# 當class的成員碰上了指標

在寫程式時難免會遇到想對資料取位址,而這資料若是某個特定的類別裡面的成員,例如:

|  |
| --- |
| class MyClass {  public:  //…  int mData;  //…  }; |

要定義一個指標指向MyClass的資料成員mData, 有兩種方法

|  |
| --- |
| int MyClass::\*ptrData = &MyClass::mData; //利用類別名稱和範圍運算子來定義,代表此指標是指向該類別的資料成員.  /\*\*  \* 先宣告類別的物件, 再直接定義一個指標指向物件的資料成員  \*/  MyClass myObject;  int \*ptrSimple = &myObject.mData; |

這兩個定義的形式差別在於使用方式.第一種定義是由類別來指定,所以使用的時候需要由該類別的物件來使用,例如

|  |
| --- |
| cout << "myObject.\*ptrData = " << myObject.\*ptrData << endl; |

第二種定義就直接用指標的方式來使用就行了

|  |
| --- |
| cout << "\*ptrSimple = " << \*ptrSimple << endl; |

以上是指向物件的資料成員的使用方法,然而還有一種資料成員是屬於類別的資料成員也就是靜態成員,例如

|  |
| --- |
| class MyClass {  public:  //...  static int sData;  //...  };  int MyClass::sData = 123; |

靜態成員就不分各物件了,一班會用來當全域資料來用.定義如下

|  |
| --- |
| int \*ptrStaticData = &MyClass::sData; //取指的方式一樣需要用類別加上範圍運算子 |

這次的指標定義就直接取個指標變數就好,前面無須再多個類別和範圍運算子.

使用方式就跟一般使用指標的方式一樣.

|  |
| --- |
| cout << "\*ptrStaticData = " << \*ptrStaticData << endl; |

類別的成員除了資料還有就是函數.所以有關函數指標的寫法如下所示:

假如有個類別

|  |
| --- |
| class MyClass {  public:  // ...  void sayHello() {  cout << "Hello in class" << endl;  }  // ...  static void sayHi() {  cout << "I am say hi in class" << endl;  }  // ...  }; |

函數成員也是有分靜態和物件型態.函數指標定義的方式如下

|  |
| --- |
| void (MyClass::\*ptfn)() = &MyClass::sayHello;  void (\*ptStaticFn)() = &MyClass::sayHi; |

跟資料成員一樣, 物件函數成員指標在定義的時候是需要類別加上範圍運算子,而類別函數成員指標只要跟一般的指標定義一樣就好.使用的方式也有差別.

|  |
| --- |
| (myObject.\*ptfn)(); // 需要由物件帶出使用  (\*ptStaticFn)(); // 跟一般的函數指標一樣的方式使用 |