```
struct test
{
    int i;
    int j;
};
```

方法1：

```
struct test a = {1, 2};
```

方法2：

```
struct test a;
    a = (struct test){1,2};
```

方法3：

```
struct test a;
    a.i = 1;
    a.j = 2;
```

方法4：

```
struct test a = {
     .i = 1,
     .j = 2, // , 可有可为
};
```

方法5：

```
struct test a = {
     i :  1,
     j :  2,
};
```

分析：

类1：方法1，2，

优势：精简，代码量少，

劣势：没指定 所以只能按照顺序来

方法 2 相对于 方法 1就显得多此一举

类2：方法3，4，5

优势：清晰，指定了变量 可以打乱顺序

方法3 相对于 方法4，5 就多此一举了， 4 和 5 差不多

问题点：

1：当 结构体比较复杂时，类 2比 类1要好，因为 清晰写明了初始化的变量，不会弄混

例如：FD_CONFIG_ST fd_info = {65, 0, 0, 1920, 1080, 50, 1920, 1080, 0, 0, 0, 0, 0}；鬼知道少写了没，或者对应关系错乱了，所以应当采用 方法4 或 5

2：当 结构体比较简单时，类 1比类 2好，因为简单明了，没必要多写这么多变量