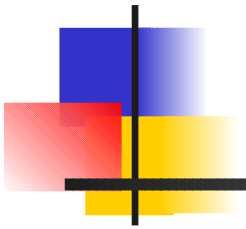
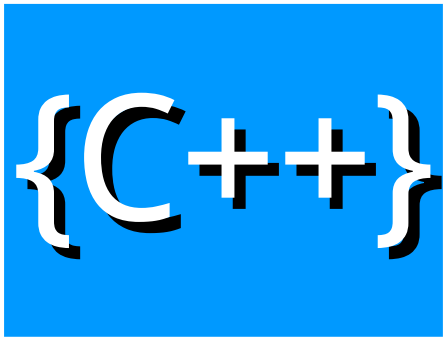
Operator Overloading



Week 5



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

Assignment 5

- 以下爲 Class Matrix 的定義

```
class Matrix
{
public:
   Matrix();

   double& operator()(int i, int j);
   double operator()(int i, int j) const;
   Matrix& operator+=(const Matrix& right);

   static const int ROWS = 3;
   static const int COLUMNS = 3;
private:
   double elements[ROWS * COLUMNS];
};
```

Assignment 5

- 完成 Class Matrix 的成員函式

```
Matrix& Matrix::operator+=(const Matrix& right);
```

- 完成以下運算子的多載

```
Matrix operator*(const Matrix& left, const Matrix& right);
Matrix operator*(const Matrix& left, double right);
Matrix operator*(double left, const Matrix& right);
ostream& operator<<(ostream& left, const Matrix& right);
```

Matrix::operator()

- 這兩個 operator() 有何不同？

```
double& Matrix::operator()(int i, int j);

double Matrix::operator()(int i, int j) const;
```

- 分別對應不同的操作

給值

```
double& Matrix::operator()(int i, int j);
```

```
Matrix m;

m(0,0)=m(1,1)=m(2,2)=1;
```

相當於

```
&(m->elements+0*COLUMNS+0)=
&(m->elements+1*COLUMNS+1)=
&(m->elements+2*COLUMNS+2)=1
```

取值

```
double Matrix::operator()(int i, int j) const;
```

```
Matrix m;

m(0,0)=m(1,1)=m(2,2)=1;

cout << m(0,0);
```

相當於

```
cout << m->elements[0*COLUMNS+0]
```

operator+() 與 operator*()

- 注意這兩個運算子都不是 Class Matrix 的成員函式
- 需要使用 Class Matrix 的成員函式來完成操作
- 例如

```
Matrix operator+(const Matrix& left, const Matrix& right);
```

使用 `Matrix& Matrix::operator+=(const Matrix& right);` 來完成

```
Matrix operator*(const Matrix& left, const Matrix& right);
```

取得 matrix 的某個元素值需要使用

```
double Matrix::operator()(int i, int j) const;
```