Lab of Object-Oriented Programming: More on Class

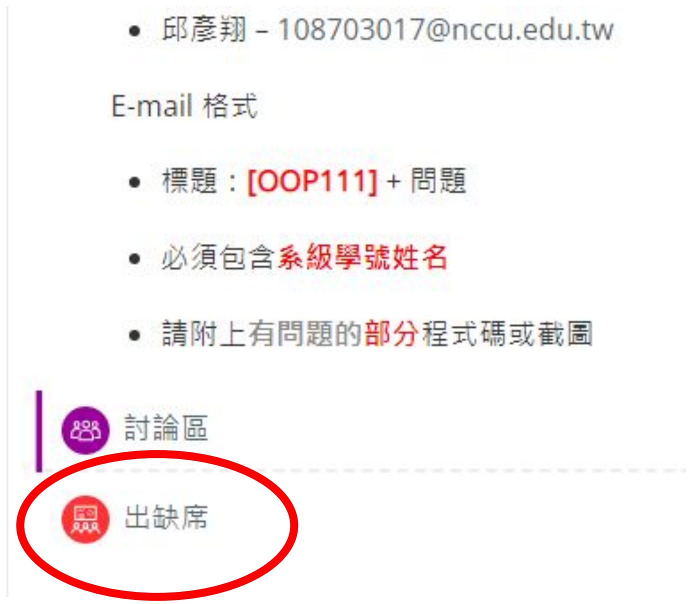
黃威、陳岳紘、邱彥翔

2022

使用 moodle 點名

請登入實習課的 moodle 課程

點擊出缺席並完成今日的點名



- 邱彥翔 – 108703017@nccu.edu.tw

E-mail 格式

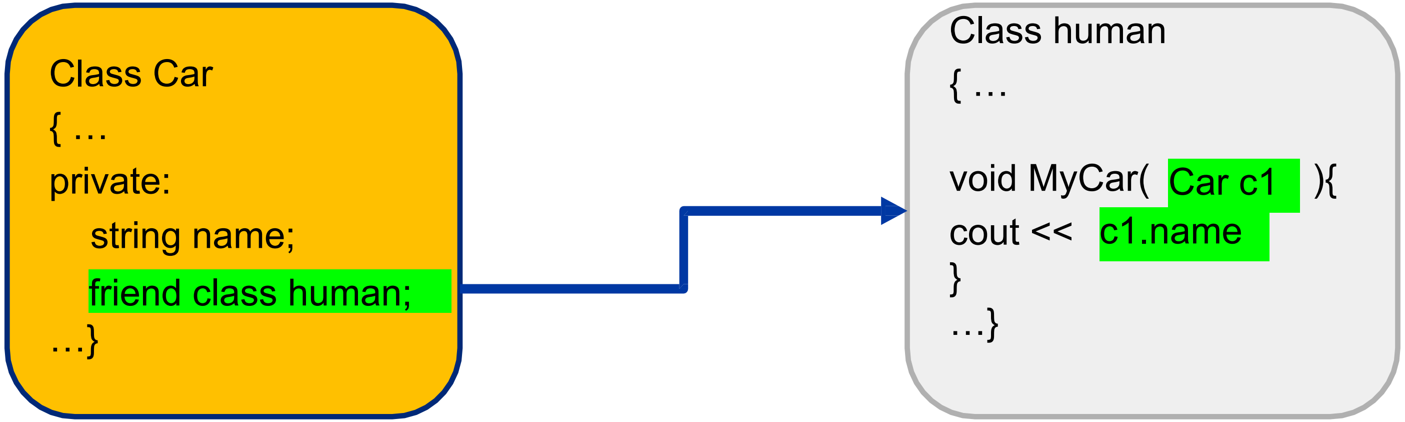
- 標題：[OOP111] + 問題
- 必須包含系級學號姓名
- 請附上有問題的部分程式碼或截圖

Outline

- Friend Class
- Copy Constructor
- Inline function
- Exercise5

Friend Class

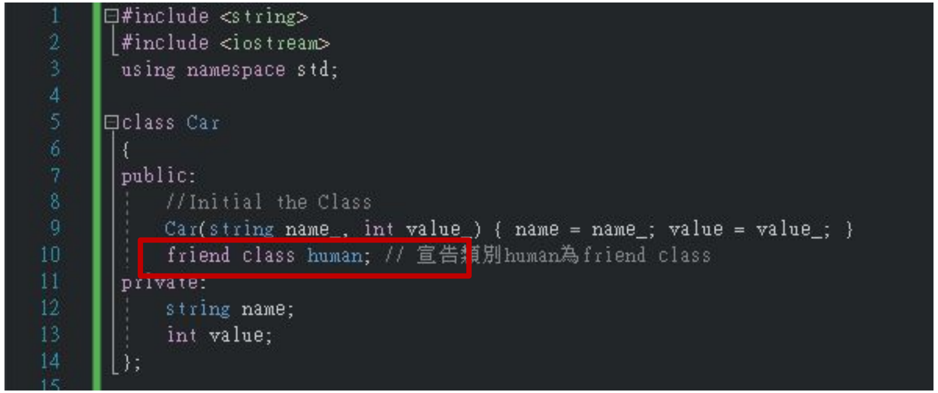
- 正常情況下，無法從外部存取Class 的 private 變數
- “ **Friend Class** ” 作為一種 “ 機制 ” 幫助我們達成不同class 互動情形
    - 讓其他 class 可以 **access 不同 class 的 private** 變數



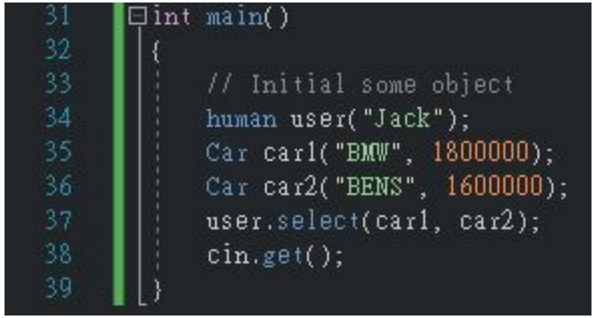
```
Class Car
{ …
private:
    string name;
    friend class human;
…}
```

```
Class human
{ …

void MyCar( Car c1 ){
cout << c1.name
}
…}
```

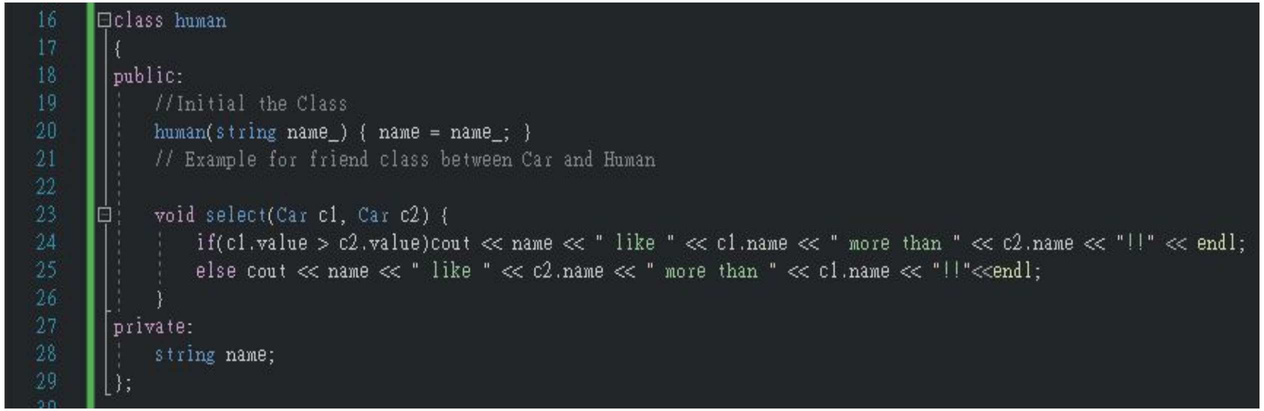
Friend Class



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

class Car
{
public:
    //Initial the Class
    Car(string name_, int value_) { name = name_; value = value_; }
    friend class human; // 宣告類別human為friend class
private:
    string name;
    int value;
};
```

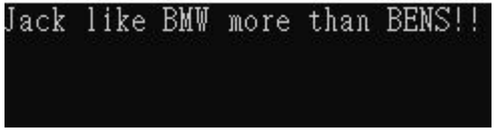


```cpp
int main()
{
    // Initial some object
    human user("Jack");
    Car car1("BMW", 1800000);
    Car car2("BENS", 1600000);
    user.select(car1, car2);
    cin.get();
}
```



```cpp
class human
{
public:
    //Initial the Class
    human(string name_) { name = name_; }
    // Example for friend class between Car and Human

    void select(Car c1, Car c2) {
        if(c1.value > c2.value)cout << name << " like " << c1.name << " more than " << c2.name << "!!" << endl;
        else cout << name << " like " << c2.name << " more than " << c1.name << "!!"<<endl;
    }
private:
    string name;
};
```

Output:



```
Jack like BMW more than BENS!!
```

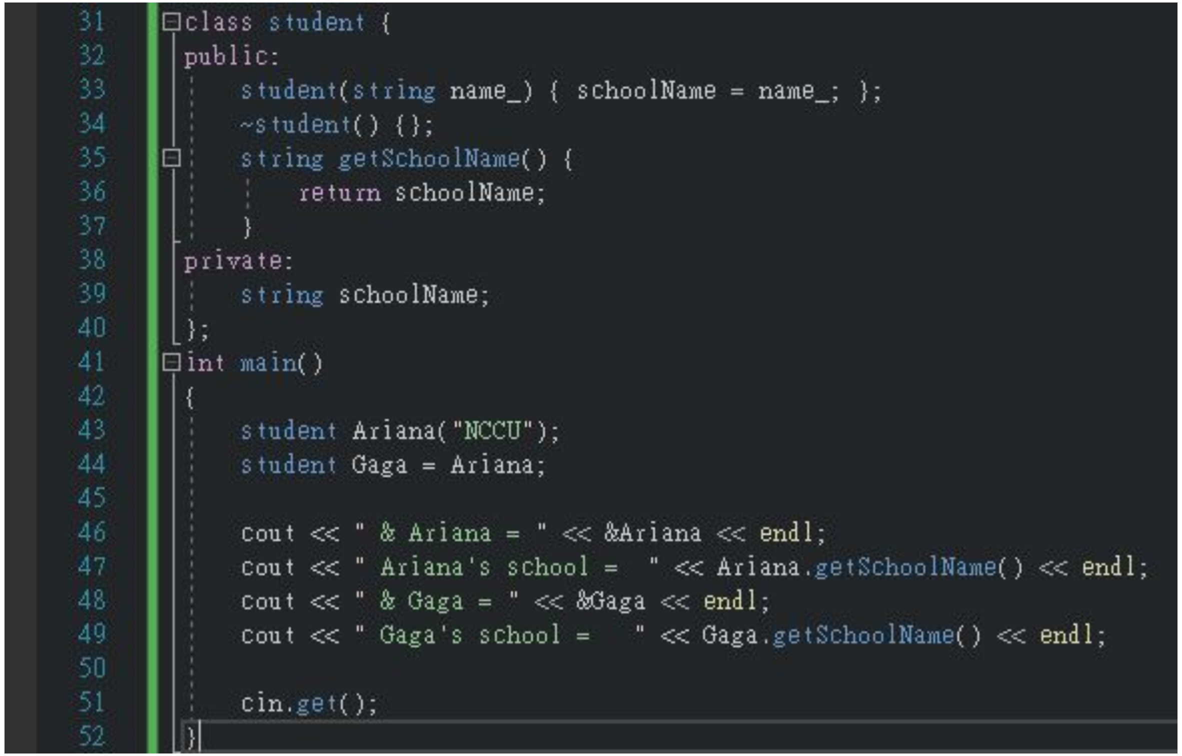
Copy Constructor

- 宣告一個物件，並使用另一個物件的內容當作此物件的初始內容

```
SomeClass obj1;
SomeClass obj2 = obj1;
```

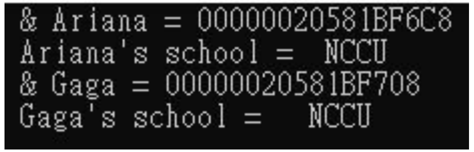
- obj1 的**所有屬性**都會被**複製**到obj2的每一個屬性。兩物件的**記憶體位置不同**。

Copy Constructor



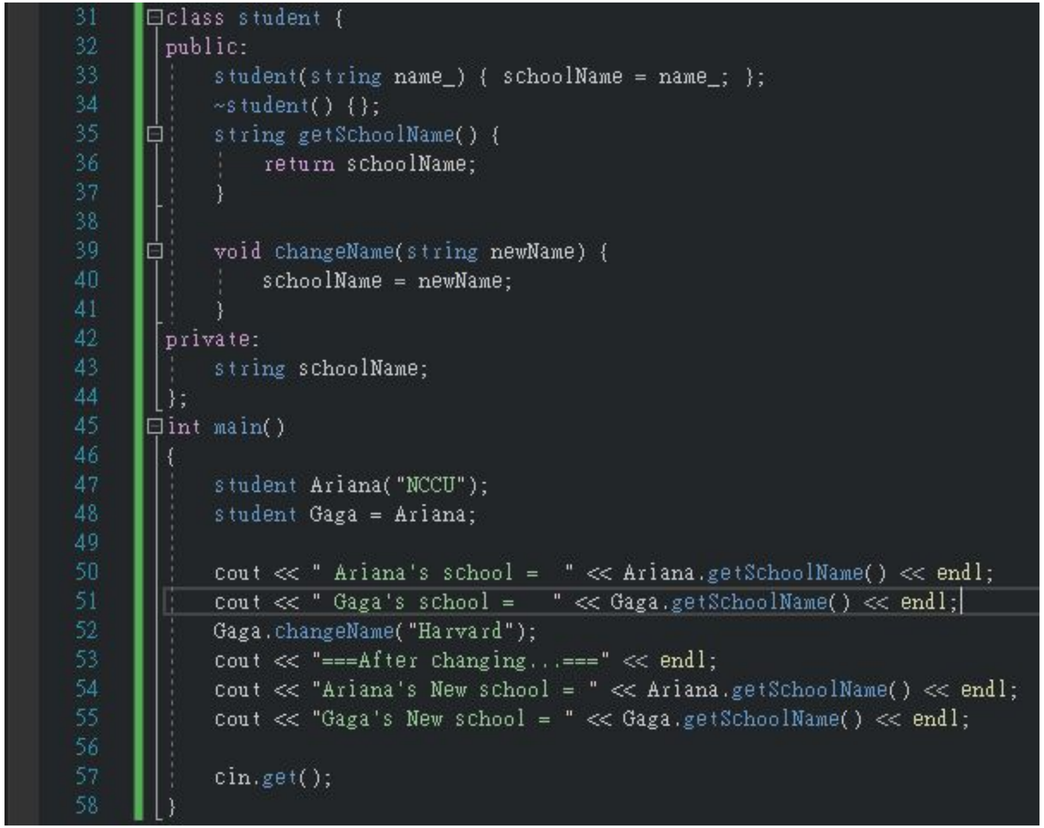
```
class student {
public:
    student(string name_) { schoolName = name_; };
    ~student() {};
    string getSchoolName() {
        return schoolName;
    }
private:
    string schoolName;
};
int main()
{
    student Ariana("NCCU");
    student Gaga = Ariana;

    cout << " & Ariana = " << &Ariana << endl;
    cout << " Ariana's school =  " << Ariana.getSchoolName() << endl;
    cout << " & Gaga = " << &Gaga << endl;
    cout << " Gaga's school =   " << Gaga.getSchoolName() << endl;

    cin.get();
}
```

Output:



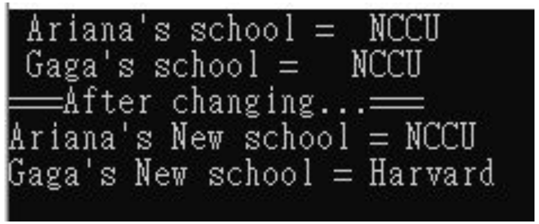
```
& Ariana = 00000020581BF6C8
Ariana's school =  NCCU
& Gaga = 00000020581BF708
Gaga's school =   NCCU
```

Copy Constructor



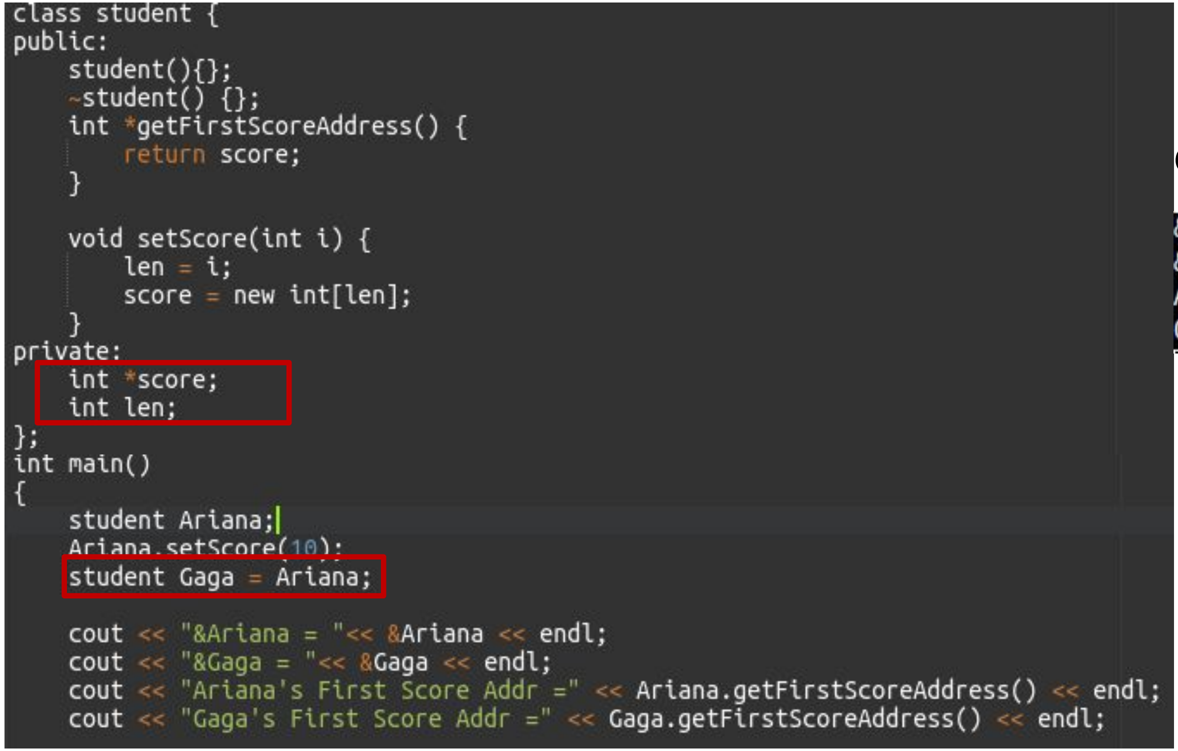
```cpp
class student {
public:
    student(string name_) { schoolName = name_; };
    ~student() {};
    string getSchoolName() {
        return schoolName;
    }

    void changeName(string newName) {
        schoolName = newName;
    }
private:
    string schoolName;
};
int main()
{
    student Ariana("NCCU");
    student Gaga = Ariana;

    cout << " Ariana's school =  " << Ariana.getSchoolName() << endl;
    cout << " Gaga's school =    " << Gaga.getSchoolName() << endl;
    Gaga.changeName("Harvard");
    cout << "===After changing...===" << endl;
    cout << "Ariana's New school = " << Ariana.getSchoolName() << endl;
    cout << "Gaga's New school = " << Gaga.getSchoolName() << endl;

    cin.get();
}
```

Output:



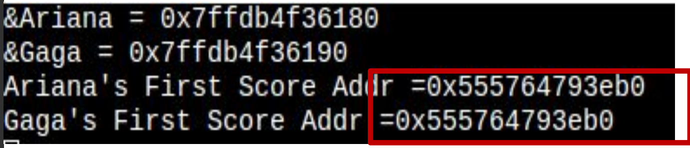
```
 Ariana's school =  NCCU
 Gaga's school =    NCCU
===After changing...===
Ariana's New school = NCCU
Gaga's New school = Harvard
```

Problem : Pointer Address



```cpp
class student {
public:
    student(){};
    ~student() {};
    int *getFirstScoreAddress() {
        return score;
    }

    void setScore(int i) {
        len = i;
        score = new int[len];
    }
private:
    int *score;
    int len;
};
int main()
{
    student Ariana;
    Ariana.setScore(10);
    student Gaga = Ariana;

    cout << "&Ariana = "<< &Ariana << endl;
    cout << "&Gaga = "<< &Gaga << endl;
    cout << "Ariana's First Score Addr =" << Ariana.getFirstScoreAddress() << endl;
    cout << "Gaga's First Score Addr =" << Gaga.getFirstScoreAddress() << endl;
```

Output:

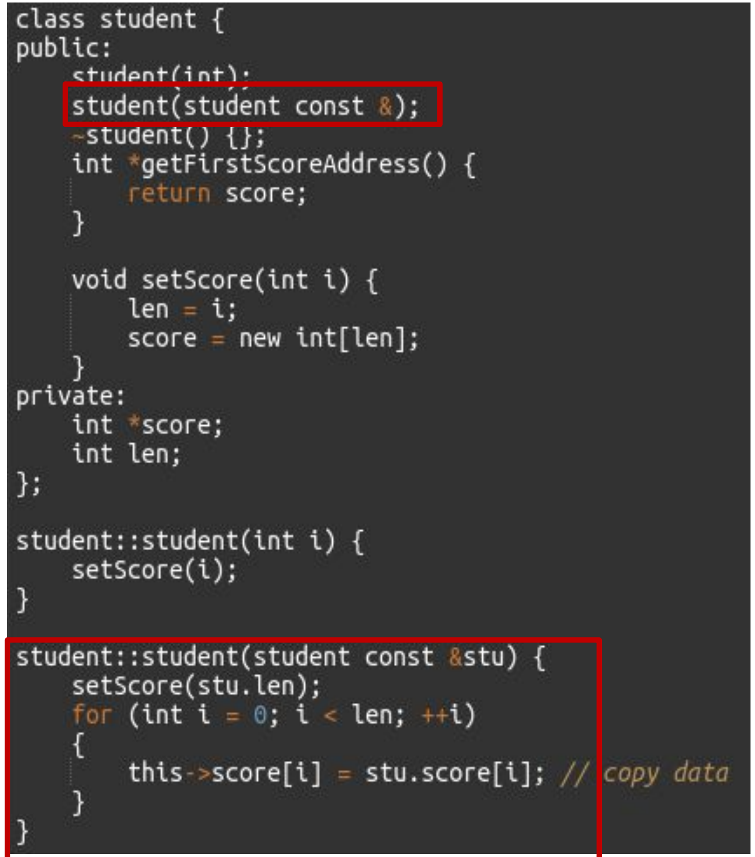


```
&Ariana = 0x7ffdb4f36180
&Gaga = 0x7ffdb4f36190
Ariana's First Score Addr =0x555764793eb0
Gaga's First Score Addr =0x555764793eb0
```

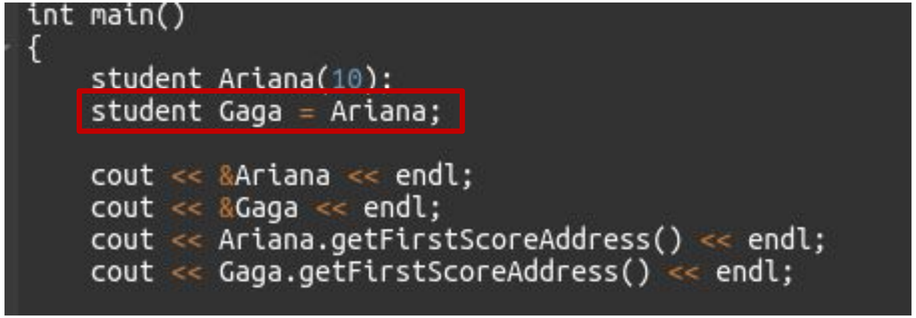
Problem : Pointer Address

- Class中有**指標屬性**，使用前面的 Copy 的方法也會將同樣的指標位址複製到後來的物件中。若**原本的物件被 delete掉之後，後來的物件中的指標屬性將會指向一個被釋放掉的記憶體空間** 。
- (e.g.,Gaga複製了Ariana的屬性，當然也包括了score指標，如果Ariana資源先被回收了，但Gaga的score仍然參考至一個已被回收資源的位址，這時再存取該位址的資料就有危險)
- **解決辦法**：用**Menber function**來避免動態配置屬性時有可能發生的問題。當遇到指標成員時，**產生一個新的資源並指定位址** 給該成員。

Solve with Copy Constructor : Pointer Address

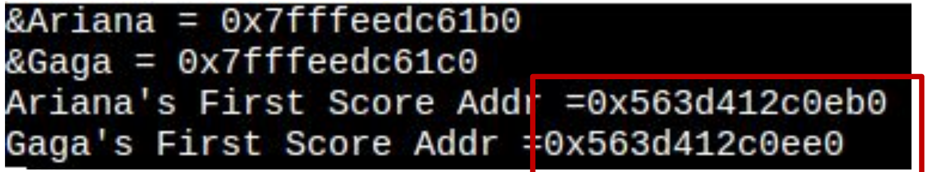


```
class student {
public:
    student(int);
    student(student const &);
    ~student() {};
    int *getFirstScoreAddress() {
        return score;
    }

    void setScore(int i) {
        len = i;
        score = new int[len];
    }
private:
    int *score;
    int len;
};

student::student(int i) {
    setScore(i);
}

student::student(student const &stu) {
    setScore(stu.len);
    for (int i = 0; i < len; ++i)
    {
        this->score[i] = stu.score[i]; // copy data
    }
}
```



```
int main()
{
    student Ariana(10);
    student Gaga = Ariana;

    cout << &Ariana << endl;
    cout << &Gaga << endl;
    cout << Ariana.getFirstScoreAddress() << endl;
    cout << Gaga.getFirstScoreAddress() << endl;
```

Output:

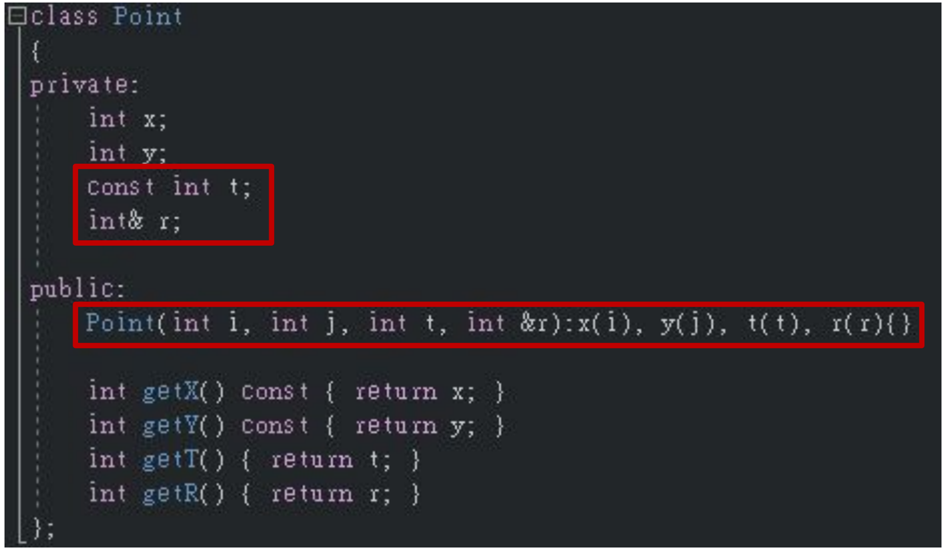


```
&Ariana = 0x7fffeedc61b0
&Gaga = 0x7fffeedc61c0
Ariana's First Score Addr =0x563d412c0eb0
Gaga's First Score Addr =0x563d412c0ee0
```

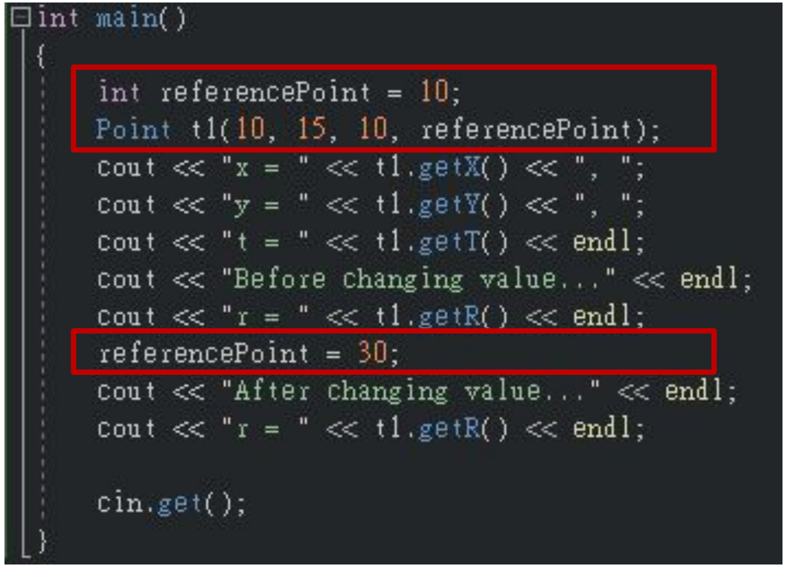
Initialization list

- 用於物件初始化。不能在建構式用指定的方式給予初值。
- 除了在建構函式中設定初始值，member variable是**其他類別的物件**、**reference**，**const 型別**的pointer，要使用initialization list來設定初始值。

initialization list

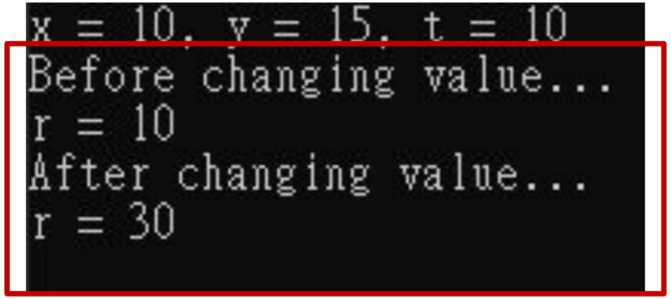


```
class Point
{
private:
    int x;
    int y;
    const int t;
    int& r;

public:
    Point(int i, int j, int t, int &r):x(i), y(j), t(t), r(r){}

    int getX() const { return x; }
    int getY() const { return y; }
    int getT() { return t; }
    int getR() { return r; }
};
```



```
int main()
{
    int referencePoint = 10;
    Point t1(10, 15, 10, referencePoint);
    cout << "x = " << t1.getX() << ", ";
    cout << "y = " << t1.getY() << ", ";
    cout << "t = " << t1.getT() << endl;
    cout << "Before changing value..." << endl;
    cout << "r = " << t1.getR() << endl;
    referencePoint = 30;
    cout << "After changing value..." << endl;
    cout << "r = " << t1.getR() << endl;

    cin.get();
}
```

Output:

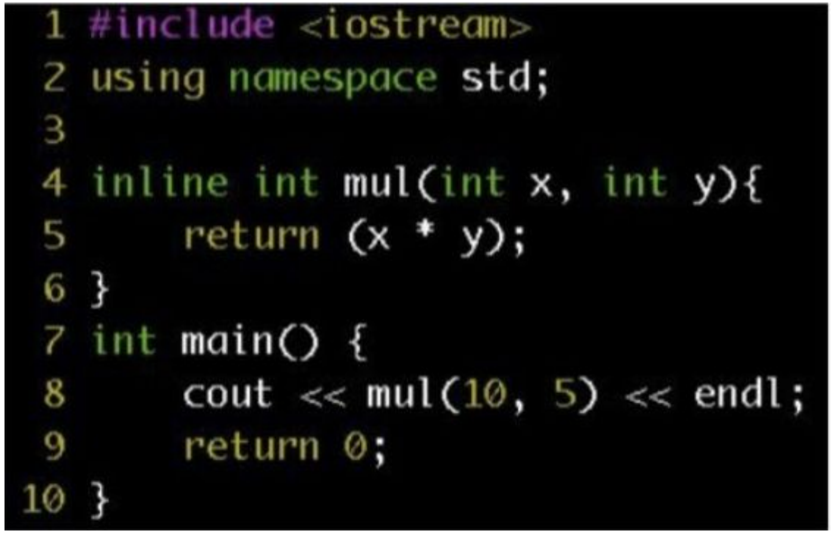


```
x = 10, y = 15, t = 10
Before changing value...
r = 10
After changing value...
r = 30
```

Inline Function

- inline放在**function前**, 稱為內嵌函式, 目的是為了**增加執行速度** 。
- Compiler在**編譯時會將 內嵌函式展開**, 因此不必花大量CPU time處理函式呼叫 和 回傳, 執行速度上升。
- 如果函式**短且常被呼叫**, 可以考慮加上inline。
- 但compiler會自己評估效率來決定是否忽略inline。

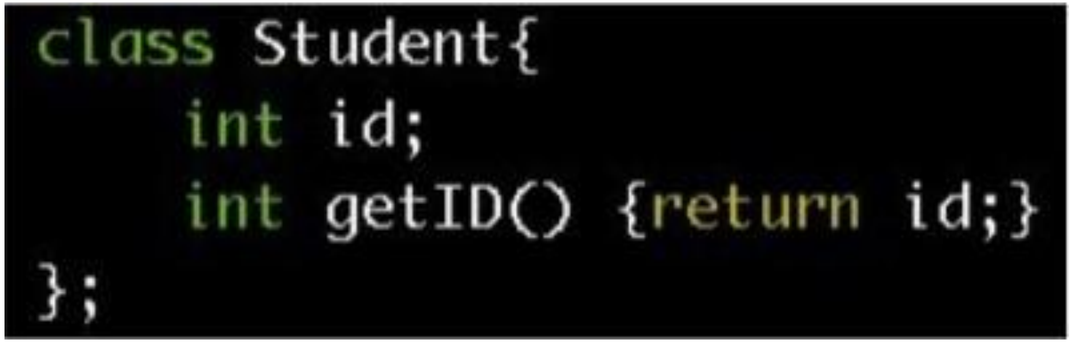
Inline Function



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

inline int mul(int x, int y){
    return (x * y);
}
int main() {
    cout << mul(10, 5) << endl;
    return 0;
}
```

Inline Function

- 如果在class宣告，直接將function定義在class宣告中，compiler會自動把它當 作 inline function。
- 如下，getID()會被當作inline function使用。



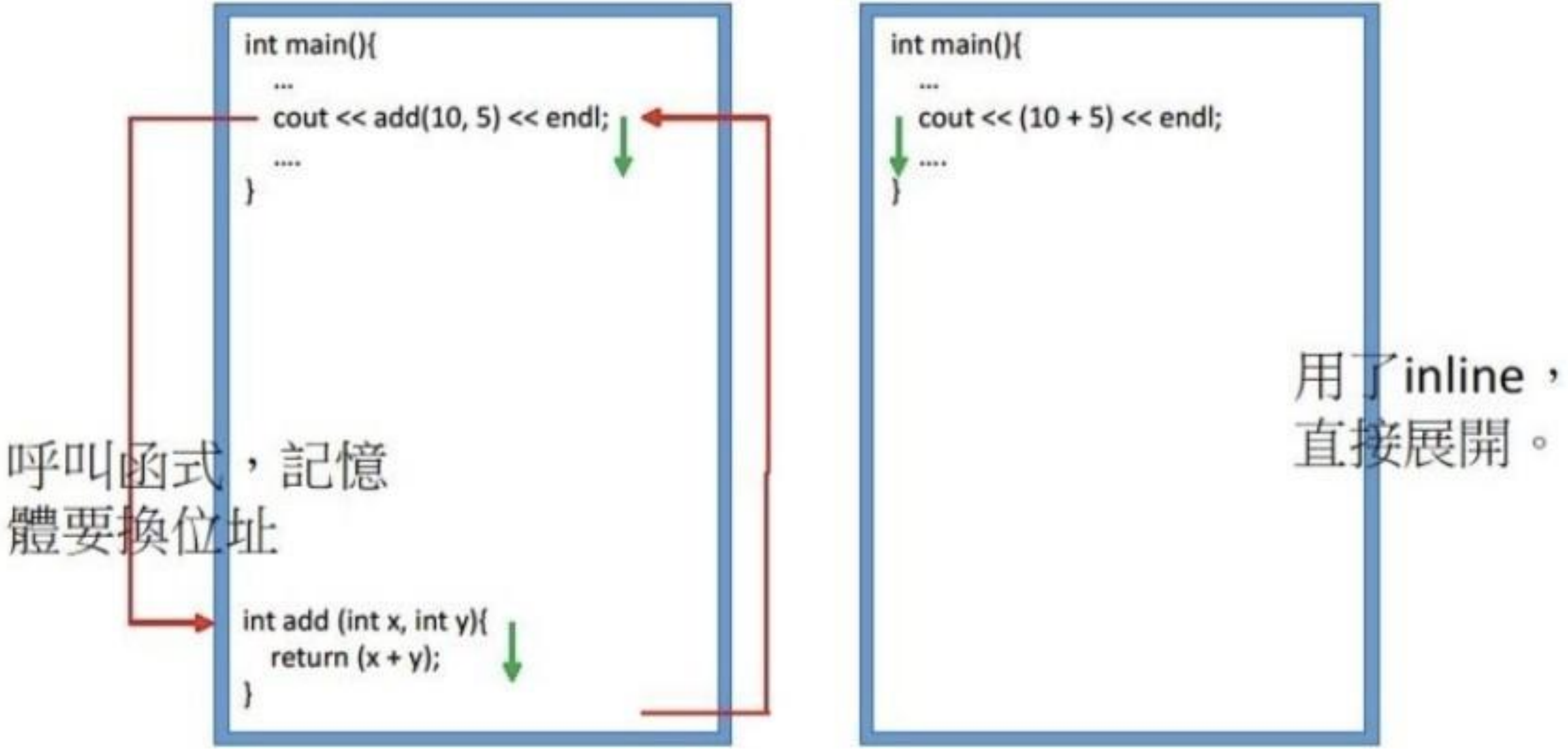
```
class Student{
    int id;
    int getID() {return id;}
};
```

Inline Function

- 一般而言，當我們呼叫函式，電腦會紀錄目前的記憶體位址，然後**跳到函式的記憶體位址** 去執行函式，執行結束後**再跳回**原先的位址。

- 用了inline function，編譯器會直接展開程式碼，就**不需要花費額外時間在函式切換位址** 。

Inline Function v.s. Function



```
int main(){
    ...
    cout << add(10, 5) << endl;
    ....
}

int add (int x, int y){
    return (x + y);
}
```

呼叫函式，記憶體要換位址

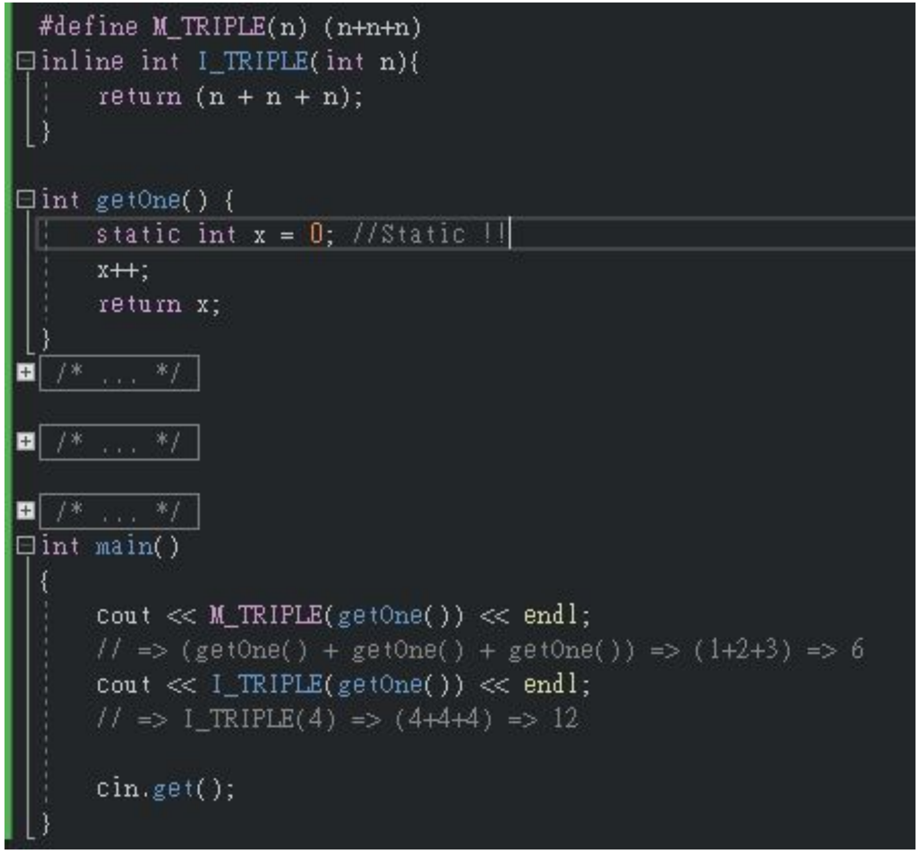
```
int main(){
    ...
    cout << (10 + 5) << endl;
    ....
}
```

用了inline，直接展開。

Inline v.s. Macro

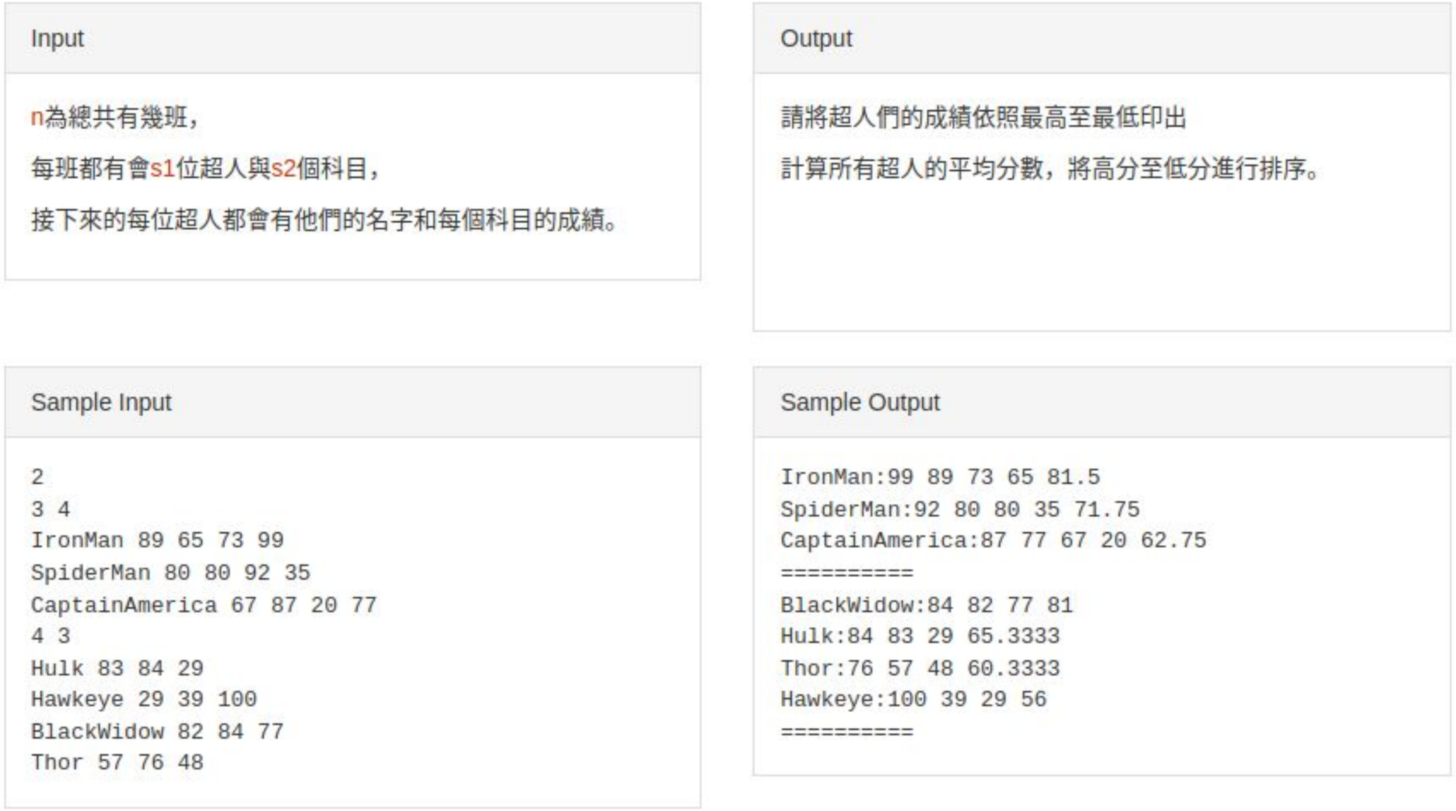
- Macro由**preprocessor**(前置處理器)處理, inline由**compiler**處理。
- Macro**不會檢查傳入參數型別**，而**inline會檢查**。

Inline v.s. Macro



```
#define M_TRIPLE(n) (n+n+n)
inline int I_TRIPLE(int n){
    return (n + n + n);
}

int getOne() {
    static int x = 0; //Static !!
    x++;
    return x;
}
/* ... */

/* ... */

/* ... */
int main()
{
    cout << M_TRIPLE(getOne()) << endl;
    // => (getOne() + getOne() + getOne()) => (1+2+3) => 6
    cout << I_TRIPLE(getOne()) << endl;
    // => I_TRIPLE(4) => (4+4+4) => 12

    cin.get();
}
```

Exercise 5



Input

n為總共有幾班,

每班都有會s1位超人與s2個科目,

接下來的每位超人都會有他們的名字和每個科目的成績。

Output

請將超人們的成績依照最高至最低印出

計算所有超人的平均分數,將高分至低分進行排序。

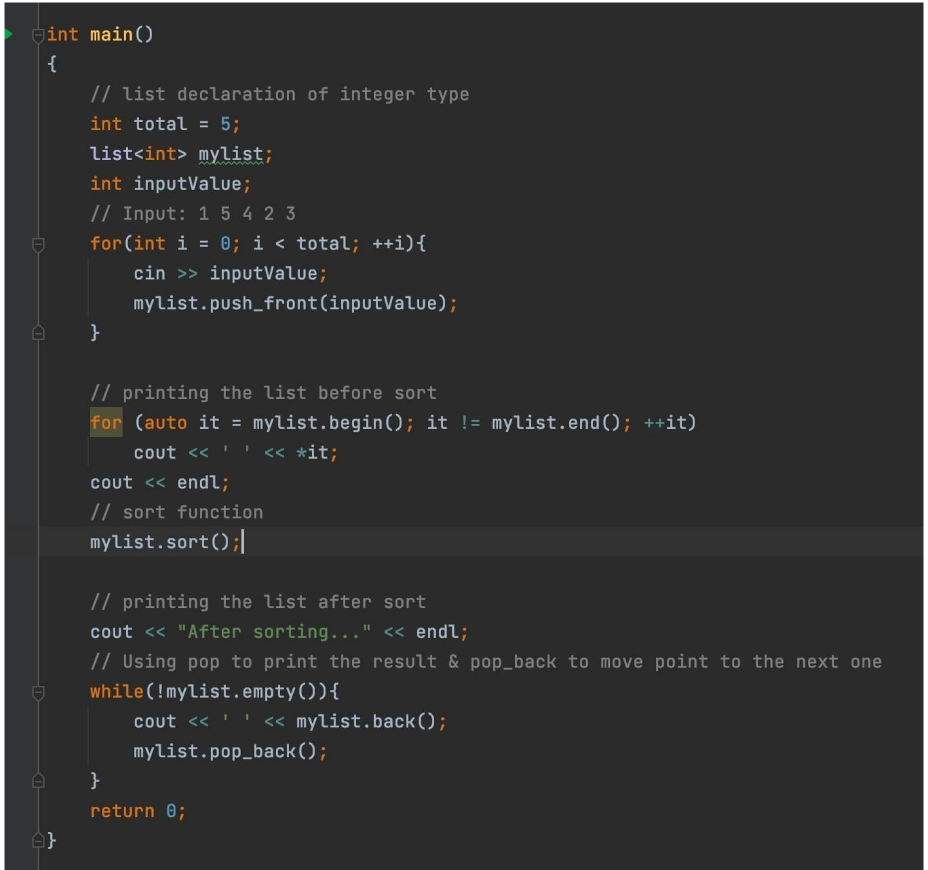
Sample Input

```
2
3 4
IronMan 89 65 73 99
SpiderMan 80 80 92 35
CaptainAmerica 67 87 20 77
4 3
Hulk 83 84 29
Hawkeye 29 39 100
BlackWidow 82 84 77
Thor 57 76 48
```

Sample Output

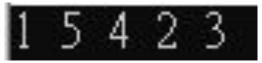
```
IronMan:99 89 73 65 81.5
SpiderMan:92 80 80 35 71.75
CaptainAmerica:87 77 67 20 62.75
==========
BlackWidow:84 82 77 81
Hulk:84 83 29 65.3333
Thor:76 57 48 60.3333
Hawkeye:100 39 29 56
==========
```

std::list



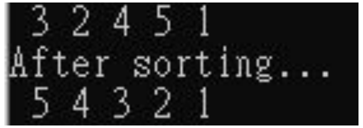
```
int main()
{
    // list declaration of integer type
    int total = 5;
    list<int> mylist;
    int inputValue;
    // Input: 1 5 4 2 3
    for(int i = 0; i < total; ++i){
        cin >> inputValue;
        mylist.push_front(inputValue);
    }

    // printing the list before sort
    for (auto it = mylist.begin(); it != mylist.end(); ++it)
        cout << ' ' << *it;
    cout << endl;
    // sort function
    mylist.sort();

    // printing the list after sort
    cout << "After sorting..." << endl;
    // Using pop to print the result & pop_back to move point to the next one
    while(!mylist.empty()){
        cout << ' ' << mylist.back();
        mylist.pop_back();
    }
    return 0;
}
```

Input :



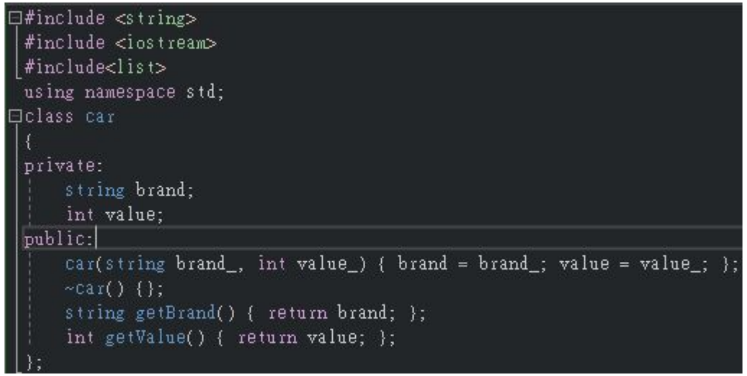
```
1 5 4 2 3
```

Output:

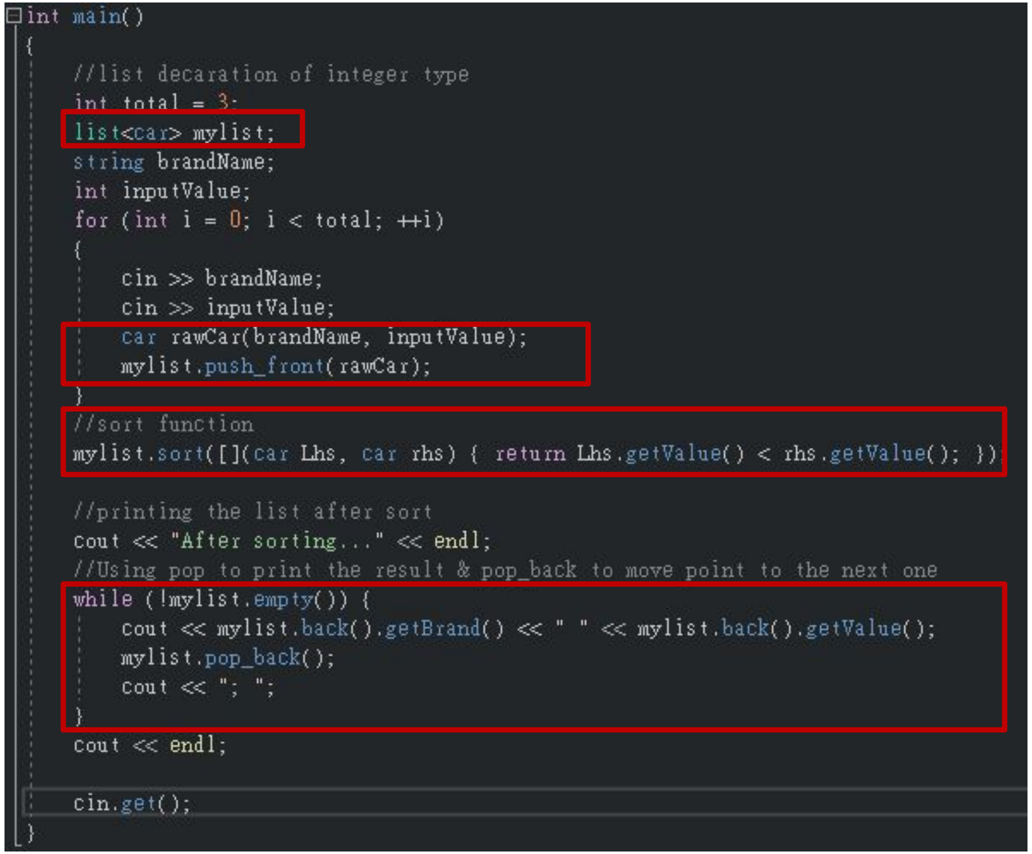


```
 3 2 4 5 1
After sorting...
 5 4 3 2 1
```

std::list

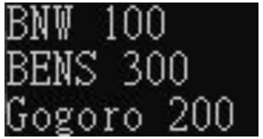


```
#include <string>
#include <iostream>
#include<list>
using namespace std;
class car
{
private:
    string brand;
    int value;
public:
    car(string brand_, int value_) { brand = brand_; value = value_; };
    ~car() {};
    string getBrand() { return brand; };
    int getValue() { return value; };
};
```



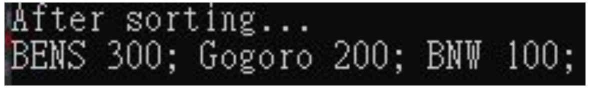
```
int main()
{
    //list decaration of integer type
    int total = 3;
    list<car> mylist;
    string brandName;
    int inputValue;
    for (int i = 0; i < total; ++i)
    {
        cin >> brandName;
        cin >> inputValue;
        car rawCar(brandName, inputValue);
        mylist.push_front(rawCar);
    }
    //sort function
    mylist.sort([](car Lhs, car rhs) { return Lhs.getValue() < rhs.getValue(); });

    //printing the list after sort
    cout << "After sorting..." << endl;
    //Using pop to print the result & pop_back to move point to the next one
    while (!mylist.empty()) {
        cout << mylist.back().getBrand() << " " << mylist.back().getValue();
        mylist.pop_back();
        cout << "; ";
    }
    cout << endl;

    cin.get();
}
```

Input :



```
BNW 100
BENS 300
Gogoro 200
```

Output:



```
After sorting...
BENS 300; Gogoro 200; BNW 100;
```

std::list

- Init: **list<XXX> myList**;
    - XXX could be **int**, **float** …and **class** (like **superman**)
- list.**sort**(), list.**push** (pop)**_back** (front), list.**sort**(_Comp)
- 記得要 **#include <list>**

Assignment 3 配分

項目	配分
有交（含屍體）	20
可以編譯	15
按 1 可以拿牌	10
按 2 放棄這一輪並顯示點數	10
按 3 可以重新開始	10
可以完成一輪完整的遊戲	20
按 4 可以離開並印出學號	10
印出玩家及莊家的名字	5

Any
questions?