一个正方体上三组相对面的面上4个顶点的和都相等

可以先求出a1-a8这8个数字的所有排列，然后判断有没有某一个的排列符合题目设定的条件，即a1+a2+a3+a4 = a5+a6+a7+a8,且a1+a3+a5+a7 = a2+a4+a6+a8，且a1+a2+a5+a6=a3=a4+a7+a8。求8个数字的排列和“面试题28：字符串的排列”中求字符串的排列类似，可以将求8个数字的排列的问题分解下，将8个数字中的1个轮流固定放在数组的第一个位置，然后求剩下7个数字的排列，再依次递归下去。

```
//设置一个全局变量来存储是否存在组合
bool flag = false;
//接收维度固定为8的数组，这样如果传入数组元素不足8个会自动补0，也可以手动传入数组长度，更便于拓展
auto func(int numbers[8]) -> bool{
    allArray(numbers, 0);
    return flag;
}
auto canBeCube(int numbers[8]) -> bool{
    bool flag1 = numbers[0]+numbers[1]+numbers[2]+numbers[3] ==
numbers[4]+numbers[5]+numbers[6]+numbers[7];
    bool flag2 = numbers[0]+numbers[2]+numbers[4]+numbers[6] ==
numbers[1]+numbers[3]+numbers[5]+numbers[7];
    bool flag3 = numbers[0]+numbers[1]+numbers[4]+numbers[5] ==
numbers[2]+numbers[3]+numbers[6]+numbers[7];

    //只要有任意一个排列符合要求就将全局变量flag设置为真
    if(flag1 && flag2 && flag3){
        flag = true;
    }
    return (flag1 && flag2 && flag3);
}


auto allArray(int numbers[8], int begin) -> void{
    //本算法将所有符合要求的排列都求出来了，题目要求只需要判定是否存在符合要求的排列，如果只是完成题目要求可以在这里加上，如果全局变量flag为真，直接返回。
    //达到数组末尾，判定当前排列是否符合条件，符合的话就打印出来。
    if(begin >= 8){
        if(!canBeCube(numbers)) return;
        for(int i = 0; i < 8; ++i){
            cout << numbers[i] << " ";
        }
        cout << endl;
        return;
    }
```

```
    //依次将数组中的元素与首元素进行交换，完成递归后，再交换回来
    for(int i = begin;i < 8; ++i){
        int tmp = numbers[i];
                numbers[i] = numbers[begin];
        numbers[begin] = tmp;

        allArray(numbers, begin + 1);

        tmp = numbers[i];
        numbers[i] = numbers[begin];
        numbers[begin] = tmp;
    }
}
```