複合變數型別

```
#include <iostream>
using namespace std;
struct complex
{
      double r; double i;
};
complex add(complex a, complex b)
{
      complex c;
      c.r = a.r + b.r; c.i = a.i + b.i;
      return c;
}
complex sub(complex a, complex b)
{
      complex c;
      c.r = a.r – b.r; c.i = a.i - b.i;
      return c;
}
complex mul(complex a, complex b)
{
      complex c;
      c.r = a.r * b.r - a.i * b.i;
      c.i = a.r * b.i + a.i * b.r;
      return c;
}
```

```
complex operator+(complex a, complex b)
```

```
complex operator-(complex a, complex b)
```

```
complex operator*(complex a, complex b)
```

operator overload

```
int main()
{
      complex x = {0.8, 1.5};
      complex y = {3.9, 2.7};
      x = sub(mul(x, y), x);
      cout << "x = " << x.r
          << (x.i > 0 ? "+" : "") << x.i << "i\n";
      return 0;
}
```

```
x = x * y - x;
```

具體的資料型別

```
#include <iostream>
using namespace std;
class complex
{
        double r;
        double i;
public:
        complex() {}
        complex(double rr, double ii) :
                r(rr), i(ii)
        {
                i = ii;
        }
        complex operator+(complex b)
        {
                return complex(r + b.r, i + b.i);
        }
        complex operator-(complex b)
        {
                return complex(r - b.r, i – b.i);
        }
        complex operator*(complex b)
        {
                return complex(r * b.r - i * b.i,
                        r * b.i + i * b.r);
        }
};
```

default constructor

- 簡化主程式
- 增加程式可讀性
- 易於驗證測試
- 建立可重用程式碼

```
ostream & operator<<(ostream & s, complex a)
{
        return s << a.r << (a.i > 0 ? "+" : "")
                << a.i << "i";
}

int main()
{
        complex x(0.8, 1.5), y;
        y = x * complex(3.9, 2.7) - x;
        cout << "y = " << y << '\n';
        return 0;
}
```

標頭檔及程式庫

檔案:　cmp.hh

```
// cmp.hh: Complex Variables
#ifndef CMP_HH
#define CMP_HH
#include <iostream>
class Complex
{
        double r;
        double i;
public:
        Complex() {}
        Complex(double rp, double ip);
        Complex(double rp);
        // operators
        Complex operator+(Complex b) const;
        Complex operator-(Complex b) const;
        Complex operator*(Complex b) const;
        Complex operator/(Complex b) const;
        // utilities
        void stream_output(std::ostream & st) const;
};
// overload, inline functions
inline Complex operator+(double a, Complex b) {return b + a;}
inline Complex operator-(double a, Complex b) {return Complex(a) - b;}
inline Complex operator*(double a, Complex b) {return b * a;}
inline Complex operator/(double a, Complex b) {return Complex(a) / b;}
inline std::ostream & operator<<(std::ostream & st, const Complex & c)
{
        c.stream_output(st);
        return st;
}
#endif
```

- 僅宣告，不定義 (inline 除外)
- 獨立檔案
- 可重複含入 (#include)

成員定義 型別使用

```
// cmp.cc: Complex Variables, definition
#include "cmp.hh"

Complex::Complex(double rp, double ip) :
        r(rp), i(ip)
{}

Complex::Complex(double rp) :
        r(rp), i(0)
{}

Complex Complex::operator+(Complex b) const
{
        return Complex(r + b.r, i + b.i);
}

        … snipped …

void Complex::stream_output(std::ostream & stream) const
{
        stream << r;
        if (i > 0) stream << '+' << i << 'I';
        else if (i < 0) stream << i << 'I';
}
```

```
// cmp_test.cc
#include "cmp.hh"
using namespace std;
int main()
{
        Complex a(1.6, 2.4);
        Complex b = Complex(3.1, -1.6) / a;
        Complex c = b + 3;
        cout << "a=" << a << '\n';
        cout << "b=" << b << '\n';
        cout << "c=" << c << '\n';
        cout << "(a*c)/(1-b)=" << (a * c) / (1 - b) << '\n';
}
```

```
cp1@area:~$ g++ cmp_test.cc cmp.cc -o cmp_test
cp1@area:~$ ./cmp_test
a=1.6+2.4I
b=0.134615-1.20192I
c=3.13462-1.20192I
(a*c)/(1-b)=6.18521-2.11945I
cp1@area:~$
```

程式編譯和連結

程式的編譯

```
$ c++ -c cmp.cc -I. -Wall
$ c++ -c cmp_test.cc -I. -Wall
```

得到物件檔: `cmp.o` 和 `cmp_test.o`

程式的連結

```
$ c++ cmp_test.o cmp.o -o cmp_test
```

得到執行檔: `cmp_test`

實例：矩陣類型

```
// mat.hh
#ifndef MAT_HH
#define MAT_HH
#include <iostream>
class Matrix
{
        size_t n_row;
        size_t n_col;
        double * vals;
public:
        Matrix(size_t nr, size_t nc);
        Matrix(const Matrix & m); // copy constructor
        Matrix(size_t nr, size_t nc, double const * v);
        ~Matrix(); // destructor
        double & elem(size_t ri, size_t ci); // element access
        // overloading operators
        Matrix operator+(const Matrix & m) const;
        Matrix operator-(const Matrix & m) const;
        Matrix operator*(const Matrix & m) const;
        Matrix operator*(double v) const;
        // making unit Matrix
        static Matrix unit(size_t sz);
        // output
        void stream_out(std::ostream & o) const;
};
// external overloads
Matrix operator*(double v, const Matrix & m);
std::ostream & operator<<(std::ostream & o, const Matrix & m);
#endif
```

constructors

```
Matrix::Matrix(size_t nr, size_t nc) :
        n_row(nr), n_col(nc)
{vals = new double[n_row * n_col];}
```

```
Matrix::Matrix(const Matrix & m) :
        n_row(m.n_row), n_col(m.n_col)
{
        vals = new double[n_row * n_col];
        for (size_t i = n_row * n_col; i --;) vals[i] = m.vals[i];
}
```

destructor

```
Matrix::~Matrix()
{delete [] vals;}
```

```
double & Matrix::elem(size_t ri, size_t ci)
{return vals[ri * n_col + ci];}
```

矩陣應用

```cpp
// mat_exp.cc
#include <mat.hh>
using namespace std;
int main()
{
    double v1[] = {
        1,2,3,4,5,
        5,4,3,2,1,
        0,1,3,1,5,
        9,8,7,6,4,
        7,1,8,2,6};
    double v2[] = {
        1,0,3,0,0,
        5,2,3,0,1,
        0,1,3,1,1,
        0,0,7,4,1,
        1,2,3,9,5};
    Matrix m1(5, 5, v1);
    Matrix m2(5, 5, v2);
    Matrix mm = m1 * m2 - 10 * Matrix::unit(5);
    cout << mm;
    return 0;
}
```

```
cp1@area:~/hw5$ ls
mat.cc  mat_exp.cc  mat.hh
cp1@area:~/hw5$ c++ -c mat.cc -I. -Wall
cp1@area:~/hw5$ c++ -c mat_exp.cc -I. -Wall
cp1@area:~/hw5$ c++ mat_exp.o mat.o -o mat_exp
cp1@area:~/hw5$ ./mat_exp
6       17      61      64      34
26      3       53      20      14
10      15      24      52      30
53      31      126     57      41
18      22      80      70      31
cp1@area:~/hw5$
```

標準模式庫

container classes

```
#include <vector>
#include <iostream>
using namespace std;
int main()
{
        double l[] = {1.2, 3.4, 5.78, 2.22};
        vector<double> a(l, l + sizeof(l) / sizeof(double));
        a.push_back(4.1);
        a.erase(a.begin());
        for (vector<double>::iterator i = a.begin(); i != a.end(); i ++) {
                cout << ' ' << * i;
        }
        cout << '\n';
        a.resize(8);
        for (int i = 0; i < a.size(); i ++) {
                cout << ' ' << a[i];
        }
        cout << '\n';
        return 0;
}
```

vector： 會自己長的陣列
list： 可隨意插取的串列
queue： 先進先出的佇列
map： 關聯映射
set： 沒有重複的集合

微分到差分

牛頓運動定律

$$f = ma = m\frac{dv}{dt}$$

$$v = \frac{dx}{dt}$$

初始條件： $x_{0,} v_0$

1.給定初條件
2.計算加速度
3.計算新的速度及位置
4.至 2 重覆

差分方程

$$v_i \approx \frac{x_{i+1} - x_i}{t_{i+1} - t_i} = \frac{x_{i+1} - x_i}{\tau}$$

$$a_i \approx \frac{v_{i+1} - v_i}{v_{i+1} - v_i} = \frac{v_{i+1} - v_i}{\tau}$$

$$x_{i+1} = x_i + v_i \tau$$

$$v_{i+1} = v_i + a_i \tau$$

萬有引力

$$f = -\frac{x}{|x|^3} GMm$$

本週作業

1. 完成矩陣的實作：建立 mat.cc 檔，配合 mat.hh 及 mat_exp.cc （存在 CP SSH 伺服器的 /usr/local/src/hw5 目錄中）編譯及連結出可執行檔。（輸出結果如前）
2. 行星軌道：給定下列方程的初始條件

$$\frac{d^2 x}{dt^2} = \frac{x}{\left(x^2 + y^2\right)^{3/2}} \qquad \frac{d^2 y}{dt^2} = \frac{y}{\left(x^2 + y^2\right)^{3/2}}$$

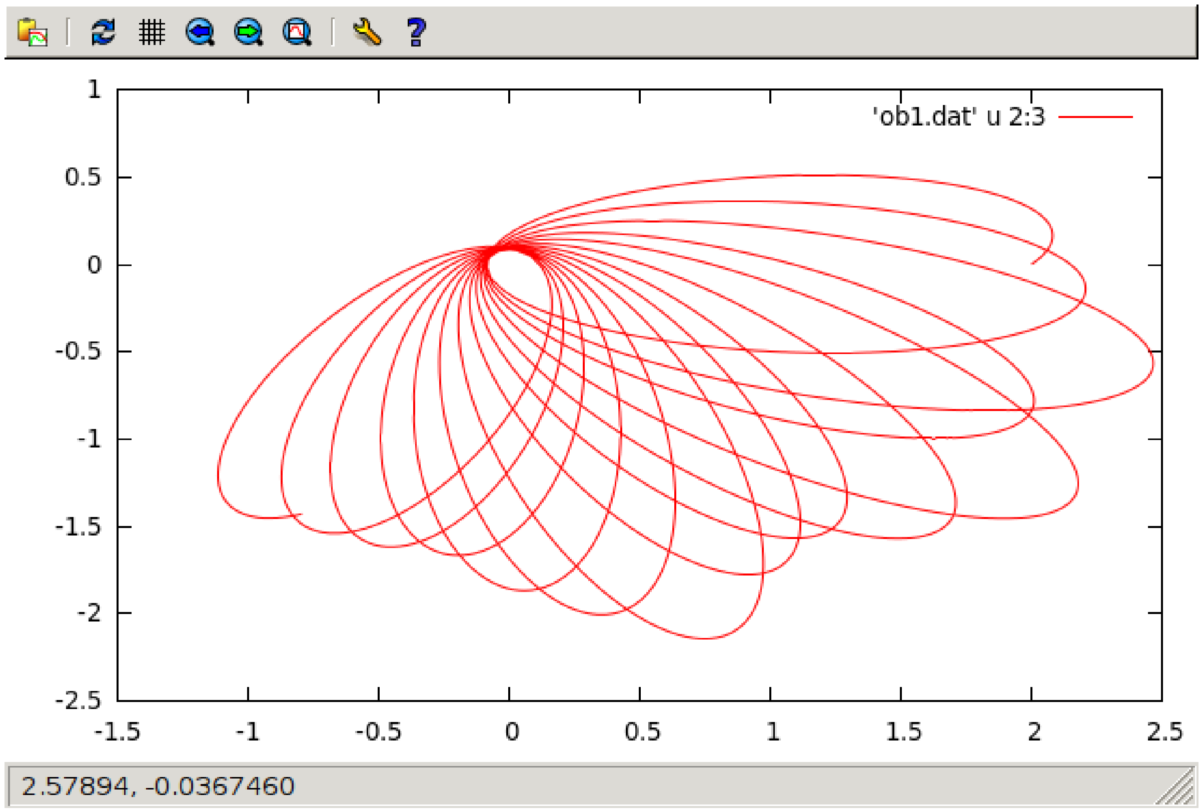
以及差分時間 τ，寫一程式計算 x 及 y 在 $0 < t < T$ 間的軌跡。將 $x(0) = 2$, $y(0) = 0$, $v_x(0) = v_y(0) = 0.2$ 及 $\tau = 0.02$, 0.01, 0.001; $T = 100$ 的運動軌跡作成圖形檔。

程式執行範例

```
cp1@area:~$ hw5-orbit
# Input: time_step x0 y0 vx0 vy0 T
0.02 2 0 0.2 0.2 0.1
# Got: 0.02 2 0 0.2 0.2 0.1
0.02     2.0039      0.004
0.04     2.0077      0.0079998
0.06     2.0114      0.0119992
0.08     2.015       0.015998
0.1      2.01851     0.0199961
cp1@area:~$
```

```
cp1@area:~$ echo 0.02 2 0 0.2 0.2 100 | hw5-orbit > ob1.dat
cp1@area:~$ gnuplot

        G N U P L O T
        Version 4.4 patchlevel 0
        last modified March 2010
        System: Linux 2.6.35-27-generic

        Copyright (C) 1986-1993, 1998, 2004, 2007-2010
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help seeking-assistance"
        immediate help:   type "help"
        plot window:      hit 'h'

Terminal type set to 'wxt'
gnuplot> set st da lines
gnuplot> plot 'ob1.dat' u 2:3
gnuplot>
```

圖型輸出



2.57894, -0.0367460