阅读程序题：

递归：

列表格总结规律。如斐波那契数列（单参数）代码：

```
int fei(int n){
    if(n==1 || n==2) return 1;
    else return f(n-1)+f(n-2);
}
```

n 值	1	2	3	4	5	6
返回值	1	1	2	3	5	8

双参数代码（选自 2024 年 CSP-J 初赛）：

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

int customFunction(int a, int b) {
    if (b == 0) {
        return a;
    }
    return a + customFunction(a, b-1);
}

int main() {
    int x, y;
    cin >> x >> y;
    int result = customFunction(x, y);
    cout << pow(result, 2) << endl;
    return 0;
}
```

a \ b	0	1	2	3	4	5
0	0	0	0	0	0	0
1	1	2	3	4	5	6
2	2	4	6	8	10	12
3	3	6	9	12	15	18
5	5	10	15	20	25	30

其他算法：

第一招：随机设值代入程序，理解程序每一步的意图和代码的用途（用于完全不理解代码）
第二招：对程序进行片段划分，逐层理解（用于理解代码每一步，但不理解整体用途）
第三招：设特殊值代入（用于全面理解程序、做含特殊值的题）
第四招：排除/估计法（用于题干出现极大极难运算的值）
第五招：英文识别意思，或暂时忽略，看完看懂程序整体用途后再来理解函数作用。（用于遇到不认识的函数，如 STL 容器）
第六招：如果是以前学过的代码，大概率是模板代码的修改版本，照搬即可。

完善程序题：

这部分真的没有说特别见效的技巧，大部分是要靠个人代码能力、算法思维的，不管什么人，用什么方法，都必须要将自己代入其中，想象成自己在主观地写这个代码，遵从原题的思路来构建。如果实在不会了，那就蒙题吧。