计算机二级辅导：链表的c语言实现 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/166/2021_2022__E8_AE_A1_E7_AE_97_E6_9C_BA_E4_c97_166367.htm 准备：动态内存分配 一、为什么用动态内存分配 但我们未学习链表的时候，如果要存储数量比较多的同类型或同结构的数据的时候，总是使用一个数组。比如说我们要存储一个班级学生的某科分数，总是定义一个float型（存在0.5分）数组： float score[30]. 但是，在使用数组的时候，总有一个问题困扰着我们：数组应该有多大？ 在很多的情况下，你并不能确定要使用多大的数组，比如上例，你可能并不知道该班级的学生的人数，那么你就要把数组定义得足够大。这样，你的程序在运行时就申请了固定大小的你认为足够大的内存空间。即使你知道该班级的学生数，但是如果因为某种特殊原因人数有增加或者减少，你又必须重新去修改程序，扩大数组的存储范围。这种分配固定大小的内存分配方法称之为静态内存分配。但是这种内存分配的方法存在比较严重的缺陷，特别是处理某些问题时：在大多数情况下会浪费大量的内存空间，在少数情况下，当你定义的数组不够大时，可能引起下标越界错误，甚至导致严重后果。 那么有没有其它的方法来解决这样的外呢体呢？有，那就是动态内存分配。 所谓动态内存分配就是指在程序执行的过程中动态地分配或者回收存储空间的分配内存的方法。动态内存分配不象数组等静态内存分配方法那样需要预先分配存储空间，而是由系统根据程序的需要即时分配，且分配的大小就是程序要求的大小。从以上动、静态内存分配比较可以知道动态内存分配相对于景泰内存分配的特

点： 1、不需要预先分配存储空间； 2、分配的空间可以根据程序的需要扩大或缩小。 二、如何实现动态内存分配及其管理 要实现根据程序的需要动态分配存储空间，就必须用到以下几个函数 1、malloc函数 malloc函数的原型为： void *malloc(unsigned int size) 其作用是在内存的动态存储区中分配一个长度为size的连续空间。其参数是一个无符号整形数，返回值是一个指向所分配的连续存储域的起始地址的指针。还有一点必须注意的是，当函数未能成功分配存储空间（如内存不足）就会返回一个NULL指针。所以在调用该函数时应该检测返回值是否为NULL并执行相应的操作。 下例是一个动态分配的程序： #include #include main() { int count,*array. /*count是一个计数器，array是一个整型指针，也可以理解为指向一个整型数组的首地址*/ if((array(int *) malloc(10*sizeof(int)))==NULL) { printf("不能成功分配存储空间。"). exit(1). } for (count=0.count〈10.count ) /*给数组赋值*/ array[count]=count. for(count=0.count〈10.count ) /*打印数组元素*/ printf("-",array[count]). } 上例中动态分配了10个整型存储区域，然后进行赋值并打印。例中if((array(int *) malloc(10*sizeof(int)))==NULL)语句可以分为以下几步： 1）分配10个整型的连续存储空间，并返回一个指向其起始地址的整型指针 2）把此整型指针地址赋给array 3）检测返回值是否为NULL 2、free函数 由于内存区域总是有限的，不能不限制地分配下去，而且一个程序要尽量节省资源，所以当所分配的内存区域不用时，就要释放它，以便其它的变量或者程序使用。这时我们就要用到free函数。 其函数原型是： void free(void *p) 作用是释放指针p所指向的内存区。 其参数p必须

是先前调用malloc函数或calloc函数（另一个动态分配存储区域的函数）时返回的指针。给free函数传递其它的值很可能造成死机或其它灾难性的后果。 注意：这里重要的是指针的值，而不是用来申请动态内存的指针本身。例： int *p1,*p2. p1=malloc(10*sizeof(int)). p2=p1. …… free(p2) /*或者free(p2)*/ malloc返回值赋给p1，又把p1的值赋给p2，所以此时p1，p2都可作为free函数的参数。 malloc函数是对存储区域进行分配的。 free函数是释放已经不用的内存区域的。 所以由这两个函数就可以实现对内存区域进行动态分配并进行简单的管理了。一、单链表的建立有了动态内存分配的基础，要实现链表就不难了。所谓链表，就是用一组任意的存储单元存储线性表元素的一种数据结构。链表又分为单链表、双向链表和循环链表等。我们先讲讲单链表。所谓单链表，是指数据接点是单向排列的。一个单链表结点，其结构类型分为两部分：1、数据域：用来存储本身数据2、链域或称为指针域：用来存储下一个结点地址或者说指向其直接后继的指针。例：typedef struct node{char name[20].struct node *link.}stud.这样就定义了一个单链表的结构，其中char name[20]是一个用来存储姓名的字符型数组，指针*link是一个用来存储其直接后继的指针。定义好了链表的结构之后，只要在程序运行的时候爱数据域中存储适当的数据，如有后继结点，则把链域指向其直接后继，若没有，则置为NULL。下面就来看一个建立带表头（若未说明，以下所指链表均带表头）的单链表的完整程序。#include #include /*包含动态内存分配函数的头文件*/#define N 10 /*N为人数*/ typedef struct node{char name[20].struct node *link.}stud. stud * creat(int n) /*建立单链表

的函数，形参n为人数*/{stud *p,*h,*s. /* *h保存表头结点的指针，*p指向当前结点的前一个结点，*s指向当前结点*/int i. /*计数器*/if((h=(stud *)malloc(sizeof(stud)))==NULL) /*分配空间并检测*/{printf("不能分配内存空间!").exit(0).}h->name[0]=\0. /*把表头结点的数据域置空*/h->link=NULL. /*把表头结点的链域置空*/p=h. /*p指向表头结点*/for(i=0.i{if((s= (stud *) malloc(sizeof(stud)))==NULL) /*分配新存储空间并检测*/{printf("不能分配内存空间!").exit(0).}p->link=s. /*把s的地址赋给p所指向的结点的链域，这样就把p和s所指向的结点连接起来了*/printf("请输入第%d个人的姓名",i 1).scanf("%s",s->name). /*在当前结点s的数据域中存储姓名*/s->link=NULL.p=s.}return(h).}main(){int number. /*保存人数的变量*/stud *head. /*head是保存单链表的表头结点地址的指针*/number=N.head=creat(number). /*把所新建的单链表表头地址赋给head*/}这样就写好了一个可以建立包含N个人姓名的单链表了。写动态内存分配的程序应注意，请尽量对分配是否成功进行检测。 