



C++

字元字串與陣列



字的處理-可分為字元與字串

- 先說字元，字元是一個字母
以下是輸入一個字元，然後輸出一個字元

```
#include <iostream>
using namespace std;

int main(){
    char cl;
    cin>>cl;
    cout<<(char)cl<<endl;
    return 0;
}
```

輸入：

p

輸出：

p



C++

字串(二個字元以上)



字串的宣告

- 字串也就是多個字元，因此字串會使用字元陣列處理
- 字串的宣告有兩種方式
 - 依照字串長度宣告：

```
char 字串變數[字串長度]
```

- 依照字串內容宣告：

```
char 字串變數[] = "字串"
```

• • •
程式編譯時，會自動從等號指定的字串內容取得字串長度，使用符合的記憶體空間。

字元陣列的定義格式

- 定義格式與陣列一樣，例如：
 - **char ch1[5];**
 - ch1是一個具有5個字元元素的一維字串陣列
 - **char ch2[3][5];**
 - ch2是一個具有15個字元元素的二維字串陣列
 - ch2是一個可以存放三個字串的二維字串陣列

字元陣列的賦值-用字元初始化陣列

- 方法

- `char chr1[5]={ 'a' , 'b' , 'c' , 'd' , 'e' };`

- 說明

- 從首元素開始給予賦值，剩餘字元預設認為空字元。
 - 字串陣列中可以視為是若干個字元，也可以視為一個字串。
 - 區別是字串有一個結束符號(`'\0'`)
 - `char char2[5 ]={ 'a' , 'b' , 'c' , 'd' , '\0' };`
代表在陣列ch2中存放著一個字串" abcd"

字元陣列的賦值-用字串初始化陣列

- 方法
 - `char char2[5]= "abcd" ;`
- 說明
 - 使用此格式時要注意，字串的長度應小於字元陣列的大小，或等於字元陣列的大小減1。
 - 對二維陣列來講，可存放若干個字串。可使用若干個字串組成的初始值，表達二維陣列初始化。
 - `char ch3[3][4]={ "abc" , "def " , " ghe" };`
//在陣列ch3中存放3個字串，每一個字串的長度不得大於3

字元陣列的賦值-陣列元素賦值

- 方法

```
char chr[3];
chr[0] = 'a' ;
chr[1] = 'b' ;
chr[2] = 'c' ;
```
- 注意事項：字元與字串的區別
 - 兩者的定界符不同，字元由單引號括起來，字串則是雙引號。
 - 字元只能是單個字符，字串則可以多個字符。
 - 可以把一個字元常數賦給一個字元變數，但不能把一個字串常數賦給一個字元變數。
 - `char ch1 = 'a' // 正確`
 - `char str[] = "abc" // 正確`
 - `char ch2 = "str" // 錯誤`
 - 字元常數佔一個位元組（byte），而字串常數佔用位元組數等於字串的字節數加1。增加的一個位元組存放字串結束標誌'\0'。例如：字元常數 'a' 佔一個位元組，字串常數 "a" 佔二個位元組。



C++

字串的輸入與輸出



輸入-cin語句

- 格式：

`cin >> 字串名稱;`

- 說明：
 - 讀取字串資料時，會將空白、跳格或換行(\n)符號作為字串結束字元。讀到結束字元後，系統會自動在輸入的字串常數後添加 '\0'，因此輸入時，僅輸入字串的內容即可。
 - 輸入多個字串，且各個字串以空格或跳格等結束字元作為區隔時，則 cin 讀入各個字串變數以 " >> " 分隔。
 - cin 讀資料時，不需要在字串變數前面加 & 這個取址符。

```
#include <iostream>
using namespace std;

int main(){
    char s1[5], s2[5], s3[5], s4[5];

    cin >> s1 >> s2 >> s3;
    // 當鍵盤輸入 "Let us go" 則三個字串分別獲取三個單字。即 s1="Let", s2="us", s3="go"

    cout << s1 << endl; // s1 輸出 Let
    cout << s2 << endl; // s2 輸出 us
    cout << s3 << endl; // s3 輸出 go

    cin >> s4; //當鍵盤輸入 "Let us go" 只會獲取第一個單字。即 s4="Let"
    cout << s4 << endl; // s4 輸出 Let
    return 0;
}
```

輸入-cin.get 語句

- 格式：

字元名稱 = cin.get();

- 說明：
 - 回傳讀取到的下一個字元

```
#include <iostream>
using namespace std;
```

```
int main(){
    char a[5], ch;
    // 當鍵盤輸入"Hello"
    ch = cin.get();
    a[0] = cin.get();
```

```
    cout << ch << endl; // ch 輸出 H
    cout << a << endl; // a 輸出 e
    return 0;
```

```
}
```

輸入-cin.getline語句

- 格式：

`cin.getline(字串名稱, 最多字元數);`

- 說明：
 - 讀取整行資料（包含空白、跳格等字元），只會將換行(\n)符號作為字串結束字元。

```
#include <iostream>
using namespace std;

int main(){
    char s1[20], s2[15];
    // 當鍵盤輸入
    // Hello World
    // Hello C++ Program
    cin.getline(s1, 20);
    cin.get(s2, 15);
    cout << s1 << endl; // s1 輸出 Hello World
    cout << s2 << endl; // s2 輸出 Hello C++ Prog
    return 0;
}
```

輸入-gets語句

- 格式：

gets(字串名稱);

- 說明：
 - 使用 gets 時，要引入：`#include <stdio.h>`
 - 使用gets只能輸入一個字串
 - 每次讀取時會從游標開始的地方讀到換行符，也就是說讀入的是一整行。

```
#include <iostream>
#include <cstdio>

using namespace std;
int main(){
    char s1[20];
    // 鍵盤輸入 Hello World

    gets(s1);
    cout << s1 << endl;
    // s1 輸出 Hello World
    return 0;
}
```

若使用 `cin << s1`
s1 會是 "Hello"

輸入-gets語句

- 格式：

```
gets(字串名稱);
```

- 說明：
 - 使用 gets 時，要引入：`#include <cstdio>`
 - 使用gets只能輸入一個字串
 - 每次讀取時會從游標開始的地方讀到換行符，也就是說讀入的是一整行。
- 例子：
 - `gets(s1, s2);` // 這是錯誤的， gets只能輸入一個字串
 - `gets(s1);` // 當鍵盤輸入"Let us go"，則s1="Let us go"
 - `cin << s2;` // 當鍵盤輸入"Let us go"，則s2="Let"

gets有缺陷，已被拋棄

- gets 函式會讀入整行，直到遇到換行符號才結束輸入，但不會檢查字元陣列邊界問題，因此有可能引發緩衝區溢位的安全問題。

```
#include <iostream>
#include <cstdio>
using namespace std;

int main(){
    char a[5];
    gets(a);
    cout << a;
    return 0;
}
```

當鍵盤輸入 “Let us go” ，gets 讀入此字串時，輸入的字串長度為9卻要放入只有5個空間的字元陣列時，程式會發生什麼事呢？

因此在C99標準中不建議使用，甚至在C11標準中已直接棄用。
※ C99 與 C11 皆為 C 語言標準的一種版本，
C11：2011年發布，C99：1999年發表

輸入-fgets語句

- 格式：

`fgets(字串名稱, 大小, 輸入來源);`

- 說明：
 - 使用 fgets 時，要引入 `#include <stdio.h>`
 - 使用 fgets 只能輸入一個字串
 - 每次讀取時會從游標開始的地方讀到換行符，也就是說讀入的是一整行，但最多只會讀入第二個參數指定的大小，因此可以避免 `gets()` 的緩衝區溢位問題。

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main(){
    char s1[10];
    // 當鍵盤輸入 Hello World
    fgets(s1, sizeof(s1), stdin);
    cout << s1 << endl;
    // s1 輸出 Hello Wor
    return 0;
}
```


輸出-cout語句

- 格式：

cout <<字串名稱

- 說明：
 - 會輸出整個字串。
 - 輸出字串不包括字串結束的識別符號' \0' 。

```
#include <iostream>
#include <cstdio>
using namespace std;

int main(){
    char s1[10], s2[10]="C++";

    // 當鍵盤輸入 Hello World
    fgets(s1, sizeof(s1), stdin);
    cout << s1 << endl;
    // s1 輸出 Hello Wor
    cout << s2 << endl;
    // s2 輸出 C++
    return 0;
}
```

輸出-puts語句

- 格式：

puts(字串名稱);

- 說明：
 - puts語句輸出一個字串和一個換行符號。
 - 對於已經聲明過的字串a，
「cout<<a<< "\n" ;」和「puts(a);」
是相等的。

```
#include <iostream>
#include <cstdio>
using namespace std;

int main(){
    char s1[10]="C++";
    cout << s1 << endl;
    puts(s1);
    // 輸出
    // C++
    // C++
    cout << s1 << endl;
    cout << s1;
    puts(s1);
    cout << s1 << endl;
    // 輸出
    // C++
    // C++C++
    // C++
    return 0;
}
```



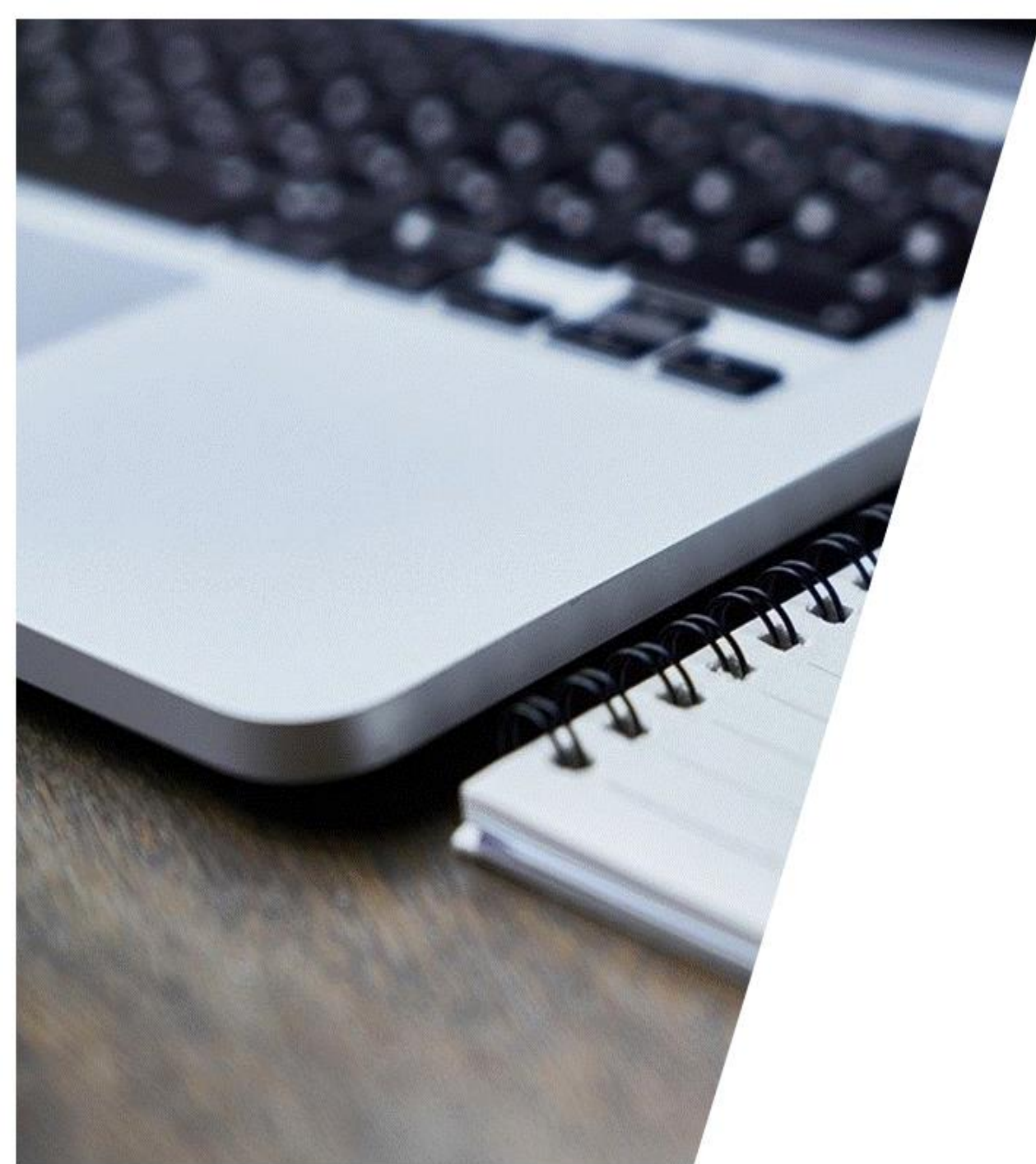
C++

字元陣列處理函式



一些字串函式

函式格式	函式功能
strcat(字串名1, 字串名2)	將字串2連接到字串1後邊，返回字串1的值
strncat(字串名1, 字串名2, 長度n)	將字串2前n個字串連接到字串1後邊，返回字串1的值
strcpy(字串名1, 字串名2)	將字串2複製到字串1，返回字串1的值
strncay(字串名1, 字串名2, 長度n)	將字串2前n個字串複製到字串1，返回字串1的值
strcmp(字串名1, 字串名2,)	比較字串1和字串2的大小，比較的結果由函數帶回； 如果字串1>字串2，返回一個正整數 如果字串1=字串2，返回0 如果字串1<字串2，返回一個負整數
strncmp(字串名1, 字串名2, 長度n)	比較字串1和字串2的前n個字串進行比較，函數返回值的情況同strcmp函數
strlen(字串名)	計算字串的長度，終止符'\0' 不算在長度之內
strlwr(字串名)	將字串中大寫字母換成小寫字母 註：只能在 windows（VC、MinGW等）環境下呼叫
strupr(字串名)	將字串中小寫字母換成大寫字母 註：只能在 windows（VC、MinGW等）環境下呼叫



C++

字串string類型



字元操作的缺點

- 使用字元陣列進行字串操作是比較低階的行為，就如之前所說的，陣列本身對自己的長度沒有意識，所以無法判斷自己是否為空字串，而陣列也不能直接指定給另一個陣列，所以您無法直接將字串指定給另一個字串，您也無法對兩個字串直接進行連接的操作，例如：

```
char str1[ ] = "text1";  
char str2[ ] = "text2";  
str1 = str2 ; // error  
cout << str1 + str2 << endl; // error
```


#include <string>

- string 是一個由 C++ 標準函式庫提供的類別，用來保存 char 的序列容器，把字串的記憶體管理責任交由 string 負責，而不是操作者，減輕了 C 語言處理字串的麻煩。
- C++ 可以使用 string 類別來建立實例，並進行各項高階的字串抽象行為，像是字串的複製、連接等，要使用string類別，您要先引入string標頭檔：

```
#include <string>
```

string 定義(1)

- 第一種 string 定義：

```
string 字串名稱;
```

- 例如：

```
string str1;
```

- 定義一個string變量，內容為空字串
- 這種建構字串的方法會建立一個空字串，空字串也是字串，只是長度為0

string 定義(2)

- 第二種 string 定義：

```
string 字串名稱(字串內容);
```

- 例如：

```
string str2("hello");
```

- 以字串常量建立字串
- 這種建構字串的方法會以字串常數內容來建立string實例

string 定義(3)

- 第三種 string 定義：

```
string 字串名稱(要被複製的字串名稱);
```

- 例如：

```
string str3(str2);
```

- 以string實例建立字串
- 這種建構字串的方法會「複製」另一個字串 str2 的內容，並建立一個新的string實例。



C++

字串string處理函式



一些字串函式

函式格式	函式功能
getline (來源, 字串名)	讀取一行字串
字串名.size()	計算字串的長度，終止符'\0'不算在長度之內
字串名.length()	計算字串的長度，同 字串名.size() 功能
字串名.substr(起始位置i, 長度s)	為字串[i]到字串[i+s]的子字串
字串名1.append(字串名2)	將字串名2的內容複製到字串名1內 ※ 也可以使用 += 運算子連接兩個字串
字串名.empty()	檢查字串名是不是空字串
字串名1.reverse()	字串反轉
字串名1.clear()	將字串內容清空，清空後字串長度會是0
字串名1.swap(字串名2)	字串交換
字串名1.find(字串名2)	在字串名1內搜尋字串名2，如果有找到會回傳找到的位置，如果沒找到會回傳 string::npos

string 範例程式(1)

```
#include <iostream>
#include <string>
using namespace std;

int main(){
    string str, str2="No";
    getline(std::cin, str);

    cout << "長度: " << str.length() << endl;
    cout << "子字串: " << str.substr(2,3) << endl;
    cout << "append: " << str.append("Yes") << endl;
    return 0;
}
```

長度: 11
子長度: llo
append: Hello WorldYes

string 範例程式(2)

```
#include <iostream>
#include <string>
using namespace std;

int main(){
    string str, str2="No";
    getline(std::cin, str);

    str.swap(str2);
    cout << "swap: " << str << endl;
    cout << "swap2: " << str2 << endl;

    str.clear();
    cout << "是不是空的: "<< str.empty() << endl;
    cout << "長度: "<< str.size() << endl;
    return 0;
}
```

```
swap: No
swap2: Hello World
是不是空的: 1
長度: 0
```