简单矩阵的加法

#对导出类型进行简单探索

给定两个 $m \times n$ 的矩阵以及一个常数 k
设计一段程序，计算两个矩阵的加法，以及加和后乘上常数 k 的结果

输入格式

第一行为三个整数 m, n, k，接下来为两个矩阵

样例输入

```
2 3 4

1 2 3
4 5 6

4 5 6
7 8 9
```

样例输出

```
5 7 9
11 13 15

20 28 36
44 52 60
```

测试样例

```
3 3 7
5 3 8
7 2 6
4 9 1
2 9 4
6 5 3
8 1 7
```

```
7 12 12
13 7 9
12 10 8

49 84 84
91 49 63
84 70 56
```

```
4 4 8
3  7  2  8
5  1  6  4
9  0  3  5
4  6  7  1
8  2  5  1
3  6  4  7
1  9  0  2
7  5  8  3
```

```
11 9 7 9
8 7 10 11
10 9 3 7
11 11 15 4

88 72 56 72
64 56 80 88
80 72 24 56
88 88 120 32
```

```
1 1 7
1
1

2

14
```

```
3 3 0
5 3 4
1 8 6
6 9 1
3 5 3
9 5 6
8 1 7
```

```
0 0 0
0 0 0
0 0 0

0 0 0
0 0 0
0 0 0
```

Codes

```
#include <iostream>
#include <cmath> // 用于 sqrt 和 pow 函数

// 枚举类型，定义图形种类
enum ShapeType {
    Circle,
    Rectangle,
    Triangle
};

// 共用体，用于存储不同类型图形的数据
union ShapeData {
    struct {
        double radius; // 圆形的半径
    } circle;
    struct {
        double width, height; // 矩形的宽度和高度
    } rectangle;
    struct {
        double base, height; // 三角形的底边和高
    } triangle;
};

// 结构体，表示一个图形
struct Shape {
    ShapeType type;
    ShapeData data;

    // 计算面积的成员函数
    double area() const {
        switch (type) {
            case Circle:
                if (data.circle.radius < 0) return -1;
                return M_PI * data.circle.radius * data.circle.radius;
            case Rectangle:
                if (data.rectangle.width < 0 || data.rectangle.height < 0) return
-1;
                return data.rectangle.width * data.rectangle.height;
            case Triangle:
                if (data.triangle.base < 0 || data.triangle.height < 0) return -1;
                return 0.5 * data.triangle.base * data.triangle.height;
            default:
```

```
                return 0.0; // 默认情况处理
        }
    }
};

int main() {
    int a, b, k;
    std::cin >> a >> b >> k;
    int **arr1 = new int*[a];
    for(int i = 0; i < a; i++){
        arr1[i] = new int[b];
    }

    int **arr2 = new int*[a];
    for(int i = 0; i < a; i++){
        arr2[i] = new int[b];
    }

    int **ans = new int*[a];
    for(int i = 0; i < a; i++){
        ans[i] = new int[b];
    }

    for(int i = 0; i < a; i++){
        for(int j = 0; j < b; j++){
            std::cin >> arr1[i][j];
        }
    }

    for(int i = 0; i < a; i++){
        for(int j = 0; j < b; j++){
            std::cin >> arr2[i][j];
            ans[i][j] = arr1[i][j] + arr2[i][j];
        }
    }

    for(int i = 0; i < a; i++){
        for(int j = 0; j < b; j++){
            std::cout << ans[i][j] << " \n"[j == b - 1];
            ans[i][j] *= k;
        }

    }
    std::cout << std::endl;
    for(int i = 0; i < a; i++){
        for(int j = 0; j < b; j++){
            std::cout << ans[i][j] << " \n"[j == b - 1];
        }
    }

    return 0;
}
```