字符串旋转结果

题目内容:

写一个函数，判断一个字符串是否为另外一个字符串旋转之后的字符串,

例如:给定S1=AABCD和s2=BCDAA，返回1给定s1=abcd和s2=ACBD，返回0

AABCD左旋一个字符得到ABCDA

AABCD左旋两个字符得到BCDAA

AABCD右旋一个字符得到DAABC

利用库函数的简洁实现方法：

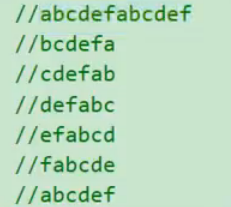
我们注意到，



如果str1自己追加自己后



我们发现 这个主串拥有abcdef左旋后的所有子串



因此，我们只需要先将str1自加

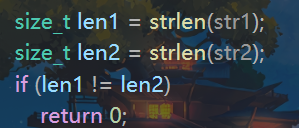


然后再用strstr寻找有无str2子串



没有的话就会返回空指针

当然，子串的长度要和一开始的str1相等，不然根本不用寻找了



int is\_left\_move(char\* str1, const char\* str2)

{

size\_t len1 = strlen(str1);

size\_t len2 = strlen(str2);

if (len1 != len2)

return 0;

strncat(str1, str1, len1);

char\* ret=strstr(str1, str2);

if (ret == NULL)

return 0;

else

return 1;

}

void test2()

{

char str1[20] = "abcdefabcdef";

char str2[] = "cdefab";

int ret=is\_left\_move(str1, str2);

if (ret == 1)

printf("是\n");

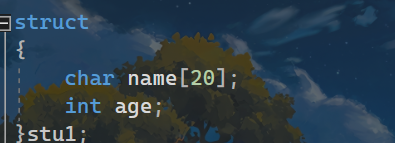
else

printf("否\n");

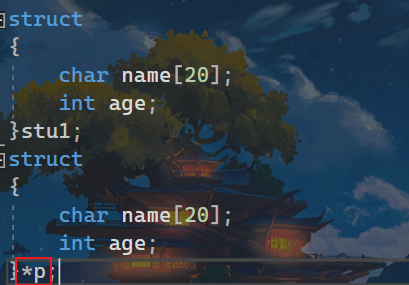
}

# 自定义类型：结构体 枚举 联合

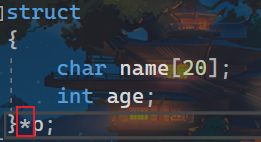
匿名结构体类型：



这种匿名结构体只能用一次 需在自定义类型的时候就在大括号后面创建，之后就不能使用了。（听说嵌入式用的很多）



若是分开匿名自定义一个结构体类型，注意



这个\*是和前面的struct连在一起的，表明这个p是一个struct\*型的指针

尽管上面的匿名定义struct的成员变量一致，但若



编译器编译时报的警告为：



说明站在编译器的角度上看，p的类型和&stu1的类型并不一致

## 结构体自引用

数据结构：数据在内存中的存储结构

树形：

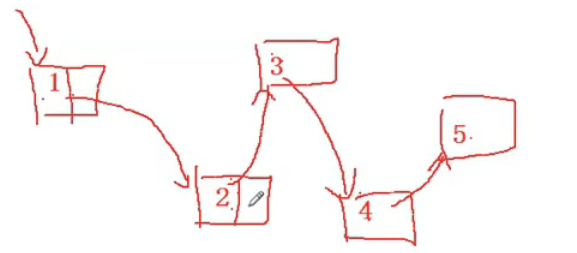
二叉树

线形：

顺序表

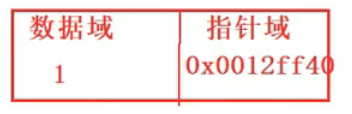


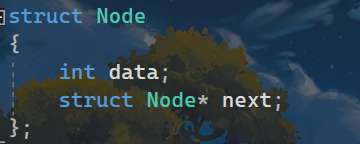
链表



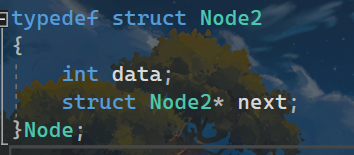
数据在内存中随机存储，但每个数据节点都包含着下一个节点的信息（地址），我们称之为这种存储方式为链表。

这种时候，我们将每个节点分成了两个区域

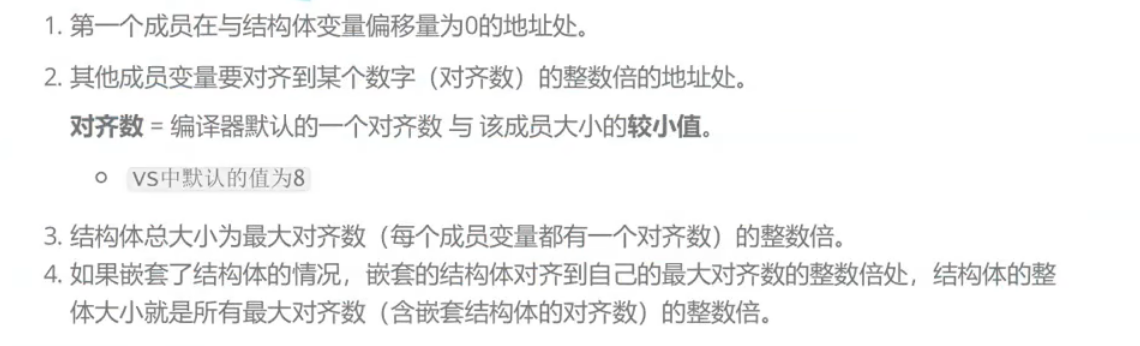




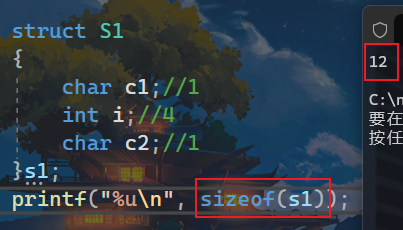
如果感觉每次定义的时候都要写struct Node麻烦，就可以用typedef定义一手



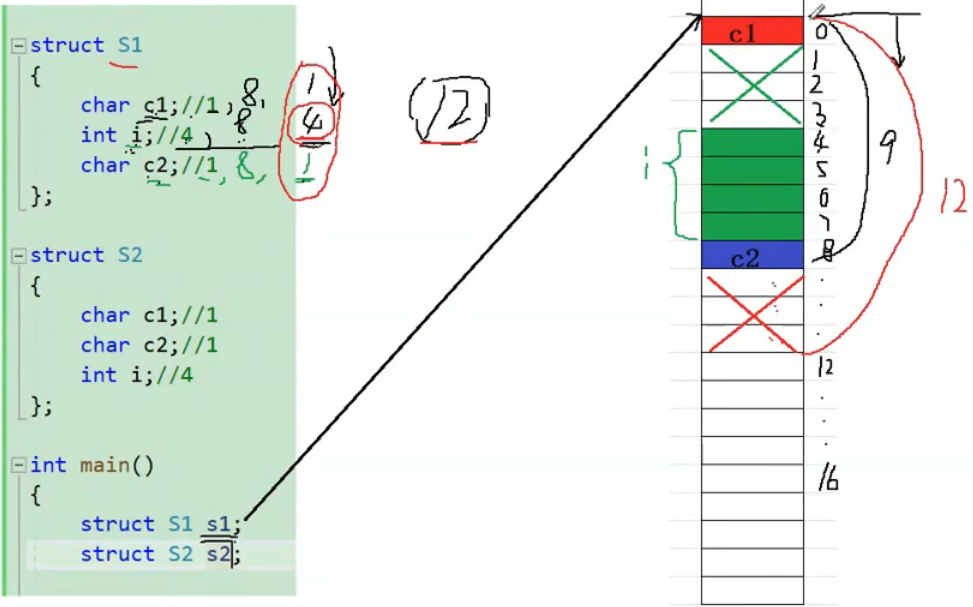
## 结构体内存对齐



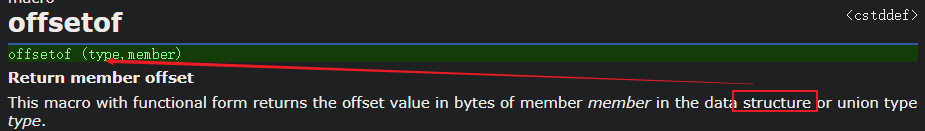
注：若其他编译器上没有默认对齐数（gcc），那么对齐数就是成员变量本身的大小



偏移量：内存中（以字节为单位）相对于起始位置的偏移量。

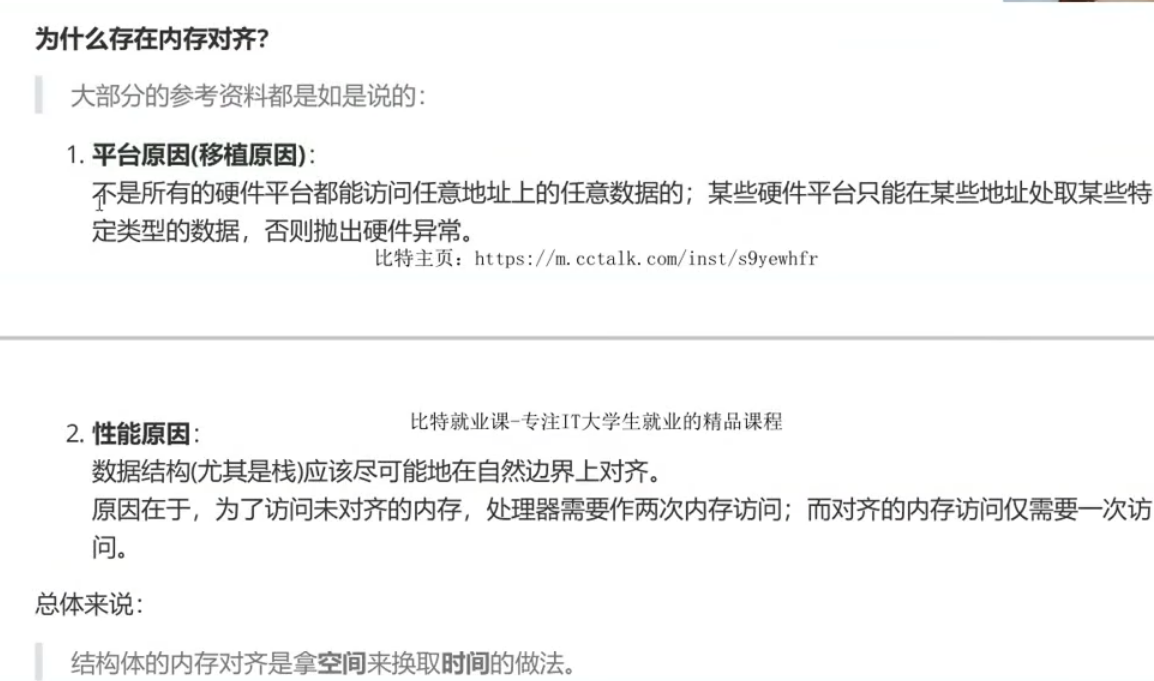


offsetof

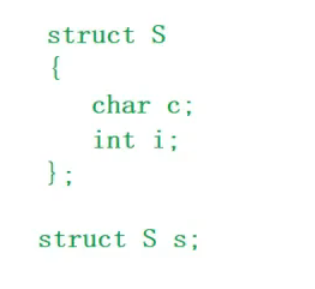


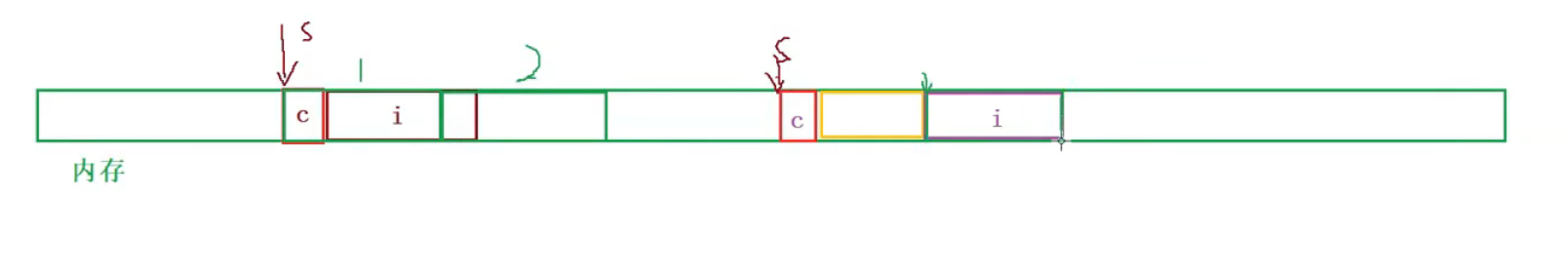


type填的是自定义的结构体类型，第二个传的是自定义结构体类型的成员变量，返回的是此成员变量对于此结构体变量的偏移量。



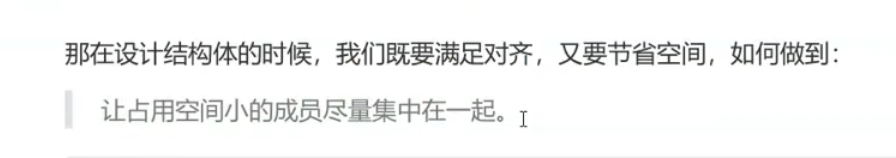
在32位机器中，处理器能够一次读取32位，也就是四个字节





如果没有对齐而是紧挨着存储的话，那处理器需要读两次才能将i数据读完，但是如果内存对齐后，读完c后，处理器可以直接跳过中间空着的部分，直接一次读完i

因此 设计成员变量时：



## 修改默认对齐数



void test8()//修改对齐数

{

struct S1

{

char c1;

int i;

char c2;

};

printf("未修改对齐数时（vs默认8）struct S1大小：%d\n", sizeof(struct S1));

#pragma pack(4)

struct S2

{

char c1;

int i;

char c2;

};

#pragma pack()

printf("修改对齐数为4时（vs默认8）struct S1大小：%d\n", sizeof(struct S2));

#pragma pack(1)

struct S3

{

char c1;

int i;

char c2;

};

#pragma pack()

printf("未进行内存对齐时（对齐数为1时）struct S1大小：%d\n", sizeof(struct S3));

}

一般对齐数写的是2的某次方

## 结构体传参