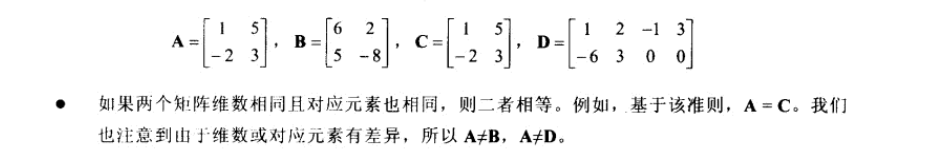
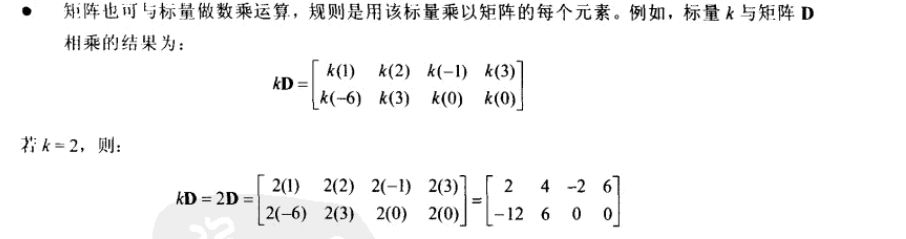
## 矩阵相等

如果两个矩阵的维数相同，所有成员也相同，那么就说这两个矩阵相等。只要有一个条件不满足，这两个矩阵就不等。



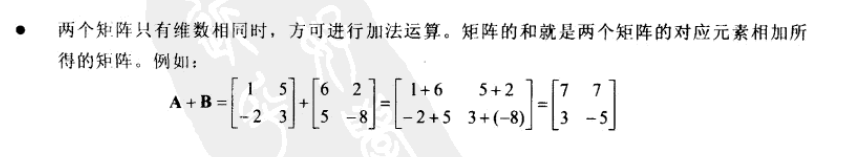
## 标量乘于矩阵

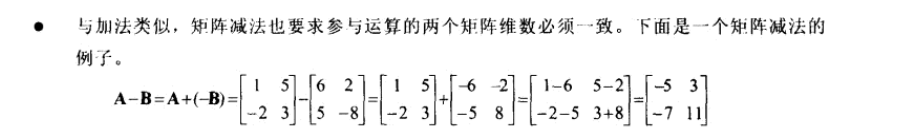
**就是用标量乘于矩阵的每一个元素。**



## 矩阵的加法

**两个矩阵只有维数相同的时候才能进行相加。两个矩阵的和就是两个矩阵对应元素分别相加得到的矩阵。**

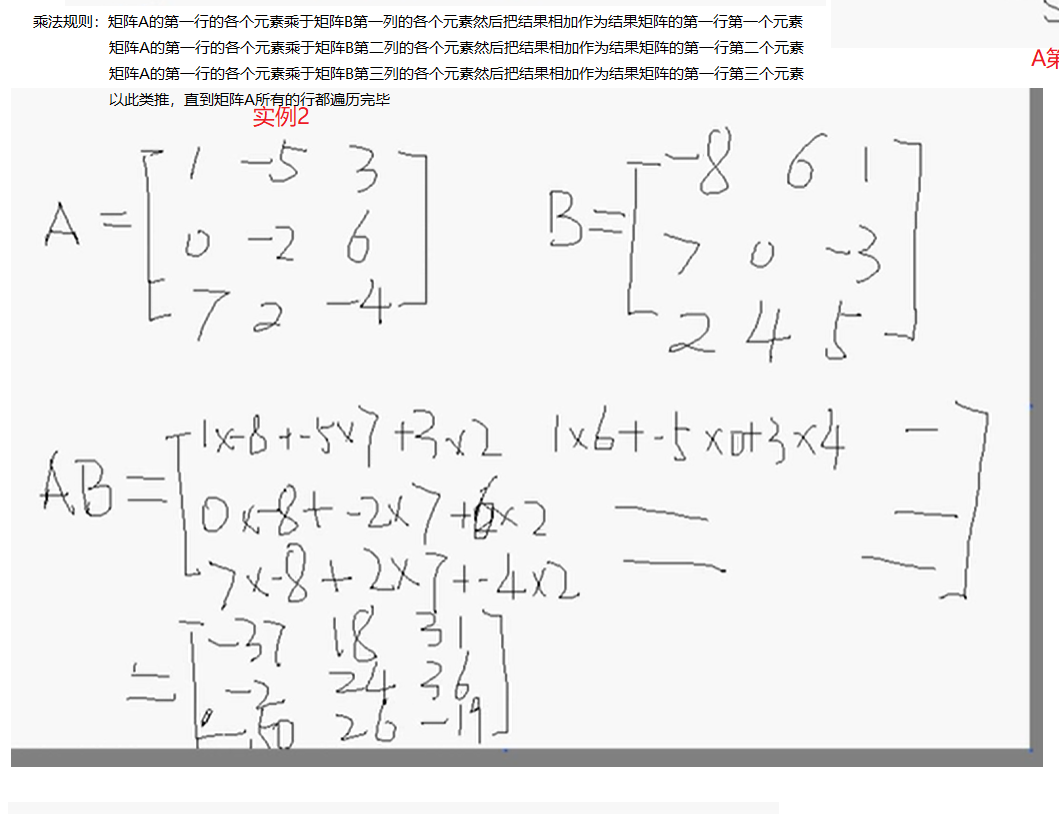


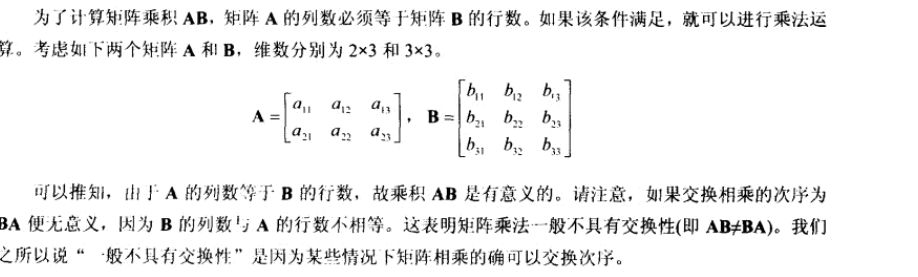


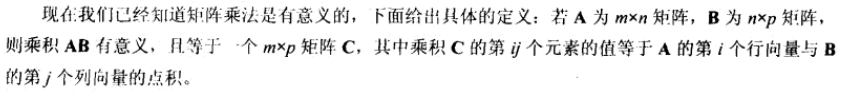
## 矩阵的乘法

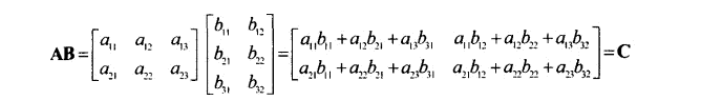
**必须满足如下条件：矩阵A乘于矩阵B，则A的行数必须等于矩阵B的列数，而且AB!=BA**

**具体运算方法**

****

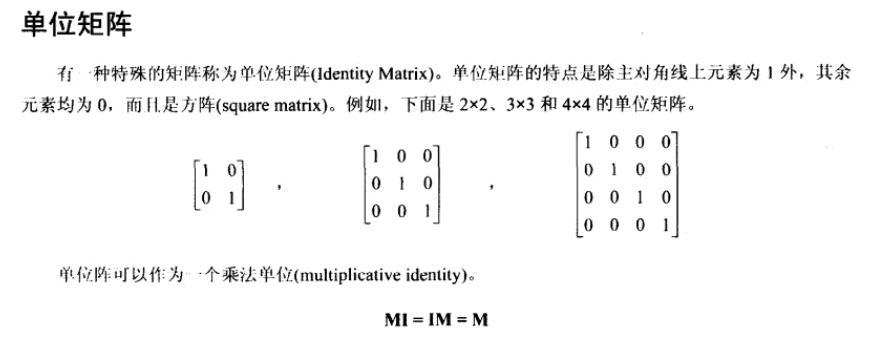






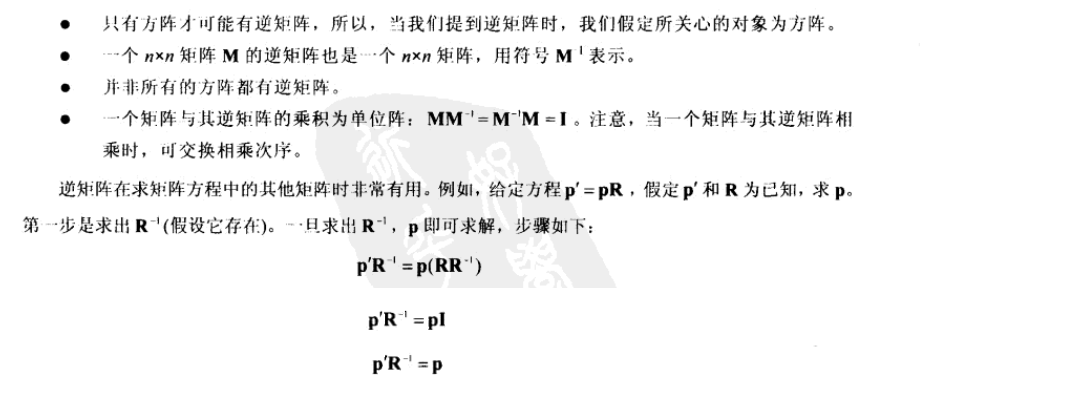
## 5.单位矩阵

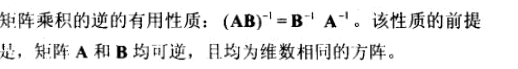
就是只有主对角线的元素的值是1，其余的元素都是0的矩阵（Identity Matrix）



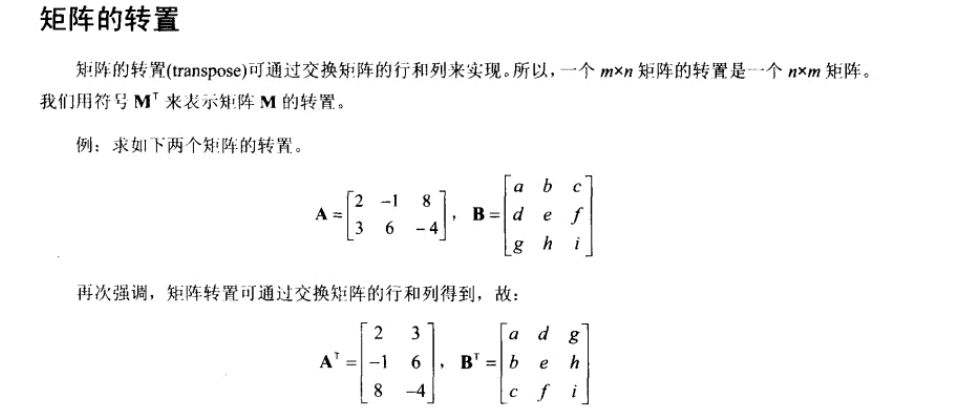
## 6矩阵的逆

**注意：只有方阵（行数和列数相等的矩阵）才于逆矩阵**

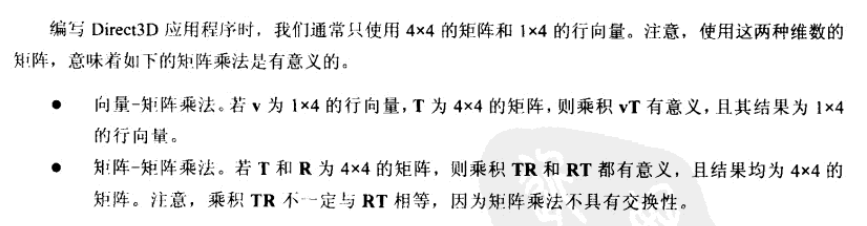
****

****

## 7．矩阵的转置



## 8.D3DX矩阵



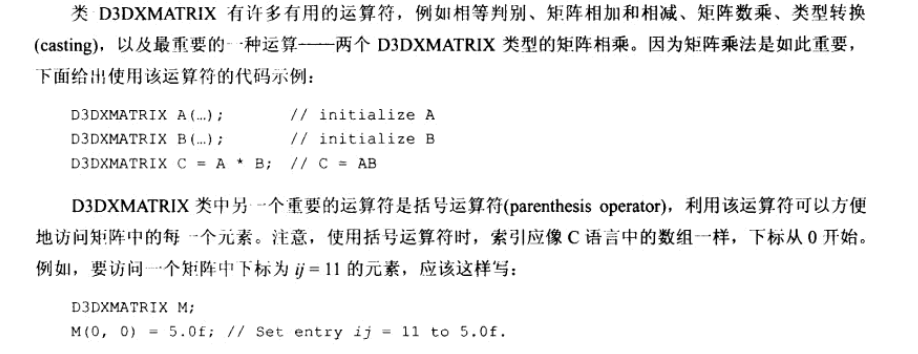
D3DX库中，1x4向量使用D3DXVECTOR3和D3DXVECTOR4。D3DXVECTOR3只包含三个分量，但是可以理解为它的第四个分量是1或0 .

4X4矩阵可以使用D3DXMATRIX:

|  |
| --- |
| typedef struct D3DXMATRIX : public D3DMATRIX  {  public:  D3DXMATRIX() {};  D3DXMATRIX( CONST FLOAT \* );  D3DXMATRIX( CONST D3DMATRIX& );  D3DXMATRIX( CONST D3DXFLOAT16 \* );  D3DXMATRIX( FLOAT \_11, FLOAT \_12, FLOAT \_13, FLOAT \_14,  FLOAT \_21, FLOAT \_22, FLOAT \_23, FLOAT \_24,  FLOAT \_31, FLOAT \_32, FLOAT \_33, FLOAT \_34,  FLOAT \_41, FLOAT \_42, FLOAT \_43, FLOAT \_44 );  // access grants  FLOAT& operator () ( UINT Row, UINT Col );  FLOAT operator () ( UINT Row, UINT Col ) const;  // casting operators  operator FLOAT\* ();  operator CONST FLOAT\* () const;  // assignment operators  D3DXMATRIX& operator \*= ( CONST D3DXMATRIX& );  D3DXMATRIX& operator += ( CONST D3DXMATRIX& );  D3DXMATRIX& operator -= ( CONST D3DXMATRIX& );  D3DXMATRIX& operator \*= ( FLOAT );  D3DXMATRIX& operator /= ( FLOAT );  // unary operators  D3DXMATRIX operator + () const;  D3DXMATRIX operator - () const;  // binary operators  D3DXMATRIX operator \* ( CONST D3DXMATRIX& ) const;  D3DXMATRIX operator + ( CONST D3DXMATRIX& ) const;  D3DXMATRIX operator - ( CONST D3DXMATRIX& ) const;  D3DXMATRIX operator \* ( FLOAT ) const;  D3DXMATRIX operator / ( FLOAT ) const;  friend D3DXMATRIX operator \* ( FLOAT, CONST D3DXMATRIX& );  BOOL operator == ( CONST D3DXMATRIX& ) const;  BOOL operator != ( CONST D3DXMATRIX& ) const;  } D3DXMATRIX, \*LPD3DXMATRIX; |

D3DMATRIX等于如下：

|  |
| --- |
| typedef struct \_D3DMATRIX {  union {  struct {  float \_11, \_12, \_13, \_14;  float \_21, \_22, \_23, \_24;  float \_31, \_32, \_33, \_34;  float \_41, \_42, \_43, \_44;  };  float m[4][4];  };  } D3DMATRIX; |



D3DXMATRIX常用函数

1. 转换为单位矩阵

|  |
| --- |
| D3DXINLINE D3DXMATRIX\* D3DXMatrixIdentity ( D3DXMATRIX /\*需要转换的矩阵\*/\*pOut ) |

1. 矩阵转置

|  |
| --- |
| D3DXMATRIX\* WINAPI D3DXMatrixTranspose( D3DXMATRIX /\*接收结果的矩阵\*/\*pOut, CONST D3DXMATRIX /\*需要转置的矩阵\*/\*pM ); |

1. 求矩阵的行列式，只有方阵才有行列式

|  |
| --- |
| FLOAT WINAPI D3DXMatrixDeterminant( CONST D3DXMATRIX \*pM ); |

4）矩阵的逆矩阵

|  |
| --- |
| D3DXMATRIX\* WINAPI D3DXMatrixInverse ( D3DXMATRIX \*pOut, FLOAT \*pDeterminant, CONST D3DXMATRIX \*pM );  // pOut接收返回值，pDeterminant：determinant，行列式可以为0，pM需要转置的矩阵指针 |

矩阵的行列式计算方法：

