

云南大学数学与统计学院

上机实践报告

课程名称：数据结构与算法实验	年级：2013	上机实践成绩：
指导教师：陆正福	姓名：金洋	
上机实践名称：线性表实验	学号：20131910023	上机实践日期： 3.26
上机实践编号：No. 3	组号：	上机实践时间： 16: 12

一、实验目的

- 1.熟悉与线性表有关的数据结构与算法
- 2.熟悉主讲教材 Chapter 3 的代码片段
- 3.熟悉实验教材第 1 章的问题

二、实验内容

1. 线性表有关的数据结构设计 with 算法设计
2. 调试主讲教材 Chapter 3 的 Java 程序
3. 阅读实验教材第 1 章的问题，将 C 程序转化为 Java 程序（每组至少完成 1 个问题）

三、实验平台

个人计算机； Oracle/Sun Java 7 SE 或 EE

四、实验记录与实验结果分析

（注意记录实验中遇到的问题。实验报告的评分依据之一是实验记录的细致程度、实验过程的真实性、实验结果的解释和分析。如果涉及实验结果截屏，应选择白底黑字。）

① 游戏存储

先新建一个类，该类用于存储玩家信息

```
public class GameEntry{
    protected String name;//name of the person
    protected int score;
    //Constructor
    public GameEntry(String n,int s){
        name=n;
        score=s;
    }
    public String getName(){return name;}
    public int getScore(){return score;}
    public String toString(){
        return "("+name+","+score+")";
    }
}
```

再新建一个 Score 类，包含对于游戏玩家列表的基本操作

```

public class Score{
    public static final int maxEntries=10;//number of high scores we keep
    protected int numEntries;//现成绩榜人数
    protected GameEntry[] entries;//声明 GameEntry 类的数组 entries[]
    //Constructor
    public Score(){
        entries=new GameEntry[maxEntries];
        numEntries=0;
    }
    public String toString(){
        String s="[";
        for (int i=0;i<numEntries;i++){
            if (i>0) s+=", ";
            s+=entries[i];
        }
        return s+"]";
    }

    //Add a new high score to the collection
    public void add(GameEntry e){
        int newScore=e.getScore();//得到欲新增节点的成绩
        if (numEntries==maxEntries){//成绩表已经满了，将新的成绩与现有最后一位比较
            if (newScore<=entries[numEntries-1].getScore())
                //该节点的成绩不够高，没有资格排入，该方法下面的步骤不再执行
                return;
        }
        //表未满，将其排入
        numEntries++;
        int i=numEntries-1;
        for (;(i>=1) &&(newScore>entries[i-1].getScore());i--)
            // 从最后部开始比较： 排名榜上至少有一位，且新增者成绩大于当前指向者的成绩——则
            // 继续往前找
            entries[i]=entries[i-1];//后移一位
        entries[i]=e;//将新增节点放入找到的位置
    }

    //Remove and return the high score at index i
    public GameEntry remove(int i) throws IndexOutOfBoundsException{
        if ((i<0)|| (i>=numEntries))
            throw new IndexOutOfBoundsException("Invalid index:"+i);//下表不合
            //法，抛出异常
        GameEntry temp=entries[i];//将要移去的单元暂时存储到别处
        //元素 i 后面的元素各向前移一位；
        for (int j=i;j<numEntries-1;j++)
            entries[j]=entries[j+1];
        entries[numEntries-1]=null;//最后一个单元赋为空值
        numEntries--;//元素量减一
        return temp;//返回移除值
    }
}

```

② 插入排序

```

public class InsertionSort {
    static char[] a=new char[]{'A','C','Y','O','B'};
    public static void insertionSort(char[] a){
        int n=a.length;
        for (int i=1;i<n;i++){
            char cur=a[i];
            int j=i-1;
            while ((j>=0)&&(a[j]>cur)){
                a[j+1]=a[j];
                j--;
            }
            a[j+1]=cur;
        }
    }
    public static void output(char[] a,String s){
        System.out.print(s+":");
        for (int i=0;i<a.length;i++){
            System.out.print(a[i]+" ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        output(a,"Before Sorting:");
        insertionSort(a);
        output(a,"After Sorting:");
    }
}

```

命令提示符: C:\Windows\system32\cmd.exe

-parameters	生成元数据以用于方法参数的反射
-d <目录>	指定放置生成的类文件的位置
-s <目录>	指定放置生成的源文件的位置
-h <目录>	指定放置生成的本机标头文件的位置
-implicit:{none,class}	指定是否为隐式引用文件生成类文件
-encoding <编码>	指定源文件使用的字符编码
-source <发行版>	提供与指定发行版的源兼容性
-target <发行版>	生成特定 VM 版本的类文件
-profile <配置文件>	请确保使用的 API 在指定的配置文件中可用
-version	版本信息
-help	输出标准选项的提要
-A<关键字[=值]>	传递给注释处理程序的选项
-X	输出非标准选项的提要
-J<标记>	直接将 <标记> 传递给运行时系统
-Werror	出现警告时终止编译
@<文件名>	从文件读取选项和文件名

```

G:\DSA>javac InsertionSort.java

G:\DSA>java InsertionSort
Before Sorting::A C Y O B
After Sorting::A B C O Y

G:\DSA>

```

③随机

```

import java.util.Arrays;
import java.util.Random;
public class ArrayTest {
    public static void main(String[] args){
        int[] num=new int[10];
        Random rand=new Random();
        rand.setSeed(System.currentTimeMillis());

        for (int i=0;i<num.length;i++){
            num[i]=rand.nextInt(100);
        }
        int[] old=num.clone();
        System.out.println("arrays equal before sort:"+Arrays.equals(old, num));
        Arrays.sort(num);
        System.out.println("arrays equal after sort:"+Arrays.equals(old,num));
        System.out.println("old="+Arrays.toString(old));
        System.out.println("num="+Arrays.toString(num));
    }
}

```

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\King Yang>E:

E:\>E:\Computer\Java\Project\DSA\src
'E:\Computer\Java\Project\DSA\src' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

E:\>cd E:\Computer\Java\Project\DSA\src

E:\Computer\Java\Project\DSA\src>javac ArrayTest.java

E:\Computer\Java\Project\DSA\src>java ArrayTest
arrays equal before sort:true
arrays equal after sort:false
old=[88, 19, 85, 86, 79, 20, 90, 37, 54, 11]
num=[1, 19, 20, 37, 54, 79, 85, 86, 88, 90]

E:\Computer\Java\Project\DSA\src>

```

④

```

public class Caesar {
    public static final int ALPHASIZE=26;
    public static final char[]
alpha={'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S',
'T','U','V','W','X','Y','Z'};
    protected char[] encrypt=new char[ALPHASIZE]; //加密数组;
    protected char[] decrypt=new char[ALPHASIZE]; //解密数组

    public Caesar(){

```

```

        for (int i=0;i<ALPHASIZE;i++) encrypt[i]=alpha[(i+3)%ALPHASIZE];
        for (int i=0;i<ALPHASIZE;i++) decrypt[encrypt[i]-'A']=alpha[i];
    }

    public String encrypt(String secret){
        char[] mess=secret.toCharArray();
        for (int i=0;i<mess.length;i++)
            if ((mess[i]>='A') && (mess[i]<='Z'))
                mess[i]=encrypt[mess[i]-'A'];
        return new String(mess);
    }

    public String decrypt(String secret){
        char[] mess=secret.toCharArray();
        for (int i=0;i<mess.length;i++)
            if ((mess[i]>='A') && (mess[i]<='Z'))
                mess[i]=decrypt[mess[i]-'A'];
        return new String(mess);
    }

    public static void main(String[] args){
        Caesar cipher=new Caesar();
        System.out.println("Encryption order="+new String(cipher.encrypt));
        System.out.println("Decryption order="+new String(cipher.decrypt));
        String secret="THE EAFGLE IS IN PLAY;MEET AT JOE'S.";
        secret=cipher.encrypt(secret);
        System.out.println(secret);
        secret=cipher.decrypt(secret);
        System.out.println(secret);
    }
}

```

The screenshot shows a Windows XP command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays the following commands and their outputs:

```

Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>D:

D:\>cd D:\Using Arrays\src

D:\Using Arrays\src>javac Caesar.java

D:\Using Arrays\src>java Caesar
Encryption order=DEFGHIJKLMNOPQRSTUVWXYZABC
Decryption order=XYZABCDEFGHIJKLMNPOQRSTUVWXYZ
WKH HDIJOH LU LQ SODB;PHHW DW MRH'U.
THE EAFGLE IS IN PLAY;MEET AT JOE'S.

D:\Using Arrays\src>

```

⑤新建结点类

```

public class Node {
    private String element;
    private Node next;
    public Node(String s, Node n){
        element=s;
        next=n;
    }
    public String getElement(){
        return element;
    }
    public Node getNext() {
        return next;
    }
    public void setElement(String newEle){
        element=newEle;
    }
    public void setNext(Node newNext){
        next=newNext;
    }
}

```

循环列表

```

public class CircleList {
    protected Node cursor;
    protected int size;
    public CircleList(){
        cursor=null;
        size=0;
    }
    public int size(){
        return size;
    }
    public Node getCursor(){
        return cursor;
    }
    public void advance(){cursor=cursor.getNext();}
    public void add(Node newNode){
        if (cursor==null) {
            newNode.setNext(newNode);
            cursor=newNode;
        }
        else {
            newNode.setNext(cursor.getNext());
            cursor.setNext(newNode);
        }
        size++;
    }
    public Node remove(){
        Node oldNode=cursor.getNext();
        if (oldNode==cursor) cursor=null;
        else{
            cursor.setNext(oldNode.getNext());
            oldNode.setNext(null);
        }
        size--;
    }
}

```

```

        return oldNode;
    }
    public String toString(){
        if (cursor==null) return "[]";
        String s="["+cursor.getElement();
        Node oldCursor=cursor;
        for (advance();oldCursor!=cursor;advance())
            s+=","+cursor.getElement();
        return s+"]";
    }
}

```

主程序

```

import java.util.Random;
public class DuckGoose {
    public static void main(String[] args) {
        CircleList C=new CircleList();
        int N=3;//重复数;
        Node it;
        Node goose;
        Random rand=new Random();
        rand.setSeed(System.currentTimeMillis());
        String[] names={"Bob","Jen","Pam","Tom","Ron","Vic","Sue","Joe"};
        for (int i=0;i<names.length;i++){
            C.add(new Node(names[i],null));
            C.advance();
        }
        for (int i=0;i<N;i++){
            System.out.println("Playing Duck,Duck,Goose for"+C.toString());
            it=C.remove();
            System.out.println(it.getElement()+"is it.");
            while (rand.nextBoolean()||rand.nextBoolean()){
                C.advance();
                System.out.println(C.getCursor().getElement()+"is a duck.");
            }
            goose=C.remove();
            System.out.println(goose.getElement()+"is the goose!");
            if (rand.nextBoolean()){
                System.out.println("The goose won!");
                C.add(goose);
                C.advance();
                C.add(it);
            }
            else{
                System.out.println("The goose lost!");
                C.add(it);
                C.advance();
                C.add(goose);
            }
        }
        System.out.println("Final circle is "+C.toString()+"...");
    }
}

```



```
}

```

```

C:\Windows\system32\cmd.exe

F:\DSA\Code\src>java DuckGoose
Playing Duck,Duck,Goose for[...Joe,Bob,Jen,Pam,Tom,Ron,Vic,Sue]
Bob is it.
Jen is a duck.
Pam is the goose!
The goose won!
Playing Duck,Duck,Goose for[...Pam,Bob,Tom,Ron,Vic,Sue,Joe,Jen]
Bob is it.
Tom is a duck.
Ron is a duck.
Vic is a duck.
Sue is a duck.
Joe is the goose!
The goose won!
Playing Duck,Duck,Goose for[...Joe,Bob,Jen,Pam,Tom,Ron,Vic,Sue]
Bob is it.
Jen is a duck.
Pam is a duck.
Tom is a duck.
Ron is the goose!
The goose won!
Final circle is [...Ron,Bob,Vic,Sue,Joe,Jen,Pam,Tom]...

F:\DSA\Code\src>

```

⑥实现在带头结点的双向循环链表中的插入和删除运算

```

public class DNode<E> {
    protected E element; //element 为储存在 DNode 的元素，元素类型为泛型 E;
    protected DNode<E> next, prev; //next、prev 分别为后继指针、前趋指针;
    //构造方法
    public DNode(E e, DNode<E> p, DNode<E> n) {
        element = e;
        prev = p;
        next = n;
    }
    //得到 element 元素值:
    public E getElement() {
        return element;
    }
    //得到后继节点
    public DNode<E> getNext() {
        return next;
    }
    //得到前趋节点
    public DNode<E> getPrev() {
        return prev;
    }
    //element 赋值
    public void setElement(E newEle) { element = newEle; }
    //next 赋值
    public void setNext(DNode<E> newNext) { next = newNext; }
}

```



```

//prev 赋值
public void setPrev(DNode<E> newPrev){prev=newPrev;}

}

public class DList<E> {
    protected int size;//元素总数
    protected DNode<E> header;
    //构造方法
    public DList(){
        size=0;
        header=new DNode<E>(null,null,null); //头节点初始化
    }
    public int size(){return size;}//得到元素总数;
    public boolean isEmpty(){return size==0;}//判断是否为空，空则返回 TRUE;
    public DNode<E> getPrev(DNode<E> v){return v.getPrev();};//得到 v 的前趋节点;
    public DNode<E> getNext(DNode<E> v){return v.getNext();};//得到 v 的后继节点;
    public DNode<E> getHeader(){return header;}//得到双向循环链表的头节点

    //在带头节点的双向循环链表中寻找第 i 个节点，找出返回该节点;
    public DNode<E> getDNodeI(int i) throws IllegalStateException{
        if (isEmpty()) throw new IllegalStateException("List is empty");
        if ((i>size()+1)|| (i<1)) throw new IllegalStateException("i is invalid.");
        int k=1;//k 从头结点开始计数，直到找到第 i 个节点;
        DNode<E> p=header;//p 为工作指针
        while (k<i){
            k++;
            p=p.getNext();
        }//但 i=size+1 时，则工作指针再次指向 header
        return p;
    }

    //在 header 节点中存值
    public void addFirst(E ele){
        header.setElement(ele);
        //只有一个元素时，前后节点都为其本身，需要设置好 prev 和 next 值，防止出现
        NullPointerException 异常
        header.setNext(header);
        header.setPrev(header);
        size++;
    }

    //在带头节点的双向循环链表的位置 i 前插入新节点 v;
    public void addBefore(int i,E ele){
        DNode<E> y=getDNodeI(i);//y 为第 i 个节点;
        DNode<E> x=y.getPrev();//x 为第 i-1 个节点;
        DNode<E> v=new DNode<E>(ele,null,null) ;
        v.setPrev(x);
        v.setNext(y);
        y.setPrev(v);
        x.setNext(v);
        size++;//表长加 1;
    }
}

```

```

    }

    //在带头节点的双向循环链表中删除第 i 个位置上的节点
    public void remove(int i){
        DNode<E> v=getDNodeI(i);//v 为第 i 个节点;
        DNode<E> y=v.getNext();
        DNode<E> x=v.getPrev();
        y.setPrev(x);
        x.setNext(y);
        //v 的前后指向置为空;
        v.setPrev(null);
        v.setNext(null);
        size--;//表长减 1;
    }

    //输出带头节点的双向循环链表中的元素;
    public String toString(String s){
        DNode<E> v=getHeader();
        int k=1;//k 统计已输出元素的个数
        while (k<=size()){
            s=s+v.getElement()+" ";
            v=v.getNext();
            k++;
        }
        return s;
    }

    public void freeDList(){
        DNode<E> q=getHeader(),p;
        do{
            p=q;
            p.setPrev(null);
            p.setElement(null);
            q=p.getNext();
            p.setNext(null);
        }while (q!=null);
    }
}

import java.io.*;
import java.util.Scanner;
public class Problem4 {
    public static void main(String[] args) {

        DList<Character> L;//对象 L 中的 element 类型为 char;
        //初始化
        System.out.println("(1)初始化双链表 L");
        L=new DList<Character>();

        //插入元素
        System.out.println("(2)依次插入元素 a,b,c,d,e");
        char ele='a';

```

```

L.addFirst(ele);//header 节点设为首节点, 在 header 中存值
int j=1;
//继续插入 b,c,d,e
do{
    j++;
    ele++;
    L.addBefore(j,ele);//依次在位置 j 前插入新节点 v
}while (j<5);
System.out.println(L.toString("(3)输出双链表 L: "));

//删除第 i 个节点
System.out.println("(4)请输入删除节点的序号: ");
Scanner s=new Scanner(System.in);
int NumRemove=s.nextInt();//NumRemove 为输入的删除节点序号
L.remove(NumRemove);
System.out.println(L.toString("(5)输出删除节点"+NumRemove+"后的双链表 L: "));

System.out.println("(6)释放双链表");
L.freeDList();
}
}

```

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\King Yang>E:

E:\> cd E:\Computer\Java\Project\DSA\src

E:\Computer\Java\Project\DSA\src>javac Problem4.java

E:\Computer\Java\Project\DSA\src>java Problem4
<1>初始化双链表L
<2>依次插入元素a,b,c,d,e
<3>输出双链表L: a b c d e
<4>请输入删除节点的序号:
2
<5>输出删除节点2后的双链表L: a c d e
<6>释放双链表

E:\Computer\Java\Project\DSA\src>

```

⑦约瑟夫问题

```
import java.io.*;
import java.util.Scanner;

public class Josephu{
protected static int m,n,k;
public static void inputParameter(){
    Scanner s=new Scanner(System.in);
    System.out.println("请输入人数 n: ");
    n=s.nextInt();
    System.out.println("请输入开始报号人的编号 k: ");
    k=s.nextInt();
    System.out.println("请输入出列人数到的号 m: ");
    m=s.nextInt();
}
public static void gameStart(){
    String s="";
    int t = 0;
    int[] a;
    int[] b;
    a=new int[n];
    b=new int[n];
    for(int i=0;i<b.length;i++)
        b[i]=0;
    for(int i=0;i<a.length-1;i++)
        a[i]=i+1;
    a[a.length-1]=0;
    k=k-1;
    for(int j=0;j<b.length;j++){
        for(int i=0;i<m-1;i++){
            t=k;
            k=a[k];
        }
        b[j]=k+1;
        a[t]=a[k];
        k=a[k];
        s=s+b[j]+" ";
    }
    System.out.println(s);
}

public static void main(String[] args){
    inputParameter();
    gameStart();
}

}
```



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\King Yang>E:

E:\>cd E:\Computer\Java\Project\DSA\src

E:\Computer\Java\Project\DSA\src>javac Josephu.java

E:\Computer\Java\Project\DSA\src>java Josephu
请输入人数n:
5
请输入开始报号人的编号k:
2
请输入出列人数到的号m:
3
4 2 1 3 5

E:\Computer\Java\Project\DSA\src>
```

搜狗拼音输入法 全 :

五、实验体会

1.对于以下代码片段

```
int i=numEntries-1;

for (;i>=1) &&(newScore>entries[i-1].getScore());i--)
```

i 的值在 for 循环结束后可以使用；

若改为 for (int i=numEntries-1;i>=1) &&(newScore>entries[i-1].getScore());i--)

则 i 的值在 for 循环结束后不能使用，因为此时 i 是属于 for 循环的局部变量；

2. Java 泛型的主要目的是可以建立具有类型安全的集合框架，如链表、散列表等数据结构，最重要的一个优点就是：在使用这些泛型类建立的数据结构时，不必进行强制类型转换，即不要求进行运行时类型检查。

3. 对于撤销链表，java 中因为有垃圾回收机制程序员可以不必处理；

4. 在链表中增加一个新的结点时，必须在 DList 类中的 addBefore(int i,E ele)方法中 new NewDnode，将其加入；而不是整个程序只 new 一个新节点，每次改变其 element,next,pre 值，这样会使每个节点与最后一个节点一模一样；

5. 双向循环链表 head 中可以存放元素，也可以不存放，两种情况注意 size 的值；

6. 类中的属性经常修饰为 private，保证了封装性，但可以通过 public 的方法来访问变量；

7. 用尾指针 rear 表示的单循环链表对开始结点 a1 和终端结点 an 查找时间都是 $O(1)$ 。而表的操作常常是在表的首尾位置上进行，因此，实用中多采用尾指针表示单循环链表；判断空链表的条件为 $rear==rear->next$ ；

8. 循环链表中没有 NULL 指针。涉及遍历操作时，其终止条件就不再是像非循环链表那样判别 p 或 $p->next$ 是否为空，而是判别它们是否等于某一指定指针，如头指针或尾指针等。

9. 在单链表中，从一已知结点出发，只能访问到该结点及其后续结点，无法找到该结点之前的其它结点。而在单循环链表中，从任一结点出发都可访问到表中所有结点，这一优点使某些运算在单循环链表上易于实现。

六、参考文献

1. 主讲课英文教材 **Goodrich, Tamassia: Data Structures and Algorithms in Java, 5th Edition International Student Version chapter 3**

2. 实验教材：汪萍，陆正福等编著 数据结构与算法的问题与实验 第 1 章

3. java 编程基础、应用与实例（人民邮电出版社，徐浩明【著】 武传海【译】