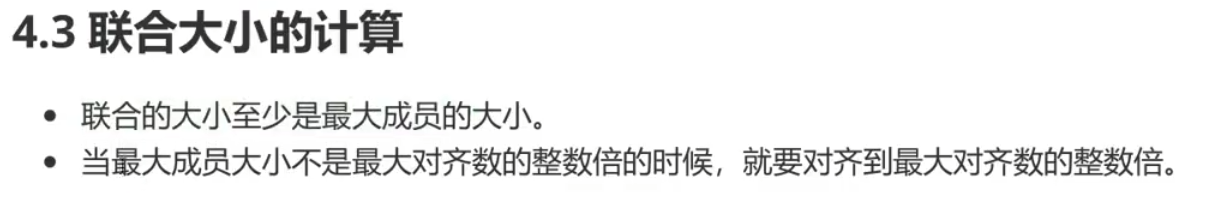
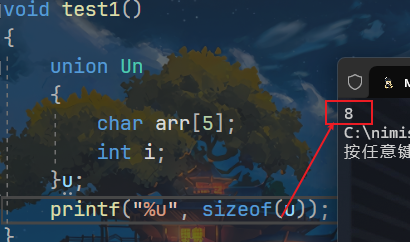
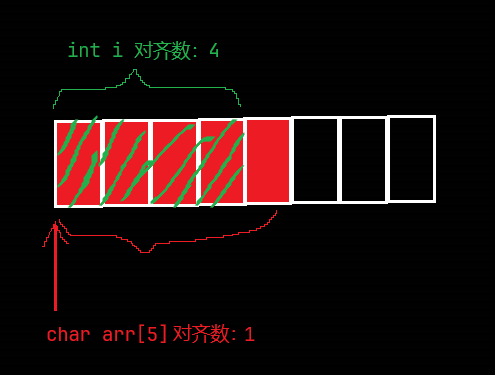
联合也存在内存对齐





虽然这个char arr[5]和i确实是共用一块内存空间的。

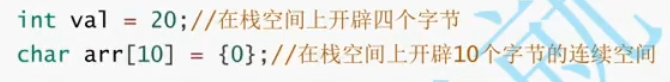
但是因为



没对齐之前，分配的空间为5，最大对齐数为4 ，总的大小应该为最大对齐数的整数倍，因此为8。 但是空间上char和int还是共用的。

# 动态内存管理

我们传统的内存开辟方式：

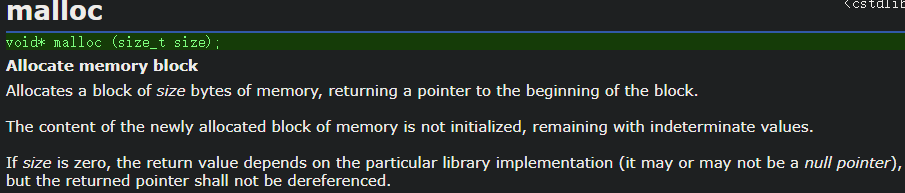


但是上述开辟的方式有两个特点

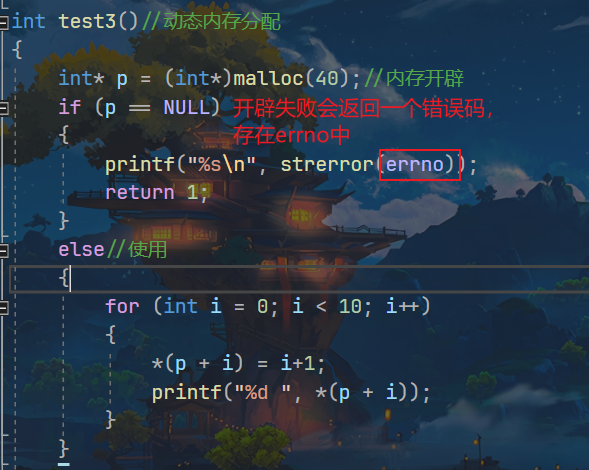
1.空间开辟大小是固定的。

2.数组在申明的时候，必须指定数组的长度，它所需要的内存在编译时分配

Malloc:memory allocation 内存分配<stdlib.h>



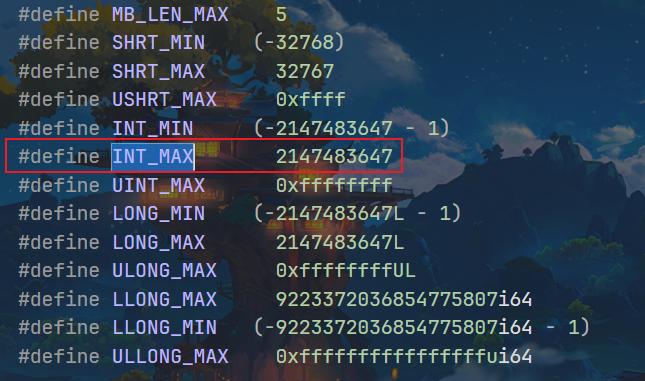
分配一块内存空间，大小为size字节，返回这块内存空间的起始地址。如果开辟失败的话（你让malloc开辟的空间大于你的内存了）就会返回空指针NULL。（可以用strerror来接收这个错误码）



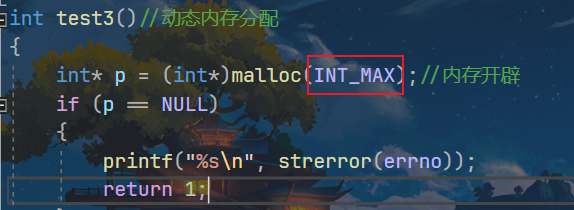
C语言有个历史习惯，在主函数int main 中，如果代码成功运行，最后返回值默认为0，若哪里报错一般都让他返回其他数字。

我们让malloc报错一下

在C语言中



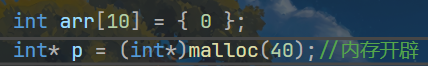
将INT\_MAX定义为2的21次方(32位系统下的int最大值)



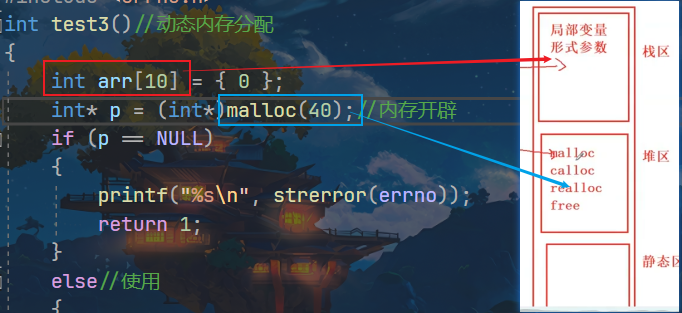
环境换成x86 试一下



完美报错哦~



这俩申请空间的有什么差别呢？



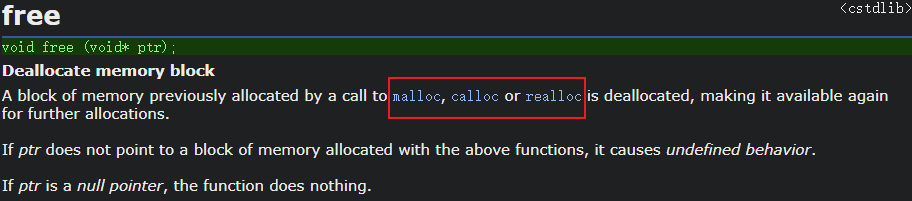
前者申请的临时变量存放于栈区；后者用malloc申请的空间位于堆区。在堆区申请的空间大小是能够调整的，但是栈区的空间大小是不能变的。

malloc对应的是free释放空间，若我们没有主动free空间的话，当程序关闭后，系统会自动回收内存空间的。

## 内存泄漏

当我们申请了一块内存后，用完之后并没有free还回去，这时候系统也没有回收，那这块内存谁都用不了，就丢在那边占空间，就是内存丢失了。

Free



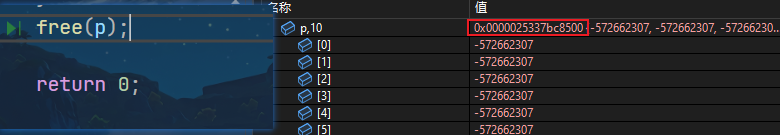
释放先前用malloc，calloc，realloc所开辟的内存空间

如果传给他的指针并不是上述三个函数所开辟的起始地址，那么(it cause undefined behavior？)它执行的行为在标准中是没有定义的，取决于编译器。

如果传给free的是空指针，那么free就does nothing



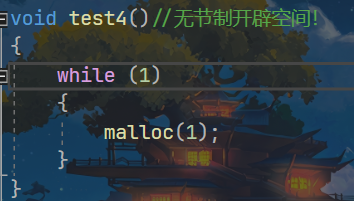
还未释放内存前p指向的地址

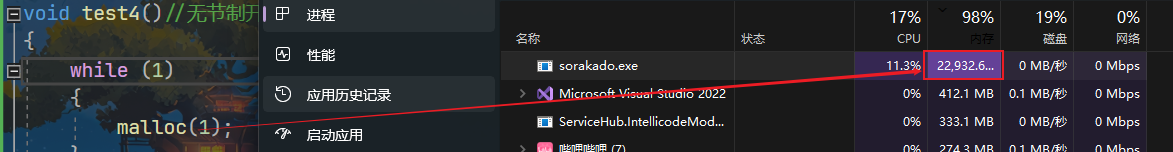


释放后p指向的地址，可见它并没有发生改变；但是p所指向的地址的那块空间已经被系统回收，我们并不能使用；如果不将p赋为空指针的话，p就成为了野指针，也是相当的危险啊



因此，这两步操作是必须的

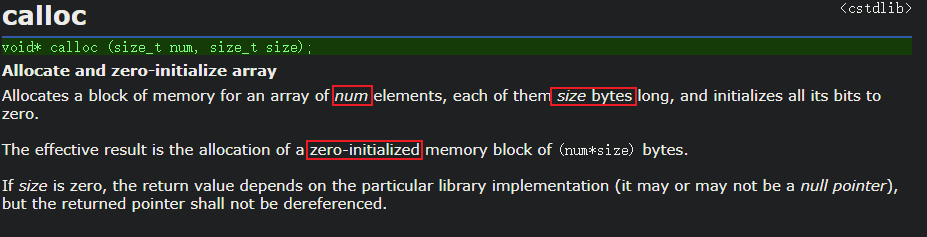


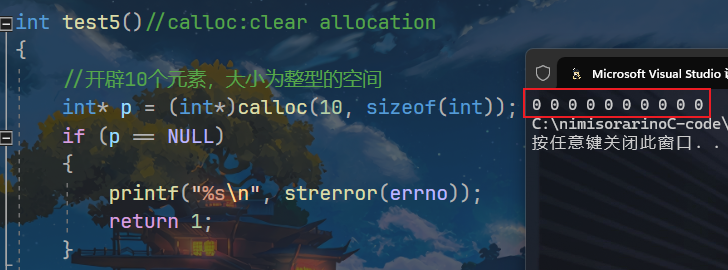


现在电脑比较聪明，不会让你无节制的开辟空间，也是成功防止死机了。

## Calloc: Clear allocation

动态内存分配并清零(clear)





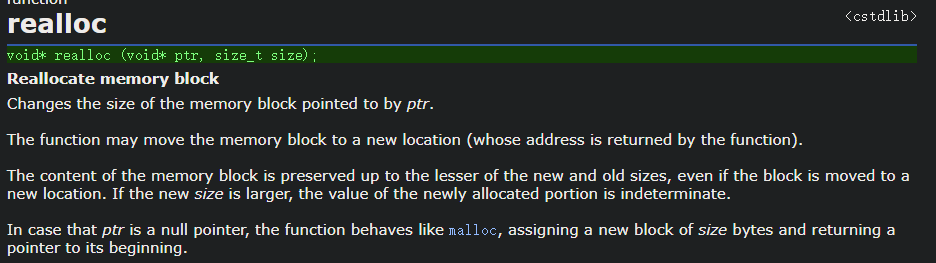
可以看见，我们没做任何事，他自己就初始化了。

可以理解为calloc=malloc+memset/memmove

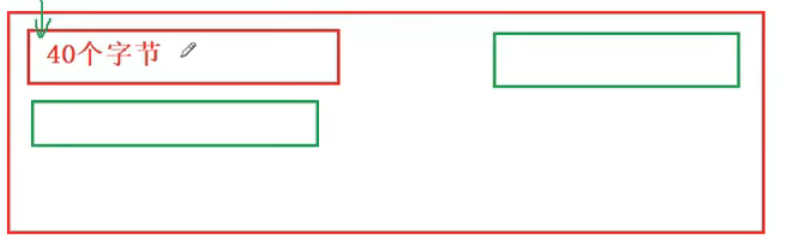
## realloc :Reallocate memory block

realloc函数的出现让动态内存管理更加灵活。

有时会我们发现过去申请的空间太小了，有时候我们又会觉得申请的空间过大了，那为了合理的时候内存，我们一定会对内存的大小做灵活的调整。那realloc函数就可以做到对动态开辟内存大小的调整。



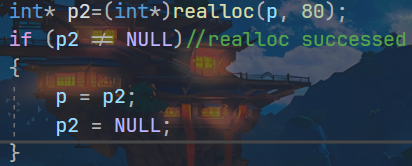
第一个参数放的是要调整内存空间的地址，第二个传的是调整后的总大小。



因为此块内存空面的后的高地址部分可能已经被占用了，因此当你增加内存空间的时候，可能会重新分配一块空间给你，返回的地址也可能会发生改变，旧的地址它会给你free。



在重新分配一块更大的空间给你时，可能会申请失败，realloc会返回空指针，那这时候如果你用p接收的话，就会将原有p指向的旧地址也置零了……所以应用新的指针来接收。记得转换类型



等判断新指针不是空指针后，再将新指针赋予原有的p。

如果传一个NULL的话，那realloc的功能就和malloc一样

realloc(NULL,40)等价于malloc(40)

若传0给realloc的话，那就相当于free一样释放内存，并返回NULL。

当我们很频繁的用malloc/calloc/realloc让系统向堆区申请内存空间时，很可能会让内存空间碎片化，而内存池就很好的解决了这个问题(少女施工中)