《数据结构》实验报告二

|  |  |  |
| --- | --- | --- |
| 题目： 约瑟夫环 | | 班级： 通信1601 |
| 学号： 29 |  | 姓名： 郑祖闯 |
| 日期： |  | 程序名： 约瑟夫环问题 |

### 一、上机实验的问题和要求：

约瑟夫环（《数据结构题集》P79 1.2）：约瑟夫（Joseph）问题的一种描述是：编号为1，2，3，…，n的n个人按顺时针方向围坐一圈，每个人持有一个密码（正整数）。一开始任选一个正整数作为报数上限值，从第一个人开始按顺时针方向自1开始顺序报数，报到m时停止报数。报m的人出列，将他的密码作为新的m值，从他在顺时针方向上的下一个人开始重新报数，如此下去，直至所有人全部出列为止。设计一个程序求出出列顺序。

利用单向循环链表存储结构模拟此过程，按照出列的顺序印出各人的编号。

### 二、程序设计的基本思想，原理和算法描述：

（包括程序的结构，数据结构，输入/输出设计，符号名说明等）

此程序实现的方法是建立一个没有表头结点循环链表，然后对循环链表进行相关操作，模拟整个报数及出列过程。

将每个人的信息用一个结点存储，包括三个信息，一是他的编号number, 二是

他持有的密码 code, 三是指向下一结点的指针。其中编号是一开始按照相邻顺序编号

的，密码可以设置，需要输入数值。

typedef struct LNode

{

int number; //编号

int code; //持有密码

struct LNode \*next; //指向下一结点的指针

}LNode,\*LinkList;

主要过程由以下两个函数实现

void CreateList(LinkList &L,int &n); //创建循环链表

void Joseph(LinkList &L,int &n); //约瑟夫环解决方案

主函数为

void main(void)

{

LinkList L;

int n;

printf("请输入人数：");

scanf("%d",&n); //设置人数

CreateList(L,n); //创建循环链表

Joseph(L,n); //约瑟夫环解决方案

}

首先要创建循环链表，在确定人数基础上编号，并输入每个人的密码，连接成一个循环链表，头指针 L 指向编号为 1 号的结点。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

创建循环链表

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CreateList(LinkList &L,int &n)

{

printf("将这%d个人编号为1-%d号。\n请依次输入这%d个人的密码:",n,n,n);

LinkList q;

for(int i=1;i<=n;i++)

{

LinkList p=(LinkList)malloc(sizeof(LNode)); //创建新结点用p指向它

p->number=i; //编号

scanf("%d",&p->code); //输入持有密码

p->next=NULL;

if(i==1) L=q=p; //若是第一个结点，直接用头指针指向它

else

{

q->next=p; //将p指向的结点加入链表

q=q->next; //q始终保持指向最后的结点

}

}

q->next=L; //q指向的最后的结点指针域指向头结点，则成为循环链表

}

约瑟夫环解决方案具体实现方案：

第一次报数 默认为 从1 号开始报数，需设置报数上限值m ,报m 的人出列，然后从出列的人后面一个人开始下一轮报数，且报数上限值m变为出列的人持有密码。即第一次由头指针 L 开始向后数到第 m-1 个结点，删掉 第 m-1 个结点的后继结点，即第 m 个结点，（可由一个指针q 反馈回第m 个结点，以便读取被删除结点的信息），这时 第 m-1 个结点 直接指向第 m+1 个结点，同时让头指针 L 也指向第 m+1 个结点。因为下一轮报数时将从第 m+1 个结点开始，即还是 L指向的结点开始。

第二轮报数首先将m值修改为刚刚被删除结点的持有密码，即m=q->code 再进行和第一轮报数相同的操作。

以后每轮报数如次反复执行，直到所有结点被删除结束。

这个过程需要一个删除函数来实现，函数功能是

删除以头指针L开始第i个结点，q返回结点指针，并改变头指针位置为下一轮报数的第一个结点。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

删除以L为头第i个结点，q返回结点指针,改变头指针位置

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DeleteList(LinkList &L,int i,LinkList &q)

{

if(i==1) DeleteList(L,i+LengthList(L),q);//i=1时需要特殊处理

else

{

LinkList p=L;

int j=0;

while(j<i-2) {p=p->next;j++;} //p指向第i-1个结点，即被删除结点的前驱结点

q=p->next; //p指向被删除结点，返回信息

p->next=p->next->next; //删除第i个结点

L=p->next; //修改头指针

}

}

其中i=1时情况比较特殊，需要特殊处理，因为i=1时表示要删除头指针本身指向的结点，无法直接找到其前驱结点。可以改为删除第i+LengthList(L)个结点，同样也是删除头指针本身指向的结点，但需要加一个求表长的函数。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

约瑟夫环解决方案

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Joseph(LinkList &L,int &n)

{

int m;

printf("请输入第一个报数上限值:");//设置第一轮报数上限值

scanf("%d",&m);

printf("\n出列顺序:\n");

LinkList q;

for(int i=1;i<n;i++)

{

DeleteList(L,m,q); //删除L开始第m个结点，修改头指针位置，q返回信息

printf("%d.\t%d号\t他的密码为:%d\n",i,q->number,q->code);//打印相关信息

m=q->code; //修改报数上限值为被删除结点的密码值

free(q);

}

printf("%d.\t%d号\t他的密码为:%d\n",n,L->number,L->code);

}

### 三、源程序及注释：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//定义链表单元

typedef struct Node

{

int num; //数据域

struct Node \*next; //指针域

}LinkList;

//根据参数生成所需要长度的链表

LinkList \*creat(int n)

{

int i = 1;

LinkList \*p,\*q,\*head;

p = (LinkList\*)malloc(sizeof(LinkList));

p->num = i;

head = p;

for(i = 2;i<=n;i++)

{

q = (LinkList\*)malloc(sizeof(LinkList)); //malloc(m)函数新开一片m字节大小的空间 ,并把该区首地址作为函数值

q->num = i;

p->next = q;

p = q;

//p->next = head;

}

p->next = head;

return head;

}

//约瑟夫环问题的具体实现

void fun(LinkList \*L,int k,int m)

{

int i,j;

LinkList \*p,\*q,\*s;

p = L;

q = p; //我修改添加代码

/\* for(i = 1;i<k;i++)

{

q=p;

p = p->next;

} \*/ //原作者代码

while(p->next!=p)

{

for(j = 1;j<m;j++)

{

q=p;

p = p->next;

}

printf("%5d",p->num);

s = p;

q->next = p->next;

p = p->next;

free(s);

}

printf("%5d",p->num);

}

void main()

{

LinkList \*L;

int n,k,m;

n = 9; //总人数

m = 4; //要求的第m个人出列

k = 2;

L=creat(n); //生成刚好所需长度的线性链表

fun(L,k,m);

}

### 四、运行输出结果：

### 2

### 五、调试和运行程序过程中产生的问题及采取的措施：

删除以头指针L开始第i个结点时，从L指向第结点一直向后找到第i个结点后，若p指向第i个结点，则删除时遇到困难，因为无法找到它的前驱结点，不方便操作，后来我改变了方案，从L向后找到第i-1个结点，用p指向此结点，然后删除p的后继结点

p->next=p->next->next;

但是这样又遇到一个问题：当i=1时无法找到其前驱结点。因为这是一个循环链表，所以我想到了删除第i个结点也就是删除第i+LengthList(L)个结点，这样就可以解决问题了，需要加一个求表长函数

int LengthList(LinkList L)

{

int i=1;

LinkList p=L->next;

while(p!=L)

{

i++;

p=p->next;

}

return i; //返回长度

}

### 六、对算法的程序的讨论、分析，改进设想，其它经验教训：

我感觉自己的算法还是比较灵活的，每次删除时都改变头指针位置，并返回删除信息，修改报数上限值，然后以新的头指针进行下一次操作，在实现上也不困难，都能自己解决。

第一轮报数时候我 默认为时从1号开始，其实程序可以改进以下，将第一轮报数的起始编号也由 用户设定，这样需要在第一次删除操作前修改头指针位置，实现起来也比较简单。

### 七、对实验方式、组织、设备、题目的意见和建议：

这个试验比较好，趣味性较强。要想写完这个程序，需要对链表操作很熟悉，并且在细节上不能出错。写完成后感觉很好，觉得自己用那个书本上的知识完成了一些实际的问题，比较有意义。建议老师多选取有趣的试验题。