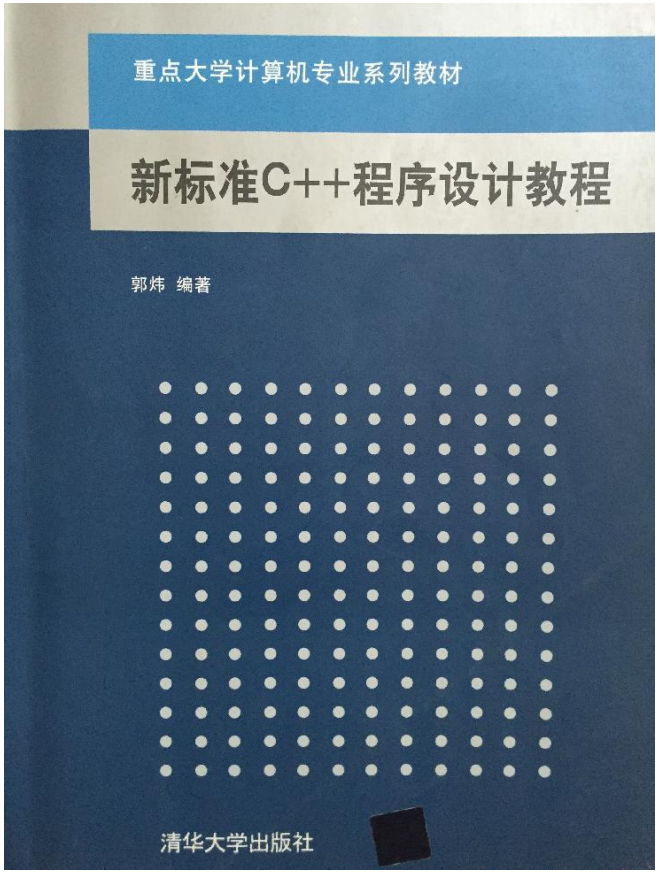
程序设计与算法(一)

李文新 郭炜

主讲教师互动微博：
http://weibo.com/guoweiofpku

指定教材:



二维数组

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m, n,
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p, q，表示下一个矩阵是p行q列（n=p）
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m, n, p, q <= 8)。

输入样例：

```
2 3
2 4 5
2 1 3
3 3
1 1 1
2 3 2
0 1 4
```

输出样例：

```
10 19 30
4 8 16
```

矩阵乘法

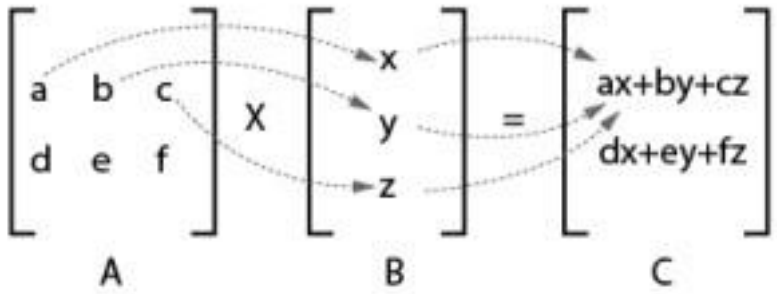
m行n列的矩阵

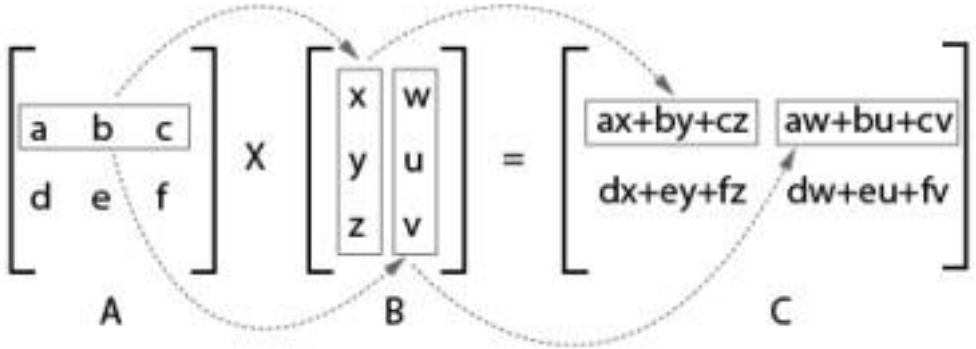
X

n行k列的矩阵

=

m行k列的矩阵





矩阵的乘法

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m, n,
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p, q，表示下一个矩阵是p行q列的（n=p）。
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m, n, p, q <= 8)。

输入样例：

```
2 3
2 4 5
2 1 3
3 3
1 1 1
2 3 2
0 1 4
```

输出样例：

```
10 19 30
4 8 16
```

用什么存放矩阵？

二维数组

●定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

二维数组

- 定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

- 每个元素都是一个类型为T的变量

二维数组

●定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

●每个元素都是一个类型为T的变量

●N×M个元素在内存里是一个挨一个连续存放的。

二维数组

●定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

●每个元素都是一个类型为T的变量

●N×M个元素在内存里是一个挨一个连续存放的。

●数组占用了一片连续的、大小总共为 N×M×sizeof(T)字节的存储空间。

二维数组

●定义N行M列的二维数组：

```
T a[N][M];    // T :类型名，如char , double, int等。
              // M、N : 正整数，或值为正整数的常量表达式
```

●每个元素都是一个类型为T的变量

●N×M个元素在内存里是一个挨一个连续存放的。

●数组占用了一片连续的、大小总共为 N×M×sizeof(T)字节的存储空间。

●表达式“sizeof(a)”的值就是整个数组的体积，即N×M×sizeof(T)。

二维数组

- 访问数组元素的方法：

  数组名[行下标][列下标]

  例如：a[i][j]

二维数组

- 访问数组元素的方法：

  数组名[行下标][列下标]

  例如：a[i][j]

- 行下标和列下标都从0开始

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

- 第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
  同一行的元素，在内存中是连续存放的。

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

- 第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
  同一行的元素，在内存中是连续存放的。

- 第j列的元素的元素，就是a[0][j]、a[1][j]……a[N-1][j]。

二维数组的存放方式

- 数组T a[N][M] 每一行都有M个元素

- 第i行的元素就是a[i][0]、a[i][1]……a[i][M-1]。
  同一行的元素，在内存中是连续存放的。

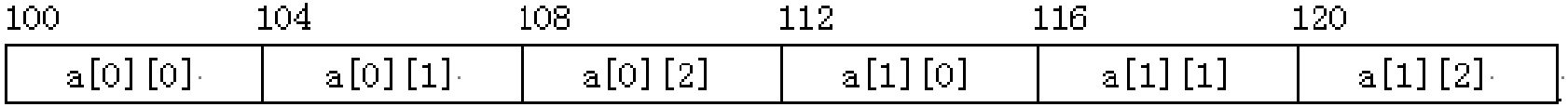
- 第j列的元素的元素，就是a[0][j]、a[1][j]……a[N-1][j]。

- a[0][0]是数组中地址最小的元素。如果a[0][0]存放在地址n，则a[i][j]存放的地址就是

$$n + i \times M \times sizeof(T) + j \times sizeof(T)$$

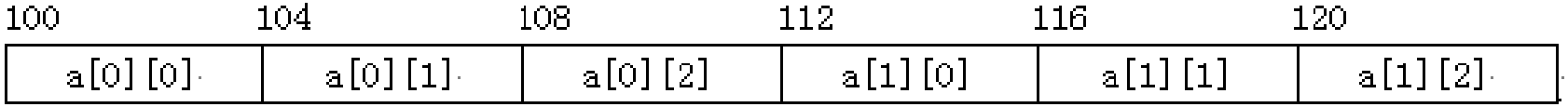
二维数组的存放方式

- int a[2][3] 在内存中的存放方式：



二维数组的存放方式

- int a[2][3] 在内存中的存放方式：



- 二维数组的每一行，实际上都是一个一维数组。

a[0]，a[1]都可以看作是一个一维数组的名字，可以直接当一维数组使用。

二维数组的初始化

```
int a[5][3]={{80,75,92},{61,65},{59,63,70},{85,90},{76,77,85}};
```

每个内层的{}，初始化数组中的一行。

二维数组的初始化

```
int a[5][3]={{80,75,92},{61,65},{59,63,70},{85,90},{76,77,85}};
```

每个内层的{}，初始化数组中的一行。

●二维数组初始化时，如果对每行都进行了初始化，则也可以不给出行数：

```
int a[][3]={ {80,75,92},{61,65} };
```

a 是一个2行3列的数组，a[1][2]被初始化成0。

遍历二维数组

遍历一个二维数组，将其所有元素逐行依次输出：

```
#define ROW 20
#define COL 30
int a[ROW][COL];
for( int i = 0; i < ROW ; ++i) {
        for( int j = 0; j < COL ; ++j )
                cout << a[i][j] << " ";
        cout << endl;
}
```

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m, n,
表示第一个矩阵是m行n列的。接下来时一个m×n的矩阵。
再下一行的输入是整数p, q，表示下一个矩阵是p行q列（n=p）
再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵(1 < m, n, p, q <= 8)。

输入样例：

```
2 3
2 4 5
2 1 3
3 3
1 1 1
2 3 2
0 1 4
```

输出样例：

```
10 19 30
4 8 16
```

矩阵乘法

```
#include <iostream>
using namespace std;
#define ROWS 8
#define COLS 8
int a[ROWS][COLS];
int b[ROWS][COLS];
int c[ROWS][COLS]; //结果
int main()
{
        int m,n,p,q;
        cin >> m >> n;
        for(int i = 0;i<m; ++i) //读入a矩阵
                for(int j = 0; j < n; ++j)
                        cin >> a[i][j];
        cin >> p >> q;
        for(int i = 0;i<p; ++i) //读入b矩阵
                for(int j = 0; j < q; ++j)
                        cin >> b[i][j];
```

矩阵乘法

```
        for(int i = 0; i < m; ++i) {
                for(int j = 0; j < q; ++j)  {
                        c[i][j] = 0;
                        for(int k = 0; k < n; ++k)
                                c[i][j] += a[i][k] * b[k][j];
                }
        }
        for(int i = 0;i<m;++i){
                for(int j = 0; j < q; ++j) {
                        cout << c[i][j] << " ";
                }
                cout << endl;
        }
        return 0;
}
```

```
2 4 5
2 1 3

1 1 1
2 3 2
0 1 4
```