

第一题:

```
#include <iostream>
using namespace std;

class Matrix{
private:
    int matrix[4][4]; //私有成员变量

public:
    Matrix();    //构造函数,将各元素的值设为 1
    Matrix(bool frominput);    //构造函数, 传入 bool 值 true 时, 由键盘输入 matrix 矩阵的值,
    这里用到了函数重载
    void Display_matrix();    //在屏幕上显示矩阵
};

Matrix::Matrix(){
    for(int i = 0; i<4; i++){
        for(int j = 0; j<4; j++){
            matrix[i][j] = 1;
        }
    }
} //构造函数, 将矩阵的初始值各元素设为 1

Matrix::Matrix(bool frominput){
    if(frominput){    //当传入的 bool 值为 true 时, 从键盘上输入元素
        for(int i = 0; i<4; i++){
            for(int j = 0; j<4; j++){
                cout<<"请输入["<<i<<" "<<j<<"]位置上的元素: "; //给用户的提示信息
                cin>>matrix[i][j];    //从键盘上输入 matrix 中元素的值
            }
        }
        cout<<"输入完成! "<<endl;
    }else{
        Matrix();    //否则, 调用没有传入参数的构造函数, 将矩阵的元组值设为 1
    }
} //重载构造函数, 由键盘输入元素的值

void Matrix::Display_matrix(){
    cout<<"Matrix:"<<endl;
    for(int i = 0; i<4; i++){
        for(int j = 0; j<4; j++){
            cout<<matrix[i][j]<<" "; //将矩阵显示在屏幕上
        }
        cout<<endl;    //4 个元素后换行
    }
} //在屏幕上显示 matrix 矩阵

int main(){

    //matrix1 在构造时自动设为 1
```

```

Matrix matrix1;
cout<<"matrix1:"<<endl;
matrix1.Display_matrix();    //显示 matrix1

//matrix2 在构造时由键盘输入
Matrix matrix2(true);
cout<<"matrix2:"<<endl;
matrix2.Display_matrix();    //显示 matrix2

return 0;
}

```

第二题:

```

#include <iostream>

using namespace std;

class cuboid{
private:
    double length;    //长
    double width;     //宽
    double height;    //高
    double SinglePrice; //单个长方体的单独造价

public:
    cuboid(double l=5, double w=5, double h=5); //构造函数，默认参数都为 5
    double get_SinglePrice(); //返回单个长方体的造价
    void Display(); //显示该长方体的长、宽、高和单个造价
};

cuboid::cuboid(double l, double w, double h){
    length = l;
    width = w;
    height = h;
    SinglePrice = 10*length*width*height; //计算单个长方体的造价
} //构造函数，由用户给入长宽高，默认为 5，并计算单个造价

double cuboid::get_SinglePrice(){
    return SinglePrice;
} //返回单个长方体的造价

void cuboid::Display(){
    cout<<"长: "<<length<<", 宽: "<<width<<", 高: "<<height<<", 单个造价:
"<<SinglePrice<<endl;
} //显示该长方体的长、宽、高和单个造价

double Add_Price(double SinglePrice){
    static double total_price = 0; //定义总造价为静态变量
    total_price += SinglePrice; //静态变量累加
    return total_price; //返回目前总造价
}

```

```
}

int main(){

    cuboid cuboid_1;//长方体 1 不传入参数，长宽高因分别为 5, 5, 5
    cout<<"长方体 1 的属性: "<<endl;
    cuboid_1.Display(); //显示长方体 1 的属性

    cuboid cuboid_2(3); //长方体 2 传入一个参数 3，长宽高因分别为 3, 5, 5
    cout<<"长方体 2 的属性: "<<endl;
    cuboid_2.Display();      //显示长方体 2 的属性

    cuboid cuboid_3(7.2, 8.9);    //长方体 3 传入 2 个参数 7.2 和 8.9，长宽高因分别为 7.2, 8.9, 5
    cout<<"长方体 3 的属性: "<<endl;
    cuboid_3.Display();      //显示长方体 3 的属性

    cuboid cuboid_4(10, 9, 5.9); //长方体 4 传入 3 个参数 10,9 和 5.9，长宽高因分别为 10,9,5.9
    cout<<"长方体 4 的属性: "<<endl;
    cuboid_4.Display();      //显示长方体 2 的属性

    //总造价计算，在函数 Add_Price()函数中使用静态变量实现
    Add_Price(cuboid_1.get_SinglePrice());
    Add_Price(cuboid_2.get_SinglePrice());
    Add_Price(cuboid_3.get_SinglePrice());
    double sum = Add_Price(cuboid_4.get_SinglePrice());

    cout<<"总造价: "<<sum;

    return 0;
}
```