线性表的实现和基本操作

1. 需求分析

线性表（亦作顺序表）是最基本、最简单、也是最常用的一种数据结构。线性表的逻辑结构简单，便于实现和操作。因此，线性表这种数据结构在实际应用中是广泛采用的一种数据结构。本实验的主要目的在于使读者熟悉线性表的基本操作在两种存储结构上的实现，其中以熟悉各种链表的操作为重点。本实验还可帮助读者复习高级编程语言的使用方法。

约瑟夫问题的一种描述是，编号为$1, 2, \ldots, n$的n个人按顺时针方向围坐一圈，每人持有一个密码（正整数）。一开始任选一个正整数作为报数上限值m，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，位于他顺时针方向上的下一个人开始重 新从1报数，如此下去，直至所有人全部出列。试设计一个程序求出出列顺序。

线性表的主要操作有：按值查找（Search）、删除（Delete）、插入（Insert）、输出（show）等。

(1) 输入的形式和输入值的范围：系统定义的基本类型。

(2) 输出的形式：系统定义的基本类型。

(3) 程序所能达到的功能：以链表（或循环链表）的形式储存数据并解决约瑟夫环问题。

(4) 测试数据：依次输入 1111、2222、3333、4444、5555，查看输出结果。

2. 概要设计

(1)链表类的基本操作：

void InitList(Linklist &L)

操作结果：构造了一个空链表。

int Search(Linklist &L,int pwd0)

初始条件：表已存在且非空（为空则在控制台打印“链表为空”）

操作结果：按值在表中查找，若查到返回位置，未查到则在控制台打印“未找到”。

void Delete(Linklist &L,int num0,int &pwd0)

初始条件：表已存在且非空（为空则在控制台打印“链表为空”）

操作结果：删除指定的元素，并将值赋给传入的变量引用。

void Insert(Linklist &L,int i,int data)

初始条件：表已存在

操作结果：头插法插入一个元素，并返回修改后的链表 。

void show(Linklist L)

初始条件：表已存在

操作结果：依次输出链表的数据。

(2)链表简单测试

void insert()

初始条件：表已存在

操作结果：依次输入每个结点所含信息。

void output()

初始条件：表已存在。

操作结果：依次输出每个结点所含信息，并测试查找、删除功能是否正常。

main()

初始条件：表已存在。

操作结果：调用相应函数，执行相应操作，输出所有信息。

(3)约瑟夫环测试

main()

初始条件：无。

操作结果：调用相应函数，执行相应操作，输出所有信息。

void start()

初始条件：表已存在。

操作结果：执行约瑟夫环测试。

3. 详细设计

（1）链表类基本内容————头文件

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

typedef struct Lnode{
    int num;
    int pwd;
    Lnode *next;
}Node,*Linklist;

 Linklist L;

void InitList(Linklist &L){
     Node *head = (Node*)malloc(sizeof(Node));
    L=head;
    head->num=head->pwd=0;
    head->next=NULL;


}



void Insert(Linklist &L,int i,int data)
{
    Node *newnode = (Node*)malloc(sizeof(Node));

    newnode->num=i;
```

```
    newnode->pwd=data;
    newnode->next=L->next;
    L->next=newnode;//头插

}

void Delete(Linklist &L,int num0,int &pwd0)
{
    if(L->next==NULL){cout<<"链表为空！";return ;}
    Node *find = L;
    while(find->next->num!=num0)
    {

       find=find->next;
        if(find->next==NULL ){cout<<"未找到！";return;}
    }

    Node *q = find->next;
    pwd0=q->pwd;
    find->next=q->next;
    free(q);


}



int Search(Linklist &L,int pwd0)
{
    if(L->next==NULL){cout<<"链表为空！";return 0;}
    Node *find = L->next;
    while(find->pwd!=pwd0){

       find=find->next;
        if(find==NULL)
        {
            cout<<"未找到！";
           return 0;
        }

    }
    return find->num;
```

```
}

void show(Linklist L)
{
    Node *find = L->next;
    while(find!=NULL&&L->next!=NULL)
    {
        cout<<find->num<<"  "<<find->pwd<<endl;
        find=find->next;
    }
}
```

（2）链表类基本内容———可执行 cpp 文件

```
#include "LinkList.hpp"

void insert()
{
    cout<<"请输入总人数：";
    int sum=0;
    cin>>sum;


    int i=1;

    while(i<=sum)
    {
        cout<<"请输入第"<<i<<"个人的密码：";
        int data;
        cin>>data;
        Insert(L,i,data);
        i++;

    }
}


void output()
{
    int i;
```

```
    cout<<"请输入要查找的密码：";
    cin>>i;
    cout<<"密码对应序号为："<< Search(L,i)<<endl;
    int pwd0;
    int j;
    cout<<"请输入要删除的序号：";
    cin>>j;

    Delete(L,j,pwd0);
    cout<< pwd0<<endl;
    show(L);
}



int main()
{
    InitList(L);
    insert();
    show(L);cout<<endl<<endl;
    output();
    return 0;
}
```

（3）约瑟夫环中链表类基本内容（稍作改进）

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

typedef struct Lnode{
    int num;
    int pwd;
    Lnode *next;
}Node,*Linklist;

Node *rear;//尾指针
Linklist L;

void Insert(Linklist &L,int i,int data)
{
    Node *newnode = (Node*)malloc(sizeof(Node));

    newnode->num=i;
```

```
    newnode->pwd=data;
    newnode->next=rear->next;
    rear->next=newnode;
    rear=newnode;

}

void InitList(Linklist &L){

    int sum=0;
    while(1){
    cout<<"请输入总人数（不超过 30 人）：";

    cin>>sum;
        if(sum>30){
            cout<<"输入人数超过 30！请重新输入！" ;
            cout<<endl<<endl;
            continue;
        }
    cout<<endl<<endl;
        break;
    }
    int i=1;

    while(i<=sum)
    {
        cout<<"请输入第"<<i<<"个人的密码：";
        int data;
        cin>>data;
        if(i==1){
            Node *newnode = (Node*)malloc(sizeof(Node));
            L=newnode;
            newnode->num=i;
            newnode->pwd=data;
            newnode->next=newnode;//循环链表构建
            rear=L;

        }else{
            Insert(L,i,data);
        }

        i++;

    }
```

```
}



void Delete(Linklist &L,Node *N)
{
    if(L==NULL){cout<<"链表为空！";return ;}
    Node *find = N;
    while(find->next!=N)
        find=find->next;


    Node *q = N;
    find->next=q->next;

}


void Show(){
    Node *find = rear;
    do{
        cout<<"  第 "<<find->next->num<<" 个 人 ， 密 码 为
"<<find->next->pwd<<endl;

        find=find->next;
    } while(find!=rear);

}
```

（4）约瑟夫环可执行 cpp 文件

```
#include <iostream>
#include "List.h"

using namespace  std;
```

```
void start(){

    cout<<"          约瑟夫环实验：          "<<endl<<endl;

    InitList(L);
    cout<<endl<<endl;
    Show();
    cout<<endl<<endl;
    cout<<"请指定报数上限 M 值："；

    int M;
    cin>>M;
    Node *visit=rear->next;
    int no=1;
    while(visit->next!=visit)
    {

        int count=1;

        while(true)
        {
            if(count==M){

                Node *p=visit;
                visit=visit->next;


        cout<<"第"<<no++<<"个出列的人编号为"<<p->num<<"，密码为
"<<p->pwd<<endl;
                Delete(L,p);
                M=p->pwd;//修改每次报数的上限
                break;
                        }
            visit=visit->next;
            count++;



        }


    }

cout<<"第"<<no<<"个出列的人编号为"<<visit->num<<"，密码为
```

```
″<<visit->pwd<<endl;


}



int main() {

    start();
    free(L);

    return 0;
}
```

4. 调试分析

1、设置总数据为 5 个，依次输入数据 1111、2222、3333、4444、5555，输出数据：

```
请输入总人数: 5
请输入第1个人的密码: 1111
请输入第2个人的密码: 2222
请输入第3个人的密码: 3333
请输入第4个人的密码: 4444
请输入第5个人的密码: 5555
5   5555
4   4444
3   3333
2   2222
1   1111
```

2、查询密码为 2222 的序列号，输出：

```
请输入要查找的密码：2222
密码对应序号为：2
```

3、然后删除 3 号元素，输出：

```
请输入要删除的序号：3
3333
5   5555
4   4444
2   2222
1   1111
```

测试完毕，链表可正常使用。

<1>调试过程中遇到的问题：

第一次运行的时候，发现输入的数据和输出的数据是反着的，开始没想明白，到后来才知道，采用头插法插入，依次遍历链表的时候必然会倒序输出。而且，由于数据定义的基本类型为 int 类型，所以密码不能含有字符，若要实现此功能，可以改变基本类型为 string 类型。

<2>时间和空间分析；

查询功能所用时间为遍历链表所耗时间，删除功能也为遍历链表所

耗时间，均为O(n)。

链表占用空间不定，为每个结点所占用空间之和。

<3>改进设想：

考虑到头插法可能导致输出结果不符合正常逻辑情况，可以采用尾插法。

<4>经验、心得和体会：

第一次设计链表，终于学会了基本链式结构的使用，可以说进一步理解了指针的使用方法，同时也巩固了C语言的基本语法和基本操作，收益颇深。

5. 使用说明

1、输入参赛者的总人数，依次输入他们所持有的密码：1—n，n>1。

2、输入淘汰数 m（依次报数 m 次后，该人出局），以 20 为例。

6. 测试程序的运行结果

（1）输入参赛人员信息：

```
约瑟夫环实验:

请输入总人数（不超过30人）：7


请输入第1个人的密码：3
请输入第2个人的密码：1
请输入第3个人的密码：7
请输入第4个人的密码：2
请输入第5个人的密码：4
请输入第6个人的密码：8
请输入第7个人的密码：4


第1个人，密码为3
第2个人，密码为1
第3个人，密码为7
第4个人，密码为2
第5个人，密码为4
第6个人，密码为8
第7个人，密码为4
```

（2）输出被淘汰的序号以及胜利者的序号。

```
请指定报数上限M值：20
第1个出列的人编号为6，密码为8
第2个出列的人编号为1，密码为3
第3个出列的人编号为4，密码为2
第4个出列的人编号为7，密码为4
第5个出列的人编号为2，密码为1
第6个出列的人编号为3，密码为7
第7个出列的人编号为5，密码为4
```

7. 心得体会

以前代码不够规范，编程能力也不强，也没有用过结构体，所以这次任务刚刚开始的时候是很困难的，结构体定义一直在报错，经过不断地调试和查阅资料，终于成功定义了第一个结构体，之后链表的实现也迎刃而解。从这次实验体会到了，要想提高自己编程能力，还得多读代码，多动手写代码。

附录：源程序文件清单

各程序源代码文件随本实验报告电子版一起打包，存放在 bin 子目录。

文件清单如下：

LinkList.h……基本链表定义及基本操作，头文件

List.h……约瑟夫环实验链表类定义，头文件

JosephCircle.cpp ……约瑟夫环算法实现

LinkList.cpp……链表的基本操作，测试文件

CENTRAL SOUTH UNIVERSITY
中南大学