

# 继承 上机练习

李宇豪 21305412

将上一次上机作业里完成的矩阵和向量类，matrix.h, vector.h 拿出来，本次作业要基于上次作业继续，所以一定要将上次作业写对!!!!!!

如果你对自己上次作业完成的质量没信心，请用老师提供的 matrix.h, vector.h.

## 1. 继承 Matrix 类编写一个方阵 SquareMatrix 的派生类

要求：

- 1) 将原 Matrix 类的成员数据 row,col,p\_data 的属性改为 protected, 我们这样设计的目的是啥？  
因为派生类不能访问基类的 private 数据成员，因此要修改为 protected 以方便派生类对其访问
- 2) 方阵的行列是相等的，故构造函数只提供一个参数；
- 3) 增加方阵求逆的成员函数，SquareMatrix& inverse(). 请使用 Gauss Jordan 消去法，这个方法应该在线性代数课程已经讲过。通过例子来检查这个函数是否编写正确。

```
class SquareMatrix : public Matrix
{
public:
    SquareMatrix(int _n): Matrix(_n, _n)
    {
        row = _n;
        col = _n;
        p_data = new double[row * col];
    }
    SquareMatrix& inverse();
};

SquareMatrix& SquareMatrix::inverse()
{
    int i, j, k;
    double ratio;

    SquareMatrix *inv = new SquareMatrix(row);
    inv->init(0);
    for(i=0; i<row; i++)
        inv->data(i,i)= 1.;

    // inv->print();

    // 对增广阵进行操作 [this.p_data inv.p_data]
    /* Applying Gauss Jordan Elimination */
```

```

for(i=0; i<row; i++)
{
    if(data(i,i) == 0.0)
    {
        printf("Mathematical Error!");
        exit(0);
    }
    for(j=0; j<row; j++)
    {
        if(i != j)
        {
            ratio = data(j,i)/data(i,i);
            for(k=0; k<row; k++)
            {
                data(j,k) -= ratio * data(i,k);
                inv->data(j,k) -= ratio * inv->data(i,k);
            }
        }
    }
}
/* Row Operation to Make Principal Diagonal to 1 */
for(i=0; i<row; i++)
{
    for(j=0; j<row; j++)
    {
        inv->data(i,j) /= data(i,i);
    }
}

return *inv;
}

```

## 2. 编写一个 LinearSystem 类

该类为线性方程组，包括一个方阵和右端项的向量。

要求：

- 1) 提供一个构造函数，LinearSystem(SquareMatrix &ma, Vector &vec)
- 2) 编程一个求解线性方程的函数，Vector& solve()。无需形参，因为都在成员对象了，返回向量为方程组的解。算法是先对方阵求逆，再用矩阵和向量相乘。
- 3) 利用如下数据测试上面两个函数的正确性。

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & -2 \\ 2 & -2 & 1 \end{bmatrix} \quad b = \begin{pmatrix} 6 \\ 1 \\ 1 \end{pmatrix}$$

LinearSystem 类如下:

```
class LinearSystem
{
public:
    LinearSystem(SquareMatrix &ma, Vector &vec): mat(ma), vect(vec)
    {
    }

    Vector& solve()
    {
        return mat.inverse().multiply(vect);
    }

private:
    SquareMatrix mat;
    Vector vect;
};
```

main 函数中初始化和测试代码如下:

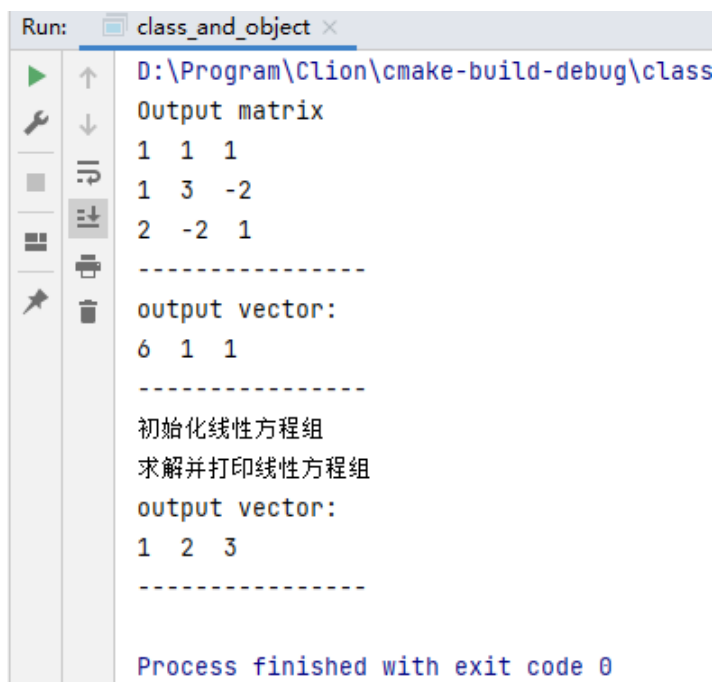
```
int main()
{
    SquareMatrix a(3);
    a.init(1.);
    a.data(1,1)= 3.; a.data(1,2)= -2.;
    a.data(2,0)= 2.; a.data(2,1)= -2.;
    a.print();
    cout << "-----" << endl;

    Vector b(3);
    b.data(0) = 6.;
    b.data(1) = 1.;
    b.data(2) = 1.;
    b.print();
    cout << "-----" << endl;

    cout << "初始化线性方程组" << endl;
    LinearSystem linear(a, b);
    cout << "求解并打印线性方程组" << endl;
    linear.solve().print();
    cout << "-----" << endl;

    return 0;
}
```

运行结果为：



```
Run: class_and_object x
D:\Program\Clion\cmake-build-debug\class
Output matrix
1 1 1
1 3 -2
2 -2 1
-----
output vector:
6 1 1
-----
初始化线性方程组
求解并打印线性方程组
output vector:
1 2 3
-----
Process finished with exit code 0
```

测试正常