C_C 中结构体(struct)知识点强化（二） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/136/2021_2022_C_C___E4_B8_AD_E7_BB_c97_136114.htm 程序种有两个组成部分 test *create() 和 void showl(test *head) 这两个函数，create是用来创建链表的，showl是用来显示链表的。 create函数的返回类型是一个结构体指针，在程序调用的时候我们用了showl(create()).，而不用引用的目的原因是引导指针是一个全局指针变量，我们不能在showl()内改变它，因为showl()函数内有一个移动操作head=head->next.，如果是引用的话我们就破坏了head指针的位置，以至于我们再也无法找会首地址的位置了。 下面我们来分解整个程序，以一个初学者的思想来思考整个程序，由浅入深的逐步解释。 首先，我们写这个程序，要考虑到由于是一个链表结构，我们不可能知道它的大小到底是多大，这个问题我们可以用动态开辟堆内存来解决，因为堆内存在程序结束前始终是有效的，不受函数栈空间生命期的限制，但要注意的是我们必须有一个指针变量来存储这一链状结构的进入地址，而在函数内部来建立这一指针变量显然是不合适的，因为函数一旦退出，这个指针变量也随之失效，所以我们在程序的开始声明了一个全局指针变量。来源：www.examda.com test *head.//创建一个全局的引导进入链表的指针 好解决了这两个问题，我们接下去思考 有输入就必然有输出，由于输出函数和输入函数是相对独立的，为了不断测试程序的正确性好调试我们先写好输出函数和main函数捏的调用，创建函数我们先约定好名为create。 我们先写出如下的代码： #include using namespace std. struct test {

char name[10]. float socre. test *next. }. test *head.//创建一个全局的引导进入链表的指针 test *create() { return head.//返回链首指针 } void showl(test *head) { cout while(head)//以内存指向为null为条件循环显示先前输入的内容 { cout head=head->next. } } void main() { showl(create()). cin.get(). cin.get(). } 程序写到这里，基本形态已经出来，输入和调用我们已经有了。 