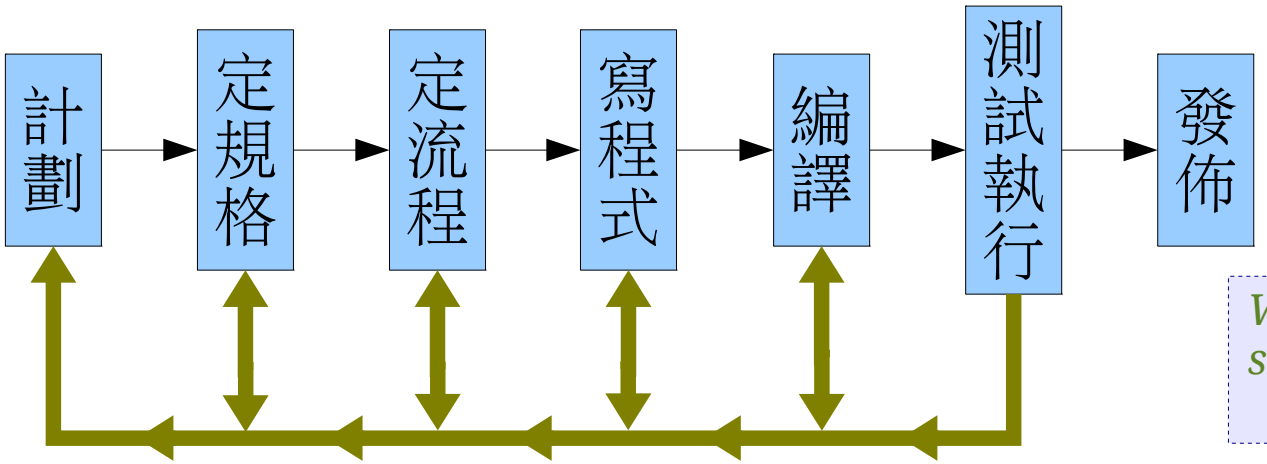
程式防／除錯的技巧

程式寫作

- 預先思考計劃
- 留下筆記
- 有結構的程式碼
- 明白的註記程式

除錯

- 編輯器的語法標示
- 編譯器的警告及錯誤訊息
- 置入的除錯輸出
- 正確條件的宣張



Weeks of programming can save hours of planning.
— Anonymous

```
#include "mat.hh"
#include "ran_nr.hh"
#include <iomanip>
using namespace std;
int main()
{
        double v[16];
        RanNR rng(7);
        for (size_t i = 0; i < 16; i ++) v[i] = 2 * rng.
        Matrix m(4, 4, v);
        cout << "det(\n" << m << ") = " << m.det() << '\
        Matrix c(m.inverse());
        cout << "inverse is " << c << '\n';
        return 0;
}
~
~
~
~
~
~
~
~
"mtest.cc" 15L, 333C                                  1,1           All
```

g++ 的錯誤訊息

```
cp1@area:~/mat$ g++ -I.  -L.  mkmat.cc  -lmat -o mkmat
mkmat.cc: In function ‘int main()’:
mkmat.cc:10: error: ‘sizxe_t’ was not declared in this scope
mkmat.cc:10: error: expected ‘;’ before ‘i’
mkmat.cc:10: error: ‘i’ was not declared in this scope
make: *** [mkmat] Error 1
cp1@area:~/mat$

cp1@area:~/mat$ g++ -I.  -L.  mkmat.cc  -lmat -o mkmat
mat.hh: In function ‘int main()’:
mat.hh:9: error: ‘double* Matrix::vals’ is private
mkmat.cc:12: error: within this context
make: *** [mkmat] Error 1
cp1@area:~/mat$
```

```
  GNU nano 2.2.4               File: mtest.cc                    Modified

#include "mat.hh"
#include "ran_nr.hh"
#include <iomanip>
using namespace std;
int main()
{
        double v[16];
        RanNR rng(7);
        for (size_t i = 0; i < 16; i ++) v[i] = 2 * rng.unifo
        Matrix m(4, 4, v);
        cout << "det(\n" << m << ") = " << m.det() << '\n';
        Matrix c(m.inverse());
        cout << "inverse is " << c << '\n';
        return 0;
}



                        [ Read 15 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Te
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut
```

宣張正確條件

```
double Matrix::det() const
{
        assert(n_row == n_col);
        if (n_row == 1) return vals[0];
        double det = 0;
        int sign = 1;
        for (size_t r = 0; r < n_row; r ++) {
                // Laplace expansion
                det += sign * elem(r, 0) * submat(r, 0).det();
                std::cerr << "in det(), *this = " << * this << '\n';
                sign = - sign;
        }
        return det;
}
```

用 std::cerr 來輸出除錯訊息

編譯程式加入“-g”選項

GNU 除錯器 (gdb)

```
cp1@area:~$ g++ -g bug.cc
cp1@area:~$ gdb ./a.out
[ snipped ]
Reading symbols from /home/cp1/a.out...done.
(gdb) break main
Breakpoint 1 at 0x8048718: file bug.cc, line 12.
(gdb) run
Starting program: /home/cp1/a.out

Breakpoint 1, main () at bug.cc:12
12          cout << "input a number: ";
(gdb) next
13          cin >> n;
(gdb)
input a number: 10
14          while (n != 1) {
(gdb)
15                  cout << '\t' << n;
(gdb)
16                  n = trans(n);
(gdb)
14          while (n != 1) {
(gdb)
15                  cout << '\t' << n;
(gdb)
16                  n = trans(n);
(gdb)
14          while (n != 1) {
(gdb)
15                  cout << '\t' << n;
(gdb)
16                  n = trans(n);
(gdb)
14          while (n != 1) {
(gdb) print n
$1 = 8
(gdb) continue
Continuing.
      10    5     16    8     4     21

Program exited normally.
(gdb) q
cp1@area:~$
```

空行會重複上個指令

可設定中止點:

```
(gdb) break trans
(gdb) break bug.cc:15
```

可以監看變數值

```
(gdb) print n
(gdb) display n
```

bug.cc

```
#include <iostream>
int trans(int n)
{
        if (n % 2) return n * 3 + 1;
        return n / 2;
}
int main()
{
        using namespace std;
        int n;
        cout << "input a number: ";
        cin >> n;
        while (n != 1) {
                cout << '\t' << n;
                n = trans(n);
        }
        cout << "1\n";
        return 0;
}
```

run： 開始執行
next： 執行下一行（不進入函式）
step： 執行下一指令（會進入函式）
continue： 繼續執行

```
(gdb) continue
Continuing.

Breakpoint 2, trans (n=10) at bug.cc:4
4           if (n % 2) return n * 3 + 1;
(gdb) backtrace
#0  trans (n=10) at bug.cc:4
#1  0x08048772 in main () at bug.cc:16
(gdb)
```

回遡呼叫的軌跡

更完整的矩陣程式庫

```
// mat.hh
#ifndef MAT_HH
#define MAT_HH
#include <iostream>
class Matrix
{
        size_t n_row;
        size_t n_col;
        double * vals;
public:
        Matrix(); // default constructor
        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
        double elem(size_t ri, size_t ci) const; // retrieval
        // 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;
        Matrix & operator=(const Matrix & m);
        // making unit Matrix
        static Matrix unit(size_t sz);
        // input/output
        void stream_in(std::istream & i);
        void stream_out(std::ostream & o) const;
        // additional operation
        Matrix submat(size_t r, size_t c) const;
        double det() const;
        Matrix transpose() const;
        Matrix inverse() const;
};
// external overloads
Matrix operator*(double v, const Matrix & m);
std::istream & operator>>(std::istream & s, Matrix & m);
std::ostream & operator<<(std::ostream & o, const Matrix & m);
#endif
```

invert.cc

```
#include "mat.hh"
using namespace std;
int main()
{
        Matrix m;
        cin >> m;
        cout << m.inverse() << '\n';
        return 0;
}
```

mtest.cc

```
#include "mat.hh"
#include "ran_nr.hh"
#include <iomanip>
using namespace std;
int main()
{
        double v[16];
        RanNR rng(7);
        for (size_t i = 0; i < 16; i ++) v[i] = 2 * rng.uniform() - 1;
        Matrix m(4, 4, v);
        cout << "det(\n" << m << ") = " << m.det() << '\n';
        Matrix c(m.inverse());
        cout << "inverse is " << c << '\n';
        return 0;
}
```

行列式及反矩陣

矩陣的行列式

$$\det(A)=\sum_{\sigma\in S_{[A]}}\operatorname{sgn}(\sigma)\prod_{i=1}^{[A]}A_{i,\sigma_i}$$

$[A]$： 矩陣 A 的線性大小

S_n： 數字 1 到 n 的所有排列

$\operatorname{sgn}(\sigma)$： 排列 σ 的交換奇偶號

Laplace 展開式

$$\det(A)=\prod_{j=1}^{[A]}A_{i,j}(-1)^{i+j}M_{i,j}=\prod_{j=1}^{[A]}A_{i,j}C_{i,j}$$

$M_{i,j}$： 矩陣 A 的 (i,j) 餘子式 (minor)

伴隨 (adjugate) 矩陣

$$\operatorname{adj}(A)=C^T \qquad A\operatorname{adj}(A)=\operatorname{adj}(A)A=\det(A)$$

如 A^{-1} 存在： $\operatorname{adj}(A)=\det(A)\,A^{-1}$

行列式及反矩陣實作

遞迴行列式

```
double Matrix::det() const
{
	assert(n_row == n_col);
	if (n_row == 1) return vals[0];
	double det = 0;
	int sign = 1;
	for (size_t r = 0; r < n_row; r ++) {
		// Laplace expansion
		det += sign * elem(r, 0) * submat(r, 0).det();
		sign = - sign;
	}
	return det;
}
```

餘子矩陣

```
Matrix Matrix::submat(size_t r, size_t c) const
{
	Matrix m(n_row - 1, n_col - 1);
	for (size_t s = 1; s < n_row; s ++) for (size_t d = 1; d < n_col; d ++) {
		m.elem(s - 1, d - 1) = elem(s <= r ? s - 1 : s, d <= c ? d - 1 : d);
	}
	return m;
}
```

轉置矩陣

```
Matrix Matrix::transpose() const
{
	Matrix m(n_col, n_row);
	for (size_t r = 0; r < n_row; r ++) for (size_t c = 0; c < n_col; c ++) {
		m.elem(c, r) = elem(r, c);
	}
	return m;
}
```

伴隨⟶反矩陣

```
Matrix Matrix::inverse() const
{
	double d = det();
	assert(d);
	Matrix m(n_row, n_col);
	for (size_t r = 0; r < n_row; r ++) for (size_t c = 0; c < n_col; c ++) {
		int sign = (r + c) % 2 ? - 1 : 1;
		m.elem(r, c) = sign * submat(r, c).det() / d;
	}
	return m.transpose();
}
```

```
// mat.cc
# include < mat.hh>
# include < iomanip>
# include < cassert>
Matrix::Matrix() : n_row(0), n_col(0), vals(0) {}
Matrix::Matrix(size_t nr, size_t nc) :
        n_row(nr), n_col(nc)
{
        if (n_row * n_col) vals =  new double[n_row * n_col];
        else vals =  0;
}
Matrix::Matrix(const Matrix & m) : // copy constructor
        n_row(m.n_row), n_col(m.n_col)
{
        if (m.vals) {
                assert(n_row * n_col);
                vals =  new double[n_row * n_col];
                for (size_t i =  n_row * n_col; i --;) vals[i] =  m.vals[i];
        }
        else vals =  0;
}
Matrix::Matrix(size_t nr, size_t nc, double const * v) :
        n_row(nr), n_col(nc)
{
        if (n_row * n_col) {
                vals =  new double[n_row * n_col];
                for (size_t i =  n_row * n_col; i --;) vals[i] =  v[i];
        }
        else vals =  0;
}
Matrix::~ Matrix()
{
        if (vals) delete [] vals;
}
double & Matrix::elem(size_t ri, size_t ci) // element access
{
        if (ri > =  n_row || ci > =  n_col) throw;
        return vals[ri * n_col +  ci];
}
double Matrix::elem(size_t ri, size_t ci) const // retrieval
{
        if (ri > =  n_row || ci > =  n_col) throw;
        return vals[ri * n_col +  ci];
}
Matrix Matrix::operator+ (const Matrix & m) const
{
        if (n_col !=  m.n_col || n_row !=  m.n_row) throw;
        Matrix r =  * this;
        for (size_t i =  n_row * n_col; i --;) r.vals[i] + =  m.vals[i];
        return r;
}
Matrix Matrix::operator-(const Matrix & m) const
{
        if (n_col !=  m.n_col || n_row !=  m.n_row) throw;
        Matrix r =  * this;
        for (size_t i =  n_row * n_col; i --;) r.vals[i] -=  m.vals[i];
        return r;
}

Matrix Matrix::operator*(const Matrix & m) const
{
        if (n_col !=  m.n_row) throw;
        Matrix r(n_row, m.n_col);
        for (size_t i =  0; i <  n_row; i + + ) for (size_t j =  0; j <  m.n_col; j + + ) {
                double v =  0;
                for (size_t k =  0; k <  n_col; k + + )
                        v + =  vals[i * n_col +  k] * m.vals[k * n_col +  j];
                r.vals[i * r.n_col +  j] =  v;
        }
        return r;
}

Matrix Matrix::operator*(double v) const
{
        Matrix r =  * this;
        for (size_t i =  n_row * n_col; i --;) r.vals[i] *=  v;
        return r;
}

Matrix & Matrix::operator= (const Matrix & m)
{
        if (vals) delete [] vals;
        n_row =  m.n_row;
        n_col =  m.n_col;
        if (size_t sz =  n_row * n_col) {
                vals =  new double [sz];
                for (size_t i =  sz; i --;) vals[i] =  m.vals[i];
        }
        else vals =  0;
}

Matrix Matrix::unit(size_t sz)
{
        Matrix mm(sz +  1, sz +  1);
        Matrix m(sz, sz);
        for (size_t i =  0; i <  sz; i + + )
                for (size_t j =  0; j <  sz; j + + ) m.elem(i, j) =  (i = =  j ? 1 : 0);
        return m;
}
```

```
void Matrix::stream_in(std::istream& s)
{
	std::string n;
	s >> n;
	if (n != "(Matrix") throw;
	size_t nr;
	size_t nc;
	s >> nr >> nc;
	double* v = 0;
	if (nr * nc) {
		v = new double[nr * nc];
		for (size_t i = 0; i < nr * nc; i++) s >> v[i];
	}
	s >> n;
	if (n != "Matrix)") {
		if (v) delete[] v;
		throw;
	}
	if (vals) delete[] vals;
	n_row = nr;
	n_col = nc;
	vals = v;
}


void Matrix::stream_out(std::ostream& o) const
{
	o << "(Matrix " << n_row << ' ' << n_col << '\n';
	for (size_t ri = 0; ri < n_row; ri++) {
		double* row = vals + ri * n_col;
		for (size_t ci = 0; ci < n_col; ci++) {
			o << std::setw(16) << row[ci];
		}
		o << '\n';
	}
	o << "Matrix)";
}


Matrix Matrix::submat(size_t r, size_t c) const
{
	Matrix m(n_row - 1, n_col - 1);
	for (size_t s = 1; s < n_row; s++) for (size_t d = 1; d < n_col; d++)
{
		m.elem(s - 1, d - 1) = elem(s <= r ? s - 1 : s, d <= c ? d - 1 : d);
	}
	return m;
}
```

```
double Matrix::det() const
{
	assert(n_row == n_col);
	if (n_row == 1) return vals[0];
	double det = 0;
	int sign = 1;
	for (size_t r = 0; r < n_row; r++) {
		// Laplace expansion
		det += sign * elem(r, 0) * submat(r, 0).det();
		sign = -sign;
	}
	return det;
}

Matrix Matrix::transpose() const
{
	Matrix m(n_col, n_row);
	for (size_t r = 0; r < n_row; r++) for (size_t c = 0; c < n_col; c++) {
		m.elem(c, r) = elem(r, c);
	}
	return m;
}

Matrix Matrix::inverse() const
{
	double d = det();
	assert(d);
	Matrix m(n_row, n_col);
	for (size_t r = 0; r < n_row; r++) for (size_t c = 0; c < n_col; c++) {
		int sign = (r + c) % 2 ? -1 : 1;
		m.elem(r, c) = sign * submat(r, c).det() / d;
	}
	return m.transpose();
}

Matrix operator*(double v, const Matrix& m)
{
	return m * v;
}

std::istream& operator>>(std::istream& s, Matrix& m)
{
	m.stream_in(s);
	return s;
}

std::ostream& operator<<(std::ostream& o, const Matrix& m)
{
	m.stream_out(o);
	return o;
}
```

定義資料輸出 / 入格式

```
cpl@area:~/mat$ ./mkmat<<<1
(Matrix 4 4
        0.943497       -0.177621        0.271653      -0.0365508
        0.418603       -0.405261      -0.0643052       0.0249313
       -0.780158        0.557913        0.972258        0.606967
       0.0592293       -0.0255439        0.58889       -0.388066
Matrix)
cpl@area:~/mat$
```

```
void Matrix::stream_in(std::istream & s)
{
        std::string n;
        s >> n;
        if (n != "(Matrix") throw;
        size_t nr;
        size_t nc;
        s >> nr >> nc;
        double * v = 0;
        if (nr * nc) {
                v = new double [nr * nc];
                for (size_t i = 0; i < nr * nc; i ++) s >> v[i];
        }
        s >> n;
        if (n != "Matrix)") {
                if (v) delete [] v;
                throw;
        }
        if (vals) delete [] vals;
        n_row = nr;
        n_col = nc;
        vals = v;
}
```

```
void Matrix::stream_out(std::ostream & o) const
{
        o << "(Matrix " << n_row << ' ' << n_col << '\n';
        for (size_t ri = 0; ri < n_row; ri ++) {
                double * row = vals + ri * n_col;
                for (size_t ci = 0; ci < n_col; ci ++) {
                        o << std::setw(16) << row[ci];
                }
                o << '\n';
        }
        o << "Matrix)";
}
```

資料的轉換需要有 well-defined 的格式

文字模式 ASCII

二進模式:

```
ostream file("filename", ostream::binary);
file.write(& var, sizeof(var));

istream file("filename", istream::binary);
file.read(& var, sizeof(var));
```

程式庫的包裝

header files (.h .hh .H)： 函式宣告，全域變數宣示，類別宣告， inline 函式
program files (.cpp .cc .C)： 函式定義，全域變數宣告

［編譯 compile ］

object files (.o)： 從 program files 編譯好的程式碼

［連結 link ］

executables： 所有會用到的程式碼連結成含有主函式 main() 的可執行檔案

程式庫 library (.a .so)： 程式碼的集合，用“ar” 來管理（見 man ar）

```
ar r libsome.a file1.o file2.o file3.o
```

建立程式庫 libsome

```
c++ -I. -L. prog.cc -lsome -o exec
```

使用程式庫

等同於：

```
c++ -I. -L. prog.cc file1.o file2.o file3.o -o exec
```

自動化編譯： Makefile

```
all:
CXXFLAGS := -I.
LDFLAGS := -L.
LDLIBS := -lmat
libmat.a: mat.o
	ar r $@ $^
.PHONY: clean
EXECS :=
EXECS += mtest
EXECS += mkmat
EXECS += invert
EXECS += multip
all: $(EXECS)
$(EXECS): | libmat.a
clean:
	rm -f libmat.a *.o *~ $(EXECS)
```

語法

< 標的 target>: < 必備物 prerequisite>
< 命令 command1>
< 命令 command2>

＊＊＊命令行必須以 tab 字元開頭

makefile 變數

< 變數 variable> := < 字串 >
< 變數 variable> += < 字串 >
使用： $(< 變數 variable>)
$< 變數 variable>

特別變數
$@： *target*
$<： *first prerequisite*
$^： *all prerequisite*

＊＊＊命令行中使用 BASH 變數，得以“$$”來代表“$”。

GNU make 的 implicit rules
file.cc ⟶ file.o
	$(CXX) -c $(CXXFLAGS) file.cc
prog.cc ⟶ exec
	$(CXX) $(CXXFLAGS) $(LDFLAGS) prog.cc $(LDLIBS) -o exec

執行範例

```
cp1@area:~$ cp -r /usr/local/src/mat .
cp1@area:~$ cd mat
cp1@area:~/mat$ ls -l
total 36
-rw-r--r-- 1 cp1 cp1  117 2011-05-02 02:51 invert.cc
-rw-r--r-- 1 cp1 cp1  239 2011-05-02 02:51 Makefile
-rw-r--r-- 1 cp1 cp1 4293 2011-05-02 02:51 mat.cc
-rw-r--r-- 1 cp1 cp1 1143 2011-05-02 02:51 mat.hh
-rw-r--r-- 1 cp1 cp1  258 2011-05-02 02:51 mkmat.cc
-rw-r--r-- 1 cp1 cp1  333 2011-05-02 02:51 mtest.cc
-rw-r--r-- 1 cp1 cp1  136 2011-05-02 02:51 multip.cc
-rw-r--r-- 1 cp1 cp1  897 2011-05-02 02:51 ran_nr.hh
cp1@area:~/mat$ make
g++ -I.   -c -o mat.o mat.cc
ar r libmat.a mat.o
ar: creating libmat.a
g++ -I.  -L.  mtest.cc  -lmat -o mtest
g++ -I.  -L.  mkmat.cc  -lmat -o mkmat
g++ -I.  -L.  invert.cc  -lmat -o invert
g++ -I.  -L.  multip.cc  -lmat -o multip
cp1@area:~/mat$ ./mkmat <<<1
(Matrix 4 4
        0.943497       -0.177621        0.271653      -0.0365508
        0.418603       -0.405261      -0.0643052       0.0249313
       -0.780158        0.557913        0.972258        0.606967
       0.0592293      -0.0255439         0.58889       -0.388066
Matrix)
cp1@area:~/mat$ ./mkmat <<<1|./invert
(Matrix 4 4
         1.29139       -0.757259       -0.158043       -0.417474
         1.33383        -3.28427       -0.198221       -0.646662
         0.10404        0.623501        0.525008        0.851413
        0.267183         1.04677        0.785625        -1.30601
Matrix)
cp1@area:~/mat$ (./mkmat <<<1|./invert;./mkmat <<<1)|./multip
(Matrix 4 4
               1    -7.73741e-07     8.24163e-07    -2.09216e-07
     1.69002e-06        0.999999    -1.78004e-07      2.5577e-07
      2.2172e-07     5.21723e-08               1     3.87273e-08
     1.73442e-06    -1.37915e-06     2.27164e-06        0.999998
Matrix)
cp1@area:~/mat$
```

登入 CP1 SSH 伺服器後可以在 /usr/local/src/mat 目錄下找到程式

mkmat： 讀入 random seed， 產生 random matrix
invert： 讀入 matrix， 產生 inverse matrix
multip： 讀入兩個 matrix， 產生 product matrix