## 实验3 狐狸逮兔子实验

【问题描述】

围绕着山顶有10个圆形排列的洞，狐狸要吃兔子，兔子说：“可以，但必须找到我，我就藏身于这十个洞中，你先到1号洞找，第二次隔1个洞（即3号洞）找，第三次隔2个洞（即6号洞）找，以后如此类推，次数不限。”但狐狸从早到晚进进出出了1000次，仍没有找到兔子。问兔子究竟藏在哪个洞里？

这实际上是一个反复查找线性表的过程，用以前学过的程序设计的知识也很容易实现，这里希望同学们能体会**数据结构**的思想。

【数据描述】

定义一个顺序表，用具有10个元素的顺序表来表示这10个洞。每个元素分别表示围着山顶的一个洞，下标为洞的编号。

#define LIST\_INIT\_SIZE 10 //线性表存储空间的初始分配量

typedef struct{

ElemType \*elem; //存储空间基址

int length; //当前长度

int listsize; //当前分配的存储容量（以sizeof(ElemType)为单位）

}SqList;

【算法描述】

status InitList\_Sq(SqList &L) { //构造一个线性表L

L.elem=(ElemType )malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

if(!L.elem) //存储分配失败

return OVERFLOW;

L.length=0; //空表长度为0

L.listsize=LIST\_INIT\_SIZE; //初始存储容量

return OK;

} //InitList\_Sq

status Rabbit(SqList &L){ //构造狐狸逮兔子函数

int current=0; //定义一个当前洞口号的记数器，初始位置为第一个洞口

for(i=0;i<LIST\_INIT\_SIZE;i++)

L.elem[i]=1; //给每个洞作标记为1，表示狐狸未进之洞

L.elem[LIST\_INIT\_SIZE-1]=L.elem[0]=0; //首先进入第一个洞，标记进过的洞为0

for(i=2;i<=1000;i++){

current=(current+i)%LIST\_INIT\_SIZE; //实现顺序表的循环引用

L.elem[i]=0; // 标记进过的洞为0

}//第二次隔1个洞找，第三次隔2个洞找，以后如此类推，经过1000次

printf("兔子可能藏在如下的洞中：")

for(i=0;i<LIST\_INIT\_SIZE;i++)

if(L.elem[i]==1)

printf(" 第%d个洞\n ",i+1); //输出未进过的洞号

return OK;

}//end

【C源程序】

#include "stdio.h"

#include "stdlib.h"

#define OK 1

#define OVERFLOW -2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 10 //线性表存储空间的初始分配量

typedef struct {

ElemType \*elem; //存储空间基址

int length; //当前长度

int listsize; //当前分配的存储容量（以sizeof(ElemType)为单位）

}SqList;

status InitList\_Sq(SqList \*L){ //构造一个线性表L

(\*L).elem=(ElemType \*)malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

if(!((\*L).elem)) return OVERFLOW; //存储分配失败

(\*L).length=0; //空表长度为0

(\*L).listsize=LIST\_INIT\_SIZE; //初始存储容量

return OK;

} //InitList\_Sq

status Rabbit(SqList \*L){ //构造狐狸逮兔子函数

int i,current=0; //定义一个当前洞口号的记数器，初始位置为第一个洞口

for(i=0;i<LIST\_INIT\_SIZE;i++)

(\*L).elem[i]=1; //给每个洞作标记为1，表示狐狸未进之洞

(\*L).elem[LIST\_INIT\_SIZE-1]=0;

(\*L).elem[0]=0; //第一次进入第一个洞，标记进过的洞为0

for(i=2;i<=1000;i++){

current=(current+i)%LIST\_INIT\_SIZE; //实现顺序表的循环引用

(\*L).elem[current]=0; //标记进过的洞为0

}//第二次隔１个洞找，第三次隔２个洞找，以后如此类推，经过1000次

printf("\n兔子可能藏在如下的洞中：") ;

for(i=0;i<LIST\_INIT\_SIZE;i++)

if((\*L).elem[i]==1)

printf(" \n此洞是第%d号洞 !",i+1); //输出未进过的洞号

return OK;

}

void main(){

SqList \*L=( SqList \*)malloc(sizeof(SqList));

InitList\_Sq(L);

Rabbit(L);

}

【测试数据】

此洞是第2号洞 !

此洞是第4号洞 !

此洞是第7号洞 !

此洞是第9号洞 !

【说明】

本算法思路比较简单，采用了顺序表表示围着山顶的10个洞，首先对所有洞设置标志为1，然后通过1000次循环，对每次所进之洞修改标志为0，最后输出标志为1的洞。

【实验题】

1. 了解、熟悉、掌握结构体。
2. 学习顺序表、单链表基本概念。
3. 看懂顺序表解决狐狸逮兔子问题的源码。
4. 程序设计题：同上面的狐狸逮兔子问题，如果山上有5个洞或20个洞，狐狸从早到晚进进出出了500次或10000次，仍没有找到兔子。问兔子究竟可能藏在哪个洞里？
5. 程序设计题：同上面的问题，将表示洞的线性表采用单链表实现，并比较其差异。