实验八：结构体和共用体

批注 [管1]: 10 分

韦璐 PB16000702

一．学习重点

结构体类型的定义

结构体变量的定义，存储特点与引用

结构体数组与应用

指向结构体变量或结构体数组的指针变量的定义和引用

结构体与函数（数据传递）

动态内存分配函数 malloc，calloc，free 的使用

单向链表的概念和结点的定义

链表应用（单向链表的建立，插入，删除与输出）

共用体和枚举类型的定义

共用体和枚举类型变量的定义和赋值

类型定义

二．要点综述

1．结构体类型

结构体（结构变量）是不同类型的数据元素的有序集合。相当于其他高级语言的记录型，用以组织和处理复杂的数据（客观世界的实体）。

结构体类型定义的一般形式：

```
Struct 结构体名{
    类型标识符 1    成员名 1;
    类型标识符 2    成员名 2;
    ……
    类型标识符 n    成员名 n;
};
```

其中，struct 是关键字，是结构体类型的标志；结构体名是由用户定义的标识符。Struct 和结构体名两者构成结构体特定类型标识符。结构体类型的组成成分称为成员，成员说明表列中的成员名应符合永和标识符的命名规则，即与变量相同。

（1） 类型定义只是说明了一个客观实体相应的属性描述：包括属性个数，所属类型，存储顺序，所占存储空间的大小等。

（2） 结构体成员可以是任何基本数据类型的变量，如 int，char，float 和 double 类型等，这些成员的类型可以相同，但往往是不同的。结构成员也可以是数组，指针类型的变量。

（3） 有定义可以知道，相应的结构变量的存储单元的大小为各成员所需容量的总和。

（4） 结构体类型可以嵌套定义，即允许一个结构中的一个或多个成员是其他结构类型的变量。嵌套的子结构类型及其变量的定义可以放在外层结构的成员表中，也可以采用把各个子结构类型单独定义，而在结构成员表中仅出现相应结构类型的变量说明。

（5） 结构体类型不允许递归定义，

（6） 在同一个结构中成员名不可相同，但不同结构中的成员名可以相同，并且结构的成员名可以和程序中的变量名相同。

2. 结构体变量的定义和初始化

结构体类型变量要先定义后使用

（1） 结构体类型变量定义的三种常用形式

先定义结构体类型再定义相应变量

在定义结构体类型的同时定义变量

利用无名结构类型定义变量，即直接定义（省略结构体名）结构体类型变量

（2） 结构变量的初始化同数组初始化，传统的 C 规定，只有结构变量为全局变量或静态的局部结构变量才能初始化，不能对动态局部结构变量进行初始化。心得标准中可以，但是实在相应函数被调用执行时进行。

结构变量的初始化就是在定义结构体变量时，将结构体变量各成员的初值顺序的组成初值表，并用逗号分隔放在一对大括号中。

对结构变量的初始化的一般类型:

结构类型 结构变量名={初值表};

3. 结构成员的引用

对结构成员的引用可以分为对结构体变量各成员的引用和对整个结构体变量的引用。

（1） 表达式中只能引用结构变量成员，不能整体引用结构变量。

对结构变量中的成员引用的一般形式:

结构体变量名.成员名

1）“.”为成员运算符，整个引用格式相当于一个变量

2）如果结构体变量为嵌套结构，就应使用成员运算符层层结合，一直找到最低一级的成员，即只能对最低一级的成员进行赋值或存取运算

3）结构体成员都属于某种数据类型，可以像同类变量一样进行各种操作

4）可以引用成员的地址，也可以引用结构变量的地址。

（2） 直接使用结构变量呢名，是对整个结构体变量能的引用。相同类型的结构体变量之间可以进行整体赋值，此时允许直接使用结构变量名。其他操作如输入输出等必须通过引用结构体变量的成员进行相应的操作。

4. 结构体数组

是一个数组，元素的数据类型为结构类型。

（1） 定义结构体数组的方法和定义其它类型的数组的方法类似，数据不同而已。

（2） 对结构体数组的初始化时，可以分局结构体数组及其结构体数组元素的成员存储，将数组每个元素的初值，(元素是一个结构变量，其初值的写法参见结构变量的初始化) 都放在一堆大括号中。如果给出了全部元素的初值，则数组的长度可以不置顶，由系统根据初值的数目来确定数组的长度。

（3） 对结构体数组的引用一般是对数组元素成员的引用。只要遵循对数组元素的引用规则核对结构体变量成员的引用规则即可。结构体数组名【下标】。成员名

5. 指向结构体变量的指针变量

指向结构体变量的指针变量的定义和赋值与一般指针变量的定义和赋值相同，只是其所指存储单元的数据类型是结构体类型，它只能富裕定义时规定的所知类型的结构变量或结构体结构体数组元素的地址以及同类型的指针变量之。

若一个结构体指针变量指向一个结构体变量，则对该结构体变量成员的饮用方式有如下三种形式：1，根据结构体变量名使用成员运算符引用成员。

2，利用指向该结构体的指针变量，使用成员运算符引用成员（*结构体指针变量名）.成员名

3，利用指向该结构体的指针变量，使用指向成员运算符引用成员。

注：在 C 语言中，结构的成员运算符，指向成员运算符，表示函数的运算符，表示数组下标的运算符都具有最高的优先级。

设有 struct date{ int year ； int month ； int day }d【5】，*pd；pd=d；则有:

. pd->year 表示 pd 所指结构数组元素成员 year 的值

.++pd->year 等价于++(pd->year)，是把 pd 所指结构数组元素成员 year 的值加 1，并不是

pd 的地址加 1
Pd->year++等价于<pd->year>++，是把 pd 所指结构数组元素成员 year 的值加 1,，属于后缀表达式
（++pd）->year 是把 pd 的值先加 1，使它指向下一个（后继）结构数组元素，然后取得 pd 所新指向结构数组元素成员的 值
Pd++->year 是先取出 pd 当前所指向结构数组元素成员 year 的值，然后 pd 加 1，指向下一个（后继）结构数组元素
如果 p 是指向结构变量的指针，xp 是该结构中的一个指针变量，则有：
p->xp 等价于（p->xp），表示间接访问 p 所指结构变量中 xp 所指的对象。
p->xp++等价于((p->xp)++)，表示先简介访问 p 所指结构变量中 xp 所指对象的内容，然后该成员 xp 的值加 1.
*p++->xp 表示先简介访问 p 所指结构变量中 xp 所指对象的内容，然后 p 加 1，指向下一个（后缀）结构变量
实际使用中类似的表达式还很多，关键是注意掌握运算符的优先级，结合性以及++或者—前缀和后追星是的功能差别
6. 指向结构体数组的指针变量
指针变量也可以指向结构体类型的数组，此时它所指的类型应该与结构体数组的类型一致，可以将数组名或其数组元素地址赋给该指针，这样通过该指针就可以引用所指结构体数组的元素，其中利用指廊成员运算符的引用形式较为常用。
若指针 p 是指向结构的指针，则对 p 所施加的运算军事按结构体的实际长度计算的，因此 p++总是指向结构数组的下一个元素，而不是指向某一元素中的某一成员，
7. 结构体数据做函数参数
（1）早起的 C 语言版本中不允许用结构变量作为函数的参数，解决的办法是
用结构变量的成员作参数，属于传值方式，具体应用规则与同类型变量相同
利用取地址运算符取结构变量的地址，以指向结构体变量(或结构体数组)的指针作为实参，将结构体变量的地址传递给形参，属于产地直的方式
函数的返回值也可以是一个只想结构的指针
根据标准，可以使用指向结构体变量的指针作为函数的参数，也允许直接使用结构体变量作为函数的参数，但是这个是传值的方式，系统将实参结构变量相应的内存单元内容全部顺序产地给形参，即传递的是结构体变量全部成员的值。形参也要是同类型的结构变量
指向结构体的指针做函数的参数的时候，函数的实参和形参是指向同一种结构体类型的指针变量，或者同一种结构体类型的地址，地址传递的规则不变，即形参和实参共享存储空间，当形参钟祥的结构体变量的值发生变化后，对应的实参指向的结构体变量的值也同样发生变化。
8. 动态存储分配
在 C 语言中，数组和结构的大小都必须是固定的，不支持动态数组和动态结构之类的类型，但在实际编程中，往往会遇到这种情况，所需要的内存空间的大小无法预先确定下来，要根据实际得到数据点多少来决定，解决这类问题的办法是，动态分配存储空间，动态分配存储空间必须通过指针，并利用 C 编译程序提供的内存动态分配函数实现
（1） 常用到的内存管理函数有：malloc（）calloc（）和 free（）
Malloc
Void *malloc （unsigned size)；
功能是分配 size 字节的连续存储区，返回值为所分配内存去的起始地址，如不成功，则返回 0.

例如在内存中需要开辟一个存放 4 个 double 型数据的 32 个自洁空间。并让定义为 double *pd；的指针指向它，则可以采用如下形式的语句实现

Pd=（double *）malloc（4*sizeof（double））；

Calloc 函数

Void *calloc（unsigned n，unsigned size）；

功能：分配一个 n 个数据项的内存连续空间，每个数据项的大小为 size。返回值为所分配内存单元的起始地址，并将所分配的各个单元全部清零。如不成功，则返回 0.例如，在内存中需要开辟一个存放 4 个 double 型数据的空间。并让定义为 double*pd 的指针指向它，可以

Pd=（double *）calloc（4，sizeof（double））；

Free

Void free（void *p）；

功能：释放指针变量 p 所指从由动态函数分配的存储空间，返回值无。

（2） 说明：ANSI 标准要求动态分配系统返回 void 指针。Void 指针具有一般性，它可以指向任何类型的数据，但目前大多数 C 编译所提供的这类函数返回的是 cahr 指针。无论上述情况中的哪一种，都需要用强制类型转换的方法将它们转换为所需要的类型。

9 . 链表

（1）链表的基本特征：

1. 有一个头指针变量，一般取名为 head，用来指向链表的第一个元素。

2 . 链表中的每一个元素称为结点，结点包含两部分内容，一是实际的数据信息，二十指向下一个结点的指针。

链表的最后一个称为表尾结点，它不指向其他节点，因此将它的指针部分置位 null，标志链表的结束。

链表中，一个结点的后续结点是由该节点包含的指针指出的，，与后续节点在内存中的位置无关，也就是链表的各个节点在内存中可以不连续存放，但如果要查找该链表中某个节点，则必须由头指针所指的第一个节点开始顺序查找。

（3） 链表与结构体数组的异同

相似之处

都是由若干相同类型的结构体变量组成

结构体变量间有一定顺序关系

连标语结构提升胡祖的区别

结构体数组中各元素是连续存放的，而链表中的节点可以不连续存放，

结构体数组元素可通过下表运算或相应指针变量的移动进行顺序或随机访问，而链表中节点不便随机访问，只能从头

结构体素组在定义时就确定元素的个数，不能动态声场，而链表的长度往往是不确定的，根据问题求解过程中的实际需要，动态地创建节点并分配存储空间。

单项链表的基本操作主要有链表的建立，输出，插入，删除

（4） 链表的建立

在建立链表之前必须定义该链表要使用的结构体类型，并要定义一个头指针变量 head，由于链表最初是空的，所以头指针变量 head 应初始化为 null，为了在链表上添加新的节点，还要定义一个附加指针变量设为 p1，它所指向对象也是节点的类型。

如果一个新的节点要加入到链表中，可以根据头指针变量 head 是否是 null 分为两种状态，if head 为 null，则表示链表是空的，此时加入到该链表中的新节点称为链表中位移的成员，

若 head 不是 null 则该链表中已经有成员，不论哪种情况，将一个新的成员加入链表的揩油所要完成的动作都是相同的。
调用 malloc 函数，申请一个结点的存储空间，创建一个新的节点实体，结构体变量，将新的节点的 next 指针变量置位 head 指针变量的当前值，如果是心得 链表，即链表为空，那么 head 的当前值为 null，否则，head 的当前值是目前链表中第一个成员的地址。
使 head 指向新成员
在 head 已经初始化的前提下，可以
P1=(STUDENT*)malloc(LEN);//动态申请存储空间
P1->next=head;//新节点的 next 置位 head 的当前值
Head=p1;//使 head 指向新成员上面是表头插入，还可以表尾插入和在链表中查找定位插入算法等。

（5） 从链表中删除结点

从一个链表中删除节点并不是从内存中删除，而是将它从链表中分离出去，即只需要改变链表中节点的关系。利用指针实现从链表中删除结点是很简单的事情，具体的过程取决于被删除结点在链表中处于什么位置，可以归纳为

1. 删除链表中第一个节点，此时只要把 head 指向链表中的第二个结点：p1=head;//就是 p1 指向链表的第一个结点 head=p1->next;//head 指向链表的第二个结点 free（p1）；释放 p1 所指结点的存储空间
2. 删除链表的为节点，此时要通过查找定位链表尾部，并把链表尾部的前驱结点中的 next 置位 null。这个操作要两个辅助的指针变量
   定义两个辅助的指针变量，将其中一个指向链表的第一个结点，查找定位被删为节点尾结点，指针一前一后移动，将表尾结点的前驱结点的 next 置为 null，释，放指向最后一个结点地址处的值的内存空间，然后释放指向最后一个地址的地方存的空间
3. 删除链表其他位置的任一结点，此时通过查找定位被删除结点，并使该结点的前驱结点的 next 指向被删结点之后的结点，定义两个辅助的指针变量，然后第一个指向链表的第一个结点，然后查找定位被删的尾结点，指针一前一后往后移动，被删除点的前面成员的 next 指向被删除节点的后面的结点的开头，释放第一个结点此时所指向的结点的内存空间

12. 类型定义

用来对已经存在的数据类型定义新的类型名（称用户自定义类型）

（1） 基本数据类型，数组类型，结构体类型，指针类型定义成用户自定义类型

基本数据类型的自定义：
Typedef 基本类型名 用户自定义类型名；
数组类型的定义：
Typedef 类型说明符 用户类型名【数组长度】；
结构体类型的自定义：
Typedef struct{
数据类型 1 成员 1；
数据类型 1 成员 1；
……
数据类型 1 成员 1；
}用户自定义类型名；
指针类型定义的自定义：

Typedef 类型说明符 *用户自定义类型名;

（2） 使用用户自定义类型来定义变量，数组，指针。

定义基本数据类型的变量

用户自定义类型名 变量名表列;

定义数组

用户自定义类型名 数组名;

注：数组名后不需要数组下标运算符和制定数组的大小

定义指针变量

用户自定义类型名 指针变量名表列

指针变量名前不需要加“*”

范例一

题目

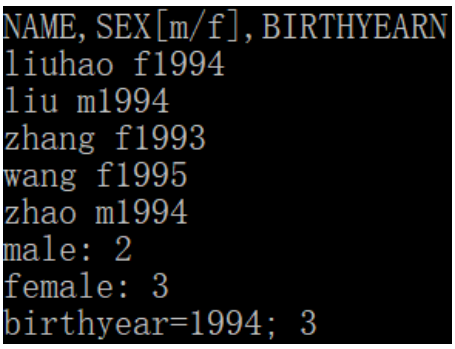
打印 1994 年出生的学生人数，并统计全班男女生各多少

实验代码：

```
#include<stdio.h>
#define N 5
struct student{
    char sex,name[16];
    int birthyear;
};
main(){
    struct student class[N];
    void input(struct student *p,int n);//函数原型说明
    void count(struct student s[], int n,int *c1994,int *cm,int *cf);
    int i,count1994=0,countm=0,countf=0;
    input(class,N);
    count(class,N,&count1994,&countm,&countf);
    printf("male: %d\n",countm);
    printf("female: %d\n",countf);
    printf("birthyear=1994; %d\n",count1994);
}
void input(struct student *p,int n){
    struct student *pend;
    pend=p+n-1;
    printf("NAME,SEX[m/f],BIRTHYEAR\N\n");
    for( ;p<=pend;p++){
        scanf("%s %c%d",p->name,&p->sex,&p->birthyear);
        getchar();
    }
}
void count(struct student s[],int n,int *c1994,int *cm,int *cf){
    int i;
```

```
    for(i=0;i<n;i++){
        if(s[i].birthyear==1994)
        ++*c1994;
        if(s[i].sex=='m')
        ++*cm;
        else
        ++*cf;
    }
}
```

实验结果：



习题

第一题

题目

定义一个图书相关信息的结构体类型和结构变量，其中包括成员书号，书名，作者，出版社和价格；从键盘输入 10 本图书信息，计算并输出这 10 本图书的平均价格。

实验代码：

```
#include <stdio.h>
#include <string.h>
#define NAME_MAX 20
struct book{
int nu;
char name[NAME_MAX];
char author[NAME_MAX];
char publications[NAME_MAX];
int price;
}booklist[3];
void input(struct book *p,int n){
    struct book *pend;
    pend=p+n-1;
    for( ;p<=pend;p++){
        scanf("%d %s %s %s %d",&p->nu,p->name,p->author,p->publications,&p->price);
        getchar();
    }
```

```
}
void count(struct book s[],int n,int *j){
	int i;
	for(i=0;i<n;i++){
		(*j)+=s[i].price;
	}
	(*j)=(*j)/3;
}
main(){
	int i=0;
	input(booklist,3);
	count(booklist,3,&i);
	printf("%d",i);
}
```

实验结果：

```
8 vkwbsvu vukb vsdhzhubd 6
5 cwhaue cbzks vgdb 8
7 cskjvbj vskz.n h 10
8
```

调试分析：无，一次过

实验总结：可以在取值的时候加一个括号，然后就可以进行正常的运算了，然后在使用指针的时候可以全程使用指针，就是将某个数据的地址传过去即可。

第二题

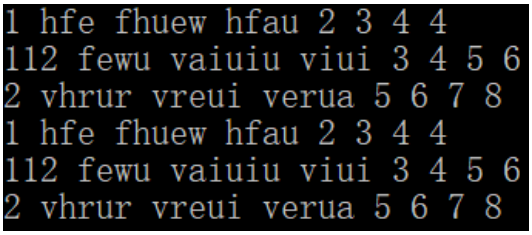
题目：在第一题定义的结构体类型中增加一个成员出版日期，该日是一个嵌套的结构类型变量，其中包括年月日，设计一个输入输出图书信息的函数 read 和 print；并写主函数定义一个 10 个元素的结构数组，分别调用输入输出图书信息的函数输入和输出图书信息。

实验代码：

```
#include <stdio.h>
#include <string.h>
#define NAME_MAX 20
struct book{
int nu;
char name[NAME_MAX];
char author[NAME_MAX];
char publications[NAME_MAX];
struct{
	int nian;
	int yue;
	int ri;
}shijian;
int price;
}booklist[3];
void input(struct book *p,int n){
```

```
    struct book *pend;
    pend=p+n-1;
    for( ;p<=pend;p++){

    scanf("%d %s %s %s %d %d %d %d",&p->nu,p->name,p->author,p->publications,&p->shij
ian.nian,&p->shijian.yue,&p->shijian.ri,&p->price);
        getchar();
    }
}
void shuchu(struct book *p){
    struct book *pend;
     for( ;p<=pend;p++){

    printf("%d %s %s %s %d %d %d %d\n",p->nu,p->name,p->author,p->publications,p->shiji
an.nian,p->shijian.yue,p->shijian.ri,p->price);
     }
}
main(){
    int i=0;
    input(booklist,3);
    shuchu(booklist);
}
```

实验结果：



调试分析：

我们在使用 printf 的时候后面的部分全部是地址，所以对于指向结构体的部分就直接使用不用取值了。

实验总结：

结构体在定义的时候写在结构后面的就相当于属性名字了，在结尾定义的才是这个结构体在引用的时候的指针地址，然后只要在后面加一个方括号就可以定义结构体数组了。

第三题

题目：在第二题的基础上，增加一个按书号递增排序函数 sort，在主函数中调用排序函数后再输出图书信息。

实验代码：

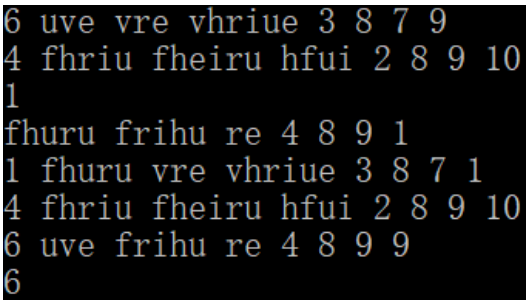
```
#include <stdio.h>
#include <string.h>
#define NAME_MAX 20
```

```
struct book{
int nu;
char name[NAME_MAX];
char author[NAME_MAX];
char publications[NAME_MAX];
struct{
	int nian;
	int yue;
	int ri;
}shijian;
int price;
}booklist[3];
void input(struct book *p,int n){
	struct book *pend;
	pend=p+n-1;
	for( ;p<=pend;p++){

	scanf("%d %s %s %s %d %d %d %d",&p->nu,p->name,p->author,p->publications,&p->shij
ian.nian,&p->shijian.yue,&p->shijian.ri,&p->price);
		getchar();
	}
}
void output(struct book *p,int n){
	struct book *pend;
	pend=p+n-1;
	for( ;p<=pend;p++){

	printf("%d %s %s %s %d %d %d %d\n",p->nu,p->name,p->author,p->publications,p->shiji
an.nian,p->shijian.yue,p->shijian.ri,p->price);
	}
}
void count(struct book s[],int n,int *j){
	int i;
	for(i=0;i<n;i++){
		(*j)+=s[i].price;
	}
	(*j)=(*j)/3;
}
void swap(struct book * a, struct book * b) {

	struct book tmp;

	tmp.nu = a->nu;
	tmp.price = a->price;
```

```
    strcpy(tmp.name, a->name);
    strcpy(tmp.author, a->author);
    strcpy(tmp.publications, a->publications);
    tmp.shijian.nian = a->shijian.nian;
    tmp.shijian.yue = a->shijian.yue;
    tmp.shijian.ri = a->shijian.ri;

    a->nu = b->nu;
    a->price = b->price;
    strcpy(a->name, b->name);
    strcpy(tmp.author, a->author);
    strcpy(tmp.publications, a->publications);
    tmp.shijian.nian = a->shijian.nian;
    tmp.shijian.yue = a->shijian.yue;
    tmp.shijian.ri = a->shijian.ri;

    b->nu = tmp.nu;
    b->price = tmp.price;
    strcpy(b->name, tmp.name);
    strcpy(tmp.author, a->author);
    strcpy(tmp.publications, a->publications);
    tmp.shijian.nian = a->shijian.nian;
    tmp.shijian.yue = a->shijian.yue;
    tmp.shijian.ri = a->shijian.ri;

}
void paixu(struct book *p){
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<(3-i-1);j++)
        {
            if((p[j].nu)>(p[j+1].nu))
            {
                swap(&p[j], &p[j+1]);
            }
        }
    }
}
main(){
    int i=0;
    input(booklist,3);
    count(booklist,3,&i);
    paixu(booklist);
    output(booklist,3);
```

```
    printf("%d",i);
}
```

实验结果：



调试分析：就是在 printf 后面的东西地址和内容可以混用的。

实验总结：无

第六题

题目：13 个人围成一圈，从第 1 个人开始顺序报号 1，2，3。凡报到“3”者退出圈子。编程找到最后留在圈子中的人原来的序号。

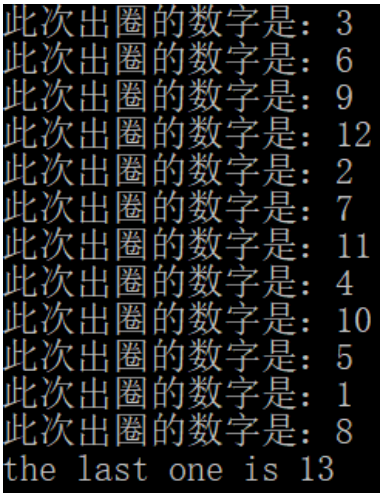
实验代码：

```
#include <stdio.h>

#include <stdlib.h>

#define N 13

struct person

{

    int number;

    int nextp;

}link[N+1];//创建结构体数组

int main(int argc, char *argv[]) {

    int i;

    for(i=1;i<=N;++i)

    {
```

```
        if(i==N)
            link[i].nextp=1; //在数字 13 的邻近点为 1，可以形成一个循环
        else
            link[i].nextp=i+1; //在数字 i 的邻近点为 i+1，依次进行赋值的操作。
            link[i].number=i; //number 即为该数字
    }
    //进行报道，数到三的人退出圈子
    int j=N;
    i=0;
    int count=0;//i 进行清 0 操作
    while(count<N-1)  //注意，此处循环应该是出圈的人数应该小于总的人数。否则的就停止出圈、循环报数
    {
        while(i<3)
        {
            j=link[j].nextp; //必须写在 if（i==3）的前面
            if(link[j].number)
            {
                i++; //如果这个数字没有出圈的话 将其计入在其中
            }
                //如果 j 这个数字出圈的话，就查看他的邻位数字
        }
        if(i==3)
```

```
            {
                printf("此次出圈的数字是：%d\n",link[j].number);//出圈
                link[j].number=0; //  将该数字为 0，代表出圈
                count++; //进行 count++，计数出圈的总人数
                i=0;//i 进行归 0。再次进行循环的出圈的操作
            }
        }
        printf("the last one is "); //输出最后一个数。
        for(i=1;i<=N;i++)//循环遍历
        {
            if(link[i].number)//如果 link[i].number 存在的话，那么其即为最后一个数字
                printf("%d",link[i].number);//输出
        }
        printf("\n");
        return 0;
}
```

实验结果：


此次出圈的数字是：3
此次出圈的数字是：6
此次出圈的数字是：9
此次出圈的数字是：12
此次出圈的数字是：2
此次出圈的数字是：7
此次出圈的数字是：11
此次出圈的数字是：4
此次出圈的数字是：10
此次出圈的数字是：5
此次出圈的数字是：1
此次出圈的数字是：8
the last one is 13

调试分析:

无

结果分析:

无