C/C++基礎程式設計

C++: 物件的使用、參考、重載函式

講師：張傑帆
CSIE NTU

成就別人認為不可能的事。
Do what nobody else considered possible. -Steve Jobs

C++相較於C的特色

- 向下相容
  - 在C語言中，我們學了許多程式語法，所有學過的東西，在C++中都可以使用
- 高階的程式描述方式
  - 更利於用來開發大型專案，讓程式設計師在分工時更能快速的開發程式，並減少錯誤的產生
- 物件導向程式設計
  - 讓開發程式者簡單的使用物件所提供的功能，來達到所需要的效果

課程大綱

- **C++基礎語法**
  - 第一個C++程式: Hello World !
  - 輸入輸出
  - 動態記憶體配置
  - 字串
  - 參考 (Reference)
  - 重載函式 (Over Loading)

C++的程式架構

- C++是由C延伸出來的，因此C++當然包含C語言的所有功能，另外提供完整物件導向程式設計(Object-Oriented Programming：OOP)功能
- C++可說是C語言的加強版
- C++的程式架構和C語言很類似，而且程式的進入點都是由main()函式開始

新式標頭

- C程式中若有使用到**printf()**和**scanf()**輸出入函式時，由於這些函式的宣告都在**stdio.h**標頭檔內，因此必須先將此標頭檔含入到程式的最前面，寫法如下：
- **#include<stdio.h>**

新式標頭

- C++逐漸取代傳統的C語言，為了能向下與C語言相容，標準的ANSI/ISO C++(簡稱標準C++)仍支援傳統C語言的標頭檔
- 只不過在C++程式中，若有使用到這些C語言所提供的標頭檔時，我們建議使用新式標頭寫法。
- 其做法就是在標頭檔名稱最前面加上小寫的c和省略副檔名*.h即可。例如：
- #include<cstdio>

- 標準C++為了提升功能，引進新的C++標準程式庫函式，當然也使用新式標頭檔寫法，以和ANSI C有所區別。
- 由於新式標頭不是檔名而是一個標頭名稱，是由識別字組成，因此在含入新式標頭名稱時不要加上 .h副檔名。
- 下列即是標準C++ 所提供一些新式標頭名稱寫法：

- #include<iostream>
  #include<string>
  #include<fstream>
  #include<list>
  #include<vector>
  #include<map>

輸入/出資料

- 標準的C++程式，則是使用**cout**和**cin**物件來輸出入資料。
- 由於這兩個物件的宣告都定義在C++標準程式庫中的**iostream**。
- 在C++ 的程式中，若有使用到cin或cout物件，都必須在程式最前面先將iostream含入進來。寫法如下：
- **#include<iostream>**

- C++寫一個簡單cout程式，將 “Hello,” 和 “這是第一個C++程式” 兩個字串分別用cout逐行輸出：

- Hello,
  這是第一個C++程式

```
// Program : greeting.cpp
# include <iostream>
int main(int argc, char *argv[])
{
    std::cout << "Hello,\n ";
    std::cout << "這是第一個 C++程式." <<std::endl;
    system("PAUSE");
    return 0;
}
```

註解開頭

程式中使用到 cout 必須含入此標題名

程式由此開始執行，int 表示傳回值的資料型別為整數

輸出雙引號括住的字串常數，並將游標移到下一行的開頭

將 0 傳回給系統表示系統正常結束
若傳回非零值代表失敗

- C++寫一個簡單cout程式，將 “Hello,” 和 “這是第一個C++程式” 兩個字串分別用cout逐行輸出：

- Hello,
  這是第一個C++程式

命名空間

- 已標準化後的C++，標頭檔的所有內容放在std命名空間中，因此程式中使用到cout或cin時，就必須如上面敘述在前面加上 std::，此種寫法造成撰寫上的不方便。
- 為解決此問題C++ 提供using directive方式，即在程式之前加上 **using namespace std;** 敘述把std完全打開，讓std命名空間變得可見，如此上面範例在程式中便可將 std:: 拿掉。

```
#include <cstdlib>

#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout << "Hello,\n ";
    cout << "這是第一個 C++程式." <<endl;
    system("PAUSE");
    return 0;
}
```

C++: Helloworld

- 第一個C++程式:Helloworld

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

int main()
{
	cout << "Hello World!" << endl;
	return 0;
}
```

C++宣告變數

- 在C++中,變數宣告不一定要在函式一開始就通通宣告,只要在使用前宣告即可
- 以下範例在C語言中無法正確編譯，但C++可以

```
int main()
{
        int a;
        a = 10;

        double b;
        b = 10.5;

        return 0;
}
```

C++輸入輸出

- **cout物件: 輸出**
  - << : 將一個指定的內容傳給cout輸出
- **cin物件: 輸入**
  - >> : 將cin輸入值傳給某個儲存單位

(有幾個內容就用幾個<<或>>)

```
#include <iostream>
using namespace std;
int main()
{
        int num;

        cin >> num;
        cout << num << endl;

        return 0;
}
```

C++輸入輸出

- 試著輸入輸出不同型態的資料
- 請觀察跟printf函式使用上的差異

```
#include <iostream>
using namespace std;
int main()
{
        int a;
        double b;
        char c;
        char d[80];

        cout << "請分別輸入整數, 小數, 字元, 字串" << endl;
        cin >> a;
        cin >> b;
        cin >> c;
        cin >> d;

        cout << "輸入的內容為: " << endl;
        cout << a << endl;
        cout << b << endl;
        cout << c << endl;
        cout << d << endl;
        return 0;
}
```

C++輸入輸出

- **思考: 為什麼cin/cout物件都不用跟告訴它我要印到螢幕的資料型態，scanf/printf函式就要?**
  - 因為C++的cin/cout物件比較聰明，所以比較容易使用
  - 在聰明的背後...
    (使用物件很容易，自己設計物件比較困難!)

輸入含有空白字元的字串

- 使用cin物件提供的getline函式 (類似gets函式)
- 語法: cin.getline(字元陣列名稱，最大長度);
  - 輸入字串放到指定字元陣列中
  - 若輸入字串長度超過 最大長度-1 則自動捨去

```
#include <iostream>
using namespace std;
int main()
{
        char a[80];

        cin.getline(a, 80);
        cout << a << endl;

        return 0;
}
```

輸入含有空白字元的字串

- 使用cin的>>與getline會產生像在C語言中使用scanf與gets的問題
- 使用cin物件提供的cin.ignore()解決

```
#include <iostream>
using namespace std;
int main()
{
        char a[80];
        char b[80];

        cin >> a;
        cin.ignore();
        cin.getline(b, 80);

        cout << a << endl << b << endl;
        return 0;
}
```

C++動態記憶體配置

- 動態記憶體配置: **new**
  - 配置一個資料空間，並傳回該空間的位址，語法:
    - 指標 = new 資料型態;
  - 配置一個給定初始值的空間，並傳回該空間的位址，語法:
    - 指標 = new 資料型態(初始值);

- 釋放記憶體: **delete**
  - 配置一個空間的釋放
    - delete 指標;

```
#include <iostream>
using namespace std;
int main()
{
    int *ptr = new int(100);
    cout << "空間位置:" << ptr << endl;
    cout << "空間儲存值：" << *ptr << endl;

    *ptr = 200;
    cout << "空間位置:" << ptr << endl;
    cout << "空間儲存值：" << *ptr << endl;
    delete ptr;

    return 0;
}
```

C++動態記憶體配置

- 動態記憶體配置: **new**
  - 配置多個資料空間，並傳回該空間的位址，語法:
    - 指標 = new 資料型態[個數];
- 釋放記憶體: **delete**
  - 配置多個空間的釋放
    - delete [] 指標;

```
#include <iostream>
using namespace std;
int main()
{
    int *ptr;
    int size, i;

    cout << "請輸入個數：";
    cin >> size;
    ptr = new int[size];
    cout << "請輸入內容：" << endl;
    for(i = 0; i < size; i++) {
        cout << "ptr[" << i << "] = ";
        cin >> ptr[i];
    }
    for(i = 0; i < size; i++)  {
        cout << "ptr[" << i << "] = " << ptr[i] << endl;
    }
    delete [] ptr;
    return 0;
}
```

C++字串: string

- 使用C++提供的特殊字串型態**string**可以用來宣告字串物件，方便我們做字串處理
- 宣告語法：
  - **string 字串物件名稱;**
- 產生的字串物件提供下面語法可以使用：
  - [索引]: 取得索引值代表的字元
  - =: 字串複製
  - == : 字串比對
  - += : 字串連結
  - length(): 計算字串長度
  - c_str(): 回傳字串位置 (常用在字串函式)

```
getline(cin, str, '\n');
可輸入含"空白"的字串
```

C++字串: string

- 範例：輸入字串後印出長度與所有字元

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string str;
        int n, i;

        cout << "請輸入字串: ";
        cin >> str;

        n = str.length();
        cout << "輸入的長度為: " << n << endl;
        cout << "輸入的字元為: " << endl;
        for(i=0; i<n; i++)
            cout << "[" << i << "]:" << str[i] << endl;

        return 0;
}
```

C++字串: string

- 範例：string常用之運算
  - 字串比對: a==b //比對a與b是否相等
  - 字串複製: a = b // 將b複製到a
  - 字串連結: a+=b // 將b連結到a後面

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string a, b;

        a = "Hello";
        cout << "請輸入b字串: ";
        cin >> b;
        if(a==b)
                cout << "b字串a字串相同" << endl;
        else
                cout << "b字串a字串不同" << endl;
        a+=b;
        cout << "a,b字串連結結果: " << a << endl;
        return 0;
}
```

C++字串: string

- 範例：輸入b字串內容後儲存a

string → 字元陣列

```
#include <iostream>
#include <string>
#include <string.h>
using namespace std;
int main()
{
    char a[80];
    string b;

    cout << "輸入string字串b: ";
    cin >> b;

    strcpy(a, b.c_str());

    cout << "輸出字元陣列a: " << a << endl;
    return 0;
}
```

字元陣列 → string

```
#include <iostream>
#include <string>
#include <string.h>
using namespace std;
int main()
{
    char a[80];
    string b;

    cout << "輸入字元陣列a: ";
    cin >> a;

    b = a;

    cout << "輸出string字串b: " << b << endl;
    return 0;
}
```

進階練習

- 請撰寫一程式令
- 使用C++動態記憶體配置
- 令使用者輸入陣列大小ex:5
- 然後令使用者輸入n個整數或字串後使用泡沫排序法排序
- 最後將結果列出

課程大綱

- **C++基礎語法**
  - 第一個C++程式: Hello World !
  - 輸入輸出
  - 動態記憶體配置
  - 字串
  - 參考 (Reference)
  - 重載函式 (Over Loading)

C++ 參考（Reference）

- 參考(Reference)型態代表了變數或物件的一個別名(Alias)
- 參考型態可以直接取得變數或物件的位址，並間接透過參考型態別名來操作物件
- 作用類似於指標，但卻不必使用指標語法，也就是不必使用*運算子來提取值。
- 要定義參考型態，在定義型態時於型態關鍵字後加上&運算子，例如：
  - `int var = 10;` // 定義變數
  - `int *ptr = &var;` // 定義指標，指向var的位址
  - `int &ref = var;` // 定義參考，代表var變數

C++ 參考（Reference）

- 為何參考型態一定要初始化？
  - 因為參考初始化後就不能改變它所代表的物件，任何指定給參考的值，就相當於指定給原來的物件

```
#include <iostream>
using namespace std;
int main()
{
        int var = 10;
        int *ptr = &var;
        int &ref = var;
        cout << "var: " << var << endl;
        cout << "*ptr: " << *ptr << endl;
        cout << "ref: " << ref << endl;
        ref = 20;

        cout << "var: " << var << endl;
        cout << "*ptr: " << *ptr << endl;
        cout << "ref: " << ref << endl;
        return 0;
}
```

C++ 參考（Reference）

- 參考型態最常使用於函式的參數列上

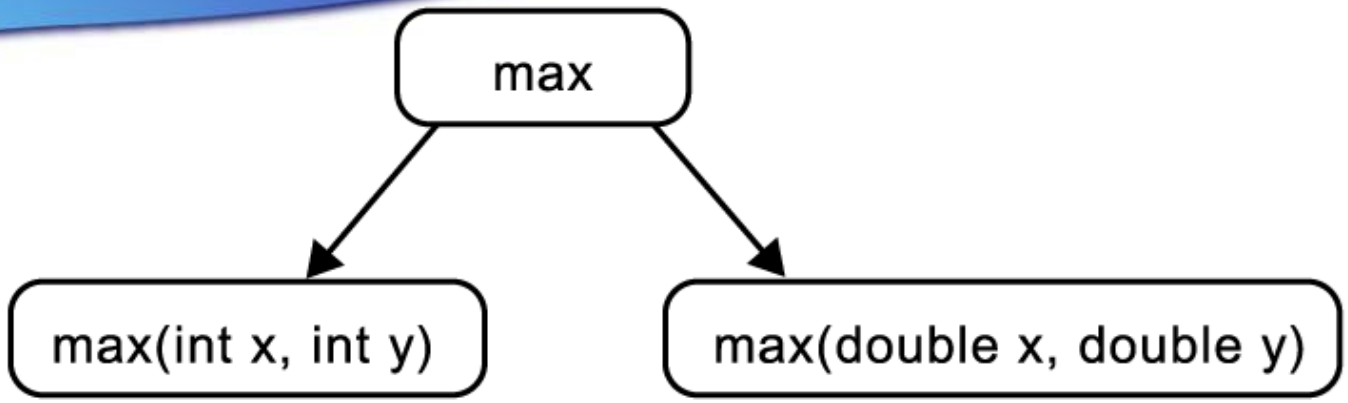
```
#include <iostream>
using namespace std;

void increment(int &n)
{
        n = n + 1;
}

int main()
{
        int x = 10;

        increment(x);
        cout << x << "\n";

        return 0;
}
```

練習

- 使用參考寫一個MySwap函式，將兩個變數內容交換
  - ex: int a=10, b=5;
    MySwap(a,b); // a=5, b=10

C++重載函式

- **理解多載（overload）的原理**
  - 定義多個名稱相同，但引數的型態和數量不同之函數，就叫作函數的多載（多重定義：function overloading）。
  - 當函數進行多載時，可以依照呼叫時所傳遞的引數，來呼叫出適當的函數。
  - 要進行多載的函數，其引數的型態或個數都必須不同，否則將無法判斷要呼叫的函數，因此也必須避免使用預設引數。




```
int main()
{
   int ans1 = max(a, b);
   double ans2 = max(da, db);
}
```

```
int max(int x, int y)
{

}
```

引數為int型態

```
double max(double x, double y)
{

}
```

引數為double型態

C++重載函式

- C++支援函式「重載」(Overload)，根據回傳值的不同或參數列個數或型態的不同，而自動呼叫對應的函式

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

void show(int x)
{
        cout << "我有一個整數int : " << x << endl;
}
void show(int x, int y)
{
        cout << "我有兩個整數int : " << x << ", " << y << endl;
}

int main()
{
        show(10);
        show(20, 30);
        return 0;
}
```

C++重載函式

- 根據參數的型態來決定要呼叫的函式

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

void show(int x)
{
        cout << "我是整數int : " << x << endl;
}
void show(double x)
{
        cout << "我是小數double : " << x << endl;
}

int main()
{
        show(12);
        show(12.5);
        return 0;
}
```

練習

- 延續上頁範例，新增一個函式show將上一章的struct Person型態中的姓名，身高，體重輸出
  - void show(struct Person x);

C++重載函式

- **思考: 為什麼要使用重載函式? 宣告成不同函式名稱不好嗎?**
  - 從使用函式角度...
    功能一樣但參數不同的函式不用取多個名字比較好記!