轉成最大值（位元全為 1）。

為了可讀性與一致性，C++ 標準建議用 string::npos 來代表「沒找到」。

npos 不只用在 std::string，其他 STL 容器（例如 std::wstring）也有定義。

## 字串：substr ()

從第 pos 個字元開始，擷取長度為 len 的子字串

#include <iostream>

#include <string>

using namespace std;

int main() {

    string path = "/usr/bin/g++";

    size\_t start = path.find\_last\_of('/');

    string filename = path.substr(start + 1);

    cout << filename; // g++

}

## 字串：reverse ()

#include <iostream>

#include <string>

using namespace std;

int main() {

    string s1 = "Hello";

    string s2 = s1;

    for (unsigned i = 0; i < s1.length() / 2; ++i)

        swap(s2[i], s2[s1.length() - 1 - i]);

    cout << s2; // olleH

}

## 字串：Sort ()