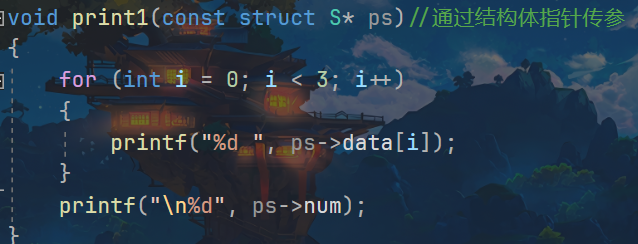
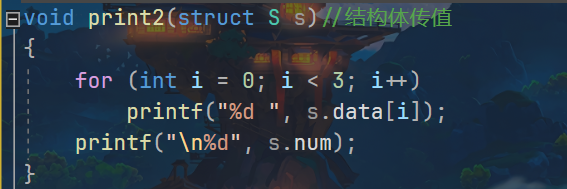
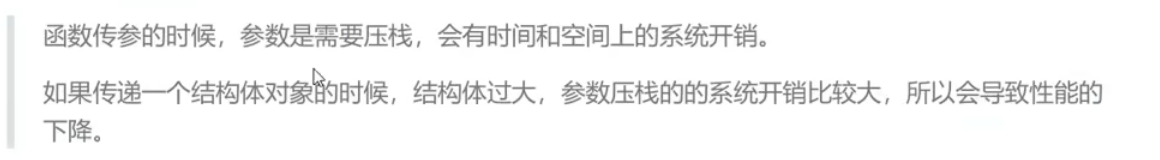
## 结构体传参





那当然是传地址好

原因：



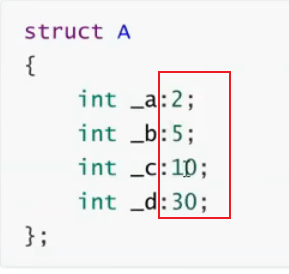
## 位段



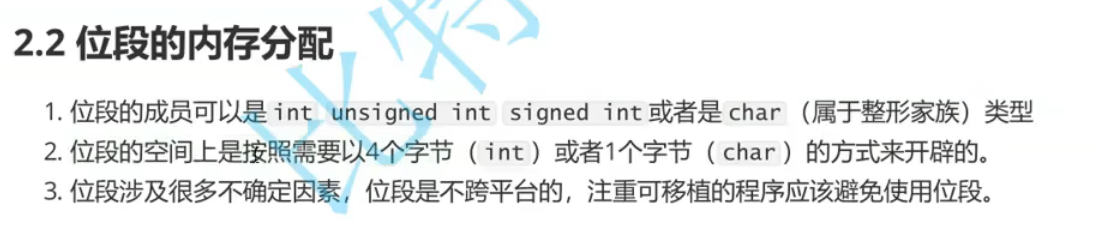
（实际上整型家族的都可以是位段的成员）

位段只能在结构体里面用

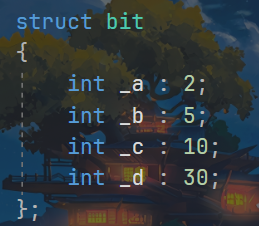
位段的位：就是比特位的意思 bit



冒号后面的数字代表，只需要分配多少的比特位。



一般情况下，位段的类型是统一的

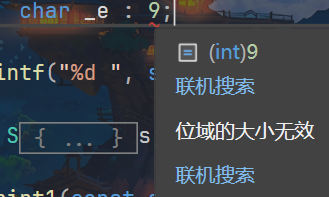


内存是这样开辟的：看到为整型—先开辟4byte（32bit）

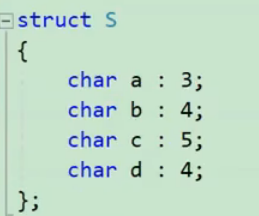
前面abc三个够用了，但是最后一个d不够用，于是又开辟4byte（这里不知道的是d有没有用上四个字节用剩下的bit），

于是此位段的大小为8

注意：冒号后面的数字是不能超过前面的这个类型的



本来只有8个bit的char怎么处理你的9bit？

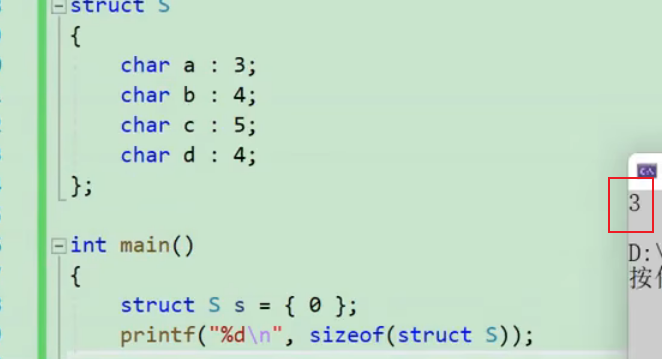


用此结构体来测一下他到底是怎么内存分配的

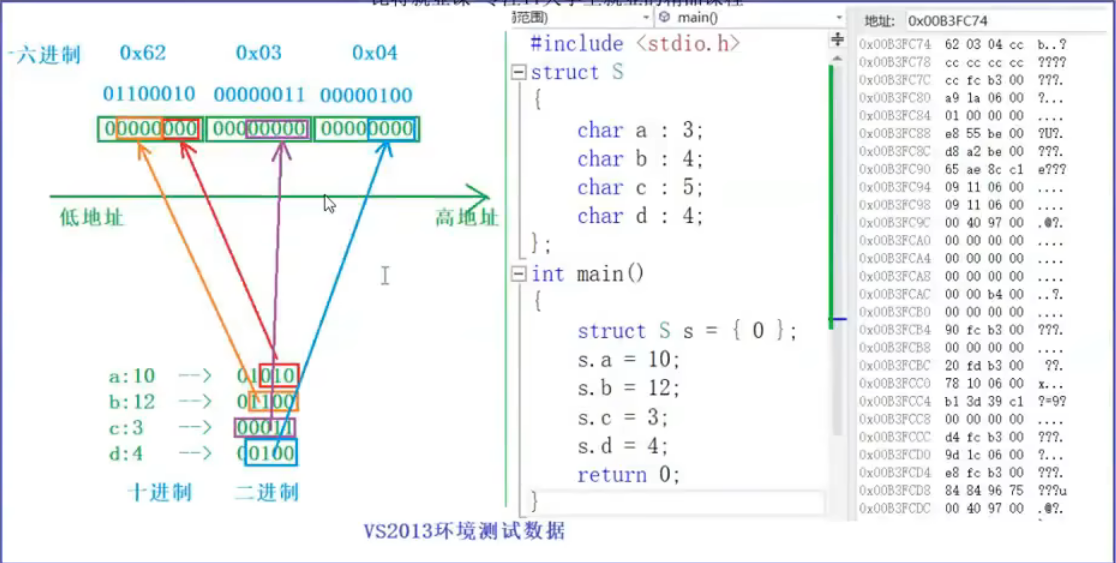
如果是2 说明他会用上一次开辟的8bit

如果是3 说明他会将上一次剩下的bit浪费

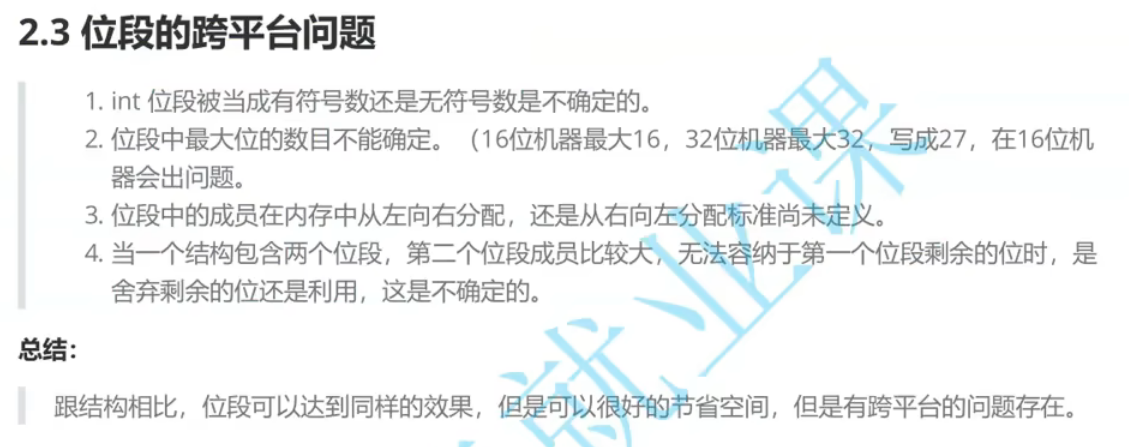
那么实际上

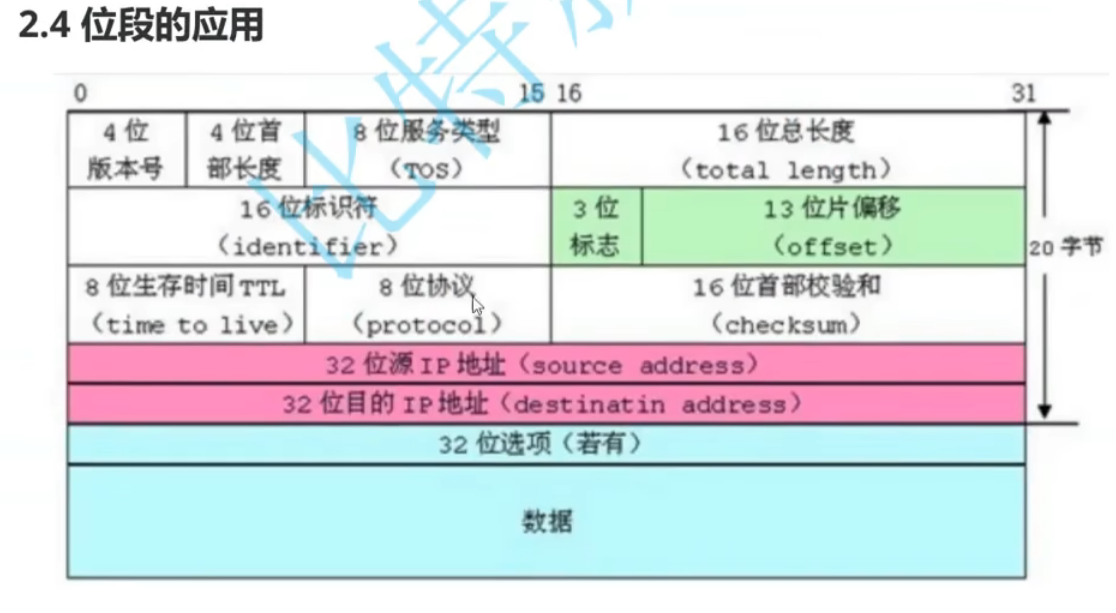


也确实是浪费掉了，说明位段并不会使用上一次开辟所剩下的空间。（只是在vs编译器下是这样的，其他ide并不确定）



也是有点没懂这个内存分配，只知道是从右往左存放，在一个字节内低地址向高地址存放，不够补零，多了截断。注意：这里没有大小端的顺序问题，大小端是以字节序去排序的，而这里是在一个字节里去存放，就是从低位向高位存放。（只在vs编译器下是这样分配的）





这是网络数据包格式

接下来去实现通讯录啦~