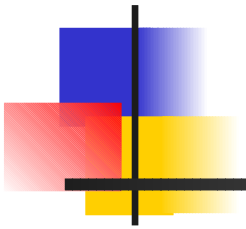
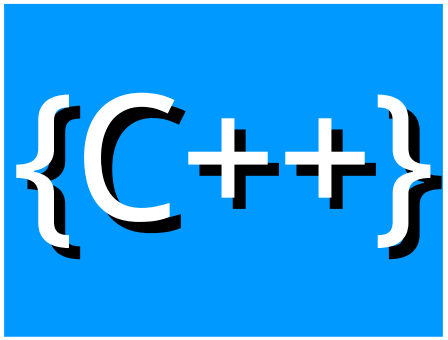
Operator Overloading



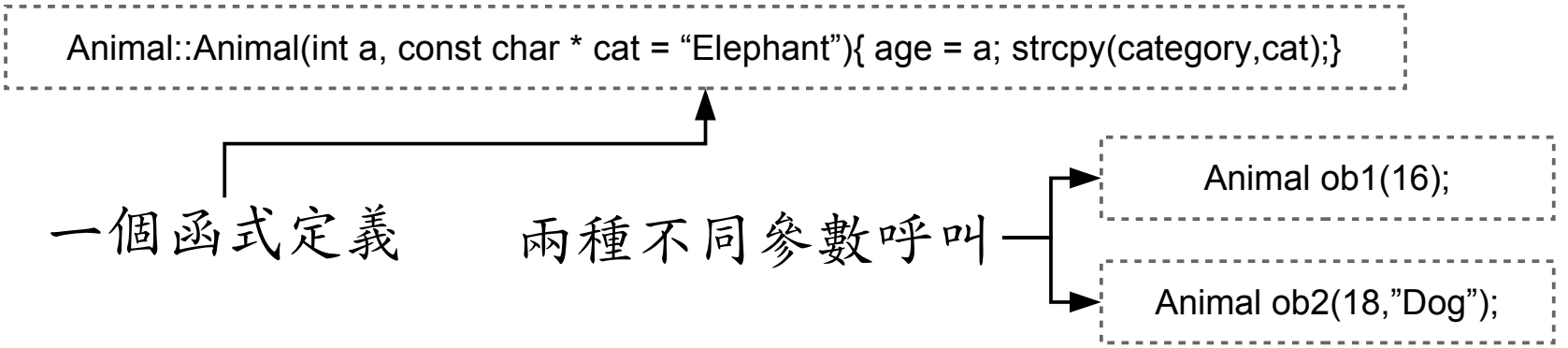
Week 4



Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

函式多載
Function Overloading

- 同樣名稱的函式有多種格式，或說多個函式共用一個函數名稱
- 可以定義兩個有相同名稱的函數，但函式簽名(Signature) 必須不同
  - 參數串列的參數數目、參數型別
- 預設參數是可以使用不同數目的參數呼叫同一個函數



函式多載範例

- 定義一組 print() 函式

```
void print(const char * str, int width);     //#1
void print(double d, int width);             //#2
void print(long l, int width);               //#3
void print(int i, int width);                //#4
void print(const char * str);                //#5
```

- 當使用 print() 函式時，編譯程式從函式原型找一個符合函式呼叫的相同簽名

```
print("Pancakes", 15);
print("Syrup");
print(1999.0,10);
print(1999,12);
print(1999L,15);
```

運算子多載

- 什麼是運算子多載？
  - 讓運算子(符號)有不同的意義
- 運算子的預設意義 (以 + 與 = 爲例)

```
class frac {.....} ;
int main() {
      int x=5, y =3, z ; z = x + y ;   // 使用 = 、 +
      frac f1(3,5), f2(2, 5), f3 ;
      f3 = f1 + f2 ;                   // 3/5 + 2/5 = 1, 可行？
      return 0;
}
```

如果沒有運算子多載能力

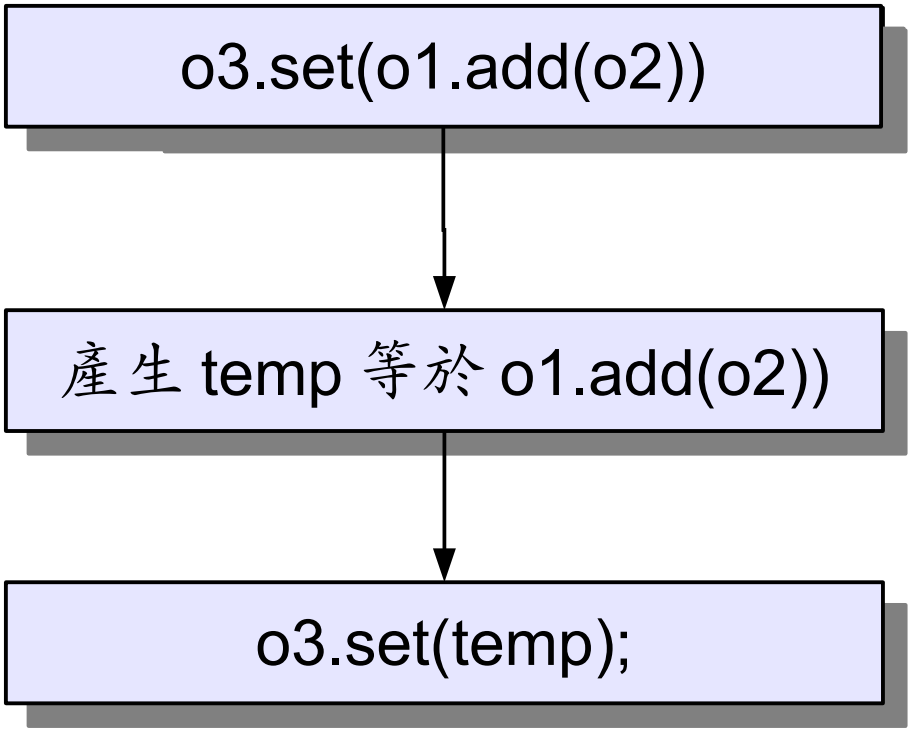
```
int main() {
    frac f1(3,5), f2(2, 5) , f3;
    f3.set(f1.add(f2));                  // 模擬 f3 = f1 + f2 ;
    if (f1.great_equal(f2))              // f1 >= f2
        cout << " f1 >= f2" ;
    cout<<"f3="; f3.print() ;            // cout << "f3="<<f3 ;
    return 0;
}
```

不用多載運算子的作法

```
class Coord {
    int x, y ;
    ......
};
int main() {
    Coord o1(10,10), o2(5,3), o3 ;
    o3.set(o1.add(o2));                    // o3 = o1 + o2 ;
    return 0;
}
```

不用多載運算子的作法

```
class Coord {
      int x, int y;
public:
      Coord(){}
      Coord(int a, int b){x=a; y=b;}
      Coord add(Coord c) {
            Coord temp ;
            temp.x = x + c.x ;
            temp.y = y + c.y ;
            return temp ;
      }
      void set(Coord c) {
            x = c.x; y=c.y;
      }
      void print(){cout<<x<<y<<endl;}
};
```



多載運算子範例

```
class Coord {
    int x, y ;
    ......
};
int main() {
    Coord o1(10,10), o2(5,3), o3 ;
    o3 = o1 + o2;
    return 0;
}
```

多載運算子 + 、 = 的實際運作

```
class Coord {
    int x, y ;
    ......
};
int main() {
    Coord o1(10,10), o2(5,3), o3 ;
    o3.operator=(o1.operator+(o2));      // o3 = o1 + o2 ;
    return 0;
}
```

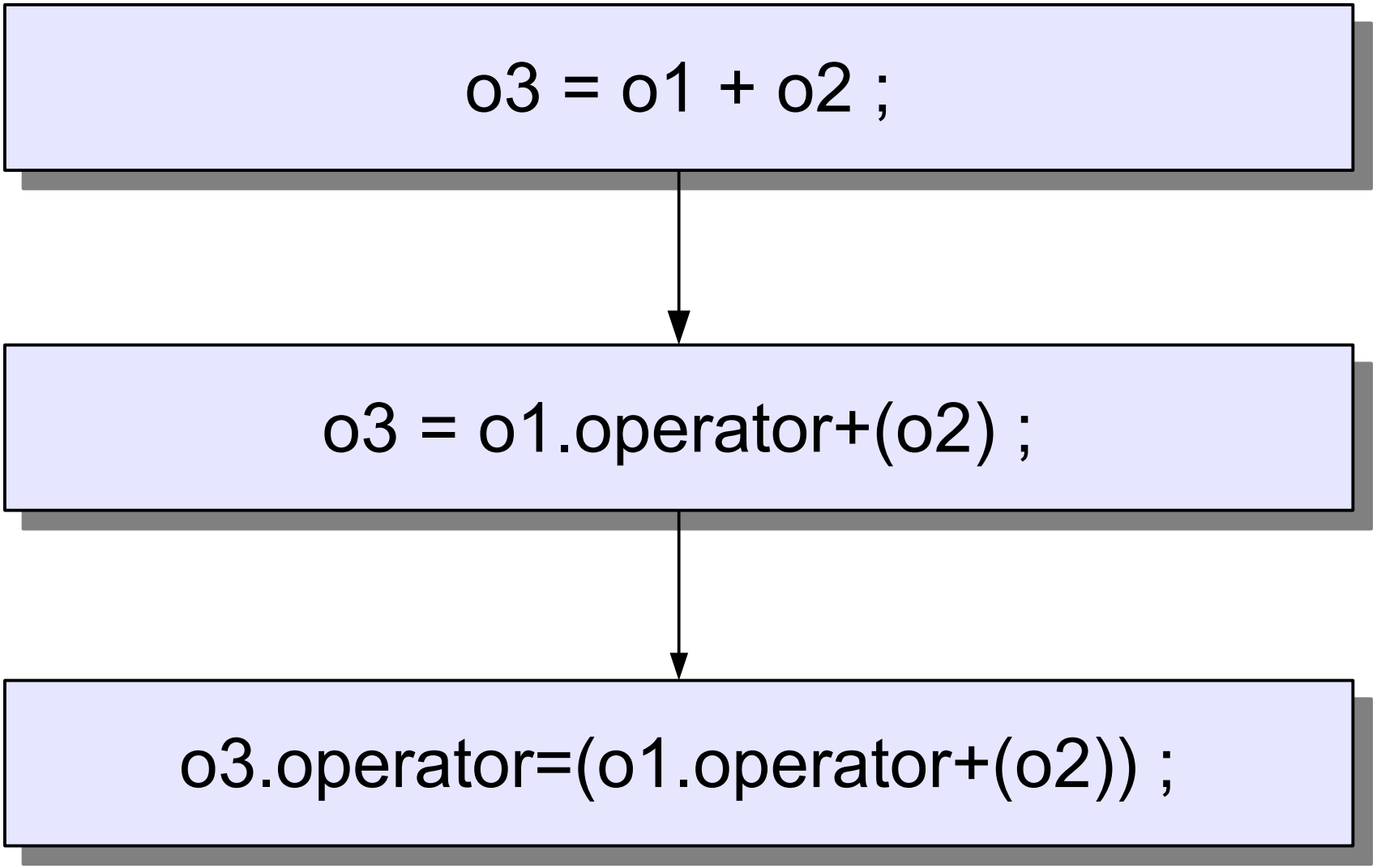
多載運算子 + 、 = 的實作

```
class Coord {
    int x, y ;
    ......
    Coord operator+(Coord c) {        // 原 add()
        Coord temp ;
        temp.x = x + c.x ;
        temp.y = y + c.y ;
        return temp ;
    }
    void operator=(Coord c) {         // 原 set()
        x = c.x; y=c.y;
    }
};
```

多載運算子 + 、 =

```
class Coord {
    int x, y ;
    ......
    Coord operator+(Coord c);
    void operator=(Coord c);
};
int main() {
    Coord o1(10,10), o2(5,3), o3 ;
    o3 = o1 + o2 ;
    return 0;
}
```

多載運算子的眞實面貌



如何多載運算子

- 在類別中建立運算子函式 (operator function)
  - 語法

```
class 類別名稱 {
    ......
    // overload 運算子 X
    回傳型態 operatorX( 參數列表 ){ ......}
};
```

多載 << 與 >> 運算子

- << 與 >> 運算子
  - 可供內建資料型別使用
  - 可供使用者自訂類別多載使用
- 多載 << 運算子
  - 左邊的運算元必須是型別 ostream&
  - 例如 : cout<< ob1
- 多載 >> 運算子
  - 左邊的運算元必須是型別 istream&
  - 例如 : cin >> ob1
- 多載這兩個運算子必須爲非成員函式，並宣告爲類別的 friend

friend 函式、 friend 類別

- 在定義類別成員時，私有成員只能被同一個類別定義的成員存取，不可以直接由外界進行存取
- 然而有些時候，希望提供私用成員給某些外部函式來存取，這時可以設定類別的「好友」，只有好友才可以直接存取自家的私用成員。

operator<<() 範例

```
#include <iostream>
using namespace std;
class Point {
    friend ostream& operator<<(ostream &out, const Point &outPoint);
    friend istream& operator>>(istream &in, Point &outPoint);

    private:
        int coordX,coordY;
    public:
        Point(int x, int y):coordX(x),coordY(y){}
        Point operator+(Point &rightPoint);
        Point operator-(Point &rightPoint);
};
```

```
ostream& operator<<(ostream &out, const Point &outPoint)
{
    out << "(" << outPoint.coordX << "," << outPoint.coordY << ")" << endl;
    return out;
}
```

operator>>() 範例

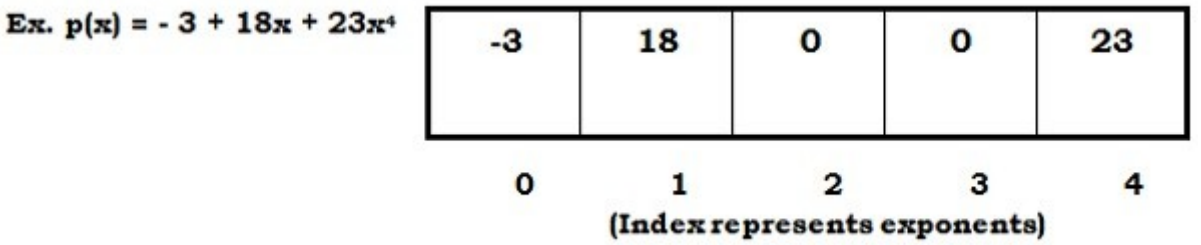
```
#include <iostream>
using namespace std;
class Point {
    friend ostream& operator<<(ostream &out, const Point &outPoint);
    friend istream& operator>>(istream &in, Point &outPoint);

    private:
        int coordX,coordY;
    public:
        Point(int x, int y):coordX(x),coordY(y){}
        Point operator+(Point &rightPoint);
        Point operator-(Point &rightPoint);
};
```

```
 istream& operator>>(istream &in, Point &outPoint)
{
    in >> outPoint.coordX >> outPoint.coordY;
    return in;
}
```

Assignment 4

- 開發 class Polynomial
  - 用 integer array 表達一個多項式
  - 當某個項次的係數爲 0，同樣佔有一個 array 的元素空間



  - 建構子需要參數來初始化多項式的項數
    - 也就是需要在建構子中使用多項式的項數來動態配置 integer array 的大小

```
Polynomial A5(5);  // 代表宣告一個具有 5 個項數的多項式
Polynomial::Polynomial(int expNum){
        terms=new int[expNum]();
}
```

Assignment 4

- 必須完成以下運算子的多載，要能夠處理項數不同時的多項式加法與減法
  - operator+()
  - operator-()
  - operator<<()
  - operator>>()

Assignment 4

- 可以使用 operator>> 輸入一個多項式

```
[ 程式碼 ]
cout<<"Input Polynomial"<<endl;
cin >> polyA;

[ 執行 ]

Input Polynomial:
-13 18 0 0 23
```

- 可以用 operator<< 輸出一個多項式

```
[ 程式碼 ]
cout<<"Output Polynomial"<<endl;
cout << poly_A<<endl;

[ 執行 ]

Output Polynomial:
-13 * X^0 + 18 * X^1 + 0 * X^2 + 0 * X^3 + 23 * X^4
```

Assignment 4

- 主程式

```
// Test Class Polynomial
#include "Polynomial.h"

int main(int argc, const char *argv[])
{
    Polynomial p(3);
    Polynomial q(3);
    Polynomial m(3);

    cout << "Please input all coefficients of polynomial p" << endl;
    cin >> p;
    cout << "Please input all coefficients of  polynomial q" << endl;
    cin >> q;

    cout << p + q;
    cout << p - q;

    m = p + q;
    cout << m;

    Polynomial n(p+q);
    cout << n;

    return 0;
}
```