# 实验报告

**题目**：编制一个模拟约瑟夫环，求解出列顺序的程序

**班级**：电信1808 **姓名**：喻越 **学号**：U201813467 **完成日期**：2019.03.27

**一、需求分析**

1. **输入的形式和输入值的范围：**初始上限值m由键盘输入，要求为正整数；每个人的密码用scanf函数连续输入，输入形式为以空格相隔，以回车符结束的一连串正整数（程序中设定输入密码个数为7）。
2. **输出的形式：**模拟程序以用户和计算机的对话方式执行，即在计算机终端显示“提示信息”后，由用户在键盘上输入模拟程序中规定的运算命令；相应的输入数据跟运算结果显示在其后。以“num：%d出局”及“the winner：”的形式输出。
3. **程序所能达到的功能**
   1. .创建及初始化无头节点单向循环链表
   2. 删去循环链表中出列的人
   3. 进行约瑟夫环游戏主体部分模拟
   4. 输出出局顺序和胜利者
4. **测试数据**

输入：3 1 7 2 4 8 4 //每个人的密码

20 //初始上限值

输出：num;6出局

num;1出局

num;4出局

num;7出局

num;2出局

num;3出局

the winer: 5

**二、概要设计**

1. **所有抽象数据类型的定义**：

为实现上述功能，需要两个抽象数据类型：链式线性表和数组

* 1. .链式线性表

**ADT List {**  
　 **数据对象：**D＝{ | ∈ ElemSet, i=1,2,...,n, n≥0 }  
 **数据关系：**R1＝{ <ai-1 ,ai >| ,∈D, i=2,...,n }

**基本操作：**  
　　　　InitList( **&**L )  
　　　 操作结果：构造一个空的线性表 L 。

　　DestroyList( **&**L )  
　　　 初始条件：线性表 L 已存在。  
　　　 操作结果：销毁线性表 L 。

ListEmpty( L )  
　　　初始条件：线性表L已存在。  
　　　操作结果：若 L 为空表，则返回 TRUE，否则返回 FALSE。

ListLength( L )  
　　　初始条件：线性表 L 已存在。  
　　　操作结果：返回 L 中元素个数。  
 PriorElem( L, cur\_e, **&**pre\_e )  
　　　初始条件：线性表 L 已存在。  
　　　操作结果：若 cur\_e 是 L 中的数据元素，则用 pre\_e 返回它的前

驱，否则操作失败，pre\_e 无定义。  
 NextElem( L, cur\_e, **&**next\_e )  
　　　初始条件：线性表 L 已存在。  
　　　操作结果：若 cur\_e 是 L 中的数据元素，则用 next\_e 返回它的后

继，否则操作失败，next\_e 无定义。

GetElem( L, i, **&**e )  
　　　初始条件：线性表 L 已存在，1≤i≤LengthList(L)。  
　　　操作结果：用 e 返回 L 中第 i 个元素的值。

LocateElem( L, e, compare( ) )  
　　　初始条件：线性表 L 已存在，compare( ) 是元素判定函数。  
　　　操作结果：返回 L 中第1个与 e 满足关系 compare( ) 的元素的位  
　　　　　　　　序，若这样的元素不存在，则返回值为0。

　ListTraverse(L, visit( ))  
　　 初始条件：线性表 L 已存在，visit( ) 为元素的访问函数。  
　　 操作结果：依次对 L 的每个元素调用函数 visit( )。  
　　　　　　　一旦 visit( ) 失败，则操作失败。  
 ClearList( **&**L )  
　　　初始条件：线性表 L 已存在。  
　　　操作结果：将 L 重置为空表。

　PutElem( **&**L, i, **&**e )  
　　　初始条件：线性表L已存在，1≤i≤LengthList(L)。  
　　　操作结果：L 中第 i 个元素赋值同 e 的值。  
　ListInsert( **&**L, i, e )  
　　　初始条件：线性表 L 已存在，1≤i≤LengthList(L)+1。  
　　　操作结果：在 L 的第 i 个元素之前插入新的元素 e，L 的长度增1。

　ListDelete(**&**L, i, **&**e )  
　　　初始条件：线性表 L 已存在且非空，**1≤i≤LengthList(L)**。  
　　　操作结果：**删除**L 的第 i 个元素，并用 e 返回其值，L 的**长度减1**。

**} ADT**List

* 1. 数组

1. **本程序包含四个模块**
   1. 主程序模块

**int main()**

**{**

初始化；

执行游戏模拟程序；

return 0;

**}**

* 1. 结点结构单元模块——实现链表的结点结构
  2. 链表单元模块——实现链表的建立和初始化
  3. 游戏模拟单元模板——模拟约瑟夫环游戏过程
  4. 链表删除操作模块——删除出列者（模拟游戏的一部分）

**各模块之间的调用关系：**

主程序模块

int main()

{

int c[20],i,first\_m;

linklist L;

printf("请输入游戏者密码:");

for(i=0; i<N; i++)

{

scanf("%d",&c[i]);

}

L=creatlist(c,N);

printf("请输入初始m值：");

scanf("%d",&first\_m);

game(L,N,first\_m);

return 0;

}

游戏模拟块

t main()

{

int c[20],i,first\_m;

linklist L;

printf("请输入游戏者密码:");

for(i=0; i<N; i++)

{

scanf("%d",&c[i]);

}

L=creatlist(c,N);

printf("请输入初始m值：");

scanf("%d",&first\_m);

game(L,N,first\_m);

return 0;

}

链表删除结点模板

int main()

{

int c[20],i,first\_m;

linklist L;

printf("请输入游戏者密码:");

for(i=0; i<N; i++)

{

scanf("%d",&c[i]);

}

L=creatlist(c,N);

printf("请输入初始m值：");

scanf("%d",&first\_m);

game(L,N,first\_m);

return 0;

}

链表单元模块

int main()

{

int c[20],i,first\_m;

linklist L;

printf("请输入游戏者密码:");

for(i=0; i<N; i++)

{

scanf("%d",&c[i]);

}

L=creatlist(c,N);

printf("请输入初始m值：");

scanf("%d",&first\_m);

game(L,N,first\_m);

return 0;

}

、

结点结构单元模块

1. **详细设计**

**1.各数据类型**

**int delete\_num；**

**int n, int first\_m；** //元素类型

**int c[20]；** //数组类型

**typedef struct lnode**

**{**

**int number;**

**int key;**

**struct lnode \*next;**

**} lnode,\*linklist;** //结点类型，指针类型

**2.主程序算法**

//不会写伪代码，怎么办

**int main()**

**{**

**int c[20],i,first\_m;**

**linklist L;**

**scanf(c[i]);** //输入数组c[i]的值

**L=creatlist(c,N)；**

**scanf(first\_m);**  //输入first\_m的值

**game(L,N,first\_m);**

**return 0;**

**}**

1. **建立及初始化链表程序算法**

**linklist creatlist(int a[20],int n)**

**{**

**linklist head=(linklist)malloc(sizeof(lnode));**

**head->next=NULL;**

**linklist r=head;**

**head->number=1;**

**head->key=a[0];**

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

**{**

**linklist p=(linklist)malloc(sizeof(lnode));**  //开辟一个空间

**p->number=i+1;**

**p->key=a[i];**

**r->next=p;**

**r=p;**  //链接结点

**}**

**r->next=head;**

**return head;**

**}**

**4.链表删除操作程序**

**linklist linkdelete(linklist head,int i)**

**{**

**if( head->next == head )**

**{**

**printf("the winer: %d\n", head->number);**

**return 0;**

**}**  //判断是否只剩下一个个体

**linklist y;**

**int j;**

**for(j=1, y=head; j<i; j++, y = y->next);**

**linklist pre = y;**

**while(pre->next != y)**

**{**

**pre = pre->next;**

**}**

**pre->next = y->next;**

**printf("num;%d出局\n", y->number);**

**delete\_num = y->key;**

**free(y);**

**return pre->next;**

**}**

**5.游戏模拟操作**

**int game(linklist L, int n, int first\_m)**

**{**

**linklist b=L;**

**int flag=1;**

**while(n)**

**{**

**if(flag)**

**{**

**b=linkdelete(b,first\_m);**

**flag=0;**

**}**  //第一次进行游戏

**else**

**{**

**b=linkdelete(b,delete\_num);**

**}**

**n--;**

**}**

**return 0;**

**}**

**函数的调用关系反映了模拟程序的层次关系：**

**main**

**creatlist game**

**linklist**

1. **调试分析**

**1.** **问题解决及经验与体会**

* 1. .首先不会创建无头结点的链式线性表，通过自己百度和与同学交流得以解决；
  2. 将函数linkdelete错写成linkdetele导致后面出现的linkdelete无定义。以后编写程序应当注意定义函数名的对应；
  3. 在函数linkdelete中，返回值错写成x，即在删除结点后未移动到新链表，导致结果错误；
  4. 在函数linkdelete中，删除结点逻辑错误，不适用当i(或者其他)等于1的情况，错误做法是采用从头遍历到需删除结点的前一个，正确做法是设定需删除结点的前一个结点pre，从头到尾在到需删除结点的前一个，并且做出n=1时停止程序的判断；
  5. 使用scanf输入，在多次调试中多次输入浪费不少时间，下次可以直接将参数编写进程序中，节省时间

1. **算法的时空分析**

程序linklist和creatlist的时间复杂度均为O（n）,程序game的时间复杂

为O（n2）,整个程序的时间复杂度O（n \*m）。

1. **用户使用说明**

1..程序设定参与游戏人员为七人

2..用户在出现"请输入游戏者密码:"时，键入七个正整数，要求以"空格"间隔，以及用"enter"结束输入

3.在出现"请输入初始m值："后键入正整数作为初始上限值，使用enter运行程序即会出现如下结果：

num; 出局

num; 出局

num; 出局

num; 出局

num; 出局

num; 出局

the winer:

1. **测试结果**
2. 键入“3 1 7 2 4 8 4（回车）”“20”

输出：num;6出局

num;1出局

num;4出局

num;7出局

num;2出局

num;3出局

the winer: 5

1. 键入“5 2 3 6 9 8 4（回车）”“30”

输出：num;2出局

num;4出局

num;5出局

num;6出局

num;1出局

num;3出局

the winer: 7

1. **附录**

**源程序文件名清单：**

**stdio.h**

**stdlib.h //**运行malloc和free函数的源程序

**malloc.h**