

<Matrix.h>

C 语言基本矩阵运算宏。需要 `typeof` 特性。

以二维数组表示矩阵，形状同数组初始化器。

对于一般矩阵运算，静态推导、检查其类型、形状。

通过定义宏 `NDEBUG` 以忽略对形状的动态检查。

由 VLA 表示的矩阵忽略对形状的静态检查。

对于矩阵运算中具有 VM 类型的 C 表达式，将额外求值 1 次以获得类型。

用户可在头文件处修改宏的名字，但仅限于下文介绍的宏。

矩阵求值

声明于<MatrixBase.h>

```
#define _Mv1    /* Matrix Evaluate With Depth 1 */
#define _Mv13   /* Matrix Evaluate With Depth 3 */
#define _Mv19   /* Matrix Evaluate With Depth 9 */
#define _Mv127  /* Matrix Evaluate With Depth 27 */
#define _Mv181  /* Matrix Evaluate With Depth 81 */
#define _Mv1243 /* Matrix Evaluate With Depth 243 */
#define _Mv1729 /* Matrix Evaluate With Depth 729 */
```

描述

矩阵

二维数组类型的 C 表达式或复合矩阵表达式。

`_Mv1NN`(矩阵)

将矩阵表达式编译为 C，展开深度为 NN。

示例

```
int A[3][3] = {};  
_Mv1(A);
```

简单地返回并丢弃 A 的值。

```
size_t N = rand();  
typedef float matNxN[N][N];  
_Mv1(*(matNxN*)malloc(sizeof(matNxN)));
```

由于该 C 表达式具有 VM 类型，实际发生 2 次分配。

矩阵加、乘法

声明于<MatrixTrivial.h>

```
#define _Mst /* Matrix Set */
```

```
#define _Mgt /* Matrix Get */
```

声明于<MatrixTrivialCall.h>

```
#define _Mcl /* Matrix Call */
```

```
#define _Mrt /* Matrix Return */
```

声明于<MatrixXT.h>

```
#define $T$ /* ^T */
```

```
#define $ /* dot */
```

```
#define $T /* dot ^T */
```

```
#define T$ /* ^T dot */
```

```
#define T$T /* ^T dot ^T */
```

描述

矩阵二次式

具有以下格式之一的参数表示。

A、B 均为矩阵。

A

表示 A

A^T

表示 A^T

$A \cdot B$

表示 AB

$A^T \cdot B$

表示 AB^T

$A \cdot B^T$

表示 $A^T B$

$A^T \cdot B^T$

表示 $A^T B^T$

连接符

在 C 中可用于连接两个数值值以形成表达式的语句片段。

`_Mst`(矩阵, 连接符, 矩阵二次式, 连接符, ... , 矩阵二次式)

执行参数表对应的矩阵操作。返回参数中的第一个矩阵。

`_Mgt`(矩阵二次式, 连接符, ... , 矩阵二次式)

执行参数表对应的矩阵运算。返回临时矩阵以表示结果。

`_Mcl`(可调用对象, 矩阵, 矩阵二次式, ... , 矩阵二次式)

逐元素调用对象, 执行参数表对应的矩阵操作。返回参数中的第一个矩阵。

`_Mrt`(可调用对象, 矩阵二次式, ... , 矩阵二次式)

逐元素调用对象, 执行参数表对应的矩阵运算。返回临时矩阵以表示结果。

示例

```
_Mv19(_Mst(A, = ,B, + ,C $T$, + ,D $ E));
```

将 A 置为 $B + C^T + DE$ 。

```
_Mv19(_Mst(A, += 2*,B, - 3*,C));
```

可在连接符中表示赋值方式和系数。

```
_Mv19(_Mst(A, = ,_Mgt(F T$ A)));
```

必须使用临时矩阵以防止 A 就地自乘。

```
_Mv19(_Mst(A, = ,F T$ _Mgt(P $ F)));
```

超过 2 次的矩阵多项式必须用临时矩阵计算。

```
#define set0(x) x = 0
```

```
_Mv13(_Mcl(set0, A));
```

将 A 置为零矩阵。

矩阵除法

声明于<MatrixDivide.h>

```
#define _Mdv /* Matrix Divide */  
#define _Mnv /* Matrix Inverse */  
#define I$ /*  $^{-1}$  dot */  
#define IT$ /*  $^{-T}$  dot */  
#define $I /* dot  $^{-1}$  */  
#define $IT /* dot  $^{-T}$  */
```

描述

A、B 均为矩阵。

`_Mdv(A I$ B)`

将 B 置为 $A^{-1}B$ ，A 置为不确定值。返回 B。

`_Mdv(A IT$ B)`

将 B 置为 $A^{-T}B$ ，A 置为不确定值。返回 B。

`_Mdv(A $I B)`

将 A 置为 AB^{-1} ，B 置为不确定值。返回 A。

`_Mdv(A $IT B)`

将 A 置为 AB^{-T} ，B 置为不确定值。返回 A。

`_Mnv(A)`

返回临时矩阵 A^{-1} ，并将 A 置为不确定值。

示例

```
_Mv19(_Mst(A, = ,_Mdv(_Mgt(C) I$ _Mgt(D))));
```

将 A 置为 $C^{-1}D$ 。使用临时矩阵保护 C、D 不被修改。

矩阵单位加法

声明于<MatrixGrow.h>

```
#define _Mgr /* Matrix Grow */
```

描述

`_Mgr(矩阵, 数值)`

将矩阵增加数值倍的单位矩阵并返回。

矩阵分块

声明于<MatrixPartition.h>

```
#define _Mpt /* Matrix Partition */
```

描述

`_Mpt(矩阵, 自然数, 自然数, 自然数, 自然数)`

返回矩阵中指定左上角、形状的子矩阵。

示例

```
_Mv13(_Mst(_Mpt(A, 0,0, 3,3), -= ,_Mpt(A, 3,3, 3,3))));
```

将 A 中左上角 3x3 的分块减去其右下方的 3x3 分块。