**南京航空航天大学**

**数据结构课程设计报告**

|  |  |
| --- | --- |
| 学生姓名 | 薛方岗 |
| 学 号 | 161630213 |
| 学 院 | 计算机科学与技术学院 |
| 专 业 | 软件工程 |
| 班 级 | 1616302 |
| 指导教师 | 高航 |

二〇一八年一月

**目 录**

[一、购物网站信息管理 4](#_Toc16524)

[1.1 题目简介 4](#_Toc24720)

[1.2数据结构 4](#_Toc28370)

[1.3 算设法计思想 5](#_Toc9562)

[1.4测试数据和结果 7](#_Toc31152)

[1.5算法时间复杂度 9](#_Toc32400)

[1.6源代码 10](#_Toc27957)

[二、 迷宫问题 24](#_Toc24745)

[2.1 题目简介 24](#_Toc25235)

[2.2 数据结构 25](#_Toc4239)

[2.3 算法设计思想 25](#_Toc28770)

[2.4测试数据和结果 25](#_Toc30801)

[2.5算法时间复杂度 27](#_Toc4610)

[2.6源代码 27](#_Toc31079)

[三、 二叉树的应用 33](#_Toc2816)

[3.1 题目简介 33](#_Toc3912)

[3.2 数据结构 33](#_Toc19359)

[3.3 算法设计思想 33](#_Toc25064)

[3.4测试数据和结果 34](#_Toc31772)

[3.5算法时间复杂度 35](#_Toc15368)

[3.6源代码 36](#_Toc8095)

[四、 Huffman编码与解码 49](#_Toc24057)

[4.1 题目简介 49](#_Toc11772)

[4.2 数据结构 49](#_Toc3759)

[4.3 算法设计思想 50](#_Toc2661)

[4.4测试数据和结果 50](#_Toc15911)

[4.56算法时间复杂度 51](#_Toc4576)

[4.6源代码 51](#_Toc24227)

[五、无线传感器网络 58](#_Toc8639)

[5.1 题目简介 58](#_Toc6319)

[5.2 数据结构 58](#_Toc4286)

[5.3 算法设计思想 58](#_Toc29013)

[5.4测试数据和结果 59](#_Toc8829)

[5.5算法时间复杂度 60](#_Toc30701)

[5.6源代码 60](#_Toc22481)

[六、 排序算法比较 68](#_Toc1727)

[6.1 题目简介 68](#_Toc150)

[6.2 数据结构 68](#_Toc14755)

[6.3 算法设计思想 68](#_Toc23480)

[6.4测试数据和结果 70](#_Toc25757)

[6.5算法时间复杂度 70](#_Toc7219)

[6.6源代码 71](#_Toc24469)

[七、 表达式求解 68](#_Toc1727)

[7.1 题目简介 68](#_Toc150)

[7.2 数据结构 68](#_Toc14755)

[7.3 算法设计思想 68](#_Toc23480)

[7.4测试数据和结果 70](#_Toc25757)

[7.5算法时间复杂度 70](#_Toc7219)

[7.6源代码 71](#_Toc24469)

八、 [课程设计总结 1](#_Toc24048)18

[8.1 完成情况（代码行数） 1](#_Toc3603)18

[8.2 心得体会 1](#_Toc6584)18

# 一、

1.1 题目简介

购物网站信息管理（必做）（链表）

[问题描述]

设计一个程序，对商铺信息管理，商铺信息包括：商铺编号，商铺名，信誉度，（商品名称1，价格1，销量1），（商品名称2，价格2，销量2），（商品名称3，价格3，销量3）…。

假设商品名称包括(毛巾，牙刷，肥皂，牙膏，洗发水，沐浴露等)，每个商铺具有其中事先确定若干商品及价格，由文件输入，销量初始为0。

[基本要求]

（1）建立一个单向链表存储所有商铺信息（至少30个），以编号为序，编号从1开始递增，从文件中读取数据，并能将数据存储在文件。商铺信息结点的数据结构自行设计。

（2）可以增、删商铺。增加商铺，编号自动加一，插入链表尾部；删除商铺以编号为准，并修改后续结点的编号，保持编号连续性。

（3）可以增、删选定商铺中的商品，修改商品价格。

（4）查询某一种商品名称，建立一个双向循环链表，结点信息是包含该商品的所有商铺编号，商铺名，信誉度，商品名称，价格，销量，按销量高至低排序，并可逐一显示。

（5）购买某一商铺的商品，修改单向链表中商品的信息的销量。

（6）任何的商铺信息变化，实现文件存储。

1.2 数据结构

主要采用链表：

商品：

**public class Storelist** {  
 **private String** id;//店铺编号  
 **private String** name;//店铺名  
 **private String** cre;//信誉值  
  
 **private Storelist** next\_StoreList;//下一个店铺  
  
 **private** List<**Merchanidsefile**> merchanidse;//店铺的商品  
  
  
 **public** List<**Merchanidsefile**> getMerchanidse() {  
 **return** merchanidse;  
 }  
  
 **public void** setMerchanidse(List<**Merchanidsefile**> merchanidse) {  
 **this**.merchanidse = merchanidse;  
 }  
  
 **public** Storelist() {  
 id = name = cre = **null**;  
 next\_StoreList = **null**;  
 }  
  
 **public String** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**String** id) {  
 **this**.id = id;  
 }  
  
 **public String** getName() {  
 **return** name;  
 }  
  
 **public void** setName(**String** name) {  
 **this**.name = name;  
 }  
  
 **public String** getCre() {  
 **return** cre;  
 }  
  
 **public void** setCre(**String** cre) {  
 **this**.cre = cre;  
 }  
  
 **public Storelist** getNext\_StoreList() {  
 **return** next\_StoreList;  
 }  
  
 **public void** setNext\_StoreList(**Storelist** next\_StoreList) {  
 **this**.next\_StoreList = next\_StoreList;  
 }  
  
 **public** Storelist(**String** id, **String** name, **String** cre, **Storelist** next\_StoreList,List<**Merchanidsefile**> merchanidseList) {  
 **this**.id = id;  
 **this**.name = name;  
 **this**.cre = cre;  
 **this**.next\_StoreList = next\_StoreList;  
 **this**.merchanidse=merchanidseList;  
 }  
}

商品：

**public class Merchanidse** {  
  
 **private String** name;//商品名称  
 **private double** price;//商品价格  
 **private int** Sales;//商品销量  
  
 **private String** id;//商品编号  
  
  
 **public Merchanidse** getPre() {  
 **return** pre;  
 }  
  
 **public void** setPre(**Merchanidse** pre) {  
 **this**.pre = pre;  
 }  
  
 **private Merchanidse** pre;  
  
  
 **public** Merchanidse() {  
 name = "机械键盘";  
 price = 100;  
 Sales = 200;  
  
 }  
  
 **public** Merchanidse(**String** name, **double** price, **int** sales, **String** id, **Merchanidse** next\_merchanidse) {  
  
 **this**.name = name;  
 **this**.price = price;  
 Sales = sales;  
 **this**.id = id;  
 **this**.next\_merchanidse = next\_merchanidse;  
 }  
  
 **public String** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**String** id) {  
 **this**.id = id;  
 }  
  
 **private Merchanidse** next\_merchanidse;  
  
 **public Merchanidse** getNext\_merchanidse() {  
 **return** next\_merchanidse;  
 }  
  
 **public void** setNext\_merchanidse(**Merchanidse** next\_merchanidse) {  
 **this**.next\_merchanidse = next\_merchanidse;  
 }  
  
 **public String** getName() {  
 **return** name;  
 }  
  
 **public void** setName(**String** name) {  
 **this**.name = name;  
 }  
  
 **public double** getPrice() {  
 **return** price;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.price = price;  
 }  
  
 **public int** getSales() {  
 **return** Sales;  
 }  
  
 **public void** setSales(**int** sales) {  
 Sales = sales;  
 }  
  
  
}

1.3 算法设计思想

增、删店铺：

从文件中读取出店铺信息并建立店铺链表，增删店铺时通过链表的增删来实现。

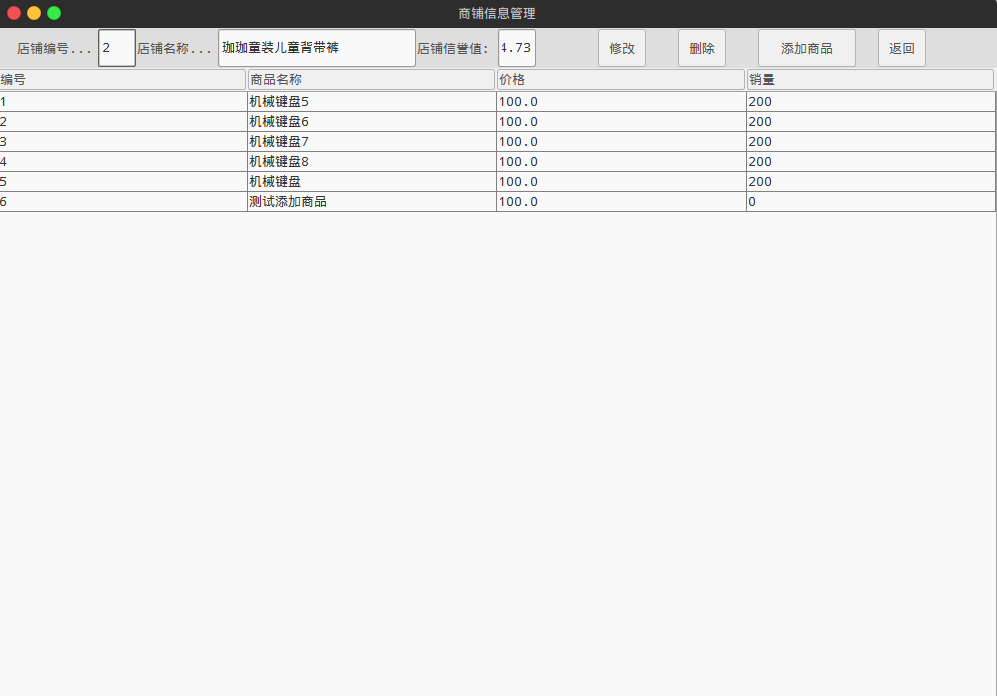
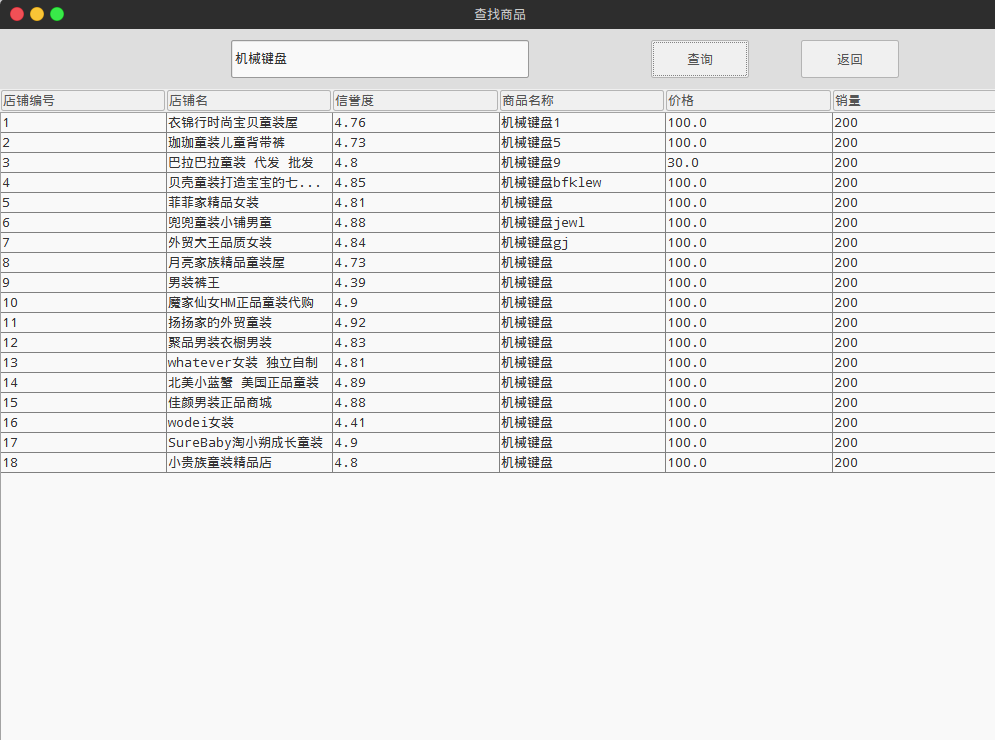
增、删商品：

从文件中读取所有的商品信息并建立商品链表，增删商品通过对商品链表节点的增删实现。

查询商品：

通过链表的查询节点操作实现商品的查找功能。

1.4测试数据和结果



1.5算法时间复杂度

初始化链表：

T（n）=O（n）

查找商品：

T（n）=O（n）

查找店铺：

T（n）=O（n）

1.6源代码

**package Bean**;  
  
**public class Merchanidse** {  
  
 **private String** name;//商品名称  
 **private double** price;//商品价格  
 **private int** Sales;//商品销量  
  
 **private String** id;//商品编号  
  
  
 **public Merchanidse** getPre() {  
 **return** pre;  
 }  
  
 **public void** setPre(**Merchanidse** pre) {  
 **this**.pre = pre;  
 }  
  
 **private Merchanidse** pre;  
  
  
 **public** Merchanidse() {  
 name = "机械键盘";  
 price = 100;  
 Sales = 200;  
  
 }  
  
 **public** Merchanidse(**String** name, **double** price, **int** sales, **String** id, **Merchanidse** next\_merchanidse) {  
  
 **this**.name = name;  
 **this**.price = price;  
 Sales = sales;  
 **this**.id = id;  
 **this**.next\_merchanidse = next\_merchanidse;  
 }  
  
 **public String** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**String** id) {  
 **this**.id = id;  
 }  
  
 **private Merchanidse** next\_merchanidse;  
  
 **public Merchanidse** getNext\_merchanidse() {  
 **return** next\_merchanidse;  
 }  
  
 **public void** setNext\_merchanidse(**Merchanidse** next\_merchanidse) {  
 **this**.next\_merchanidse = next\_merchanidse;  
 }  
  
 **public String** getName() {  
 **return** name;  
 }  
  
 **public void** setName(**String** name) {  
 **this**.name = name;  
 }  
  
 **public double** getPrice() {  
 **return** price;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.price = price;  
 }  
  
 **public int** getSales() {  
 **return** Sales;  
 }  
  
 **public void** setSales(**int** sales) {  
 Sales = sales;  
 }  
  
  
}

**package Bean**;  
  
**public class Merchanidsefile** {  
  
 **private String** name;  
 **private double** price;  
 **private int** Sales;  
  
 **private String** id;  
  
  
  
 **public** Merchanidsefile() {  
 }  
  
 **public** Merchanidsefile(**String** name, **double** price, **int** sales, **String** id, **Merchanidsefile** next\_merchanidse) {  
  
 **this**.name = name;  
 **this**.price = price;  
 Sales = sales;  
 **this**.id = id;  
 }  
  
 **public String** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**String** id) {  
 **this**.id = id;  
 }  
  
  
  
  
 **public String** getName() {  
 **return** name;  
 }  
  
 **public void** setName(**String** name) {  
 **this**.name = name;  
 }  
  
 **public double** getPrice() {  
 **return** price;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.price = price;  
 }  
  
 **public int** getSales() {  
 **return** Sales;  
 }  
  
 **public void** setSales(**int** sales) {  
 Sales = sales;  
 }  
  
  
}

**package Bean**;  
  
**import java.util.ArrayList**;  
**import java.util.**List;  
  
**public class Storefile** {  
 **private String** id;//编号  
 **private String** name;//店铺名  
 **private String** cre;//信誉值  
  
 **private** List<**Merchanidsefile**> merchanidseList;//店铺所有的商品  
  
  
 **public** Storefile() {  
 }  
  
 **public** List<**Merchanidsefile**> getMerchanidseList() {  
  
 **return** merchanidseList;  
 }  
  
 **public void** setMerchanidseList(List<**Merchanidsefile**> merchanidseList) {  
 **this**.merchanidseList = merchanidseList;  
 }  
  
 **public String** getId() {  
 **return** id;  
 }  
  
 **public void** setId(**String** id) {  
 **this**.id = id;  
 }  
  
 **public String** getName() {  
 **return** name;  
 }  
  
 **public void** setName(**String** name) {  
 **this**.name = name;  
 }  
  
 **public String** getCre() {  
 **return** cre;  
 }  
  
 **public void** setCre(**String** cre) {  
 **this**.cre = cre;  
 }  
  
  
  
 **public** Storefile(**String** id, **String** name, **String** cre, **Storefile** next\_Store) {  
 **this**.id = id;  
 **this**.name = name;  
 **this**.cre = cre;  
 }  
}

**package JFrame**;  
  
**import Bean.Merchanidsefile**;  
**import Bean.Storelist**;  
  
**import javax.swing.**\*;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.io.IOException**;  
  
**public class** AddMerchanidseJFrame **extends** JFrame {  
  
 **private** JLabel l\_usr;  
  
 **private** JLabel l\_who;  
  
  
 **private** JLabel l\_name;//商品名称  
 **private** JLabel l\_price;//商品价格  
  
  
 **private** JTextField t\_name;  
 **private** JTextField t\_price;  
  
 **private** Storelist storelist;  
  
  
 **private** JButton ok;  
 **private** JButton cancel;  
  
 **public** AddMerchanidseJFrame(String username, Storelist m\_storeList) **throws** HeadlessException {  
 **super**();  
 setTitle("新增商品");  
 setSize(600, 550);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
 storelist=m\_storeList;  
 l\_usr = **new** JLabel("店铺名称： " + m\_storeList.getName());  
  
 l\_usr.setBounds(100, 0, 240, 40);  
  
  
 l\_who = **new** JLabel("店铺信誉值：" + m\_storeList.getCre());  
 l\_who.setBounds(350, 0, 200, 40);  
  
  
 l\_name = **new** JLabel("商品名称：");  
 l\_name.setBounds(200, 80, 80, 30);  
  
 t\_name = **new** JTextField();  
 t\_name.setBounds(280, 80, 100, 30);  
  
  
 l\_price = **new** JLabel("商品价格：");  
 l\_price.setBounds(200, 120, 80, 30);  
  
 t\_price = **new** JTextField();  
 t\_price.setBounds(280, 120, 100, 30);  
  
  
 cancel = **new** JButton("取消");  
 cancel.setBounds(210, 200, 60, 30);  
  
  
 ok = **new** JButton("确定");  
 ok.setBounds(310, 200, 60, 30);  
  
  
 add(l\_usr);  
 add(l\_who);  
  
 add(l\_name);  
 add(t\_name);  
  
 add(l\_price);  
 add(t\_price);  
  
 add(ok);  
 add(cancel);  
  
 SetListener();  
  
 setSize(600, 550);  
 setVisible(**true**);  
 }  
  
  
 **private void** SetListener() {  
 ok.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String name = t\_name.getText().trim();  
 String price = t\_price.getText().trim();  
  
 **if** (name.equals("") || price.equals("")) {  
 **JOptionPane**.showMessageDialog(**null**, "商品名称和商品价格不可为空！");  
 } **else** {  
 **Merchanidsefile** merchanidsefile = **new** Merchanidsefile();  
 merchanidsefile.setName(name);  
 merchanidsefile.setPrice(**Double**.parseDouble(price));  
 merchanidsefile.setSales(0);  
  
 merchanidsefile.setId(**String**.valueOf(storelist.getMerchanidse().size() + 1));  
 storelist.getMerchanidse().add(merchanidsefile);  
  
 **try** {  
 **StoreInformationJFrame**.SaveChange(storelist);  
 } **catch** (**IOException** e1) {  
 e1.printStackTrace();  
 }  
 GoPreJFrame();  
 }  
  
 }  
 });  
 cancel.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 GoPreJFrame();  
 }  
 });  
 }  
  
  
 **private void** GoPreJFrame() {  
 **StoreInformationJFrame** storeInformationJFrame = **new** StoreInformationJFrame(storelist.getMerchanidse(), storelist);  
 storeInformationJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
  
}

**package JFrame**;  
  
**import Bean.Storelist**;  
  
**import javax.swing.**\*;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.io.IOException**;  
  
**public class** AddStoreJFrame **extends** JFrame {  
  
 **private** JLabel l\_usr;  
  
 **private** JLabel l\_who;  
  
  
 **private** JLabel l\_name;//商铺名称  
 **private** JLabel l\_cre;//商铺信誉值  
  
 **private** JTextField t\_name;  
 **private** JTextField t\_cre;  
  
  
 **private** JButton ok;  
 **private** JButton cancel;  
  
 **public** AddStoreJFrame(String username) **throws** HeadlessException {  
 **super**();  
 setTitle("新增商铺");  
 setSize(600, 550);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
 l\_usr = **new** JLabel("当前用户： " + username);  
  
 l\_usr.setBounds(140, 0, 120, 40);  
  
  
 l\_who = **new** JLabel("当前角色：管理员");  
 l\_who.setBounds(290, 0, 200, 40);  
  
  
 l\_name = **new** JLabel("商铺名称：");  
 l\_name.setBounds(200, 80, 80, 30);  
  
 t\_name = **new** JTextField();  
 t\_name.setBounds(280, 80, 100, 30);  
  
  
 l\_cre = **new** JLabel("商铺信誉值：");  
 l\_cre.setBounds(200, 120, 80, 30);  
  
 t\_cre = **new** JTextField();  
 t\_cre.setBounds(280, 120, 100, 30);  
  
  
 cancel = **new** JButton("取消");  
 cancel.setBounds(210, 200, 60, 30);  
  
  
 ok = **new** JButton("确定");  
 ok.setBounds(310, 200, 60, 30);  
  
  
 add(l\_usr);  
 add(l\_who);  
  
 add(l\_name);  
 add(t\_name);  
  
 add(l\_cre);  
 add(t\_cre);  
  
 add(ok);  
 add(cancel);  
  
 SetListener();  
  
  
 setVisible(**true**);  
 }  
  
  
 **private void** SetListener() {  
 ok.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **String** name = t\_name.getText().trim();  
 **String** cre = t\_cre.getText().trim();  
  
 **if** (name.equals("") || cre.equals("")) {  
 **JOptionPane**.showMessageDialog(**null**, "店铺名称和店铺信誉值不可为空！");  
 } **else** {  
 **Storelist** storeList = **new** Storelist();  
 storeList.setName(name);  
 storeList.setCre(cre);  
 storeList.setNext\_StoreList(**null**);  
 storeList.setId(**null**);  
 **try** {  
 **HomeJFrame**.SaveChange(storeList);  
 GoHomeJFrame();  
 } **catch** (**IOException** e1) {  
 e1.printStackTrace();  
 }  
 }  
  
 }  
 });  
 cancel.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 GoHomeJFrame();  
 }  
 });  
 }  
  
 **private void** GoHomeJFrame() {  
  
 **HomeJFrame** homeJFrame = **new** HomeJFrame("dmrf");  
 homeJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
  
 }  
}

**package** JFrame;  
  
**import** Bean.Merchanidsefile;  
**import** Bean.Storelist;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.IOException;  
  
**public class** EditMerchanidseJFrame **extends** JFrame {  
  
  
 **private** JLabel l\_name;//商品名称  
 **private** JLabel l\_price;//商品价格  
 **private** JLabel l\_sale;//商品销量  
  
 **private** JTextField t\_price;  
 **private** JTextField t\_name;  
 **private** JTextField t\_sale;  
  
  
 **private** JButton ok;  
 **private** JButton cancel;  
 **private** JButton de;  
  
 **private int** index;  
 **private** Storelist storelist;  
 **private** Merchanidsefile m\_merchanidsefile;  
  
  
 **public** EditMerchanidseJFrame(Storelist m\_storeList, Merchanidsefile merchanidsefile) **throws** HeadlessException {  
 **super**();  
 setTitle("修改商品信息");  
 setSize(400, 400);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 **this**.index = index;  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
 storelist = m\_storeList;  
 m\_merchanidsefile = merchanidsefile;  
  
 l\_name = **new** JLabel("商品名称");  
 l\_name.setBounds(100, 20, 80, 30);  
  
 t\_name = **new** JTextField();  
 t\_name.setBounds(180, 20, 100, 30);  
 t\_name.setEditable(**false**);  
 t\_name.setText(merchanidsefile.getName());  
  
  
 l\_price = **new** JLabel("商品价格：");  
 l\_price.setBounds(100, 60, 80, 30);  
  
 t\_price = **new** JTextField();  
 t\_price.setBounds(180, 60, 100, 30);  
 t\_price.setText(String.valueOf(merchanidsefile.getPrice()));  
  
  
 l\_sale = **new** JLabel("商品销量：");  
 l\_sale.setBounds(100, 100, 80, 30);  
  
  
 t\_sale = **new** JTextField();  
 t\_sale.setBounds(180, 100, 100, 30);  
 t\_sale.setText(String.valueOf(m\_merchanidsefile.getSales()));  
 t\_sale.setEditable(**false**);  
  
  
 cancel = **new** JButton("取消");  
 cancel.setBounds(70, 140, 60, 30);  
  
 de = **new** JButton("删除");  
 de.setBounds(190, 140, 60, 30);  
  
 ok = **new** JButton("确定");  
 ok.setBounds(320, 140, 60, 30);  
  
  
 add(de);  
  
 add(l\_name);  
 add(t\_name);  
  
 add(l\_price);  
 add(t\_price);  
  
 add(l\_sale);  
 add(t\_sale);  
  
 add(ok);  
 add(cancel);  
  
 SetListener();  
  
 setVisible(**true**);  
 }  
  
  
 **private void** SetListener() {  
  
 de.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 // StoreInformationJFrame.DeleteMerchandse(m\_merchanidsefile.getId(), storelist);  
 storelist.getMerchanidse().remove(Integer.valueOf(m\_merchanidsefile.getId()) - 1);  
 // System.out.println("id"+m\_merchanidsefile.getId());  
  
 **try** {  
 StoreInformationJFrame.SaveChange(storelist);  
 } **catch** (IOException e1) {  
 e1.printStackTrace();  
 }  
 GoPreJFrame();  
 }  
 });  
 cancel.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 GoPreJFrame();  
 }  
 });  
  
 ok.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **String** price = t\_price.getText().trim();  
  
 **if** (price.equals("")) {  
 **JOptionPane**.showMessageDialog(**null**, "商品价格不可为空！");  
 } **else** {  
 m\_merchanidsefile.setPrice(**Double**.parseDouble(price));  
 **StoreInformationJFrame**.SaveEdit(m\_merchanidsefile, storelist);  
 **try** {  
 **StoreInformationJFrame**.SaveChange(storelist);  
 } **catch** (**IOException** e1) {  
 e1.printStackTrace();  
 }  
 GoPreJFrame();  
 }  
 }  
 });  
 }  
  
  
 **private void** GoPreJFrame() {  
 StoreInformationJFrame storeInformationJFrame = **new** StoreInformationJFrame(storelist.getMerchanidse(), storelist);  
 storeInformationJFrame.setVisible(**true**);  
  
 **this**.setVisible(**false**);  
 }  
  
  
}

**package JFrame**;  
  
**import javax.swing.**\*;  
**import javax.swing.table.DefaultTableModel**;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.awt.event.**MouseAdapter;  
**import java.awt.event.MouseEvent**;  
**import java.io.**\*;  
**import java.util.**List;  
  
**import Bean.Merchanidse**;  
**import Bean.Merchanidsefile**;  
**import Bean.Storelist**;  
**import List.Store\_List**;  
**import Utils.JsonFileToStoreList\_Utils**;  
**import Utils.StoreListToJsonFile\_Utils**;  
**import JFrame.AddStoreJFrame**;  
  
**public class** HomeJFrame **extends** JFrame {  
  
 // 默认表格模型  
 **private** DefaultTableModel model = **null**;  
 **private** JTable table = **null**;  
  
 **private** JButton addBtn = **null**;  
 **private** JButton back = **null**;  
 **private** JButton searchmer = **null**;//查找商品  
  
 **private** JPopupMenu jpm;  
 **private** JMenuItem Browse;  
 **private** JMenuItem Delele;  
  
 **private** JLabel l\_usr;  
  
 **private** JLabel l\_who;  
  
 **private static** Store\_List store\_list;  
  
 **public** HomeJFrame(String usrname) **throws** HeadlessException {  
 **super**();  
 setTitle("商家概览");  
 //setSize(1000, 800);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
 l\_usr = **new** JLabel("当前用户： " + usrname);  
  
 l\_usr.setBounds(0, 0, 120, 40);  
  
  
 l\_who = **new** JLabel("当前角色：管理员");  
 l\_who.setBounds(150, 0, 200, 40);  
  
 addBtn = **new** JButton("添加店铺");  
 addBtn.setBounds(880, 0, 100, 40);  
  
  
 back = **new** JButton("切换用户");  
 back.setBounds(750, 0, 100, 40);  
  
 searchmer = **new** JButton("查询商品");  
 searchmer.setBounds(620, 0, 100, 40);  
  
 add(searchmer);  
 add(l\_usr);  
 add(l\_who);  
 add(addBtn, BorderLayout.NORTH);  
 add(back);  
  
  
 String[][] datas = {};  
 String[] titles = {"编号", "店铺名", "信誉度"};  
 model = **new** DefaultTableModel(datas, titles);  
 table = **new** JTable(model);  
 table.setRowHeight(20);  
 JScrollPane jScrollPane = **new** JScrollPane(table);  
 jScrollPane.setBounds(0, 40, 1000, 760);  
 add(jScrollPane);  
  
  
 store\_list = JsonFileToStoreList\_Utils.JsonFileToStoreList();  
 Storelist bean = store\_list.getHead\_storeList();  
  
  
 **while** (**true**) {  
 bean = bean.getNext\_StoreList();  
 **if** (bean == **null**) {  
 **break**;  
 } **else** {  
  
 model.addRow(**new** String[]{bean.getId(), bean.getName(), bean.getCre()});  
 }  
 }  
  
  
 addBtn.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 GoAddStoreFrame();  
  
  
 }  
 });  
  
  
 table.addMouseListener(**new** MouseAdapter() {  
 @Override  
 **public void** mouseClicked(MouseEvent e) {  
 **super**.mouseClicked(e);  
  
  
 **if** (e.getClickCount() == 2) {  
  
 }  
 **if** (e.getButton() == MouseEvent.BUTTON3) {  
 //在table显示  
 jpm = **new** JPopupMenu();  
  
 //表格 的rowAtPoint方法返回坐标所在的行号，参数为坐标类型，  
 **int** i = table.rowAtPoint(e.getPoint());  
  
 Browse = **new** JMenuItem("查看详细信息");  
 Delele = **new** JMenuItem("删除店铺信息");  
 jpm.add(Browse);  
 jpm.add(Delele);  
  
 **int** x = e.getX();  
 **int** y = e.getY();  
  
 jpm.show(table, e.getX(), e.getY());  
  
  
 Browse.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** r = table.getSelectedRow();  
 **int** c = table.getSelectedColumn();  
 **int** d = table.getRowHeight();  
 //Object value = table.getValueAt(r, c);  
 String info = r + "行" + c + "列" + " " + d;  
  
 JOptionPane.showMessageDialog(**null**, info);  
  
 }  
 });  
  
 Delele.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String info = "x:" + x + " y:" + y;  
  
 JOptionPane.showMessageDialog(**null**, info);  
  
 }  
 });  
  
  
 } **else if** (e.getButton() == MouseEvent.BUTTON1) {  
 **int** r = table.getSelectedRow();  
 **int** c = table.getSelectedColumn();  
 //得到选中的单元格的值，表格中都是字符串  
  
 */\* Object value = table.getValueAt(r, c);  
 String info = r + "行" + c + "列值 : " + value.toString();  
  
 JOptionPane.showMessageDialog(null, info);\*/* GoStoreInfoFRame(store\_list.get\_Store(r + 1).getMerchanidse(), r + 1);  
 }  
  
 }  
 });  
  
 back.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 GoToLoginJFrame();  
 }  
 });  
  
 setSize(1000, 800);  
 setLocationRelativeTo(**null**);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setVisible(**true**);  
  
  
 searchmer.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 GoSearchMJFrame(store\_list);  
 }  
 });  
 }  
  
 **private void** GoSearchMJFrame(Store\_List store\_list) {  
 SearchMJFrame searchMJFrame = **new** SearchMJFrame(store\_list);  
 searchMJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
 **private void** GoToLoginJFrame() {  
 LogInJFrame logInJFrame = **new** LogInJFrame();  
 logInJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
 **private void** GoAddStoreFrame() {  
 AddStoreJFrame addStoreJFrame = **new** AddStoreJFrame("dmrf");  
 **this**.setVisible(**false**);  
 }  
  
  
 **private void** GoStoreInfoFRame(List<Merchanidsefile> merchanidse\_list, **int** r) {  
  
 StoreInformationJFrame storeInformationJFrame = **new** StoreInformationJFrame(merchanidse\_list, store\_list.get\_Store(r));  
 storeInformationJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
 **public static void** SaveChange(**Storelist** storeList) **throws IOException** {  
 storeList.setId(**String**.valueOf(*store\_list*.getLength() + 1));  
 *store\_list*.Insert\_Store(*store\_list*.getLength() + 1, storeList);  
 **StoreListToJsonFile\_Utils**.StoreListToJsonFile(*store\_list*);  
 **System**.out.println("增加成功～");  
 }  
  
 **public static void** SaveChange2(**Storelist** storeList) **throws IOException** {  
 *store\_list*.Change(storeList.getId(), storeList);  
 **StoreListToJsonFile\_Utils**.StoreListToJsonFile(*store\_list*);  
 **System**.out.println("增加成功～");  
 }  
  
  
 **public static void** DeleleStore(**String** id) **throws IOException** {  
 *store\_list*.delete\_Store(**Integer**.parseInt(id));  
 **StoreListToJsonFile\_Utils**.StoreListToJsonFile(*store\_list*);  
  
 *store\_list* = **JsonFileToStoreList\_Utils**.JsonFileToStoreList();  
 **Storelist** bean = *store\_list*.getHead\_storeList();  
 **System**.out.println("删除成功～");  
  
  
 }  
  
  
}

**package JFrame**;  
  
**import javax.swing.**\*;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
  
**public class LogInJFrame extends JFrame** {  
  
 **JTextField** tf\_username;  
 **JPasswordField** tf\_password;  
  
  
 **public** LogInJFrame() **throws HeadlessException** {  
 **super**();  
 setTitle("登录");  
 setSize(600, 400);  
 setDefaultCloseOperation(**JFrame**.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 **Dimension** dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
  
 **final JLabel** l\_username = **new** JLabel();  
 l\_username.setText("请输入用户名：");  
 l\_username.setBounds(175, 50, 100, 25);  
 getContentPane().add(l\_username);  
  
  
 tf\_username = **new** JTextField();  
 tf\_username.setBounds(275, 50, 150, 25);  
 getContentPane().add(tf\_username);  
  
  
 **final JLabel** l\_password = **new** JLabel();  
 l\_password.setText("请输入密码：");  
 l\_password.setBounds(175, 150, 100, 25);  
 getContentPane().add(l\_password);  
  
  
 tf\_password = **new** JPasswordField();  
 tf\_password.setBounds(275, 150, 150, 25);  
 getContentPane().add(tf\_password);  
  
  
 **final JButton** bt\_login = **new** JButton();  
 bt\_login.setText("登录");  
 bt\_login.setBounds(250, 250, 100, 25);  
 getContentPane().add(bt\_login);  
  
  
 bt\_login.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **String** username = tf\_username.getText().trim();  
 **String** passworld = tf\_password.getText().trim();  
 **if** (username.equals("DMRF") && passworld.equals("161630213")) {  
 //JOptionPane.showMessageDialog(null, "登录成功！");  
 GoHomeFRame();  
  
 } **else** {  
  
 **JOptionPane**.showMessageDialog(**null**, "账号密码错误！");  
  
 }  
 }  
 });  
 }  
  
  
 **private void** GoHomeFRame() {  
  
 **HomeJFrame** homeJFrame = **new** HomeJFrame("DMRF");  
 homeJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
}

**package JFrame**;  
  
**import Bean.Merchanidse**;  
  
**import javax.swing.**\*;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
  
**import Bean.Merchanidsefile**;  
**import List.Merchanidse\_List**;  
  
**public class** MerchanidseInfoJFrame **extends** JFrame {  
  
  
 JLabel l\_id;  
 JLabel l\_cre;  
 JLabel l\_name;  
 JLabel l\_price;  
  
 JTextField t\_id;  
 JTextField t\_name;  
 JTextField t\_cre;  
 JTextField t\_price;  
  
  
 JButton edit;  
 JButton delete;  
  
 **boolean** flag=**false**;  
  
 **private** Merchanidse\_List merchanidse\_list;  
  
  
 **public** MerchanidseInfoJFrame(**Merchanidsefile** merchanidse) **throws HeadlessException** {  
 **super**();  
 setTitle("商品管理");  
  
 setDefaultCloseOperation(**JFrame**.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 **Dimension** dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
  
 l\_id = **new** JLabel("商品编号：");  
 l\_id.setBounds(200, 20, 80, 30);  
  
 t\_id = **new** JTextField(merchanidse.getId());  
 t\_id.setBounds(280, 20, 120, 30);  
 t\_id.setEditable(**false**);  
  
 add(l\_id);  
 add(t\_id);  
  
  
 l\_name = **new** JLabel("商品名称：");  
 l\_name.setBounds(200, 120, 80, 30);  
  
 t\_name = **new** JTextField(merchanidse.getName());  
 t\_name.setBounds(280, 120, 120, 30);  
 t\_name.setEditable(**false**);  
  
 add(l\_name);  
 add(t\_name);  
  
  
 l\_price = **new** JLabel("商品价格：");  
 l\_price.setBounds(200, 220, 80, 30);  
  
 t\_price = **new** JTextField(String.valueOf(merchanidse.getPrice()));  
 t\_price.setBounds(280, 220, 120, 30);  
 t\_price.setEditable(**false**);  
  
 add(l\_price);  
 add(t\_price);  
  
 l\_cre = **new** JLabel("商品销量：");  
 l\_cre.setBounds(200, 320, 80, 30);  
  
 t\_cre = **new** JTextField(merchanidse.getSales());  
 t\_cre.setBounds(280, 320, 120, 30);  
 t\_cre.setEditable(**false**);  
  
  
 add(l\_cre);  
 add(t\_cre);  
  
  
 edit = **new** JButton("修改商品信息");  
 edit.setBounds(165, 420, 115, 30);  
 add(edit);  
  
 delete = **new** JButton("删除该商品");  
 delete.setBounds(335, 420, 115, 30);  
 add(delete);  
  
  
  
 edit.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 **if** (!flag) {  
 t\_cre.setEditable(**true**);  
 t\_name.setEditable(**true**);  
 t\_price.setEditable(**true**);  
 edit.setText("确定");  
 flag = **true**;  
 } **else** {  
  
 String new\_name = t\_name.getText().trim();  
 String new\_cre = t\_cre.getText().trim();  
 String new\_price = t\_price.getText().trim();  
  
  
 **if** (!new\_cre.equals("") && !new\_name.equals("")) {  
 t\_cre.setEditable(**false**);  
 t\_name.setEditable(**false**);  
 t\_price.setEditable(**false**);  
 edit.setText("修改");  
 flag = **false**;  
  
 */\*  
 用这里得到的new\_name和new\_cre更新链表数据  
 \*/* } **else** {  
 System.out.println("空～");  
 JOptionPane.showMessageDialog(**null**, "店铺名称和店铺信誉值不可为空！");  
 }  
  
 }  
  
 }  
 });  
  
  
  
 setSize(600, 550);  
 setVisible(**true**);  
  
 }  
}

**package JFrame**;  
  
**import Bean.Merchanidsefile**;  
**import Bean.Storelist**;  
**import List.Store\_List**;  
  
  
**import javax.swing.**\*;  
**import javax.swing.table.DefaultTableModel**;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.awt.event.**MouseAdapter;  
**import java.awt.event.MouseEvent**;  
**import java.util.ArrayList**;  
**import java.util.**List;  
  
  
**public class** SearchMJFrame **extends** JFrame {  
  
 // 默认表格模型  
 **private** DefaultTableModel model = **null**;  
 **private** JTable table = **null**;  
  
 **private** JButton addBtn = **null**;  
 **private** JButton back = **null**;  
 **private** JButton searchmer = **null**;//查找商品  
  
 **private** JPopupMenu jpm;  
 **private** JMenuItem Browse;  
 **private** JMenuItem Delele;  
  
 **private** JTextField t\_search;  
  
 **private** JLabel l\_usr;  
  
 **private** JLabel l\_who;  
 List<Storelist> store\_list1 = **new** ArrayList<Storelist>();  
 **private static** Store\_List m\_store\_list;  
  
 **public** SearchMJFrame(Store\_List store\_list) **throws** HeadlessException {  
 **super**();  
 setTitle("查找商品");  
 setSize(1000, 800);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
  
 m\_store\_list = store\_list;  
  
 t\_search = **new** JTextField();  
 t\_search.setBounds(230, 10, 300, 40);  
  
 searchmer = **new** JButton("查询");  
 searchmer.setBounds(650, 10, 100, 40);  
  
 back = **new** JButton("返回");  
 back.setBounds(800, 10, 100, 40);  
  
  
 add(searchmer);  
 add(t\_search);  
  
  
 add(back);  
  
  
 String[][] datas = {};  
 String[] titles = {"店铺编号", "店铺名", "信誉度", "商品名称", "价格", "销量"};  
 model = **new** DefaultTableModel(datas, titles);  
 table = **new** JTable(model);  
 table.setRowHeight(20);  
 JScrollPane jScrollPane = **new** JScrollPane(table);  
 jScrollPane.setBounds(0, 60, 1000, 760);  
 add(jScrollPane);  
  
  
 table.addMouseListener(**new** MouseAdapter() {  
 @Override  
 **public void** mouseClicked(MouseEvent e) {  
 **super**.mouseClicked(e);  
  
  
 **if** (e.getButton() == MouseEvent.BUTTON3) {  
 //在table显示  
 jpm = **new** JPopupMenu();  
  
 //表格 的rowAtPoint方法返回坐标所在的行号，参数为坐标类型，  
 **int** i = table.rowAtPoint(e.getPoint());  
  
 Browse = **new** JMenuItem("查看详细信息");  
 Delele = **new** JMenuItem("删除店铺信息");  
 jpm.add(Browse);  
 jpm.add(Delele);  
  
 **int** x = e.getX();  
 **int** y = e.getY();  
  
 jpm.show(table, e.getX(), e.getY());  
  
  
 Browse.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **int** r = table.getSelectedRow();  
 **int** c = table.getSelectedColumn();  
 **int** d = table.getRowHeight();  
 //Object value = table.getValueAt(r, c);  
 **String** info = r + "行" + c + "列" + " " + d;  
  
 **JOptionPane**.showMessageDialog(**null**, info);  
  
 }  
 });  
  
 Delele.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **String** info = "x:" + x + " y:" + y;  
  
 **JOptionPane**.showMessageDialog(**null**, info);  
  
 }  
 });  
  
  
 } **else if** (e.getButton() == **MouseEvent**.BUTTON1) {  
 **int** r = table.getSelectedRow();  
 **int** c = table.getSelectedColumn();  
  
 GoStoreInfoFRame(store\_list.get\_Store(r + 1).getMerchanidse(), r + 1);  
 }  
  
  
 }  
 });  
  
 back.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
  
 GoPreJFrame();  
 }  
 });  
  
 searchmer.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
  
  
 **while** (model.getRowCount()>0){  
 model.removeRow(0);  
 }  
 store\_list1.clear();  
  
 List<Merchanidsefile> flag = **new** ArrayList<Merchanidsefile>();  
  
 String name = t\_search.getText().trim();  
  
 **if** (name.equals("")) {  
 JOptionPane.showMessageDialog(**null**, "商品名称不可为空！");  
 } **else** {  
 Storelist bean = m\_store\_list.getHead\_storeList();  
  
  
 **while** (**true**) {  
 bean = bean.getNext\_StoreList();  
 **if** (bean == **null**) {  
 **break**;  
 } **else** {  
 List<Merchanidsefile> merchanidse = bean.getMerchanidse();  
 **if** (merchanidse != **null**) {  
 **for** (**int** i = 0; i < merchanidse.size(); i++) {  
 **if** (merchanidse.get(i).getName().contains(name)) {  
 Storelist s = **new** Storelist();  
 s.setNext\_StoreList(bean.getNext\_StoreList());  
 s.setId(bean.getId());  
 s.setCre(bean.getCre());  
 s.setName(bean.getName());  
 s.setMerchanidse(bean.getMerchanidse());  
 store\_list1.add(s);  
 flag.add(merchanidse.get(i));  
 **break**;  
 }  
 }  
  
 }  
  
 }  
 }  
  
  
 **for** (**int** i = 0; i < store\_list1.size() - 1; i++) {  
 **for** (**int** j = i + 1; j < store\_list1.size(); j++) {  
 **if** (flag.get(i).getSales() < flag.get(j).getSales()) {  
 Storelist s;  
 s = store\_list1.get(i);  
 store\_list1.set(i, store\_list1.get(j));  
 store\_list1.set(j, s);  
  
 Merchanidsefile si;  
 si = flag.get(i);  
 flag.set(i, flag.get(j));  
 flag.set(j, si);  
 }  
 }  
 }  
  
  
 **for** (**int** i = 0; i < store\_list1.size(); i++) {  
 model.addRow(**new** String[]{store\_list1.get(i).getId(),  
 store\_list1.get(i).getName(), store\_list1.get(i).getCre(), flag.get(i).getName(),  
 String.valueOf(flag.get(i).getPrice()), String.valueOf(flag.get(i).getSales())});  
 }  
  
 }  
 }  
 });  
 }  
  
 **private void** GoPreJFrame() {  
 HomeJFrame homeJFrame = **new** HomeJFrame("DMRF");  
 homeJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
 **private void** GoStoreInfoFRame(List<Merchanidsefile> merchanidse\_list, **int** r) {  
  
 StoreInformationJFrame storeInformationJFrame = **new** StoreInformationJFrame(merchanidse\_list, store\_list1.get(r));  
 storeInformationJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
}

**package JFrame**;  
  
**import javax.swing.**\*;  
**import javax.swing.table.DefaultTableModel**;  
**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.awt.event.**MouseAdapter;  
**import java.awt.event.MouseEvent**;  
**import java.io.IOException**;  
**import java.util.**List;  
  
**import Bean.Merchanidse**;  
**import Bean.Merchanidsefile**;  
**import Bean.Storelist**;  
**import List.Merchanidse\_List**;  
  
**public class** StoreInformationJFrame **extends** JFrame {  
  
 **private boolean** flag = **false**;  
  
 // 默认表格模型  
 **private** DefaultTableModel model = **null**;  
 **private** JTable table = **null**;  
  
 **private** JButton edit = **null**;  
 **private** JButton delete = **null**;  
 **private** JButton add = **null**;  
 **private** JButton back = **null**;  
  
  
 **private** JPopupMenu jpm;  
 **private** JMenuItem Browse;  
 **private** JMenuItem Delele;  
  
 **private** JTextField t\_id;  
 **private** JTextField t\_name;  
 **private** JTextField t\_cre;  
  
  
 **private** JLabel l\_id;  
 **private** JLabel l\_name;  
 **private** JLabel l\_cre;  
  
  
 **private static** List<Merchanidsefile> m\_merchanidse\_list;  
  
 **private** Storelist m\_storeList;  
  
 **public** StoreInformationJFrame(List<Merchanidsefile> merchanidse\_list, Storelist storeList) **throws** HeadlessException {  
 **super**();  
 **if** (storeList == **null**) {  
 System.out.println("store为空！");  
 **return**;  
 } **else** {  
 m\_storeList = storeList;  
 m\_merchanidse\_list = merchanidse\_list;  
 }  
  
 setTitle("商铺信息管理");  
 //setSize(1000, 800);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = getToolkit(); // 获得Toolkit对象  
 Dimension dimension = toolkit.getScreenSize(); // 获得Dimension对象  
 **int** screenHeight = dimension.height; // 获得屏幕的高度  
 **int** screenWidth = dimension.width; // 获得屏幕的宽度  
 **int** frm\_Height = **this**.getHeight(); // 获得窗体的高度  
 **int** frm\_width = **this**.getWidth(); // 获得窗体的宽度  
 setLocation((screenWidth - frm\_width) / 2,  
 (screenHeight - frm\_Height) / 2); // 使用窗体居中显示  
  
 getContentPane().setLayout(**null**);  
  
  
 l\_id = **new** JLabel("店铺编号： ");  
 t\_id = **new** JTextField(storeList.getId());  
 t\_id.setEditable(**false**);  
  
  
 l\_name = **new** JLabel("店铺名称： ");  
 t\_name = **new** JTextField(storeList.getName());  
 t\_name.setEditable(**false**);  
  
  
 l\_cre = **new** JLabel("店铺信誉值： ");  
 t\_cre = **new** JTextField(storeList.getCre());  
 t\_cre.setEditable(**false**);  
  
  
 l\_id.setBounds(20, 0, 80, 40);  
 t\_id.setBounds(100, 0, 40, 40);  
 l\_name.setBounds(140, 0, 80, 40);  
 t\_name.setBounds(220, 0, 200, 40);  
 l\_cre.setBounds(420, 0, 100, 40);  
 t\_cre.setBounds(500, 0, 40, 40);  
  
  
 edit = **new** JButton("修改");  
 edit.setBounds(600, 0, 50, 40);  
  
 delete = **new** JButton("删除");  
 delete.setBounds(680, 0, 50, 40);  
  
  
 add = **new** JButton("添加商品");  
 add.setBounds(760, 0, 100, 40);  
  
  
 back = **new** JButton("返回");  
 back.setBounds(880, 0, 50, 40);  
  
 add(l\_name);  
 add(l\_id);  
 add(l\_cre);  
 add(t\_name);  
 add(t\_cre);  
 add(t\_id);  
 add(edit);  
 add(delete);  
 add(add);  
 add(back);  
  
  
 String[][] datas = {};  
 String[] titles = {"编号", "商品名称", "价格", "销量"};  
 model = **new** DefaultTableModel(datas, titles);  
 table = **new** JTable(model);  
 table.setRowHeight(20);  
 JScrollPane jScrollPane = **new** JScrollPane(table);  
 jScrollPane.setBounds(0, 40, 1000, 800);  
  
 add(jScrollPane);  
  
  
 **for** (**int** i = 0; i < merchanidse\_list.size(); i++) {  
 merchanidse\_list.get(i).setId(String.valueOf(i + 1));  
 model.addRow(**new** String[]{merchanidse\_list.get(i).getId(), merchanidse\_list.get(i).getName(), String.valueOf(merchanidse\_list.get(i).getPrice()), String.valueOf(merchanidse\_list.get(i).getSales())});  
 }  
  
 SetListener();  
  
  
 setSize(1000, 800);  
 setLocationRelativeTo(**null**);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setVisible(**true**);  
  
  
 }  
  
 **private void** SetListener() {  
 edit.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 **if** (!flag) {  
 t\_cre.setEditable(**true**);  
 t\_name.setEditable(**true**);  
 edit.setText("确定");  
 flag = **true**;  
 } **else** {  
  
 String new\_name = t\_name.getText().trim();  
 String new\_cre = t\_cre.getText().trim();  
 **if** (!new\_cre.equals("") && !new\_name.equals("")) {  
 t\_cre.setEditable(**false**);  
 t\_name.setEditable(**false**);  
 edit.setText("修改");  
 flag = **false**;  
  
 */\*  
 用这里得到的new\_name和new\_cre更新链表数据  
 \*/* } **else** {  
 **System**.out.println("空～");  
 **JOptionPane**.showMessageDialog(**null**, "店铺名称和店铺信誉值不可为空！");  
 }  
  
 }  
  
 }  
 });  
  
 table.addMouseListener(**new** MouseAdapter() {  
 @Override  
 **public void** mouseClicked(**MouseEvent** e) {  
 **super**.mouseClicked(e);  
 **int** r = table.getSelectedRow();  
 **int** c = table.getSelectedColumn();  
 //得到选中的单元格的值，表格中都是字符串  
 **Object** value = table.getValueAt(r, c);  
 //String info = r + "行" + c + "列值 : " + value.toString();  
 //JOptionPane.showMessageDialog(null, info);  
  
 GoMerchanidseInfoFRame(*m\_merchanidse\_list*.get(r));  
 }  
 });  
  
  
 add.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 GoAddMerchanidseJFrame(m\_storeList);  
 }  
 });  
  
 delete.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **try** {  
 **HomeJFrame**.DeleleStore(m\_storeList.getId());  
 GoHomeFrame();  
  
 } **catch** (**IOException** e1) {  
 e1.printStackTrace();  
 }  
 }  
 });  
  
 back.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 GoHomeJFRame();  
 }  
 });  
  
 }  
  
 **private void** GoMerchanidseInfoFRame(Merchanidsefile merchanidse) {  
 EditMerchanidseJFrame editMerchanidseJFrame = **new** EditMerchanidseJFrame(m\_storeList, merchanidse);  
 editMerchanidseJFrame.setVisible(**true**);  
 }  
  
 **private void** GoAddMerchanidseJFrame(Storelist m\_storeList) {  
 AddMerchanidseJFrame addMerchanidseJFrame = **new** AddMerchanidseJFrame("DMRF", m\_storeList);  
 addMerchanidseJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
 **private void** GoHomeJFRame() {  
 HomeJFrame homeJFrame = **new** HomeJFrame("DMRF");  
 homeJFrame.setVisible(**true**);  
 **this**.setVisible(**false**);  
 }  
  
  
 **private void** GoHomeFrame() {  
 **new** HomeJFrame("dmrf");  
 **this**.setVisible(**false**);  
 }  
  
 **public static void** SaveChange(Storelist storelist) **throws** IOException {  
 HomeJFrame.SaveChange2(storelist);  
 }  
  
  
 **public static void** DeleteMerchandse(String id, Storelist storelist) {  
  
 }  
  
 **public static void** SaveEdit(Merchanidsefile m\_merchanidsefile, Storelist storelist) {  
 }  
}

**package List**;  
  
**import Bean.Merchanidse**;  
  
  
**public class Merchanidse\_List** {  
 **private Merchanidse** head\_merchanidse;  
 **private int** length;  
  
  
 **public Merchanidse** getHead\_merchanidse() {  
 **return** head\_merchanidse;  
 }  
  
 **public void** setHead\_merchanidse(**Merchanidse** head\_merchanidse) {  
 **this**.head\_merchanidse = head\_merchanidse;  
 }  
  
 **public int** getLength() {  
 **return** length;  
 }  
  
 **public void** setLength(**int** length) {  
 **this**.length = length;  
 }  
  
 **public** Merchanidse\_List() {  
 head\_merchanidse = **new** Merchanidse();  
 length = 0;  
 **System**.out.println("初始化成功～");  
 }  
  
 **public boolean** ClearList() {  
 **Merchanidse** merchanidse = head\_merchanidse.getNext\_merchanidse();  
 **Merchanidse** merchanidse1;  
 **while** (merchanidse != **null**) {  
 merchanidse1 = merchanidse.getNext\_merchanidse();  
 merchanidse = **null**;  
 merchanidse = merchanidse1;  
 }  
  
 head\_merchanidse.setNext\_merchanidse(**null**);  
 **System**.out.println("清空成功～");  
 **return true**;  
  
 }  
  
 **public boolean** IsEmpty() {  
  
 **if** (length == 0) {  
 **System**.out.println("链表为空～");  
 **return true**;  
 } **else** {  
 **System**.out.println("链表不为空～");  
 **return false**;  
 }  
 }  
  
 **public Merchanidse** GetElem(**int** n) {  
 **if** (n < 1 || n > length) {  
 **System**.out.println("链表不为空～");  
 **return null**;  
 }  
 **Merchanidse** p = head\_merchanidse;  
 **while** (n != 0) {  
 p = p.getNext\_merchanidse();  
 n--;  
 }  
  
 **return** p;  
 }  
  
  
  
  
}  
  
  
  
*/\*  
//  
// Created by dmrf on 17-10-  
  
  
  
  
int LocateElem(SqList L, int e, bool compare(int a, int b)) {//返回第一个与e符合compare的元素的位序  
 if (ListEmpty(L)) {  
 cout << "链表为空！" << endl;  
 return -1;  
 }  
  
 LNode \*p = L.head->next;  
 for (int i = 0; i < L.length; ++i) {  
 if (compare(e, p->date)) {  
 cout << "找到符合条件的元素,位序为：" << i+1 << endl;  
 return i;  
 }  
 p = p->next;  
 }  
 cout << "没有找到符合条件的元素" << endl;  
 return -1;  
}  
  
bool PriorElem(SqList L, int cur\_e, int &pre\_e) {//求前驱  
 if (ListEmpty(L)) {  
 cout << "链表为空！" << endl;  
 return false;  
 }  
  
 LNode \*p = L.head;  
 LNode \*q = L.head->next;  
 if (cur\_e == q->date) {  
 cout << cur\_e << "为首元素，无前驱" << endl;  
 return false;  
 }  
  
 p = q;  
 q = q->next;  
  
 while (q != NULL) {  
 if (q->date == cur\_e) {  
 pre\_e = p->date;  
 return true;  
 }  
  
 p = q;  
 q = q->next;  
 }  
  
 cout << cur\_e << "无前驱" << endl;  
 return false;  
  
}  
  
bool NextElem(SqList L, int cur\_e, int &next\_e) {  
 if (ListEmpty(L)) {  
 cout << "链表为空！" << endl;  
 return false;  
 }  
 LNode \*p, \*q;  
 p = L.head->next;  
 while (p != NULL) {  
 q = p->next;  
 if (p->date == cur\_e) {  
 if (q != NULL) {  
 next\_e = q->date;  
 return true;  
 } else {  
 cout << cur\_e << "无后继" << endl;  
 return false;  
 }  
 }  
 p=q;  
 }  
 cout << cur\_e << "无后继" << endl;  
 return false;  
}  
  
bool ListInsert(SqList &L, int i, int q) {//在第i个元素的位置插入元素q  
  
 if (L.head == NULL) {  
 printf("L为空！\n");  
 return false;  
 }  
  
 if (i < 1 || i > L.length + 1) {  
 cout << "所要插入的元素的下标不合法！" << endl;  
 return false;  
 }  
  
 LNode \*p;  
 p = L.head;  
 while (i > 1) {  
 p = p->next;  
 if (p == NULL) {  
 cout << "插入失败" << endl;  
 return false;  
 }  
 i--;  
 }  
  
 LNode \*lNode = new LNode();  
 lNode->date = q;  
  
 lNode->next = p->next;  
 p->next = lNode;  
 L.length++;  
 cout << "插入成功" << endl;  
 return true;  
}  
  
bool ListDelete(SqList &L, int i, int &e) {//删除第i个数据元素，并用e返回其值  
 if (ListEmpty(L)) {  
 cout << "链表为空！" << endl;  
 return false;  
 }  
  
 if (i < 1 || i > L.length) {  
 cout << "所要删除的元素的下标不合法！" << endl;  
 return false;  
 }  
  
 LNode \*p;  
 LNode \*q;  
 p = L.head;  
 while (i > 1) {//将p指向要删除节点的前一个元素  
 p = p->next;  
 if (p == NULL) {  
 cout << "删除失败" << endl;  
 return false;  
 }  
 i--;  
 }  
  
 q = p->next;//q指向要删除的节点  
 p->next = q->next;  
 free(q);  
 L.length--;  
  
 cout << "删除成功" << endl;  
 return true;  
  
}  
  
  
bool ListTraverse(SqList L, bool Visit(int k)) {  
 if (ListEmpty(L)) {  
 cout << "链表为空！" << endl;  
 return false;  
 }  
 LNode \*p;  
 p = L.head->next;  
 while (p != NULL) {  
 Visit(p->date);  
 p=p->next;  
 }  
 return true;  
}  
  
  
#endif //ADTLIST\_ADT\_LIST\_SEQUENCE\_H  
 \*/*

**package List**;  
  
**import Bean.Storefile**;  
**import Bean.Storelist**;  
  
**public class Store\_List** {  
  
 **private Storelist** head\_storeList;  
 **private int** length;  
  
  
 **public** Store\_List() {  
 head\_storeList = **new** Storelist();  
 length = 0;  
 }  
  
 **public Storelist** getHead\_storeList() {  
 **return** head\_storeList;  
 }  
  
 **public void** setHead\_storeList(**Storelist** head\_storeList) {  
 **this**.head\_storeList = head\_storeList;  
 }  
  
 **public int** getLength() {  
 **return** length;  
 }  
  
 **public void** setLength(**int** length) {  
 **this**.length = length;  
 }  
  
  
 **public void** Change(**String** id, **Storelist** storelist) {  
 **Storelist** storelist1 = head\_storeList;  
 **Storelist** storelist2 = storelist1.getNext\_StoreList();  
 **for** (**int** i = 1; i <= length + 1; i++) {  
 **if** (storelist2.getId().equals(id)) {  
 storelist2 = storelist;  
  
 }  
 }  
 }  
  
  
 **public Storelist** Get(**int** i) {  
 **Storefile** storefile;  
 **if** (i < 1 || i > length + 1) {  
 **System**.out.println("要获取的元素下标不合法！");  
 **return null**;  
 } **else** {  
 **Storelist** storelist = head\_storeList;  
  
 **for** (; i > 0; i--) {  
 storelist = storelist.getNext\_StoreList();  
 }  
 **return** storelist;  
 }  
 }  
  
 **public boolean** Insert\_Store(**int** i, **Storelist** q) {//在第i个元素的位置插入元素q  
 **if** (head\_storeList == **null**) {  
 **System**.out.println("L为空！\n");  
 **return false**;  
 }  
  
 **if** (q == **null**) {  
 **return false**;  
 }  
 **if** (i < 1 || i > length + 1) {  
 **System**.out.println("所要插入的元素的下标不合法！");  
 **return false**;  
 }  
  
 **Storelist** storeList;  
 storeList = head\_storeList;  
 **while** (i > 1) {  
 storeList = storeList.getNext\_StoreList();  
 **if** (storeList == **null**) {  
 **System**.out.println("插入失败");  
 **return false**;  
 }  
 i--;  
 }  
  
 **Storelist** p = storeList.getNext\_StoreList();  
  
 q.setNext\_StoreList(p);  
  
 storeList.setNext\_StoreList(q);  
  
 length++;  
 //System.out.println("插入成功");  
 **return true**;  
  
  
 }  
  
 **public Storelist** get\_Store(**int** r) {  
  
 **if** (head\_storeList == **null**) {  
 **System**.out.println("链表为空！");  
 **return null**;  
 }  
 **if** (r < 1 || r > length) {  
 **System**.out.println("所要获取的数据下标不合法！");  
 **return null**;  
 }  
  
 **Storelist** p = head\_storeList;  
 **while** (r != 0) {  
 p = p.getNext\_StoreList();  
 r--;  
 }  
  
 **return** p;  
 }  
  
  
 **public boolean** delete\_Store(**int** i) {  
 **if** (head\_storeList == **null**) {  
 **System**.out.println("链表为空！");  
 **return false**;  
 }  
  
 **if** (i < 1 || i > length) {  
 **System**.out.println("所要删除的元素的下标不合法！");  
 **return false**;  
 }  
  
 **Storelist** p;  
 **Storelist** q;  
 p = head\_storeList;  
 **while** (i > 1) {//将p指向要删除节点的前一个元素  
 p = p.getNext\_StoreList();  
 **if** (p == **null**) {  
 **System**.out.println("删除失败！");  
 **return false**;  
 }  
 i--;  
 }  
  
 q = p.getNext\_StoreList();//q指向要删除的节点  
 p.setNext\_StoreList(q.getNext\_StoreList());  
  
 p = p.getNext\_StoreList();  
  
 **while** (p != **null**) {  
 **int** k = **Integer**.parseInt(p.getId());  
 k--;  
 p.setId(**String**.valueOf(k));  
 p = p.getNext\_StoreList();  
 }  
 length--;  
 **System**.out.println("删除成功！");  
  
 **return true**;  
 }  
}

**package** Utils;  
  
**import** Bean.Storefile;  
**import** Bean.Storelist;  
**import** List.Store\_List;  
**import** com.google.gson.Gson;  
**import** com.google.gson.JsonArray;  
**import** com.google.gson.JsonElement;  
**import** com.google.gson.JsonParser;  
  
**import** java.io.\*;  
**import** java.util.ArrayList;  
**import** java.util.Iterator;  
**import** java.util.List;  
  
**public class** JsonFileToStoreList\_Utils {  
  
  
 **public static Store\_List** JsonFileToStoreList() {  
 **Store\_List** store\_list = **new** Store\_List();  
  
 **String** json = ReadFile("/home/dmrf/文档/DataStructure/CourseDesign/StoreInformation2.json");  
  
 **Gson** gson = **new** Gson();  
 **JsonParser** jsonParser = **new** JsonParser();  
 JsonElement jsonElement = jsonParser.parse(json); //将json字符串转换成JsonElement  
 **JsonArray** jsonArray = jsonElement.getAsJsonArray(); //将JsonElement转换成JsonArray  
 Iterator it = jsonArray.iterator(); //Iterator处理  
  
  
 List<**Storefile**> storefileList = **new** ArrayList<**Storefile**>();  
  
 **while** (it.hasNext()) { //循环  
 jsonElement = (JsonElement) it.next(); //提取JsonElement  
 json = jsonElement.toString(); //JsonElement转换成String  
 **Storefile** beanf = gson.fromJson(json, **Storefile**.**class**); //String转化成JavaBean  
 // bean.setId(String.valueOf(Integer.valueOf(bean.getId())-6));  
 storefileList.add(beanf);  
 // i++;  
 //System.out.println("i：" + i);  
 }  
  
  
  
 **for** (**int** i = storefileList.size()-1; i >=0; i--) {  
 **Storelist** storelist;  
 **if** (i==storefileList.size()-1){  
 storefileList.get(i).getId();  
 storefileList.get(i).getName();  
  
 storelist = **new** Storelist(storefileList.get(i).getId(), storefileList.get(i).getName(), storefileList.get(i).getCre(),**null**,storefileList.get(i).getMerchanidseList());  
 }**else** {  
 storelist = **new** Storelist(storefileList.get(i).getId(), storefileList.get(i).getName(), storefileList.get(i).getCre(),store\_list.get\_Store(1),storefileList.get(i).getMerchanidseList());  
 }  
  
 store\_list.Insert\_Store(1, storelist);  
  
 }  
  
 **return** store\_list;  
  
 }  
  
  
 //读文件，返回字符串  
 **public static** String ReadFile(String path) {  
 File file = **new** File(path);  
 BufferedReader reader = **null**;  
 String laststr = "";  
 **try** {  
 //System.out.println("以行为单位读取文件内容，一次读一整行：");  
 //reader = new BufferedReader(new FileReader(file),"GBK");  
 reader = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(path), "gbk"));  
 String tempString = **null**;  
 //一次读入一行，直到读入null为文件结束  
 **while** ((tempString = reader.readLine()) != **null**) {  
 //显示行号  
 laststr = laststr + tempString;  
 }  
 reader.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (reader != **null**) {  
 **try** {  
 reader.close();  
 } **catch** (IOException e1) {  
 }  
 }  
 }  
 **return** laststr;  
 }  
}

**package Utils**;  
  
**import Bean.Store**;  
**import Bean.Storefile**;  
**import Bean.Storelist**;  
**import List.Store\_List**;  
**import com.google.gson.Gson**;  
**import com.google.gson.JsonArray**;  
**import com.google.gson.**JsonElement;  
**import com.google.gson.JsonParser**;  
  
**import java.io.**\*;  
**import java.util.ArrayList**;  
**import java.util.**Iterator;  
**import java.util.**List;  
  
**public class MakeDataUtiles2** {  
  
  
 **public static** List<**Store**> JsonFileToStoreList() {  
  
 **String** json = ReadFile("/home/dmrf/文档/DataStructure/CourseDesign/Data.json");  
  
 **Gson** gson = **new** Gson();  
 **JsonParser** jsonParser = **new** JsonParser();  
 JsonElement jsonElement = jsonParser.parse(json); //将json字符串转换成JsonElement  
 **JsonArray** jsonArray = jsonElement.getAsJsonArray(); //将JsonElement转换成JsonArray  
 Iterator it = jsonArray.iterator(); //Iterator处理  
  
  
 List<**Store**> storefileList = **new** ArrayList<**Store**>();  
  
 **while** (it.hasNext()) { //循环  
 jsonElement = (JsonElement) it.next(); //提取JsonElement  
 json = jsonElement.toString(); //JsonElement转换成String  
 **Store** beanf = gson.fromJson(json, **Store**.**class**); //String转化成JavaBean  
 // bean.setId(String.valueOf(Integer.valueOf(bean.getId())-6));  
 storefileList.add(beanf);  
 // i++;  
 //System.out.println("i：" + i);  
 }  
  
  
  
 **return** storefileList;  
  
 }  
  
  
 //读文件，返回字符串  
 **public static String** ReadFile(**String** path) {  
 **File** file = **new** File(path);  
 **BufferedReader** reader = **null**;  
 **String** laststr = "";  
 **try** {  
 //System.out.println("以行为单位读取文件内容，一次读一整行：");  
 //reader = new BufferedReader(new FileReader(file),"GBK");  
 reader = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(path), "gbk"));  
 **String** tempString = **null**;  
 //一次读入一行，直到读入null为文件结束  
 **while** ((tempString = reader.readLine()) != **null**) {  
 //显示行号  
 laststr = laststr + tempString;  
 }  
 reader.close();  
 } **catch** (**IOException** e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (reader != **null**) {  
 **try** {  
 reader.close();  
 } **catch** (**IOException** e1) {  
 }  
 }  
 }  
 **return** laststr;  
 }  
}

**package Utils**;  
  
**import Bean.Merchanidse**;  
**import Bean.Store**;  
**import Bean.Storefile**;  
**import Bean.Storelist**;  
**import List.Store\_List**;  
**import com.google.gson.Gson**;  
  
**import java.io.**\*;  
**import java.util.ArrayList**;  
**import java.util.**List;  
  
**public class MakeDataUtils** {  
 **public** MakeDataUtils() **throws IOException** {  
 **int** i = 1;  
  
  
 **File** file = **new** File("/home/dmrf/文档/DataStructure/CourseDesign/StoreInformation2.json");  
  
 **BufferedWriter** bw = **null**;  
 **try** {  
 bw = **new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream(file), "GBK"));  
 } **catch** (**UnsupportedEncodingException** e) {  
 e.printStackTrace();  
 } **catch** (**FileNotFoundException** e) {  
 e.printStackTrace();  
 }  
  
 List<Store> store\_list;  
 store\_list = MakeDataUtiles2.JsonFileToStoreList();  
  
  
 **int** n = 0;  
  
 **while** (**true**) {  
  
 **if** (n>=store\_list.size()) {  
 **break**;  
 } **else** {  
  
  
 Gson gson = **new** Gson();  
 Storefile storefile = **new** Storefile();  
  
 storefile.setCre(store\_list.get(n).getCre());  
 storefile.setId(store\_list.get(n).getId());  
 storefile.setName(store\_list.get(n).getName());  
  
 List<Merchanidse> merchanidseList = **new** ArrayList<Merchanidse>();  
 **Merchanidse** merchanidse = **new** Merchanidse();  
 merchanidse.setId("1");  
 merchanidseList.add(merchanidse);  
 merchanidse.setId("2");  
 merchanidseList.add(merchanidse);  
 merchanidse.setId("3");  
 merchanidseList.add(merchanidse);  
 merchanidse.setId("4");  
 merchanidseList.add(merchanidse);  
 merchanidse.setId("5");  
 merchanidseList.add(merchanidse);  
 // storefile.setMerchanidseList(merchanidseList);  
  
 **String** str = gson.toJson(storefile);  
 **if** (store\_list.get(i).getId().equals("1")) {  
 str = "[" + str + ",";  
 } **else** {  
 str = str + ",";  
 }  
 n++;  
  
  
 bw.write(str);  
 bw.newLine();  
 }  
 }  
  
 bw.flush();  
 **System**.out.println("写入文件成功~");  
  
 }  
}

**package Utils**;  
  
**import Bean.Storefile**;  
**import Bean.Storelist**;  
**import List.Store\_List**;  
**import com.google.gson.Gson**;  
  
**import java.io.**\*;  
  
**public class StoreListToJsonFile\_Utils** {  
  
 **public static void** StoreListToJsonFile(**Store\_List** store\_list) **throws IOException** {  
 **int** i = 1;  
  
  
 **File** file = **new** File("/home/dmrf/文档/DataStructure/CourseDesign/StoreInformation2.json");  
  
 **BufferedWriter** bw = **new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream(file), "GBK"));  
  
 **Storelist** s = store\_list.getHead\_storeList().getNext\_StoreList();  
  
  
 **while** (**true**) {  
  
 **if** (s == **null**) {  
 **break**;  
 } **else** {  
  
  
 **Gson** gson = **new** Gson();  
 **Storefile** storefile = **new** Storefile();  
 storefile.setCre(s.getCre());  
 storefile.setId(s.getId());  
 storefile.setMerchanidseList(s.getMerchanidse());  
 storefile.setName(s.getName());  
 **String** str = gson.toJson(storefile);  
  
 **if** (s.getId().equals("1")) {  
 str = "[" + str + ",";  
 } **else** {  
 str = str + ",";  
 }  
  
 **if** ((s = s.getNext\_StoreList()) == **null**) {  
 str=str.substring(0,str.lastIndexOf(","));  
 str = str + "]";  
 }  
  
 bw.write(str);  
 bw.newLine();  
 }  
 }  
  
 bw.flush();  
 **System**.out.println("写入文件成功~");  
 }  
  
}

# 二、

2.1 题目简介

迷宫问题（必做）（栈与队列），(选做)（深度搜索，广度搜索）

[问题描述]

利用栈操作实现迷宫问题求解。

[基本要求]

（1）从文件中读取数据，生成模拟迷宫地图，不少于10行10列。

（2）给出任意入口和出口，显示输出迷宫路线。

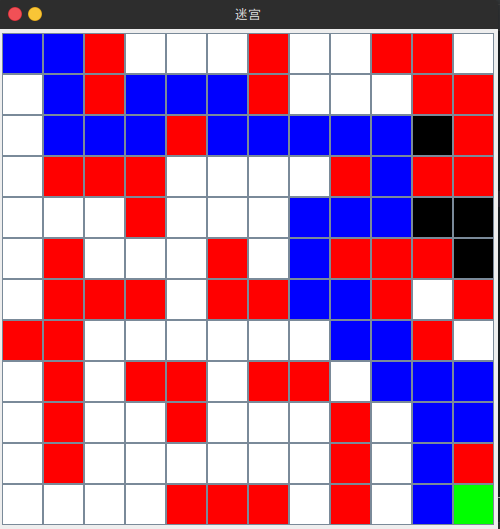
2.2 数据结构

**public class Loc** {  
 **public int** x;//点的横坐标  
  
 **public int** y;//点的纵坐标  
  
 **public** Loc(**int** x, **int** y) {  
 **this**.x = x;  
 **this**.y = y;  
 }  
  
 **public** Loc() {  
 }  
  
 **public int** getX() {  
  
 **return** x;  
 }  
  
 **public void** setX(**int** x) {  
 **this**.x = x;  
 }  
  
 **public int** getY() {  
 **return** y;  
 }  
  
 **public void** setY(**int** y) {  
 **this**.y = y;  
 }  
}

2.3 算法设计思想

使用栈对迷宫中的路径进行深度优先搜索，将可走的坐标进栈，若走到不可走的坐标则开始出栈，出栈后向未走过的方向试探，直至最后走到出口，这时栈中的坐标即为所求的路径。

2.4测试数据和结果



2.5算法时间复杂度

n为矩阵的长。

时间复杂度为O(n²)

2.6源代码

**public class Loc** {  
 **public int** x;//点的横坐标  
  
 **public int** y;//点的纵坐标  
  
 **public** Loc(**int** x, **int** y) {  
 **this**.x = x;  
 **this**.y = y;  
 }  
  
 **public** Loc() {  
 }  
  
 **public int** getX() {  
  
 **return** x;  
 }  
  
 **public void** setX(**int** x) {  
 **this**.x = x;  
 }  
  
 **public int** getY() {  
 **return** y;  
 }  
  
 **public void** setY(**int** y) {  
 **this**.y = y;  
 }  
}

**import java.awt.**\*;  
**import java.awt.event.ActionEvent**;  
**import java.awt.event.**ActionListener;  
**import java.io.**\*;  
**import java.util.Stack**;  
  
**import javax.swing.**\*;  
  
  
**public class FindPath** {  
  
 **private static double** *speech* = 1;  
 **private static int** *wid*;  
 **private static int** *hei*;  
 **private static boolean** *flag* = **false**;  
 **private static Loc** *b*;  
 **private static Loc** *en*;  
 **JFrame** frame; //窗体,是整个迷宫的容器  
 **JPanel** panel;  
 **JPanel** panel2;  
 **JButton** button[];  
 **JButton** speadreduce;  
 **JButton** ok;  
 **JLabel** tip;  
 **JButton** speadadd;  
 **private static int**[] *map*;  
 **private Stack** stack = **new** Stack();  
 **private static FindPath** *ai*;  
  
 **public static void** main(**String**[] args) **throws IOException**, **InterruptedException** {  
  
  
 **String** path = "/home/dmrf/文档/DataStructure/CourseDesign/maze.txt";  
 *map* = GetData(path);  
  
  
 *wid* = *hei* = (**int**) **Math**.sqrt(*map*.length);  
  
 **int** i = 0;  
 *b* = **new** Loc();  
 *en* = **new** Loc();  
 *ai* = **new** FindPath();  
  
  
 *b*=**new** Loc(1,1);  
 *en*=**new** Loc(12,12);  
 *ai*.domaze(*map*, *b*, *en*);  
 }  
  
 **private static int**[] GetData(**String** path) **throws IOException** {  
  
 **BufferedReader** bufferedReader = **new** BufferedReader(**new** FileReader(**new** File(path)));  
 **String** str;  
 **String** re = "";  
 **int** flag = 0;  
 **while** ((str = bufferedReader.readLine()) != **null**) {  
 re = re + " " + str;  
 flag++;  
 }  
  
 **String**[] split = re.split(" ");  
 **int**[] map = **new int**[split.length - 1];  
  
 **for** (**int** i = 0; i < map.length; i++) {  
 map[i] = **Integer**.valueOf(split[i + 1]);  
 }  
 **return** map;  
  
 }  
  
 FindPath() {  
  
 **int** i;  
 frame = **new** JFrame("迷宫");  
 frame.setBounds(300, 240, 500, 500); //调整迷宫出现的位置及大小  
 frame.setResizable(**false**);//窗体不可拉伸  
  
  
 panel = **new** JPanel();  
  
 // frame.getContentPane().add(panel); //将面板添加到窗体中  
 panel.setLayout(**new** GridLayout(*wid*, *hei*)); //panel用网格布局,10行10列  
  
  
 panel2 = **new** JPanel();  
 // frame.getContentPane().add(panel);  
 panel2.setLayout(**null**);  
 **JSplitPane** splitPane = **new** JSplitPane(**JSplitPane**.HORIZONTAL\_SPLIT);  
  
  
 speadadd = **new** JButton("加快速度");  
 speadadd.setBounds(10, 40, 100, 30);  
 speadreduce = **new** JButton("降低速度");  
 speadreduce.setBounds(120, 40, 100, 30);  
 panel2.add(speadadd);  
 panel2.add(speadreduce);  
 tip = **new** JLabel("请点击起点和终点后点击确定开始执行..");  
 tip.setBounds(10, 0, 400, 30);  
 panel2.add(tip);  
 ok = **new** JButton("确定");  
 ok.setBounds(10, 80, 80, 30);  
 panel2.add(ok);  
  
 //splitPane.setLeftComponent(panel);  
 //splitPane.setRightComponent(panel2);  
  
  
 frame.getContentPane().add(panel, **BorderLayout**.CENTER);  
  
  
 button = **new** JButton[*wid* \* *hei*];  
 **for** (i = 0; i < button.length; i++) {  
 button[i] = **new** JButton(); //创建按键的名字，Integer型的名字为i的字符串  
  
 button[i].setBackground(**Color**.white);  
  
  
 button[i].setSize(10, 10);  
 panel.add(button[i]);  
 }  
  
 **for** (**int** p = 0; p < *wid*; p++) {  
 **for** (**int** j = 0; j < *hei*; j++) {  
 **if** (*map*[p \* *wid* + j] == 1) {//通道  
 button[p \* *wid* + j].setBackground(**Color**.white);  
 } **else** {  
 button[p \* *wid* + j].setBackground(**Color**.red);  
 }  
 }  
 }  
 frame.setDefaultCloseOperation(frame.EXIT\_ON\_CLOSE); //声明点“X”图标后结束窗体所在的应用程序  
 frame.setVisible(**true**); //表明以上创建的所有窗体、按键等组件都是可见的  
  
 SetListener();  
 }  
  
 **private void** SetListener() {  
 speadadd.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
  
 **if** (*speech* >= 0.3) {  
 *speech* -= 0.3;  
 }  
 }  
 });  
 speadreduce.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **if** (*speech* <= 2) {  
 *speech* += 0.3;  
 }  
 }  
 });  
  
 **int** n = -1;  
 **for** (**int** i = 0; i < button.length; i++) {  
 n++;  
 **int** finalN = n;  
 button[i].addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **if** (*b* == **null**) {  
 button[finalN].setBackground(**Color**.GREEN);  
  
 *b*.x = finalN % *wid* + 1;  
 *b*.y = finalN / *wid* + 1;  
 tip.setText("起点坐标：（" + *b*.x + " , " + *b*.y + ")");  
 } **else** {  
 button[finalN].setBackground(**Color**.GREEN);  
  
 *en*.x = finalN % *wid* + 1;  
 *en*.y = finalN / *wid* + 1;  
 **String** s = tip.getText();  
 s = s + ",终点坐标：（" + *en*.x + " , " + *en*.y + ")";  
 tip.setText(s);  
  
 }  
 }  
 });  
 }  
  
 ok.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(**ActionEvent** e) {  
 **if** (*b* == **null** || *en* == **null**) {  
 **JOptionPane**.showMessageDialog(**null**, "请正确选择起点和终点！");  
 } **else** {  
  
 *flag* = **true**;  
  
 }  
 }  
 });  
 }  
  
  
 **public void** domaze(**int**[] map, **Loc** bebin, **Loc** end) **throws InterruptedException** {  
  
  
 **int** num = *wid*;  
 **int**[][] d = **new int**[num + 3][num + 3];  
  
 **int** x, y;  
 x = y = 1;  
  
  
 **for** (**int** j = 1; j <= num + 2; ++j) {  
 **for** (**int** i = 1; i <= num + 2; ++i) {  
 **if** (j == 1 || j == num + 2 || i == 1 || i == num + 2) {  
 d[j][i] = 0;  
 } **else** {  
 d[j][i] = map[(i - 2) \* num + j - 2];  
 }  
  
 }  
 }  
  
  
 **Stack**<**Loc**> S = **new** Stack<**Loc**>();  
  
  
 bebin.x++;  
 bebin.y++;  
 end.x++;  
 end.y++;  
  
 **Loc** cur;  
  
 S.push(bebin);  
 button[(bebin.y - 2) \* *wid* + bebin.x - 2].setBackground(**Color**.GREEN);  
  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
 d[bebin.x][bebin.y] = 2;  
  
 **int** stapnum = 1;  
  
 **Loc** t;  
 **while** (!S.empty()) {  
 cur = S.get(S.size() - 1);  
 **if** (cur.x == end.x && cur.y == end.y) {  
  
 **break**;  
 } **else** {  
 **Loc** temp = **new** Loc();  
 **if** (d[cur.x + 1][cur.y] != 0 && d[cur.x + 1][cur.y] != 2) {  
 d[cur.x + 1][cur.y] = 2;  
 temp.x = cur.x + 1;  
 temp.y = cur.y;  
  
 t = S.get(S.size() - 1);  
 S.push(temp);  
 button[(t.y - 2) \* *wid* + t.x - 2].setBackground(**Color**.blue);  
 button[(temp.y - 2) \* *wid* + temp.x - 2].setBackground(**Color**.GREEN);  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
  
 } **else if** (d[cur.x][cur.y + 1] != 0 && d[cur.x][cur.y + 1] != 2) {  
 d[cur.x][cur.y + 1] = 2;  
 temp.x = cur.x;  
 temp.y = cur.y + 1;  
  
 t = S.get(S.size() - 1);  
 S.push(temp);  
 button[(t.y - 2) \* *wid* + t.x - 2].setBackground(**Color**.blue);  
 button[(temp.y - 2) \* *wid* + temp.x - 2].setBackground(**Color**.GREEN);  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
 } **else if** (d[cur.x - 1][cur.y] != 0 && d[cur.x - 1][cur.y] != 2) {  
 d[cur.x - 1][cur.y] = 2;  
 temp.x = cur.x - 1;  
 temp.y = cur.y;  
  
 t = S.get(S.size() - 1);  
 S.push(temp);  
 button[(t.y - 2) \* *wid* + t.x - 2].setBackground(**Color**.blue);  
 button[(temp.y - 2) \* *wid* + temp.x - 2].setBackground(**Color**.GREEN);  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
 } **else if** (d[cur.x][cur.y - 1] != 0 && d[cur.x][cur.y - 1] != 2) {  
 d[cur.x][cur.y - 1] = 2;  
 temp.x = cur.x;  
 temp.y = cur.y - 1;  
  
 t = S.get(S.size() - 1);  
 S.push(temp);  
 button[(t.y - 2) \* *wid* + t.x - 2].setBackground(**Color**.blue);  
 button[(temp.y - 2) \* *wid* + temp.x - 2].setBackground(**Color**.GREEN);  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
 } **else** {  
 t = S.pop();  
 button[(t.y - 2) \* *wid* + t.x - 2].setBackground(**Color**.black);  
 **Thread**.sleep((**long**) (*speech* \* 1000));  
 }  
  
 }  
  
  
 }  
  
  
 }  
  
  
}

# 三、

3.1 题目简介

二叉树的应用 (必做)（二叉树）

[问题描述]

编程实现二叉树的建立，先序、中序、后序（递归和非递归方法）、层序遍历。求二叉树的高度、宽度，结点数。判断是否为二叉排序树。

[基本要求]

（1） 从文件中读入建树信息，树的节点数目不小于20个，树的高度不小于4。

（2） 采用二叉链表结构。

（3） 至少2组输入数据，分别是二叉排序树和不是二叉排序树，

3.2 数据结构

**typedef struct** BiNode {  
 **struct** BiNode \*lchild;  
 **struct** BiNode \*rchild;  
 **struct** BiNode \*parent;  
 **char** c;  
 **int** data;  
 **int** node\_count;  
} BiNode, \*BiTree;

3.3 算法设计思想

先序遍历递归：

先遍历根节点，然后递归遍历左子树，最后递归遍历右子树。

先序遍历非递归：

先遍历根节点，然后将右节点入栈，再将左节点入栈，然后弹栈，每出来一个元素就先遍历然后将右左子节点分别进栈，直至最后栈空。

中序遍历递归：

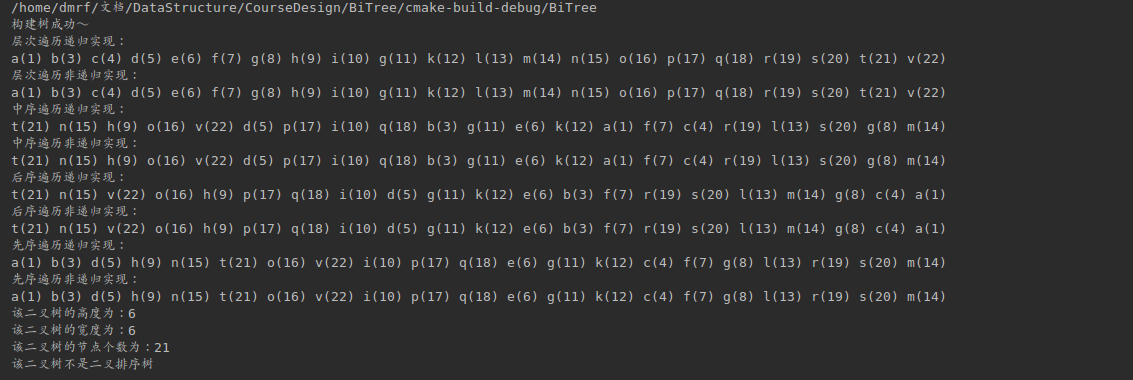
首先中序遍历左字数，然后中序遍历右子树，最后遍历根。

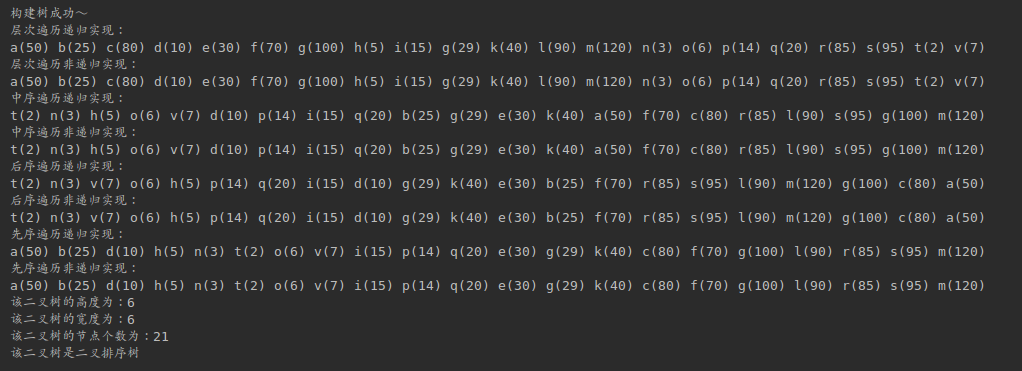
中序遍历非递归、后序遍历递归、非递归原理类似。

判断是否为二叉排序树：

先判断根和其两个直接子节点是否满足二叉排序树的要求，若满足，再递归判断左子树和右子树，若左右子树都满足，则该树为二叉排序树。

3.4测试数据和结果





3.5算法时间复杂度

遍历时间复杂度：O（n）

求深度时间复杂度：O（n）

3.6源代码

//  
// Created by dmrf on 17-11-20.  
//  
  
#ifndef INC\_1\_BITREE\_H  
#define INC\_1\_BITREE\_H  
#define TElemType **int**#include <queue>  
#include <stack>  
#include <iostream>  
#include <cmath>  
  
**using namespace** std;  
  
  
**typedef struct** BiNode {  
 **struct** BiNode \*lchild;  
 **struct** BiNode \*rchild;  
 **struct** BiNode \*parent;  
 **char** c;  
 **int** data;  
 **int** node\_count;  
} BiNode, \*BiTree;  
  
**int** Depth(BiTree biTree);  
  
**void** LevelOrderTraversePrint(BiTree biTree, **int** depth);  
**int** GetNum(BiTree biTree, **int** depth);  
  
**void** InitBitree(BiTree &biTree) {  
 biTree = **new** BiNode();  
 biTree->lchild = NULL;  
 biTree->rchild = NULL;  
 biTree->parent = NULL;  
 biTree->data = 0;  
 biTree->node\_count = 1;  
 biTree->c = NULL;  
 // cout << "初始化成功～" << endl;  
}  
  
**void** DestroyBiTree(BiTree biTree) {  
  
 **if** (biTree == NULL) {  
 **return**;  
 }  
 **if** (biTree->lchild != NULL) {  
 DestroyBiTree(biTree->lchild);  
 }  
  
 **if** (biTree->rchild != NULL) {  
 DestroyBiTree(biTree->rchild);  
 }  
  
 // cout << "销毁" << biTree->data << "成功～" << endl;  
 free(biTree);  
}  
  
  
**bool** InstertChild(BiTree biTree, **char** e, **int** LR, BiTree c) {  
 queue<BiNode \*> queue1;  
 queue1.push(biTree);  
 BiNode \*biNode;  
 biNode = queue1.front();  
 queue1.pop();  
 **if** (biNode->c == e) {  
  
 **if** (LR == 1) {//插入右边  
 BiNode \*q = biNode->rchild;  
 biNode->rchild = c;  
 c->parent = biNode;  
 c->rchild = q;  
 } **else** {  
  
 BiNode \*q = biNode->lchild;  
 biNode->lchild = c;  
 c->parent = biNode;  
 c->lchild = q;  
 }  
  
 biTree->node\_count++;  
 **return true**;  
 }  
  
  
 **while** (biNode != NULL) {  
  
 **if** (biNode->lchild != NULL) {  
 queue1.push(biNode->lchild);  
 }  
  
 **if** (biNode->rchild != NULL) {  
 queue1.push(biNode->rchild);  
 }  
  
 biNode = queue1.front();  
 queue1.pop();  
 **if** (biNode->c == e) {  
 **if** (LR == 1) {//插入右边  
 BiNode \*q = biNode->rchild;  
 biNode->rchild = c;  
 c->parent = biNode;  
 c->rchild = q;  
 } **else** {  
  
 BiNode \*q = biNode->lchild;  
 biNode->lchild = c;  
 c->parent = biNode;  
 c->lchild = q;  
 }  
 biTree->node\_count++;  
 **return true**;  
 }  
 }  
  
 cout << "没找到节点p" << endl;  
 **return false**;  
  
  
}  
  
  
//递归层次遍历  
**void** LevelOrderTraverse(BiTree biTree) {  
 **for** (**int** i = 1; i <= Depth(biTree); ++i) {  
 LevelOrderTraversePrint(biTree, i);  
  
 }  
  
  
}  
  
**void** LevelOrderTraversePrint(BiTree biTree, **int** depth) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
 **if** (depth == 1) {  
 cout << biTree->c << "(" << biTree->data << ") ";  
 **return**;  
 } **else** {  
 **if** (biTree->lchild != NULL) {  
 LevelOrderTraversePrint(biTree->lchild, depth - 1);  
 }  
  
  
 **if** (biTree->rchild != NULL) {  
 LevelOrderTraversePrint(biTree->rchild, depth - 1);  
 }  
 }  
  
  
}  
  
//中序遍历非递归  
**void** InOrderTraverse2(BiTree biTree) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 stack<BiNode \*> stack1;  
 BiNode \*biNode = biTree;  
  
 **while** (biNode != NULL || !stack1.empty()) {  
 **if** (biNode != NULL) {  
 stack1.push(biNode);  
 biNode = biNode->lchild;  
 } **else** {  
 biNode = stack1.top();  
 stack1.pop();  
 cout << biNode->c << "(" << biNode->data << ") ";  
 biNode = biNode->rchild;  
 }  
 }  
}  
  
//后序遍历非递归  
**void** PostOrderTraverse2(BiTree biTree) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 stack<BiNode \*> s;  
 BiNode \*cur; //当前结点  
 BiNode \*pre = NULL; //前一次访问的问题一转换为求源点到其他点的最短距离问题，问题二转换为求最小生成树问题。

结点  
 s.push(biTree);  
  
 **while** (!s.empty()) {  
 cur = s.top();  
 **if** ((cur->lchild == NULL && cur->rchild == NULL) ||  
 (pre != NULL && (pre == cur->lchild || pre == cur->rchild))) {  
 cout << cur->c << "(" << cur->data << ") "; //如果当前结点没有孩子结点或者孩子节点都已被访问过  
 s.pop();  
 pre = cur;  
 } **else** {  
 **if** (cur->rchild != NULL)  
 s.push(cur->rchild);  
 **if** (cur->lchild != NULL)  
 s.push(cur->lchild);  
 }  
 }  
}  
  
//层次遍历非递归  
**void** LevelOrderTraverse2(BiTree biTree) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 queue<BiNode \*> queue1;  
 queue1.push(biTree);  
 BiNode \*biNode = biTree;  
  
 **while** (biNode != NULL && !queue1.empty()) {  
  
 biNode = queue1.front();  
 queue1.pop();  
 cout << biNode->c << "(" << biNode->data << ") ";  
  
 **if** (biNode->lchild != NULL) {  
 queue1.push(biNode->lchild);  
 }  
  
 **if** (biNode->rchild != NULL) {  
 queue1.push(biNode->rchild);  
 }  
  
 }  
  
}  
  
//先序遍历非递归  
**void** PreOrderTraverse2(BiTree biTree) {//先序遍历  
 stack<BiNode \*> stack1;  
 BiNode \*biNode = biTree;  
  
 stack1.push(biNode);  
  
  
 **while** (biNode != NULL && !stack1.empty()) {  
 biNode = stack1.top();  
 stack1.pop();  
 cout << biNode->c << "(" << biNode->data << ") ";  
  
 **if** (biNode->rchild != NULL) {  
 stack1.push(biNode->rchild);  
 }  
  
 **if** (biNode->lchild != NULL) {  
 stack1.push(biNode->lchild);  
 }  
  
  
 }  
  
  
}  
  
//先序遍历递归  
**void** PreOrderTraverse(BiTree biTree) {//先序遍历  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
 cout << biTree->c << "(" << biTree->data << ") ";  
 **if** (biTree->lchild != NULL) {  
 PreOrderTraverse(biTree->lchild);  
 }  
  
 **if** (biTree->rchild != NULL) {  
 PreOrderTraverse(biTree->rchild);  
 }  
  
}  
  
//中序遍历递归  
**void** InOrderTraverse(BiTree biTree) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 **if** (biTree->lchild != NULL) {  
 InOrderTraverse(biTree->lchild);  
 }  
 cout << biTree->c << "(" << biTree->data << ") ";  
 **if** (biTree->rchild != NULL) {  
 InOrderTraverse(biTree->rchild);  
 }  
}  
  
//后序遍历递归  
**void** PostOrderTraverse(BiTree biTree) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 **if** (biTree->lchild != NULL) {  
 PostOrderTraverse(biTree->lchild);  
 }  
 **if** (biTree->rchild != NULL) {  
 PostOrderTraverse(biTree->rchild);  
 }  
 cout << biTree->c << "(" << biTree->data << ") ";  
  
}  
  
//判断是否为二叉排序树  
**bool** IsBiSort(BiTree biTree) {  
 **if** (biTree == NULL) {  
 **return false**;  
 }  
  
 **if** (biTree->lchild != NULL) {  
 **if** (biTree->lchild->data >= biTree->data) {  
 **return false**;  
 }  
 }  
  
 **if** (biTree->rchild != NULL) {  
 **if** (biTree->rchild->data <= biTree->data) {  
 **return false**;  
 }  
 }  
  
 **if** (biTree->lchild != NULL) {  
 **if** (!IsBiSort(biTree->lchild)) {  
 **return false**;  
 }  
 }  
  
  
 **if** (biTree->lchild != NULL) {  
 **if** (!IsBiSort(biTree->lchild)) {  
 **return false**;  
 }  
 }  
  
 **return true**;  
  
}  
  
//求深度  
**int** Depth(BiTree biTree) {  
  
 **if** (biTree == NULL) {  
 **return** 0;  
 }  
  
 **int** u = Depth(biTree->lchild);  
 **int** v = Depth(biTree->rchild);  
  
 **return** u > v ? u + 1 : v + 1;  
}  
  
  
//求树的节点个数  
**int** NodeNum(BiTree biTree) {  
 **return** biTree->node\_count;  
}  
  
  
//求宽度  
**int** Width(BiTree biTree) {  
 **int** max = 0;  
 **for** (**int** i = 1; i <= Depth(biTree); ++i) {  
 **int** n = GetNum(biTree, i);  
 **if** (n > max) {  
 max = n;  
 }  
  
 }  
 **return** max;  
  
}  
  
**int** GetNum(BiTree biTree, **int** depth) {  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
 **int** num = 0;  
 **if** (depth == 1) {  
 **return** 1;  
 } **else** {  
 **if** (biTree->lchild != NULL) {  
 num += GetNum(biTree->lchild, depth - 1);  
 }  
  
  
 **if** (biTree->rchild != NULL) {  
 num += GetNum(biTree->rchild, depth - 1);  
 }  
 }  
  
 **return** num;  
  
}  
  
#endif //INC\_1\_BITREE\_H

#include <iostream>  
#include <fstream>  
#include <vector>  
  
**using namespace** std;  
  
#include "BiTree.h"  
  
**typedef struct** {  
 **char** m;  
 **char** p;  
 **int** s;  
 **int** d;  
} Node;  
**void** Test1();  
**void** Test2();  
**int** main() {  
  
 Test1();  
 cout<<endl;  
 Test2();  
 **return** 0;  
}  
  
  
**void** Test1() {  
 fstream in("/home/dmrf/文档/DataStructure/CourseDesign/Tree.txt");  
 **if** (!in.is\_open()) {  
 cout << "文件打开失败！" << endl;  
 **return**;  
 }  
 vector<Node> v;  
 **char** m, p, s, d;  
 BiTree biTree;  
 InitBitree(biTree);  
 **while** (!in.eof()) {  
  
 Node n;  
 in >> n.m >> n.p >> n.s >> n.d;  
 v.push\_back(n);  
 }  
  
 **for** (**int** i = 0; i < v.size() - 1; ++i) {  
 **if** (i == 0) {  
 biTree->c = v[i].m;  
 biTree->data = v[i].d;  
 } **else** {  
 BiTree biTree1;  
 InitBitree(biTree1);  
 biTree1->data = v[i].d;  
 biTree1->c = v[i].m;  
 InstertChild(biTree, v[i].p, v[i].s, biTree1);  
 }  
  
 }  
  
 cout << "构建树成功～" << endl;  
  
  
 cout << "层次遍历递归实现：" << endl;  
 LevelOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "层次遍历非递归实现：" << endl;  
 LevelOrderTraverse2(biTree);  
  
  
 cout << endl;  
 cout << "中序遍历递归实现：" << endl;  
 InOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "中序遍历非递归实现：" << endl;  
 InOrderTraverse2(biTree);  
  
 cout << endl;  
 cout << "后序遍历递归实现：" << endl;  
 PostOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "后序遍历非递归实现：" << endl;  
 PostOrderTraverse2(biTree);  
  
 cout << endl;  
 cout << "先序遍历递归实现：" << endl;  
 PreOrderTraverse(biTree);  
  
  
 cout << endl;  
 cout << "先序遍历非递归实现：" << endl;  
 PreOrderTraverse2(biTree);  
  
  
 cout << endl;  
 cout << "该二叉树的高度为：";  
 cout << Depth(biTree) << endl;  
  
 cout << "该二叉树的宽度为：";  
 cout << Width(biTree) << endl;  
  
  
 cout << "该二叉树的节点个数为：" << biTree->node\_count << endl;  
  
  
 string ps;  
 **if** (!IsBiSort(biTree)) {  
 ps = "不是";  
 } **else** {  
 ps = "是";  
 }  
 cout << "该二叉树" << ps << "二叉排序树" << endl;  
  
  
}  
  
  
**void** Test2() {  
 fstream in("/home/dmrf/文档/DataStructure/CourseDesign/Tree2.txt");  
 **if** (!in.is\_open()) {  
 cout << "文件打开失败！" << endl;  
 **return**;  
 }  
 vector<Node> v;  
 **char** m, p, s, d;  
 BiTree biTree;  
 InitBitree(biTree);  
 **while** (!in.eof()) {  
  
 Node n;  
 in >> n.m >> n.p >> n.s >> n.d;  
 v.push\_back(n);  
 }  
  
 **for** (**int** i = 0; i < v.size() - 1; ++i) {  
 **if** (i == 0) {  
 biTree->c = v[i].m;  
 biTree->data = v[i].d;  
 } **else** {  
 BiTree biTree1;  
 InitBitree(biTree1);  
 biTree1->data = v[i].d;  
 biTree1->c = v[i].m;  
 InstertChild(biTree, v[i].p, v[i].s, biTree1);  
 }  
  
 }  
  
 cout << "构建树成功～" << endl;  
  
  
 cout << "层次遍历递归实现：" << endl;  
 LevelOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "层次遍历非递归实现：" << endl;  
 LevelOrderTraverse2(biTree);  
  
  
 cout << endl;  
 cout << "中序遍历递归实现：" << endl;  
 InOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "中序遍历非递归实现：" << endl;  
 InOrderTraverse2(biTree);  
  
 cout << endl;  
 cout << "后序遍历递归实现：" << endl;  
 PostOrderTraverse(biTree);  
  
 cout << endl;  
 cout << "后序遍历非递归实现：" << endl;  
 PostOrderTraverse2(biTree);  
  
 cout << endl;  
 cout << "先序遍历递归实现：" << endl;  
 PreOrderTraverse(biTree);  
  
  
 cout << endl;  
 cout << "先序遍历非递归实现：" << endl;  
 PreOrderTraverse2(biTree);  
  
  
 cout << endl;  
 cout << "该二叉树的高度为：";  
 cout << Depth(biTree) << endl;  
  
 cout << "该二叉树的宽度为：";  
 cout << Width(biTree) << endl;  
  
  
 cout << "该二叉树的节点个数为：" << biTree->node\_count << endl;  
  
  
 string ps;  
 **if** (!IsBiSort(biTree)) {  
 ps = "不是";  
 } **else** {  
 ps = "是";  
 }  
 cout << "该二叉树" << ps << "二叉排序树" << endl;  
  
  
}

# 四、

4.1 题目简介

Huffman编码与解码(必做)（Huffman编码、二叉树）

[问题描述]

对一篇不少于2000字符的英文文章，统计各字符出现的次数，实现Huffman编码，以及对编码结果的解码。

[基本要求]

（1） 输出每个字符出现的次数和编码。

（2） 在Huffman编码后，要将编码表和英文文章编码结果保存到文件中，编码结果必须是二进制形式，即0 1的信息用比特位表示，不能用字符’0’和’1’表示。

（3） 提供读编码文件生成原文件的功能。

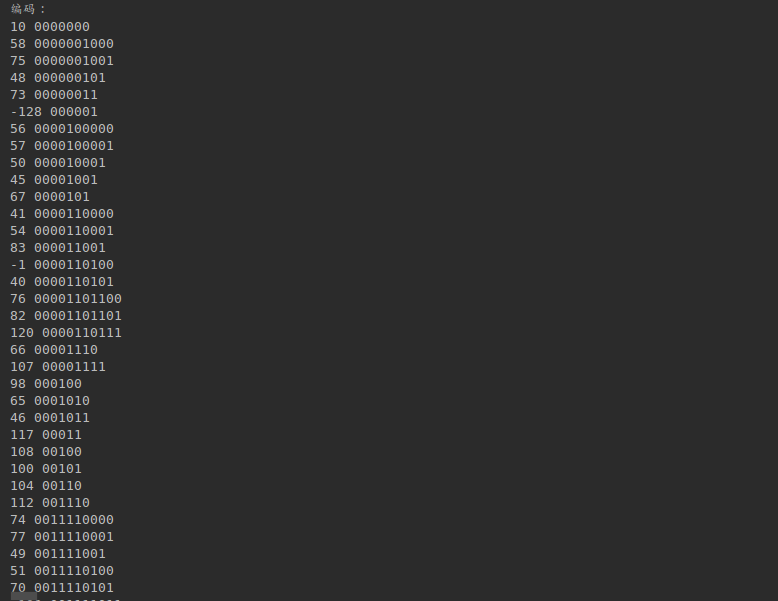
4.2 数据结构

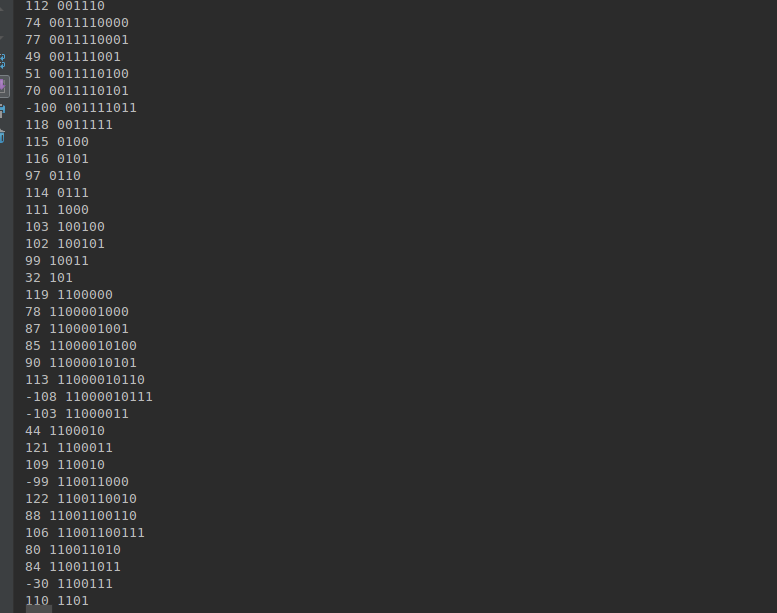
**typedef struct** {  
 **char** a;  
 **char** b;  
 **int** v;  
} Vex;  
  
**typedef struct** BiNode {  
 **struct** BiNode \*lchild;  
 **struct** BiNode \*rchild;  
 **struct** BiNode \*parent;  
 **int** v = 0;  
 **char** c = NULL;  
} BiNode, \*BiTree;

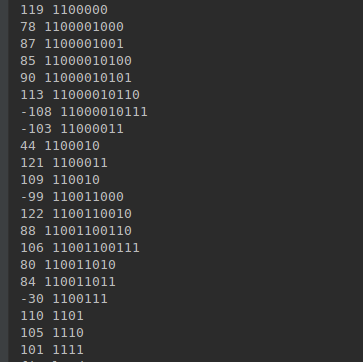
4.3 算法设计思想

将所有字符按照出现次数排序，出现次数多的字符编码到二叉树的根部部分，出现次数小的放在二叉树的叶子部分，最终得到哈弗曼树进而进行编码与解码。

4.4测试数据和结果







4.5算法时间复杂度

编码：O（n^2）

解码：O（n^2）

4.6源代码

//  
// Created by dmrf on 17-11-20.  
//  
  
#ifndef INC\_1\_BITREE\_H  
#define INC\_1\_BITREE\_H  
#define TElemType **int**#include <queue>  
#include <stack>  
#include <iostream>  
#include <cmath>  
#include <string>  
#include <fstream>  
  
**using namespace** std;  
  
**typedef struct** {  
 **char** a;  
 **char** b;  
 **int** v;  
} Vex;  
  
**typedef struct** BiNode {  
 **struct** BiNode \*lchild;  
 **struct** BiNode \*rchild;  
 **struct** BiNode \*parent;  
 **int** v = 0;  
 **char** c = NULL;  
} BiNode, \*BiTree;  
  
**int** Depth(BiTree biTree);  
  
**void** LevelOrderTraversePrint(BiTree biTree, **int** depth);  
  
**void** InitBitree(BiTree &biTree) {  
 biTree = **new** BiNode();  
 biTree->lchild = NULL;  
 biTree->rchild = NULL;  
 biTree->parent = NULL;  
 cout << "初始化成功～" << endl;  
}  
  
**void** DestroyBiTree(BiTree biTree) {  
  
 **if** (biTree == NULL) {  
 **return**;  
 }  
 **if** (biTree->lchild != NULL) {  
 DestroyBiTree(biTree->lchild);  
 }  
  
 **if** (biTree->rchild != NULL) {  
 DestroyBiTree(biTree->rchild);  
 }  
 free(biTree);  
}  
  
  
**void** GetCode(BiTree biTree, string s, ofstream &code) {//先序遍历  
 **if** (biTree == NULL) {  
 cout << "该树为空，无法遍历！" << endl;  
 }  
  
 **if** (biTree->c != NULL) {  
 **int** cs=biTree->c;  
 code<<cs<<" "<<s<<endl;  
 cout<<cs<<" "<<s<<endl;  
 /\*if (biTree->c == 32) {  
 cout << " " << " " << s << endl;  
 code << " " << " " << s << endl;  
 } else{  
 cout << biTree->c << " " << s << endl;  
 code << biTree->c << " " << s << endl;  
 }\*/  
  
 }  
 **if** (biTree->lchild != NULL) {  
 GetCode(biTree->lchild, s + "0", code);  
  
 }  
  
 **if** (biTree->rchild != NULL) {  
 GetCode(biTree->rchild, s + "1", code);  
 }  
  
 code.flush();  
}  
  
  
#endif //INC\_1\_BITREE\_H

#include <iostream>  
#include <fstream>  
#include <vector>  
#include "BiTree.h"  
  
  
**using namespace** std;  
string testjjj;  
**typedef struct** {  
 **char** m;  
 **int** n;  
} Content;  
  
**void** EnCode();  
  
vector<**int**> CharToNumber(**char** c);  
  
**void** DeCode();  
  
**void** HuffmanCode(vector<Content> v);  
  
**void** DoCode();  
  
**void** Select(BiNode \*b, **int** i, **int** &s1, **int** &s2);  
  
**int** main() {  
  
  
 /\* FILE \*te;  
 te=fopen("/home/dmrf/文档/DataStructure/CourseDesign/test.txt","w");  
 char c=' ';  
 char d=32;  
 fputc(c,te);  
 //fputc(d,te);  
  
 fflush(te);  
 fclose(te);\*/  
  
  
 /\* ofstream o("/home/dmrf/文档/DataStructure/CourseDesign/test.txt");  
 o<<' ';\*/  
 /\* ifstream i("/home/dmrf/文档/DataStructure/CourseDesign/test.txt");  
 char c;  
 c=20;  
 i>>c;  
 cout<<(int)c;\*/  
  
  
 EnCode();  
 DeCode();  
  
 **return** 0;  
}  
  
  
**void** EnCode() {//编码  
  
 FILE \*in;  
 in = fopen("/home/dmrf/文档/DataStructure/CourseDesign/HuffmanData.txt", "r");  
  
  
 vector<Content> c;  
 **while** (!feof(in)) {  
 **char** d;  
 d = fgetc(in);  
  
  
 **bool** flag = **false**;  
 **for** (**int** i = 0; i < c.size(); ++i) {  
 **if** (c[i].m == d) {  
 c[i].n++;  
 flag = **true**;  
 }  
 }  
 **if** (!flag) {  
 Content t;  
 t.m = d;  
 t.n = 1;  
 c.push\_back(t);  
 }  
 }  
  
 fclose(in);  
  
 //排序  
 **for** (**int** j = 0; j < c.size() - 1; ++j) {  
 **for** (**int** i = j + 1; i < c.size(); ++i) {  
 **if** (c[j].n > c[i].n) {  
 Content content;  
 content = c[j];  
 c[j] = c[i];  
 c[i] = content;  
 } **else if** (c[j].n == c[i].n && c[j].m > c[i].m) {  
 Content content;  
 content = c[j];  
 c[j] = c[i];  
 c[i] = content;  
 }  
  
 }  
 }  
  
 HuffmanCode(c);  
  
 DoCode();  
  
  
 /\*for (int k = 0; k < c.size(); ++k) {  
 cout << c[k].m << " " << c[k].n << endl;  
 }\*/  
}  
  
**void** DeCode() {//解码  
  
  
  
  
 FILE \*in;  
 in = fopen("/home/dmrf/文档/DataStructure/CourseDesign/NewHuffmanData.txt", "r");  
  
  
 string in\_data;  
  
  
 **int** j = 0;  
 **char** ct;  
 string code\_data = "";  
  
 vector<**char**> cc;  
 cout << endl;  
 **while** (!feof(in)) {  
 ct = fgetc(in);  
 cc.push\_back(ct);  
 }  
  
  
 **for** (**int** l = 0; l < cc.size(); ++l) {  
 //cout << (int) cc[l] << " ";  
 vector<**int**> con;  
 **if** (l == 37) {  
 cout << "d";  
 }  
 **if** (cc[l] == 0) {  
 **if** (cc[l + 1] == 0) {  
 code\_data += "0";  
 } **else if** (cc[l + 1] == 1) {  
 code\_data += "1";  
 }  
 **for** (**int** i = 0; i < 7; ++i) {  
 code\_data += "0";  
 }  
 l++;  
 } **else** {  
 con = CharToNumber(cc[l]);  
 **for** (**int** i = 0; i < 8; ++i) {  
 **if** (con[i] == 0) {  
 code\_data += "0";  
 } **else** {  
 code\_data += "1";  
 }  
 }  
 }  
  
  
 }  
  
  
 // cout << endl << "j:" << cc.size() - 1 << endl;  
 // cout << "code\_data:" << endl;  
 // cout << code\_data;  
  
 vector<string> val;  
 **for** (**int** j = 0; j <= 256; ++j) {  
 val.push\_back("");  
 }  
 fstream incode("/home/dmrf/文档/DataStructure/CourseDesign/HuffmanCode.txt");  
  
 **int** n = 0;  
 **int** c;  
 string co;  
 **while** (incode >> c >> co) {  
  
  
 **int** index;  
 **if** (c < 0) {  
 index = -c + 127;  
 } **else** {  
 index = c;  
 }  
 val[index] = co;  
 }  
  
  
 incode.close();  
  
  
 FILE \*ou;  
 ou = fopen("/home/dmrf/文档/DataStructure/CourseDesign/EnCodeHuffmanData.txt", "w");  
  
 //int et=0;  
 cout << endl;  
 string tim\_str;  
 string fi = "";  
 // code\_data=testjjj;  
 string gs = "";  
 **for** (**int** m = 0; m <= code\_data.size() - 10; ++m) {  
 gs += code\_data[m];  
 }  
 // cout<<"gs:"<<gs.size()<<endl;  
 cout << gs << endl;  
  
 **for** (**int** i1 = 0; i1 < gs.size(); ++i1) {  
 **if** (gs[i1] != testjjj[i1]) {  
 cout << "s";  
 }  
 }  
  
 **for** (**int** k = 0; k < gs.size(); ++k) {  
 tim\_str += gs[k];  
  
  
 **for** (**int** i = 0; i < val.size(); ++i) {  
 **if** (tim\_str == val[i]) {  
 **char** c;  
  
 **if** (i > 127) {  
 c = i - 127;  
 c = -c;  
 } **else** {  
 c = i;  
 }  
  
 **if** (k>=gs.size()-8){  
 **break**;  
 }  
 fi += c;  
 fputc(c, ou);  
 tim\_str = "";  
 **break**;  
 }  
 }  
  
 }  
  
 fflush(ou);  
 fclose(ou);  
  
}  
  
  
vector<**int**> CharToNumber(**char** c) {//将ascill转化为二进制  
 vector<**int**> bin;  
 **int** n;  
 **if** (c <= 0) {  
 bin.push\_back(0);  
 n = -c;  
 } **else** {  
 bin.push\_back(1);  
 n = c;  
 }  
 **int** temp[7];  
 **for** (**int** i = 0; i < 7; ++i) {  
 **int** a;  
 a = n % 2;  
 n = n / 2;  
 temp[i] = a;  
 }  
  
 **for** (**int** j = 0; j <= 7; ++j) {  
 bin.push\_back(temp[6 - j]);  
 }  
 **return** bin;  
}  
  
**void** DoCode() {  
  
  
 vector<string> val;  
 **for** (**int** j = 0; j <= 256; ++j) {  
 val.push\_back("");  
 }  
 fstream incode("/home/dmrf/文档/DataStructure/CourseDesign/HuffmanCode.txt");  
  
 **int** n = 0;  
 **int** c;  
 string co;  
 **while** (incode >> c >> co) {  
  
  
 **int** index;  
 **if** (c < 0) {  
 index = -c + 127;  
 } **else** {  
 index = c;  
 }  
 val[index] = co;  
 }  
  
  
 incode.close();  
  
  
 FILE \*out;  
 out = fopen("/home/dmrf/文档/DataStructure/CourseDesign/NewHuffmanData.txt", "w");  
  
  
 FILE \*huffman;  
 huffman = fopen("/home/dmrf/文档/DataStructure/CourseDesign/HuffmanData.txt", "r");  
  
  
 string finalcode;  
 **char** d;  
  
 **while** (!feof(huffman)) {  
 d = fgetc(huffman);  
 **int** c = d;  
 n++;  
 **int** index;  
  
 **if** (c < 0) {  
 index = -c + 127;  
 } **else** {  
 index = c;  
 }  
  
 finalcode += val[index];  
  
 }  
  
  
 //cout<<"et"<<et<<endl;  
 /\* cout << "总字数" << n << endl;  
  
 cout << endl;\*/  
 cout << "finalcode:" << endl;  
 cout << "size:" << finalcode.size() << endl;  
  
 cout << finalcode;  
  
 **int** bit[8];  
 **int** flag\_number = 0;  
 **int** for\_number = 0;  
  
 **int** n1 = 0;  
  
 **int** cont = 1;  
 **int** g;  
  
 **while** (**true**) {  
 **if** (flag\_number == 8) {  
 **int** temp = bit[1] \* (**int**) pow(2, 6) +  
 bit[2] \* (**int**) pow(2, 5) +  
 bit[3] \* (**int**) pow(2, 4) +  
 bit[4] \* (**int**) pow(2, 3) +  
 bit[5] \* (**int**) pow(2, 2) +  
 bit[6] \* (**int**) pow(2, 1) +  
 bit[7] \* (**int**) pow(2, 0);  
  
 **if** (temp == 0) {  
 **if** (bit[0] == 0) {  
 fputc(0, out);  
 fputc(0, out);  
 } **else** {  
 fputc(0, out);  
 fputc(1, out);  
 }  
 } **else** {  
 **if** (bit[0] == 0) {//0是负  
  
 temp \*= -1;  
 }  
  
 **char** cs = temp;  
 fputc(cs, out);  
 }  
 flag\_number = 0;  
 for\_number++;  
 **int** t = temp;  
 // cout << t << " ";  
 n1++;  
 **if** (flag\_number + for\_number \* 8 == finalcode.size()) {  
 **break**;  
 }  
 } **else** {  
 **if** (flag\_number + for\_number \* 8 == finalcode.size()) {  
 **int** temp = 0;  
 **for** (**int** i = 0; i < flag\_number; ++i) {  
 temp += bit[i] \* pow(2, 7 - i);  
  
 }  
  
  
 **if** (bit[0] == 0) {//0是负  
 temp \*= -1;  
 }  
 **char** cs = temp;  
 fputc(cs, out);  
 **break**;  
 } **else** {  
 **char** temp\_c = finalcode[flag\_number + for\_number \* 8];  
 **if** (temp\_c == 48) {//0  
 bit[flag\_number] = 0;  
 } **else** {  
 bit[flag\_number] = 1;  
 }  
  
 flag\_number++;  
 }  
  
 }  
 }  
  
  
 //cout << endl << "编码后的总字符数为：" << n1 << endl;  
  
  
 fflush(out);  
 fclose(out);  
  
  
 fclose(huffman);  
  
 testjjj = finalcode;  
/\*  
  
 FILE \*ou;  
 ou = fopen("/home/dmrf/文档/DataStructure/CourseDesign/EnCodeHuffmanData.txt", "w");  
  
 //int et=0;  
 cout << endl;  
 string tim\_str;  
 for (int k = 0; k < finalcode.size(); ++k) {  
 tim\_str += finalcode[k];  
  
 for (int i = 0; i < val.size(); ++i) {  
 if (tim\_str == val[i]) {  
 char c;  
  
 if (i > 127) {  
 c = i - 127;  
 c = -c;  
 } else {  
 c = i;  
 }  
  
  
 fputc(c, ou);  
 tim\_str = "";  
 break;  
 }  
 }  
  
 }  
  
 fflush(ou);  
 fclose(ou);\*/  
  
}  
  
  
**void** HuffmanCode(vector<Content> v) {  
 **int** n = v.size();  
  
  
 **int** m = 2 \* n;  
  
 BiNode \*biNode = **new** BiNode[2 \* n];  
  
  
 **for** (**int** k = 1; k <= n; ++k) {  
 biNode[k].lchild = NULL;  
 biNode[k].rchild = NULL;  
 biNode[k].parent = NULL;  
 biNode[k].c = v[k - 1].m;  
 biNode[k].v = v[k - 1].n;  
 }  
  
 **for** (**int** i = n + 1; i < m; ++i) {  
 biNode[i].lchild = NULL;  
 biNode[i].rchild = NULL;  
 biNode[i].parent = NULL;  
 }  
  
 **for** (**int** j = n + 1; j < m; ++j) {  
 **int** s1;  
 **int** s2;  
 Select(biNode, j, s1, s2);  
 biNode[j].v = biNode[s1].v + biNode[s2].v;  
 biNode[j].lchild = &biNode[s1];  
 biNode[j].rchild = &biNode[s2];  
 biNode[s1].parent = &biNode[j];  
 biNode[s2].parent = &biNode[j];  
 }  
  
  
 ofstream code("/home/dmrf/文档/DataStructure/CourseDesign/HuffmanCode.txt");  
 **if** (!code.is\_open()) {  
 cout << "HuffmanCode.txt1打开失败！" << endl;  
 }  
  
  
 cout << "编码：" << endl;  
 GetCode(&biNode[m - 1], "", code);  
 code.close();  
 free(biNode);  
  
  
}  
  
**void** Select(BiNode \*b, **int** i, **int** &s1, **int** &s2) {  
  
 **int** m2, m1;  
  
 m2 = 10000;  
 m1 = m2;  
 i--;  
 **while** (i > 0) {  
 **if** (b[i].v < m1 && b[i].parent == NULL) {  
 m2 = m1;  
 s2 = s1;  
 m1 = b[i].v;  
 s1 = i;  
 } **else if** (b[i].v == m1 && b[i].parent == NULL) {  
 **if** (b[i].c < b[s1].c) {  
 m2 = m1;  
 s2 = s1;  
 m1 = b[i].v;  
 s1 = i;  
 }  
 } **else if** (b[i].v < m2 && b[i].parent == NULL) {  
 m2 = b[i].v;  
 s2 = i;  
 } **else if** (b[i].v == m2 && b[i].parent == NULL) {  
 **if** (b[i].c < b[s2].c) {  
 m2 = b[i].v;  
 s2 = i;  
 }  
 }  
 i--;  
  
 }  
}

# 五、

5.1 题目简介

无线传感器网络(必做) （图）

[问题描述]

在一个直角坐标（设为100\*100）平面中，随机分布n无线传感器结点，汇聚点为原点（0，0），要求将每个无线传感器结点的信息传输到汇聚点，传输可以通过多跳方式实现，设无线传感器结点最大无线通信距离为10，传输能耗与距离平方成正比，传输时间与距离成正比。

问题一：建立每个结点传输到汇聚点的最短时间通道，并找出无法实现传输的结点，将其排除。

问题二：建立整个网络传输到汇聚点的平均能耗最小网络结构。

[基本要求]

输入格式： 输入的第一行包含一个正整数n，表示无线传感器结点数量。结点使用1, 2, 3, ……n依次标号。 接下来n行，每行包含三个整数ni, xi, yi，其中xi, yi表示第ni个结点的坐标，要求从文本文件中输入。

输出格式：

问题一：输出每个结点到汇聚点的最短时间通道，包括最短时间和经过结点。

问题二：输出整个网络到汇聚点的平均能耗最小网络结构，包括平均最小能耗和连接方式。

5.2 数据结构

**typedef enum** {  
 DG, DN, UDG, UDN//有向图 有向网 无向图 无向网  
} GraphKind;  
  
**typedef struct** ArcCell {  
 **double** adj;//边的关系类型，无权图值为0或1，有权图为权值  
} ArcCell, AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];  
  
**typedef struct** {  
 **int** vexs[MAX\_VERTEX\_NUM];//用来存储顶点  
 AdjMatrix arcs;//用来存储边  
 **int** vexnum, arcnum;//顶点数和边数  
 **int** graphKind;//图的类型  
} MGraph;  
  
**typedef struct** {  
 **int** code;  
 **int** x;  
 **int** y;  
} Node;

5.3 算法设计思想

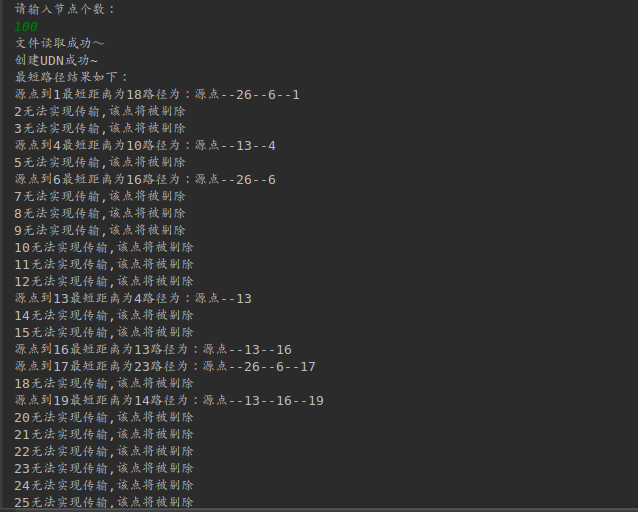
问题一：

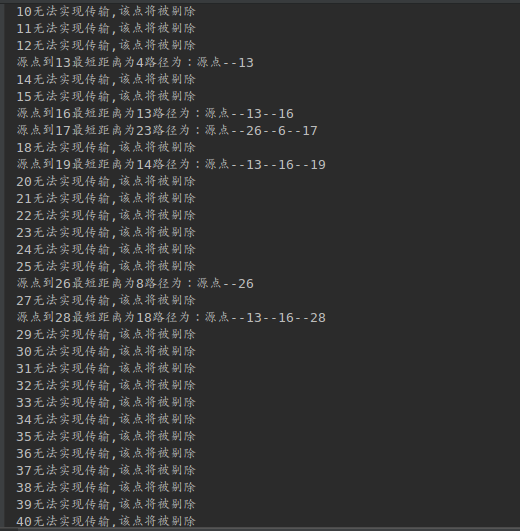
采用迪杰特斯拉算法，从源点出发，先找一条从源点出发的最短路径，然后再以这两个点的集合出发找下一条最短的路径，以此类推直到走完所有的节点。

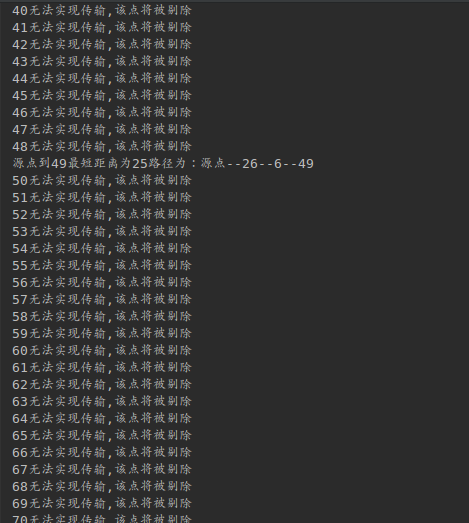
问题二：

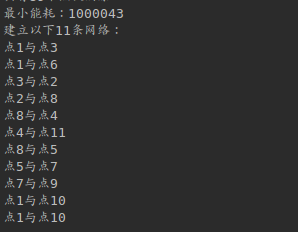
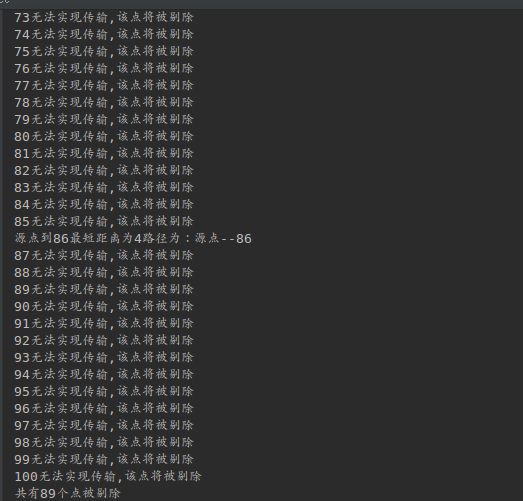
采用prim算法，先从源点出发找到一个距离源点最短的节点，再以这两个点为集合在其他点中寻找距离最进的点加入最近点集合，以此类推直到所有满足条件的点被遍历完。

5.4测试数据和结果









5.5算法时间复杂度

求最短路径：O（n）

求最小生成树：O（n^2）

5.6源代码

//  
// Created by dmrf on 17-12-6.  
//  
  
#ifndef INC\_1\_ADTGRAPH\_H  
#define INC\_1\_ADTGRAPH\_H  
  
#define MAX\_VERTEX\_NUM 500  
  
#include <math.h>  
#include <iostream>  
#include <queue>  
  
**using namespace** std;  
**typedef enum** {  
 DG, DN, UDG, UDN//有向图 有向网 无向图 无向网  
} GraphKind;  
  
**typedef struct** ArcCell {  
 **double** adj;//边的关系类型，无权图值为0或1，有权图为权值  
} ArcCell, AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];  
  
**typedef struct** {  
 **int** vexs[MAX\_VERTEX\_NUM];//用来存储顶点  
 AdjMatrix arcs;//用来存储边  
 **int** vexnum, arcnum;//顶点数和边数  
 **int** graphKind;//图的类型  
} MGraph;  
  
**typedef struct** {  
 **int** code;  
 **int** x;  
 **int** y;  
} Node;  
  
**bool** Visited[MAX\_VERTEX\_NUM];  
  
**bool** CreateDG(MGraph &G);  
  
**bool** CreateDN(MGraph &G);  
  
**bool** CreateUDG(MGraph &G);  
  
**bool** CreateUDN(MGraph &G, vector<Node> v);  
  
**int** LocateVex(MGraph g, **int** a);  
  
**void** DFS(MGraph G, **int** v);  
  
//创建图  
**bool** CreateGraph(MGraph &G) {  
 cout << "请输入图的类型（0-DG 1-DN 2-UDG 3-UDN）：" << endl;  
 cin >> G.graphKind;  
  
 **switch** (G.graphKind) {  
 **case** DG:  
 **return** CreateDG(G);  
 **break**;  
 **case** DN:  
 **return** CreateDN(G);  
 **break**;  
 **case** UDG:  
 **return** CreateUDG(G);  
 **break**;  
 **case** UDN:  
 //return CreateUDN(G);  
 **break**;  
 **default**:  
 cout << "图的类型输入错误！" << endl;  
 **return false**;  
 }  
}  
  
//销毁图  
**bool** DestroyGraph(MGraph &G) {  
 **return true**;  
}  
  
//寻找图中a点的位置  
**int** LocateVex(MGraph G, **int** a) {  
  
 **for** (**int** i = 0; i < G.vexnum; ++i) {  
  
 **if** (a == G.vexs[i]) {  
 **return** i;  
 }  
 }  
  
 **return** -1;  
}  
  
//返回下标为l的点的值  
**char** GetVex(MGraph G, **int** l) {  
 **if** (l <= G.vexnum && l >= 1) {  
 **return** G.vexs[l];  
 } **else** {  
 cout << "所获取的点的下标不合法！" << endl;  
 **return** -1;  
 }  
  
}  
  
//设置下标为l的顶点的值为v  
**bool** PutVex(MGraph &G, **int** l, **char** v) {  
 **if** (l <= G.vexnum && l >= 1) {  
 G.vexs[l] = v;  
 cout << "设置坐标" << v << "成功" << endl;  
 **return true**;  
 } **else** {  
 cout << "所设置的点的下标不合法！" << endl;  
 **return false**;  
 }  
}  
  
//返回v的第一个邻接顶点  
**char** FirstAdjVex(MGraph G, **char** v) {  
 **int** i = LocateVex(G, v);  
 **if** (i < 1 || i > G.vexnum) {  
 cout << v << "不在图中！" << endl;  
 **return** NULL;  
 } **else** {  
 **for** (**int** j = 1; j < G.vexnum; ++j) {  
 **if** (G.arcs[i][j].adj != -1) {  
 //cout << v << "的第一个邻接顶点是" << G.vexs[j] << endl;  
 **return** G.vexs[j];  
 }  
 }  
 }  
  
 cout << v << "没有邻顶点～" << endl;  
 **return** NULL;  
}  
  
//返回v的第一个邻接顶点  
**int** FirstAdjVex2(MGraph G, **char** v) {  
 **int** i = LocateVex(G, v);  
 **if** (i < 1 || i > G.vexnum) {  
 cout << v << "不在图中！" << endl;  
 **return** NULL;  
 } **else** {  
 **for** (**int** j = 1; j < G.vexnum; ++j) {  
 **if** (G.arcs[i][j].adj != -1) {  
 //cout << v << "的第一个邻接顶点是" << G.vexs[j] << endl;  
 **return** j;  
 }  
 }  
 }  
  
 // cout << v << "没有邻顶点～" << endl;  
 **return** -1;  
}  
  
//u是v的一个邻接点，寻找v的下一个邻接点并返回  
**char** NextAdjVex(MGraph G, **char** v, **char** u) {  
 **int** i = LocateVex(G, v);  
 **if** (i < 1 || i > G.vexnum) {  
 cout << v << "不在图中！" << endl;  
 **return** NULL;  
 } **else** {  
 **int** j = LocateVex(G, u);  
 **if** (j == -1) {  
 cout << u << "不在图中！" << endl;  
 **return** NULL;  
 }  
 **if** (G.arcs[i][j].adj == -1) {  
 cout << v << "和" << u << "不邻接！" << endl;  
 } **else** {  
 **for** (**int** k = j + 1; k < G.vexnum; ++k) {  
 **if** (G.arcs[i][k].adj != 0 && G.arcs[i][k].adj != -1) {  
 **return** G.vexs[k];  
 }  
 }  
  
 cout << "没有下一个临邻接点～" << endl;  
 **return** NULL;  
 }  
 }  
}  
  
//u是v的一个邻接点，寻找v的下一个邻接点并返回  
**int** NextAdjVex2(MGraph G, **char** v, **char** u) {  
 **int** i = LocateVex(G, v);  
 **if** (i < 1 || i > G.vexnum) {  
 cout << v << "不在图中！" << endl;  
 **return** NULL;  
 } **else** {  
 **int** j = LocateVex(G, u);  
 **if** (j == -1) {  
 // cout << u << "不在图中！" << endl;  
 **return** -1;  
 }  
 **if** (G.arcs[i][j].adj == -1) {  
 cout << v << "和" << u << "不邻接！" << endl;  
 } **else** {  
 **for** (**int** k = j + 1; k <= G.vexnum; ++k) {  
 **if** (G.arcs[i][k].adj != 0 && G.arcs[i][k].adj != -1) {  
 **return** k;  
 }  
 }  
  
 // cout << "没有下一个临邻接点～" << endl;  
 **return** -1;  
 }  
 }  
}  
  
//插入点v  
**bool** InserVex(MGraph &G, **char** v) {  
 **if** (G.vexnum == MAX\_VERTEX\_NUM) {  
 cout << "空间不够，无法插入！" << endl;  
 **return false**;  
 } **else** {  
 **int** i = LocateVex(G, v);  
 **if** (i != -1) {  
 cout << "该点已存在，无法插入！" << endl;  
 **return false**;  
 } **else** {  
 G.vexnum++;  
 G.vexs[G.vexnum] = v;  
 **return true**;  
 }  
  
 }  
}  
  
//删除v  
**bool** DeleteVex(MGraph &G, **int** v) {  
 **int** i = LocateVex(G, v);  
 **if** (i == -1) {  
 //cout << "要删除的点不存在！" << endl;  
 **return false**;  
 } **else** {  
 **for** (**int** j = i; j < G.vexnum-1; ++j) {  
 **for** (**int** k = 0; k < G.vexnum; ++k) {  
 G.arcs[j][k] = G.arcs[j + 1][k];  
 }  
 }  
 **for** (**int** j = i; j < G.vexnum-1; ++j) {  
 **for** (**int** k = 1; k <G.vexnum; ++k) {  
 G.arcs[k][j] = G.arcs[k][j + 1];  
 }  
 }  
  
 **for** (**int** l = i; l < G.vexnum-1; ++l) {  
 G.vexs[l] = G.vexs[l + 1];  
 }  
 G.vexnum--;  
 **return true**;  
  
 }  
}  
  
//在图中插入边v u  
**bool** InsertArc(MGraph &G, **char** v, **char** u) {  
  
  
 **int** i = LocateVex(G, v);  
 **int** j = LocateVex(G, u);  
  
 **if** (G.graphKind == UDG) {//无向图  
  
 G.arcs[i][j].adj = 1;  
 G.arcs[j][i].adj = 1;  
 G.arcnum++;  
 cout << "插入边成功～" << endl;  
 **return true**;  
 } **else if** (G.graphKind == DG) {//有向图  
 G.arcs[i][j].adj = 1;  
 G.arcnum++;  
 cout << "插入边成功～" << endl;  
 **return true**;  
 }  
 cout << "插入边失败！" << endl;  
 **return false**;  
}  
  
  
**bool** DeleteArc(MGraph &G, **char** v, **char** u) {  
 **int** i = LocateVex(G, v);  
 **int** j = LocateVex(G, u);  
 **if** (G.graphKind == UDG) {//无向图  
  
 G.arcs[i][j].adj = -1;  
 G.arcs[j][i].adj = -1;  
 G.arcnum--;  
 cout << "删除边成功～" << endl;  
 **return true**;  
 } **else if** (G.graphKind == DG) {//有向图  
 G.arcs[i][j].adj = -1;  
 G.arcnum--;  
 cout << "删除边成功～" << endl;  
 **return true**;  
 }  
 cout << "删除边失败！" << endl;  
 **return false**;  
}  
  
//深度优先遍历  
**void** DFSTraverse(MGraph G) {  
  
 **for** (**int** i = 1; i <= G.vexnum; ++i) {  
 Visited[i] = **false**;  
 }  
  
  
 **for** (**int** j = 1; j <= G.vexnum; ++j) {  
 **if** (!Visited[j]) {  
 DFS(G, j);  
 }  
 }  
  
  
}  
  
**void** DFS(MGraph G, **int** v) {  
  
 **char** c = GetVex(G, v);  
 cout << c << " ";  
 Visited[v] = **true**;  
 **for** (**int** f = FirstAdjVex2(G, c); f > 0;) {  
 **if** (!Visited[f]) {  
 DFS(G, f);  
  
 }  
  
 **char** k = GetVex(G, f);  
 f = NextAdjVex2(G, c, k);  
 }  
  
}  
  
  
//广度优先遍历  
**void** BFDTraverse(MGraph G) {  
 queue<**char**> q;  
  
 **for** (**int** i = 1; i <= G.vexnum; ++i) {  
 Visited[i] = **false**;  
 }  
  
 **for** (**int** j = 1; j <= G.vexnum; ++j) {  
 **if** (!Visited[j]) {  
 cout << GetVex(G, j) << " ";  
 q.push(GetVex(G, j));  
 **while** (!q.empty()) {  
 **char** c = q.front();  
 q.pop();  
 **int** l = LocateVex(G, c);  
  
 **for** (**int** w = FirstAdjVex2(G, c); w > 0; w = NextAdjVex2(G, c, GetVex(G, w))) {  
 **if** (!Visited[w]) {  
 cout << GetVex(G, w) << " ";  
 Visited[w] = **true**;  
 }  
 }  
 }  
 }  
 }  
}  
  
//创建有向图  
**bool** CreateDG(MGraph &G) {  
 //输入图的点数和边数以及点的代号  
 cout << "请输入点的个数和边的个数：" << endl;  
 cin >> G.vexnum >> G.arcnum;  
 cout << "请输入" << G.vexnum << "个点的代号：" << endl;  
 **for** (**int** i = 1; i <= G.vexnum; ++i) {  
 cin >> G.vexs[i];  
 }  
  
 //初始化邻接矩阵  
 **for** (**int** j = 1; j <= G.vexnum; ++j) {  
 **for** (**int** i = 1; i < G.vexnum; ++i) {  
 G.arcs[j][i].adj = -1;  
 }  
 }  
  
 cout << "请输入有联系的点（格式：点1 点2）:" << endl;  
  
 **for** (**int** k = 1; k <= G.arcnum; ++k) {  
 **char** a, b;  
 input\_arc\_UDN3:  
 cin >> a >> b;  
 **int** i = LocateVex(G, a);  
 **int** j = LocateVex(G, b);  
  
 **if** (i == -1 || j == -1) {  
 cout << "您输入的点有误，请重新输入该路径：" << endl;  
 **goto** input\_arc\_UDN3;  
 } **else** {  
 G.arcs[i][j].adj = 1;  
 }  
 }  
  
 cout << "创建DG成功~" << endl;  
 **return true**;  
}  
  
//创建有向网  
**bool** CreateDN(MGraph &G) {  
 //输入图的点数和边数以及点的代号  
 cout << "请输入点的个数和边的个数：" << endl;  
 cin >> G.vexnum >> G.arcnum;  
 cout << "请输入" << G.vexnum << "个点的代号：" << endl;  
 **for** (**int** i = 1; i <= G.vexnum; ++i) {  
 cin >> G.vexs[i];  
 }  
  
 //初始化邻接矩阵  
 **for** (**int** j = 1; j <= G.vexnum; ++j) {  
 **for** (**int** i = 1; i < G.vexnum; ++i) {  
 G.arcs[j][i].adj = -1;  
 }  
 }  
  
 cout << "请输入边的详细信息（格式：点1 点2 距离）:" << endl;  
  
 **for** (**int** k = 1; k <= G.arcnum; ++k) {  
 **char** a, b;  
 **int** v;  
 input\_arc\_UDN4:  
 cin >> a >> b >> v;  
 **int** i = LocateVex(G, a);  
 **int** j = LocateVex(G, b);  
  
 **if** (i == -1 || j == -1) {  
 cout << "您输入的点有误，请重新输入该路径：" << endl;  
 **goto** input\_arc\_UDN4;  
 } **else if** (v <= 0) {  
 cout << "您输入的距离不合法，请重新输入该路径：" << endl;  
 **goto** input\_arc\_UDN4;  
 } **else** {  
 G.arcs[i][j].adj = v;  
 }  
 }  
  
  
 cout << "创建DN成功~" << endl;  
 **return true**;  
  
}  
  
//创建无向图  
**bool** CreateUDG(MGraph &G) {  
  
 //输入图的点数和边数以及点的代号  
 cout << "请输入点的个数和边的个数：" << endl;  
 cin >> G.vexnum >> G.arcnum;  
 cout << "请输入" << G.vexnum << "个点的代号：" << endl;  
 **for** (**int** i = 1; i <= G.vexnum; ++i) {  
 cin >> G.vexs[i];  
 }  
  
 //初始化邻接矩阵  
 **for** (**int** j = 1; j <= G.vexnum; ++j) {  
 **for** (**int** i = 1; i < G.vexnum; ++i) {  
 G.arcs[j][i].adj = -1;  
 }  
 }  
  
 cout << "请输入有联系的点（格式：点1 点2）:" << endl;  
  
 **for** (**int** k = 1; k <= G.arcnum; ++k) {  
 **char** a, b;  
 input\_arc\_UDN2:  
 cin >> a >> b;  
 **int** i = LocateVex(G, a);  
 **int** j = LocateVex(G, b);  
  
 **if** (i == -1 || j == -1) {  
 cout << "您输入的点有误，请重新输入该路径：" << endl;  
 **goto** input\_arc\_UDN2;  
 } **else** {  
 G.arcs[i][j].adj = 1;  
 G.arcs[j][i].adj = 1;  
 }  
 }  
  
 cout << "创建UDG成功~" << endl;  
 **return true**;  
}  
  
//构造无向网  
**bool** CreateUDN(MGraph &G, vector<Node> v) {  
  
  
 G.vexnum = v.size();  
 G.arcnum = G.vexnum \* G.vexnum;  
  
  
 **for** (**int** i = 0; i < G.vexnum; ++i) {  
 G.vexs[i] = v[i].code;  
 }  
  
 //初始化邻接矩阵  
 **for** (**int** j = 0; j < G.vexnum; ++j) {  
 **for** (**int** i = 0; i < G.vexnum; ++i) {  
 G.arcs[j][i].adj = -1;  
 }  
 }  
  
  
 **for** (**int** l = 0; l < G.vexnum; ++l) {  
 **for** (**int** i = 0; i < G.vexnum; ++i) {  
  
 **double** val;  
 val = sqrt(pow(v[i].x - v[l].x, 2) + pow(v[i].y - v[l].y, 2));  
 **if** (val <= 10 && val > 0) {  
  
 G.arcs[l][i].adj = val;  
 }  
  
 }  
 }  
  
 G.vexnum = v.size();  
 G.arcnum = G.vexnum \* G.vexnum;  
  
 cout << "创建UDN成功~" << endl;  
 **return true**;  
}  
  
  
#endif //INC\_1\_ADTGRAPH\_H

#include <iostream>  
#include <vector>  
#include <fstream>  
  
**using namespace** std;  
  
#include "AdtGraph.h"  
  
/\*  
 \* 算法思路：  
 \* 问题一：实际上就是求从（0,0）点出发到其他所有可达点的最短路径  
 \* 问题二：求从（0,0）点出发的最小生成树  
 \*/  
  
  
**void** Qone(MGraph &G);  
  
**void** Qtwo(MGraph G);  
  
**void** GetData(vector<Node> &v, **int** n);  
  
**void** Prim(MGraph G);  
  
**void** ShortestPath(MGraph G);  
  
**int** main() {  
  
  
 cout << "请输入节点个数：" << endl;  
 **int** n;  
 cin >> n;  
  
 vector<Node> v;  
 GetData(v, n);  
  
 MGraph G;  
  
 CreateUDN(G, v);  
  
 /\* for (int i = 1; i <= 100; ++i) {  
 for (int j = 1; j <= 100; ++j) {  
 cout << G.arcs[i][j].adj << " ";  
 }  
 cout << endl;  
 }\*/  
  
  
  
 Qone(G);  
 //Qtwo(G);  
 //ShortestPath(G);  
 Prim(G);  
 **return** 0;  
}  
  
  
**void** GetData(vector<Node> &v, **int** n) {  
 ifstream in("/home/dmrf/文档/DataStructure/CourseDesign/NetData.txt");  
  
 **if** (!in.is\_open()) {  
 cout << "文件打开失败！" << endl;  
 **return**;  
 }  
  
 Node node1;  
 node1.y = 0;  
 node1.x = 0;  
 node1.code = 0;  
 v.push\_back(node1);  
  
 **while** (!in.eof() && n > 0) {  
 Node node;  
 in >> node.code >> node.x >> node.y;  
 v.push\_back(node);  
 n--;  
 }  
  
 **if** (n >= 48) {  
 v.pop\_back();  
 }  
  
 in.close();  
 cout << "文件读取成功～" << endl;  
}  
  
  
**void** Qone(MGraph &G) {  
 **int** len[G.vexnum];  
 **int** road[G.vexnum];  
 **bool** flag[G.vexnum];  
 **for** (**int** i = 0; i < G.vexnum; ++i) {  
 len[i] = G.arcs[0][i].adj;  
 flag[i] = **false**;  
 }  
 flag[0] = **true**;  
  
  
 **int** min = 1000000;  
 **int** index = 0;  
  
 **while** (**true**) {  
 min = 1000000;  
 index = 0;  
  
 **for** (**int** i = 0; i < G.vexnum; ++i) {//找到从点1出发的最短边（该边一定是最短路径的一部分）  
 **if** (len[i] < min && !flag[i] && len[i] != -1) {  
 min = len[i];  
 index = i;  
 }  
 }  
  
 **if** (index == 0) {  
 **break**;  
 }  
 flag[index] = **true**;  
  
 **for** (**int** j = 0; j < G.vexnum; ++j) {  
 **if** (!flag[j] && G.arcs[index][j].adj != -1) {//如果点j没有归入最短点的集合而且点index到点j有路径  
  
 **if** (len[j] > len[index] + G.arcs[index][j].adj || len[j] == -1) {  
  
  
 len[j] = len[index] + G.arcs[index][j].adj;  
  
 road[j] = index;  
  
 }  
 }  
 }  
  
 }  
  
 **int** si = G.vexnum;  
  
 **int** deletenum = 0;  
 cout << "最短路径结果如下：" << endl;  
 **for** (**int** k = 1; k < si; ++k) {  
  
 **if** (!flag[k]) {  
  
  
 cout << k << "无法实现传输,该点将被剔除" << endl;  
 deletenum++;  
 DeleteVex(G, k);  
 } **else** {  
 cout << "源点到" << k;  
 cout << "最短距离为" << len[k];  
 cout << "路径为：源点--";  
 **int** temp[500];  
 **int** is = 0;  
  
 temp[is++] = k;  
  
 **int** pr = road[k];  
 **while** (pr != 0) {  
 temp[is++] = pr;  
 pr = road[pr];  
 }  
 **for** (**int** i = is - 1; i > 0; --i) {  
 cout << temp[i] << "--";  
 }  
 cout << temp[0] << endl;  
  
  
 }  
 }  
  
 cout << "共有" << deletenum << "个点被剔除" << endl;  
}  
  
  
**void** Prim(MGraph G) {  
  
 **int** min;  
 vector<**int**> a;  
 vector<**int**> b;  
 vector<**int**> minvalue;  
 vector<**int**> index;  
  
 index.push\_back(1);  
  
 a.push\_back(1);  
 **bool** flag[G.vexnum + 1];  
  
 **for** (**int** k = 1; k <= G.vexnum; ++k) {  
 flag[k] = **false**;  
 }  
 **int** min\_index;  
  
 **for** (**int** i = 2; i <= G.vexnum; ++i) {  
 **if** (G.arcs[1][i].adj != -1) {  
 min = G.arcs[1][i].adj;  
 min\_index = i;  
 **break**;  
 }  
  
 }  
  
 **for** (**int** j = min\_index + 1; j <= G.vexnum; ++j) {  
 **if** (G.arcs[1][j].adj != -1 && G.arcs[1][j].adj < min) {  
 min = G.arcs[1][j].adj;  
 min\_index = j;  
 }  
 }  
  
 flag[1] = **true**;  
 flag[min\_index] = **true**;  
 b.push\_back(min\_index);  
 minvalue.push\_back(min);  
 index.push\_back(min\_index);  
  
 **while** (index.size() != G.vexnum) {  
 **int** n = index[0];  
 **int** f = 0;  
 min = 1000000;  
  
 **for** (**int** l = 0; l < index.size(); ++l) {  
  
 **for** (**int** k = 1; k <= G.vexnum; ++k) {  
 **if** (G.arcs[index[l]][k].adj != -1 && G.arcs[index[l]][k].adj < min && !flag[k]) {  
 min = G.arcs[index[l]][k].adj;  
 min\_index = k;  
 f = l;  
 }  
 }  
 }  
  
 flag[min\_index] = **true**;  
 **int** aa = a.front();  
 **int** bb = b.front();  
 a.push\_back(index[f]);  
 b.push\_back(min\_index);  
 index.push\_back(min\_index);  
 minvalue.push\_back(min);  
  
  
 }  
  
  
 **int** va = 0;  
 **for** (**int** m = 0; m < minvalue.size(); ++m) {  
 va += minvalue[m];  
 }  
  
  
 cout << "最小能耗：" << va << endl;  
 cout << "建立以下" << a.size() << "条网络：" << endl;  
 **for** (**int** i1 = 0; i1 < a.size(); ++i1) {  
 cout << "点" << a[i1] << "与点" << b[i1] << endl;  
 }  
  
 /\* for (int j1 = 0; j1 < index.size(); ++j1) {  
 cout << index[j1] << " ";  
 }  
 \*/  
}  
  
**void** Qtwo(MGraph G) {  
  
 **int** min;  
 vector<**int**> a;//起点集合  
 vector<**int**> b;//终点集合  
 vector<**int**> minvalue;//最小值集合  
 vector<**int**> index;//最小点集合  
  
 index.push\_back(1);  
  
 a.push\_back(1);  
 **bool** flag[G.vexnum + 1];  
  
 **for** (**int** k = 1; k <= G.vexnum; ++k) {  
 flag[k] = **false**;  
 }  
 **int** min\_index;  
  
 **for** (**int** i = 2; i <= G.vexnum; ++i) {  
 **if** (G.arcs[1][i].adj != -1) {  
 min = G.arcs[1][i].adj;  
 min\_index = i;  
 **break**;  
 }  
  
 }  
  
 **for** (**int** j = min\_index + 1; j <= G.vexnum; ++j) {  
 **if** (G.arcs[1][j].adj != -1 && G.arcs[1][j].adj < min) {  
 min = G.arcs[1][j].adj;  
 min\_index = j;  
 }  
 }  
  
 flag[1] = **true**;  
 flag[min\_index] = **true**;  
 b.push\_back(min\_index);  
 minvalue.push\_back(min);  
 index.push\_back(min\_index);  
  
 **while** (index.size() != G.vexnum) {  
 **int** n = index[0];  
 **int** f = 0;  
 min = 1000000;  
 **bool** br = **false**;  
  
 **for** (**int** l = 0; l < index.size(); ++l) {  
  
 **for** (**int** k = 1; k <= G.vexnum; ++k) {  
 **if** (G.arcs[index[l]][k].adj != -1 && G.arcs[index[l]][k].adj < min && !flag[k]) {  
 br = **true**;  
 min = G.arcs[index[l]][k].adj;  
 min\_index = k;  
 f = l;  
 }  
 }  
 }  
  
 **if** (!br) {  
 cout << "不存在你想要的" << endl;  
 **break**;  
 }  
 flag[min\_index] = **true**;  
 a.push\_back(index[f]);  
 b.push\_back(min\_index);  
 index.push\_back(min\_index);  
 minvalue.push\_back(min);  
  
  
 }  
  
  
 **int** va = 0;  
 **for** (**int** m = 0; m < minvalue.size(); ++m) {  
 va += minvalue[m];  
 }  
  
 cout << "最小能耗为：" << va << endl;  
 cout << "建立以下" << a.size() << "条路径：" << endl;  
 **for** (**int** i1 = 0; i1 < a.size() - 1; ++i1) {  
 cout << a[i1] << "--" << b[i1] << endl;  
 }  
  
 /\* for (int j1 = 0; j1 < index.size(); ++j1) {  
 cout << index[j1] << " ";  
 }\*/  
  
}

# 六、

6.1 题目简介

排序算法比较 （必做）（排序）

[问题描述]

利用随机函数产生10个样本的20000个随机整数（其中之一已经是正序，之一是逆序），利用直接插入排序、希尔排序，冒泡排序、快速排序、选择排序、堆排序，归并排序、基数排序8种排序方法进行排序（结果为由小到大的顺序），并统计每一种排序算法对不同样本所耗费的时间。

[基本要求]

（1） 原始数据存在文件中，用相同样本对不同算法进行测试；

（2） 屏幕显示每种排序算法对不同样本所花的时间；

6.2 数据结构

采用vector<int>存储所要排序的数列

6.3 算法设计思想

先将随机产生的数字从文件中读取出来，然后再使用不同的算法对其进行排序，并输出排序所需时间。

直接插入排序：

从第二个开始将其和前面的数据比较，找到比其大的第一个就将该数插入到比其大的数前面，做完最后一个数的对比插入就可完成排序。

希尔排序：

将所排序数列每隔5个数当做一个小的数列进行直接插入排序，排序完成后缩小间隔为3，进行直接插入排序，最后缩小间隔为1进行最终直接插入排序。

冒泡排序：

将大数向下“沉”，小数向上“升”，最终达到有序。

快速排序：

通过一趟排序将数据分成独立的两部分，然后对这两部分分别进行排序。

选择排序：

第i次找到第i小的数据将其放在第i位，当i增长到n时即可完成排序。

堆排序：

首先建立一个大顶堆，然后将堆顶元素放在堆的最后一个，然后调整堆重新成为大顶堆，以此类推直到数列有序。

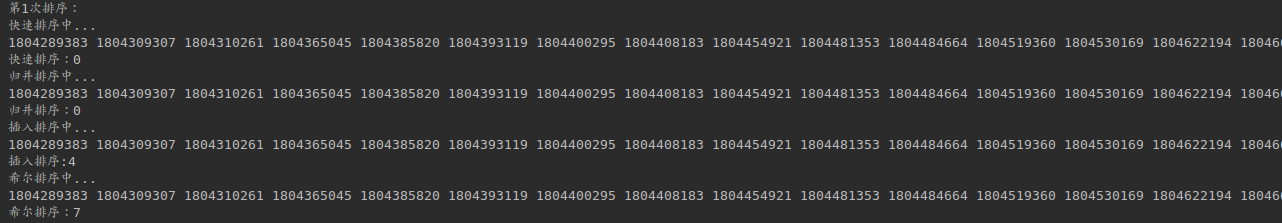
归并排序：

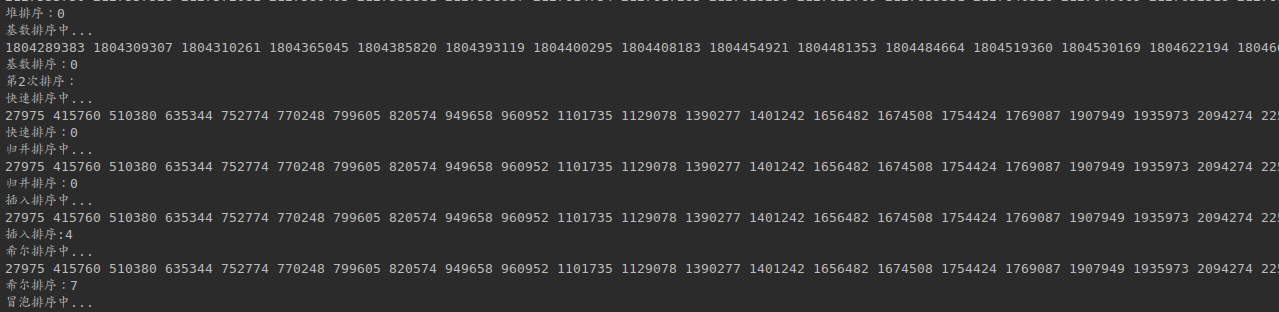
将数列看做若干个有序的子数列，对其进行合并，最终合并为一个有序的数列。

基数排序：

利用队列的思想，每次排序一个数位，最终达到有序。

6.4测试数据和结果





6.5算法时间复杂度

直接插入排序  
\* T(n)=O(n^2)  
\* S(n)=O(1)

\* 希尔排序  
\* T(n)=O(n^1.5)  
\* S(n)=O(1)

\* 冒泡排序  
\* T(n)=O(n^2)  
\* S(n)=O(1)

\* 快速排序  
\* T(n)=O(nlog2->n)  
\* S(n)=O(log2->n)

\* 选择排序  
\* S(n)=O(1)  
\* T(n)=O(n^2)

\* 堆排序  
\* T(n)=O(nlog2-->n)  
\* S(n)=O(1)

\* 归并排序  
\* T(n)=O(nlog2-->n)  
\* S(n)=O(n)

\* 基数排序  
\* T(n)=O(d\*n) d为排序数中最大数的位数  
\* S(n)=O(n)

6.6源代码

#include <iostream>  
#include <fstream>  
#include <stdlib.h>  
#include <vector>  
#include <list>  
#include <queue>  
#include <time.h>  
  
**using namespace** std;  
  
**void** MSort(vector<**int**> v);  
  
**void** getNum();  
  
**void** ShellSort(vector<**int**> v);  
  
**void** ShellInsert(vector<**int**> &v, **int** d);  
  
**void** HeapSort(vector<**int**> v);  
  
**int** Partition(vector<**int**> &v, **int** low, **int** high);  
  
**void** BubbleSort(vector<**int**> v);  
  
**void** QSort(vector<**int**> &v, **int** low, **int** high);  
  
**void** SelectSort(vector<**int**> v);  
  
**void** HeapAdjust(vector<**int**> &v, **int** s, **int** m);  
  
**void** QuickSort(vector<**int**> v);  
  
**int** GetMaxBit(vector<**int**> v);  
  
**void** InsertSort(vector<**int**> v);  
  
**void** radixSort(vector<**int**> v);  
  
**void** doSort(vector<**int**> v);  
  
**int** main() {  
  
  
 ifstream in("/home/dmrf/文档/DataStructure/CourseDesign/num.txt");  
 **if** (!in.is\_open()) {  
 cout << "文件打开失败~" << endl;  
 **return** 0;  
 }  
 **int** x = 0;  
 vector<**int**> v;  
 **int** t;  
 **int** w=1;  
 **while** (!in.eof()) {  
 in >> t;  
 v.push\_back(t);  
 **if** (v.size() == 20000) {  
 cout<<"第"<<w<<"次排序："<<endl;  
 w++;  
 doSort(v);  
 v.clear();  
 }  
 }  
  
 **return** 0;  
}  
  
**void** doSort(vector<**int**> v) {  
  
  
 vector<**int**> ve[8];  
 **for** (**int** i = 0; i < 8; ++i) {  
 ve[i] = v;  
 }  
 **double** Time;  
 time\_t end;  
 time\_t start;  
  
  
 start = clock();  
 cout << "快速排序中..." << endl;  
 QuickSort(ve[0]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "快速排序：" << Time << endl;  
  
 start = clock();  
 cout << "归并排序中..." << endl;  
 MSort(ve[1]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "归并排序：" << Time << endl;  
  
  
 start = clock();  
 cout << "插入排序中..." << endl;  
 InsertSort(ve[2]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "插入排序:" << Time << endl;  
  
 start = clock();  
 cout << "希尔排序中..." << endl;  
 ShellSort(ve[3]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "希尔排序：" << Time << endl;  
  
  
 start = clock();  
 cout << "冒泡排序中..." << endl;  
 BubbleSort(ve[4]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "冒泡排序：" << Time << endl;  
  
  
 start = clock();  
 cout << "选择排序中..." << endl;  
 SelectSort(ve[5]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "选择排序：" << Time << endl;  
  
  
 start = clock();  
 cout << "堆排序中..." << endl;  
 HeapSort(ve[6]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "堆排序：" << Time << endl;  
  
  
 start = clock();  
 cout << "基数排序中..." << endl;  
 radixSort(ve[7]);  
 end = clock();  
 Time = (end - start) / CLOCKS\_PER\_SEC;  
 cout << "基数排序：" << Time << endl;  
  
}  
  
**void** getNum() {  
  
 ofstream out("/home/dmrf/文档/DataStructure/CourseDesign/num.txt");  
  
 //生成升序数  
 **int** num = 1;  
 **int** x = rand();  
 out << x << " ";  
 **while** (**true**) {  
 **int** f = rand();  
 **if** (f > x) {  
 out << f << " ";  
 num++;  
 }  
 **if** (num == 20000) {  
 out << endl;  
 **break**;  
 }  
 }  
  
 //生成降序数  
  
  
 num = 1;  
 x = rand();  
 out << x << " ";  
 **while** (**true**) {  
 **int** f = rand();  
 **if** (f < x) {  
 out << f << " ";  
 num++;  
 }  
 **if** (num == 20000) {  
 out << endl;  
 **break**;  
 }  
 }  
  
  
 **for** (**int** i = 0; i < 8; ++i) {  
 num = 20000;  
 **while** (num > 0) {  
 out << rand() << " ";  
 num--;  
 }  
  
 out << endl;  
 }  
  
 cout << "生成成功！" << endl;  
}  
  
  
**void** show(vector<**int**> v) {  
 **for** (**int** i = 0; i < v.size(); ++i) {  
 cout << v[i] << " ";  
 }  
 cout << endl;  
}  
  
/\*  
 \* 直接插入排序  
 \* T(n)=O(n^2)  
 \* S(n)=O(1)  
 \* 稳定  
 \*/  
**void** InsertSort(vector<**int**> v) {  
  
 **for** (**int** i = 1; i < v.size(); ++i) {  
 **for** (**int** j = 0; j < i; ++j) {  
 **if** (v[j] > v[i]) {  
 **int** t = v[i];  
 **for** (**int** k = i; k > j; --k) {  
 v[k] = v[k - 1];  
 }  
 v[j] = t;  
 }  
 }  
 }  
 show(v);  
  
}  
  
  
/\*  
 \* 希尔排序  
 \* T(n)=O(n^1.5)  
 \* S(n)=O(1)  
 \* 不稳定  
 \*/  
  
  
**void** ShellSort(vector<**int**> v) {  
 **int** d[3] = {5, 3, 1};  
 **for** (**int** i = 0; i < 3; ++i) {  
 ShellInsert(v, d[i]);  
 }  
 show(v);  
}  
  
  
**void** ShellInsert(vector<**int**> &v, **int** d) {  
 **for** (**int** l = 0; l < d; ++l) {  
 **for** (**int** i = l + 1; i < v.size(); i += d) {  
 **for** (**int** j = l; j < i; j += d) {  
 **if** (v[j] > v[i]) {  
 **int** t = v[i];  
 v[i] = v[j];  
 v[j] = t;  
 }  
 }  
 }  
 }  
}  
  
  
/\*  
 \* 冒泡排序  
 \* T(n)=O(n^2)  
 \* S(n)=O(1)  
 \* 稳定  
 \*/  
  
**void** BubbleSort(vector<**int**> v) {  
  
 **for** (**int** i = 0; i < v.size(); ++i) {  
 **int** flag = 1;  
 **for** (**int** j = 0; j < v.size() - i && j < v.size() - 1; ++j) {  
 **if** (v[j] > v[j + 1]) {  
 **int** t;  
 t = v[j];  
 v[j] = v[j + 1];  
 v[j + 1] = t;  
 flag = 0;  
 }  
 }  
  
 **if** (flag == 1) {//说明已经有序了  
 **break**;  
 }  
 }  
  
 show(v);  
}  
  
  
/\*  
 \* 快速排序  
 \* T(n)=O(nlog2->n)  
 \* S(n)=O(log2->n)  
 \* 不稳定  
 \*/  
  
**void** QuickSort(vector<**int**> v) {  
 QSort(v, 0, v.size() - 1);  
 show(v);  
}  
  
**void** QSort(vector<**int**> &v, **int** low, **int** high) {  
  
 **if** (low >= high) {  
 **return**;  
 }  
 **int** t = Partition(v, low, high);  
 QSort(v, low, t - 1);  
 QSort(v, t + 1, high);  
}  
  
**int** Partition(vector<**int**> &v, **int** low, **int** high) {  
 **int** pivotkey;  
 pivotkey = v[low];  
  
 **while** (low < high) {  
 **while** (low < high && v[high] >= pivotkey) {  
 --high;  
 }  
 **int** t;  
 t = v[low];  
 v[low] = v[high];  
 v[high] = t;  
  
 **while** (low < high && v[low] <= pivotkey) {  
 ++low;  
 }  
 t = v[low];  
 v[low] = v[high];  
 v[high] = t;  
 }  
  
 **return** low;  
}  
  
  
/\*  
 \* 选择排序  
 \* S(n)=O(1)  
 \* T(n)=O(n^2)  
 \* 不稳定  
 \*/  
  
**void** SelectSort(vector<**int**> v) {  
 **for** (**int** i = 0; i < v.size(); ++i) {  
 **int** min;  
 min = i;  
 **for** (**int** j = i + 1; j < v.size(); ++j) {  
 **if** (v[j] < v[min]) {  
 min = j;  
 }  
 }  
 **int** t;  
 t = v[min];  
 v[min] = v[i];  
 v[i] = t;  
 }  
 show(v);  
}  
  
  
/\*  
 \* 堆排序  
 \* T(n)=O(nlog2-->n)  
 \* S(n)=O(1)  
 \* 不稳定  
 \*/  
  
**void** HeapSort(vector<**int**> v) {  
  
 **int** size = v.size();  
 v.push\_back(0);  
 **for** (**int** k = v.size(); k > 0; --k) {  
 v[k] = v[k - 1];  
 }  
 //建堆  
 **for** (**int** i = size / 2; i > 0; --i) {  
 HeapAdjust(v, i, size);  
 }  
  
 **for** (**int** j = size; j > 1; --j) {  
 **int** t;  
 t = v[1];  
 v[1] = v[j];  
 v[j] = t;  
 HeapAdjust(v, 1, j - 1);  
 }  
  
  
 /\*  
 \* n\_v即为排好序的vector  
 \*/  
 **for** (**int** i = 1; i < v.size(); ++i) {  
 cout << v[i] << " ";  
 }  
 cout << endl;  
}  
  
**void** HeapAdjust(vector<**int**> &v, **int** s, **int** m) {  
 /\*  
 \* 已知v[s..m]除v[s]之外均满足堆的定义，本函数调整v[s]，使得v[s..m]成为一个小顶堆  
 \*/  
  
 **int** rc;  
 rc = v[s];  
 **for** (**int** i = 2 \* s; i <= m; i = 2 \* i) {  
 **if** (i < m && v[i] < v[i + 1]) {  
 i++;//i为较大数据的下标  
 }  
 **if** (rc >= v[i]) {  
 **break**;  
 }  
  
 v[s] = v[i];  
 s = i;  
 }  
 v[s] = rc;  
}  
  
  
/\*  
 \* 归并排序  
 \* T(n)=O(nlog2-->n)  
 \* S(n)=O(n)  
 \* 稳定  
 \*/  
  
  
**void** MSort(vector<**int**> v) {  
  
  
 vector<**int**> h;  
 h = v;  
  
 **int** start, seg;  
  
  
 **for** (seg = 1; seg < v.size(); seg \*= 2) {  
 **int** k = 0;  
 **for** (start = 0; start < v.size(); start = start + seg \* 2) {  
 **int** end;  
 end = start + seg;  
 **int** low = start;  
 **while** (low < start + seg && end < start + seg + seg && low < v.size() && end < v.size()) {  
 **if** (v[low] <= v[end]) {  
 h[k++] = v[low];  
 low++;  
 } **else** {  
 h[k++] = v[end];  
 end++;  
 }  
 }  
  
 **while** (low < start + seg && low < v.size()) {  
 h[k++] = v[low++];  
 }  
 **while** (end < start + seg + seg && end < v.size()) {  
 h[k++] = v[end++];  
 }  
  
 }  
  
 v = h;  
 }  
 show(v);  
  
}  
  
  
/\*  
 \* 基数排序  
 \* T(n)=O(d\*n) d为排序数中最大数的位数  
 \* S(n)=O(n)  
 \* 稳定  
 \*  
 \*/  
  
**void** radixSort(vector<**int**> v) {  
  
 **int** d = GetMaxBit(v);  
  
 **int** \*count = **new int**[10];  
 queue<**int**> q[10];  
  
 **int** radix = 1;  
 **for** (**int** i = 0; i < d; ++i) {  
 **for** (**int** j = 0; j < v.size(); ++j) {  
 **int** t;  
 t = (v[j] / radix) % 10;  
 q[t].push(v[j]);  
 }  
  
 **int** p = 0;  
 **for** (**int** k = 0; k < 10; ++k) {  
 **while** (!q[k].empty()) {  
 v[p++] = q[k].front();  
 q[k].pop();  
 }  
 }  
 radix \*= 10;  
 }  
 show(v);  
  
}  
  
**int** GetMaxBit(vector<**int**> v) {  
 **int** max = v[0];  
 **for** (**int** i = 1; i < v.size(); ++i) {  
 max = v[i] > max ? v[i] : max;  
 }  
  
 **int** d = 1;  
 **while** (max >= 10) {  
 max /= 10;  
 d++;  
 }  
 **return** d;  
  
}

# 七、

7.1 题目简介

算术表达式求值 (选做) （栈）

[问题描述]

　　一个算术表达式是由操作数(operand)、运算符(operator)和界限符(delimiter)组成的。假设操作数是正实数，运算符只含加减乘除等四种运算符，界限符有左右括号和表达式起始、结束符“#”，如：#6+15\*（21-8/4）#。引入表达式起始、结束符是为了方便。编程利用“运算符优先法”求算术表达式的值。

[基本要求]

（1） 从键盘或文件读入一个合法的算术表达式，输出正确的结果。

（2） 显示输入序列和栈的变化过程。

（3） 考虑算法的健壮性，当表达式错误时，要给出错误原因的提示。

（4） 实现非整数的处理。

7.2 数据结构

使用string存储输入的算术表达式。

7.3 算法设计思想

使用栈的思想，将数字和运算符分别进栈，再根据运算符的优先级进行出栈运算。

同时，针对用户可能出现的常见问题，给出必要的提示，引导用户正确使用本程序。

7.4测试数据和结果





7.5算法时间复杂度

运算的时间复杂度：O（n）

7.6源代码

**package com.example.xfg.calculator**;  
  
//import java.text.NumberFormat;  
  
**import java.text.DecimalFormat**;  
  
**import android.app.Activity**;  
**import android.os.Bundle**;  
**import android.widget.**\*;  
**import android.view.View**.\*;  
**import android.view.**\*;  
  
**import java.util.**\*;  
  
**public class MainActivity extends Activity** {  
 //0~9十个按键  
 **private Button**[] btn = **new** Button[10];  
 //显示器,用于显示输出结果  
 **private EditText** input;  
 //显示器下方的记忆器，用于记录上一次计算结果  
 **private TextView** mem;  
 //三角计算时标志显示：角度还是弧度  
 **private TextView** \_drg;  
 //小提示，用于加强人机交互的弱检测、提示  
 **private TextView** tip;  
 **private Button** div, mul, sub, add, equal, // ÷ × - + =  
 sin, cos, tan, log, ln, //函数  
 sqrt, square, factorial, bksp, //根号 平方 阶乘 退格  
 left, right, dot, exit, drg, //（ ） . 退出 角度弧度控制键  
 mc, c; // mem清屏键 input清屏键  
 //保存原来的算式样子，为了输出时好看，因计算时，算式样子被改变  
 **public String** str\_old;  
 //变换样子后的式子  
 **public String** str\_new;  
 //输入控制，true为重新输入，false为接着输入  
 **public boolean** vbegin = **true**;  
 //控制DRG按键，true为角度，false为弧度  
 **public boolean** drg\_flag = **true**;  
 //π值：3.14  
 **public double** pi = 4 \* **Math**.atan(1);  
 //true表示正确，可以继续输入；false表示有误，输入被锁定  
 **public boolean** tip\_lock = **true**;  
 //判断是否是按=之后的输入，true表示输入在=之前，false反之  
 **public boolean** equals\_flag = **true**;  
  
 **public void** onCreate(**Bundle** icicle) {  
 **super**.onCreate(icicle);  
 setContentView(**R**.**layout**.main\_activity);  
 //获取界面元素  
 input = (**EditText**) findViewById(**R**.**id**.input);  
 mem = (**TextView**) findViewById(**R**.**id**.mem);  
 tip = (**TextView**) findViewById(**R**.**id**.tip);  
 \_drg = (**TextView**) findViewById(**R**.**id**.\_drg);  
 btn[0] = (**Button**) findViewById(**R**.**id**.zero);  
 btn[1] = (**Button**) findViewById(**R**.**id**.one);  
 btn[2] = (**Button**) findViewById(**R**.**id**.two);  
 btn[3] = (**Button**) findViewById(**R**.**id**.three);  
 btn[4] = (**Button**) findViewById(**R**.**id**.four);  
 btn[5] = (**Button**) findViewById(**R**.**id**.five);  
 btn[6] = (**Button**) findViewById(**R**.**id**.six);  
 btn[7] = (**Button**) findViewById(**R**.**id**.seven);  
 btn[8] = (**Button**) findViewById(**R**.**id**.eight);  
 btn[9] = (**Button**) findViewById(**R**.**id**.nine);  
 div = (**Button**) findViewById(**R**.**id**.divide);  
 mul = (**Button**) findViewById(**R**.**id**.mul);  
 sub = (**Button**) findViewById(**R**.**id**.sub);  
 add = (**Button**) findViewById(**R**.**id**.add);  
 equal = (**Button**) findViewById(**R**.**id**.equal);  
 sin = (**Button**) findViewById(**R**.**id**.sin);  
 cos = (**Button**) findViewById(**R**.**id**.cos);  
 tan = (**Button**) findViewById(**R**.**id**.tan);  
 log = (**Button**) findViewById(**R**.**id**.log);  
 ln = (**Button**) findViewById(**R**.**id**.ln);  
 sqrt = (**Button**) findViewById(**R**.**id**.sqrt);  
 square = (**Button**) findViewById(**R**.**id**.square);  
 factorial = (**Button**) findViewById(**R**.**id**.factorial);  
 bksp = (**Button**) findViewById(**R**.**id**.bksp);  
 left = (**Button**) findViewById(**R**.**id**.left);  
 right = (**Button**) findViewById(**R**.**id**.right);  
 dot = (**Button**) findViewById(**R**.**id**.dot);  
 exit = (**Button**) findViewById(**R**.**id**.exit);  
 drg = (**Button**) findViewById(**R**.**id**.drg);  
 mc = (**Button**) findViewById(**R**.**id**.mc);  
 c = (**Button**) findViewById(**R**.**id**.c);  
 //为数字按键绑定监听器  
 **for** (**int** i = 0; i < 10; ++i) {  
 btn[i].setOnClickListener(actionPerformed);  
 }  
 //为+-x÷等按键绑定监听器  
 div.setOnClickListener(actionPerformed);  
 mul.setOnClickListener(actionPerformed);  
 sub.setOnClickListener(actionPerformed);  
 add.setOnClickListener(actionPerformed);  
 equal.setOnClickListener(actionPerformed);  
 sin.setOnClickListener(actionPerformed);  
 cos.setOnClickListener(actionPerformed);  
 tan.setOnClickListener(actionPerformed);  
 log.setOnClickListener(actionPerformed);  
 ln.setOnClickListener(actionPerformed);  
 sqrt.setOnClickListener(actionPerformed);  
 square.setOnClickListener(actionPerformed);  
 factorial.setOnClickListener(actionPerformed);  
 bksp.setOnClickListener(actionPerformed);  
 left.setOnClickListener(actionPerformed);  
 right.setOnClickListener(actionPerformed);  
 dot.setOnClickListener(actionPerformed);  
 exit.setOnClickListener(actionPerformed);  
 drg.setOnClickListener(actionPerformed);  
 mc.setOnClickListener(actionPerformed);  
 c.setOnClickListener(actionPerformed);  
 }  
  
 */\*  
 \* 键盘命令捕捉  
 \*/* //命令缓存，用于检测输入合法性  
 **String**[] Tipcommand = **new** String[500];  
 //Tipcommand的指针  
 **int** tip\_i = 0;  
 **private** OnClickListener actionPerformed = **new** OnClickListener() {  
 **public void** onClick(**View** v) {  
 //按键上的命令获取  
 **String** command = ((**Button**) v).getText().toString();  
 //显示器上的字符串  
 **String** str = input.getText().toString();  
 //检测输入是否合法  
 **if** (equals\_flag == **false** && "0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1) {  
 //检测显示器上的字符串是否合法  
 **if** (right(str)) {  
 **if** ("+-×÷√^)".indexOf(command) != -1) {  
 **for** (**int** i = 0; i < str.length(); i++) {  
 Tipcommand[tip\_i] = **String**.valueOf(str.charAt(i));  
 tip\_i++;  
 }  
 vbegin = **false**;  
 }  
 } **else** {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip\_lock = **true**;  
 tip.setText("欢迎使用！");  
 }  
  
 equals\_flag = **true**;  
 }  
 **if** (tip\_i > 0)  
 TipChecker(Tipcommand[tip\_i - 1], command);  
 **else if** (tip\_i == 0) {  
 TipChecker("#", command);  
 }  
 **if** ("0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1 && tip\_lock) {  
 Tipcommand[tip\_i] = command;  
 tip\_i++;  
 }  
 //若输入正确，则将输入信息显示到显示器上  
 **if** ("0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1  
 && tip\_lock) { //共25个按键  
 print(command);  
 //若果点击了“DRG”，则切换当前弧度角度制，并将切换后的结果显示到按键上方。  
 } **else if** (command.compareTo("DRG") == 0 && tip\_lock) {  
 **if** (drg\_flag == **true**) {  
 drg\_flag = **false**;  
 \_drg.setText(" RAD");  
 } **else** {  
 drg\_flag = **true**;  
 \_drg.setText(" DEG");  
 }  
 //如果输入时退格键，并且是在按=之前  
 } **else if** (command.compareTo("Bksp") == 0 && equals\_flag) {  
 //一次删除3个字符  
 **if** (TTO(str) == 3) {  
 **if** (str.length() > 3)  
 input.setText(str.substring(0, str.length() - 3));  
 **else if** (str.length() == 3) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //依次删除2个字符  
 } **else if** (TTO(str) == 2) {  
 **if** (str.length() > 2)  
 input.setText(str.substring(0, str.length() - 2));  
 **else if** (str.length() == 2) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //依次删除一个字符  
 } **else if** (TTO(str) == 1) {  
 //若之前输入的字符串合法则删除一个字符  
 **if** (right(str)) {  
 **if** (str.length() > 1)  
 input.setText(str.substring(0, str.length() - 1));  
 **else if** (str.length() == 1) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //若之前输入的字符串不合法则删除全部字符  
 } **else** {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 }  
 **if** (input.getText().toString().compareTo("-") == 0 || equals\_flag == **false**) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 tip\_lock = **true**;  
 **if** (tip\_i > 0)  
 tip\_i--;  
 //如果是在按=之后输入退格键  
 } **else if** (command.compareTo("Bksp") == 0 && equals\_flag == **false**) {  
 //将显示器内容设置为0  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip\_lock = **true**;  
 tip.setText("欢迎使用！");  
 //如果输入的是清除键  
 } **else if** (command.compareTo("C") == 0) {  
 //将显示器内容设置为0  
 input.setText("0");  
 //重新输入标志置为true  
 vbegin = **true**;  
 //缓存命令位数清0  
 tip\_i = 0;  
 //表明可以继续输入  
 tip\_lock = **true**;  
 //表明输入=之前  
 equals\_flag = **true**;  
 tip.setText("欢迎使用！");  
 //如果输入的是”MC“，则将存储器内容清0  
 } **else if** (command.compareTo("MC") == 0) {  
 mem.setText("0");  
 //如果按”exit“则退出程序  
 } **else if** (command.compareTo("EXIT") == 0) {  
 **System**.exit(0);  
 //如果输入的是=号，并且输入合法  
 } **else if** (command.compareTo("=") == 0 && tip\_lock && right(str) && equals\_flag) {  
 tip\_i = 0;  
 //表明不可以继续输入  
 tip\_lock = **false**;  
 //表明输入=之后  
 equals\_flag = **false**;  
 //保存原来算式样子  
 str\_old = str;  
 //替换算式中的运算符，便于计算  
 str = str.replaceAll("sin", "s");  
 str = str.replaceAll("cos", "c");  
 str = str.replaceAll("tan", "t");  
 str = str.replaceAll("log", "g");  
 str = str.replaceAll("ln", "l");  
 str = str.replaceAll("n!", "!");  
 //重新输入标志设置true  
 vbegin = **true**;  
 //将-1x转换成-  
 str\_new = str.replaceAll("-", "-1×");  
 //计算算式结果  
 **new** calc().process(str\_new);  
 }  
 //表明可以继续输入  
 tip\_lock = **true**;  
 }  
 };  
  
 //向input输出字符  
 **private void** print(**String** str) {  
 //清屏后输出  
 **if** (vbegin)  
 input.setText(str);  
 //在屏幕原str后增添字符  
 **else** input.append(str);  
 vbegin = **false**;  
 }  
  
 */\*  
 \* 判断一个str是否是合法的，返回值为true、false  
 \* 只包含0123456789.()sincostanlnlogn!+-×÷√^的是合法的str，返回true  
 \* 包含了除0123456789.()sincostanlnlogn!+-×÷√^以外的字符的str为非法的，返回false  
 \*/* **private boolean** right(**String** str) {  
 **int** i = 0;  
 **for** (i = 0; i < str.length(); i++) {  
 **if** (str.charAt(i) != '0' && str.charAt(i) != '1' && str.charAt(i) != '2' &&  
 str.charAt(i) != '3' && str.charAt(i) != '4' && str.charAt(i) != '5' &&  
 str.charAt(i) != '6' && str.charAt(i) != '7' && str.charAt(i) != '8' &&  
 str.charAt(i) != '9' && str.charAt(i) != '.' && str.charAt(i) != '-' &&  
 str.charAt(i) != '+' && str.charAt(i) != '×' && str.charAt(i) != '÷' &&  
 str.charAt(i) != '√' && str.charAt(i) != '^' && str.charAt(i) != 's' &&  
 str.charAt(i) != 'i' && str.charAt(i) != 'n' && str.charAt(i) != 'c' &&  
 str.charAt(i) != 'o' && str.charAt(i) != 't' && str.charAt(i) != 'a' &&  
 str.charAt(i) != 'l' && str.charAt(i) != 'g' && str.charAt(i) != '(' &&  
 str.charAt(i) != ')' && str.charAt(i) != '!')  
 **break**;  
 }  
 **if** (i == str.length()) {  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 */\*  
 \* 检测函数，返回值为3、2、1 表示应当一次删除几个？ Three+Two+One = TTO  
 \* 为Bksp按钮的删除方式提供依据  
 \* 返回3，表示str尾部为sin、cos、tan、log中的一个，应当一次删除3个  
 \* 返回2，表示str尾部为ln、n!中的一个，应当一次删除2个  
 \* 返回1，表示为除返回3、2外的所有情况，只需删除一个（包含非法字符时要另外考虑：应清屏）  
 \*/* **private int** TTO(**String** str) {  
 **if** ((str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'i' &&  
 str.charAt(str.length() - 3) == 's') ||  
 (str.charAt(str.length() - 1) == 's' &&  
 str.charAt(str.length() - 2) == 'o' &&  
 str.charAt(str.length() - 3) == 'c') ||  
 (str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'a' &&  
 str.charAt(str.length() - 3) == 't') ||  
 (str.charAt(str.length() - 1) == 'g' &&  
 str.charAt(str.length() - 2) == 'o' &&  
 str.charAt(str.length() - 3) == 'l')) {  
 **return** 3;  
 } **else if** ((str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'l') ||  
 (str.charAt(str.length() - 1) == '!' &&  
 str.charAt(str.length() - 2) == 'n')) {  
 **return** 2;  
 } **else** {  
 **return** 1;  
 }  
 }  
  
 */\*  
 \* 检测函数，对str进行前后语法检测  
 \* 为Tip的提示方式提供依据，与TipShow()配合使用  
 \* 编号 字符 其后可以跟随的合法字符  
 \* 1 （ 数字|（|-|.|函数  
 \* 2 ） 算符|）|√ ^  
 \* 3 . 数字|算符|）|√ ^  
 \* 4 数字 .|数字|算符|）|√ ^  
 \* 5 算符 数字|（|.|函数  
 \* 6 √ ^ （ |. | 数字  
 \* 7 函数 数字|（|.  
 \*  
 \* 小数点前后均可省略，表示0  
 \* 数字第一位可以为0  
 \*/* **private void** TipChecker(**String** tipcommand1, **String** tipcommand2) {  
 //Tipcode1表示错误类型，Tipcode2表示名词解释类型  
 **int** Tipcode1 = 0, Tipcode2 = 0;  
 //表示命令类型  
 **int** tiptype1 = 0, tiptype2 = 0;  
 //括号数  
 **int** bracket = 0;  
 //“+-x÷√^”不能作为第一位  
 **if** (tipcommand1.compareTo("#") == 0 && (tipcommand2.compareTo("÷") == 0 ||  
 tipcommand2.compareTo("×") == 0 || tipcommand2.compareTo("+") == 0 ||  
 tipcommand2.compareTo(")") == 0 || tipcommand2.compareTo("√") == 0 ||  
 tipcommand2.compareTo("^") == 0)) {  
 Tipcode1 = -1;  
 }  
 //定义存储字符串中最后一位的类型  
 **else if** (tipcommand1.compareTo("#") != 0) {  
 **if** (tipcommand1.compareTo("(") == 0) {  
 tiptype1 = 1;  
 } **else if** (tipcommand1.compareTo(")") == 0) {  
 tiptype1 = 2;  
 } **else if** (tipcommand1.compareTo(".") == 0) {  
 tiptype1 = 3;  
 } **else if** ("0123456789".indexOf(tipcommand1) != -1) {  
 tiptype1 = 4;  
 } **else if** ("+-×÷".indexOf(tipcommand1) != -1) {  
 tiptype1 = 5;  
 } **else if** ("√^".indexOf(tipcommand1) != -1) {  
 tiptype1 = 6;  
 } **else if** ("sincostanloglnn!".indexOf(tipcommand1) != -1) {  
 tiptype1 = 7;  
 }  
 //定义欲输入的按键类型  
 **if** (tipcommand2.compareTo("(") == 0) {  
 tiptype2 = 1;  
 } **else if** (tipcommand2.compareTo(")") == 0) {  
 tiptype2 = 2;  
 } **else if** (tipcommand2.compareTo(".") == 0) {  
 tiptype2 = 3;  
 } **else if** ("0123456789".indexOf(tipcommand2) != -1) {  
 tiptype2 = 4;  
 } **else if** ("+-×÷".indexOf(tipcommand2) != -1) {  
 tiptype2 = 5;  
 } **else if** ("√^".indexOf(tipcommand2) != -1) {  
 tiptype2 = 6;  
 } **else if** ("sincostanloglnn!".indexOf(tipcommand2) != -1) {  
 tiptype2 = 7;  
 }  
  
 **switch** (tiptype1) {  
 **case** 1:  
 //左括号后面直接接右括号,“+x÷”（负号“-”不算）,或者"√^"  
 **if** (tiptype2 == 2 || (tiptype2 == 5 && tipcommand2.compareTo("-") != 0) ||  
 tiptype2 == 6)  
 Tipcode1 = 1;  
 **break**;  
 **case** 2:  
 //右括号后面接左括号，数字，“+-x÷sin^...”  
 **if** (tiptype2 == 1 || tiptype2 == 3 || tiptype2 == 4 || tiptype2 == 7)  
 Tipcode1 = 2;  
 **break**;  
 **case** 3:  
 //“.”后面接左括号或者“sincos...”  
 **if** (tiptype2 == 1 || tiptype2 == 7)  
 Tipcode1 = 3;  
 //连续输入两个“.”  
 **if** (tiptype2 == 3)  
 Tipcode1 = 8;  
 **break**;  
 **case** 4:  
 //数字后面直接接左括号或者“sincos...”  
 **if** (tiptype2 == 1 || tiptype2 == 7)  
 Tipcode1 = 4;  
 **break**;  
 **case** 5:  
 //“+-x÷”后面直接接右括号，“+-x÷√^”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6)  
 Tipcode1 = 5;  
 **break**;  
 **case** 6:  
 //“√^”后面直接接右括号，“+-x÷√^”以及“sincos...”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6 || tiptype2 == 7)  
 Tipcode1 = 6;  
 **break**;  
 **case** 7:  
 //“sincos...”后面直接接右括号“+-x÷√^”以及“sincos...”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6 || tiptype2 == 7)  
 Tipcode1 = 7;  
 **break**;  
 }  
 }  
 //检测小数点的重复性，Tipconde1=0,表明满足前面的规则  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo(".") == 0) {  
 **int** tip\_point = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 //若之前出现一个小数点点，则小数点计数加1  
 **if** (Tipcommand[i].compareTo(".") == 0) {  
 tip\_point++;  
 }  
 //若出现以下几个运算符之一，小数点计数清零  
 **if** (Tipcommand[i].compareTo("sin") == 0 || Tipcommand[i].compareTo("cos") == 0 ||  
 Tipcommand[i].compareTo("tan") == 0 || Tipcommand[i].compareTo("log") == 0 ||  
 Tipcommand[i].compareTo("ln") == 0 || Tipcommand[i].compareTo("n!") == 0 ||  
 Tipcommand[i].compareTo("√") == 0 || Tipcommand[i].compareTo("^") == 0 ||  
 Tipcommand[i].compareTo("÷") == 0 || Tipcommand[i].compareTo("×") == 0 ||  
 Tipcommand[i].compareTo("-") == 0 || Tipcommand[i].compareTo("+") == 0 ||  
 Tipcommand[i].compareTo("(") == 0 || Tipcommand[i].compareTo(")") == 0) {  
 tip\_point = 0;  
 }  
 }  
 tip\_point++;  
 //若小数点计数大于1，表明小数点重复了  
 **if** (tip\_point > 1) {  
 Tipcode1 = 8;  
 }  
 }  
 //检测右括号是否匹配  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo(")") == 0) {  
 **int** tip\_right\_bracket = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 //如果出现一个左括号，则计数加1  
 **if** (Tipcommand[i].compareTo("(") == 0) {  
 tip\_right\_bracket++;  
 }  
 //如果出现一个右括号，则计数减1  
 **if** (Tipcommand[i].compareTo(")") == 0) {  
 tip\_right\_bracket--;  
 }  
 }  
 //如果右括号计数=0,表明没有响应的左括号与当前右括号匹配  
 **if** (tip\_right\_bracket == 0) {  
 Tipcode1 = 10;  
 }  
 }  
 //检查输入=的合法性  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo("=") == 0) {  
 //括号匹配数  
 **int** tip\_bracket = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 **if** (Tipcommand[i].compareTo("(") == 0) {  
 tip\_bracket++;  
 }  
 **if** (Tipcommand[i].compareTo(")") == 0) {  
 tip\_bracket--;  
 }  
 }  
 //若大于0，表明左括号还有未匹配的  
 **if** (tip\_bracket > 0) {  
 Tipcode1 = 9;  
 bracket = tip\_bracket;  
 } **else if** (tip\_bracket == 0) {  
 //若前一个字符是以下之一，表明=号不合法  
 **if** ("√^sincostanloglnn!".indexOf(tipcommand1) != -1) {  
 Tipcode1 = 6;  
 }  
 //若前一个字符是以下之一，表明=号不合法  
 **if** ("+-×÷".indexOf(tipcommand1) != -1) {  
 Tipcode1 = 5;  
 }  
 }  
 }  
 //若命令式以下之一，则显示相应的帮助信息  
 **if** (tipcommand2.compareTo("MC") == 0) Tipcode2 = 1;  
 **if** (tipcommand2.compareTo("C") == 0) Tipcode2 = 2;  
 **if** (tipcommand2.compareTo("DRG") == 0) Tipcode2 = 3;  
 **if** (tipcommand2.compareTo("Bksp") == 0) Tipcode2 = 4;  
 **if** (tipcommand2.compareTo("sin") == 0) Tipcode2 = 5;  
 **if** (tipcommand2.compareTo("cos") == 0) Tipcode2 = 6;  
 **if** (tipcommand2.compareTo("tan") == 0) Tipcode2 = 7;  
 **if** (tipcommand2.compareTo("log") == 0) Tipcode2 = 8;  
 **if** (tipcommand2.compareTo("ln") == 0) Tipcode2 = 9;  
 **if** (tipcommand2.compareTo("n!") == 0) Tipcode2 = 10;  
 **if** (tipcommand2.compareTo("√") == 0) Tipcode2 = 11;  
 **if** (tipcommand2.compareTo("^") == 0) Tipcode2 = 12;  
 //显示帮助和错误信息  
 TipShow(bracket, Tipcode1, Tipcode2, tipcommand1, tipcommand2);  
 }  
  
 */\*  
 \* 反馈Tip信息，加强人机交互，与TipChecker()配合使用  
 \*/* **private void** TipShow(**int** bracket, **int** tipcode1, **int** tipcode2,  
 **String** tipcommand1, **String** tipcommand2) {  
 **String** tipmessage = "";  
 **if** (tipcode1 != 0)  
 tip\_lock = **false**;//表明输入有误  
 **switch** (tipcode1) {  
 **case** -1:  
 tipmessage = tipcommand2 + " 不能作为第一个算符\n";  
 **break**;  
 **case** 1:  
 tipmessage = tipcommand1 + " 后应输入：数字/(/./-/函数 \n";  
 **break**;  
 **case** 2:  
 tipmessage = tipcommand1 + " 后应输入：)/算符 \n";  
 **break**;  
 **case** 3:  
 tipmessage = tipcommand1 + " 后应输入：)/数字/算符 \n";  
 **break**;  
 **case** 4:  
 tipmessage = tipcommand1 + " 后应输入：)/./数字 /算符 \n";  
 **break**;  
 **case** 5:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字/函数 \n";  
 **break**;  
 **case** 6:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字 \n";  
 **break**;  
 **case** 7:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字 \n";  
 **break**;  
 **case** 8:  
 tipmessage = "小数点重复\n";  
 **break**;  
 **case** 9:  
 tipmessage = "不能计算，缺少 " + bracket + " 个 )";  
 **break**;  
 **case** 10:  
 tipmessage = "不需要 )";  
 **break**;  
 }  
 **switch** (tipcode2) {  
 **case** 1:  
 tipmessage = tipmessage + "[MC 用法: 清除记忆 MEM]";  
 **break**;  
 **case** 2:  
 tipmessage = tipmessage + "[C 用法: 归零]";  
 **break**;  
 **case** 3:  
 tipmessage = tipmessage + "[DRG 用法: 选择 DEG 或 RAD]";  
 **break**;  
 **case** 4:  
 tipmessage = tipmessage + "[Bksp 用法: 退格]";  
 **break**;  
 **case** 5:  
 tipmessage = tipmessage + "sin 函数用法示例：\n" +  
 "DEG：sin30 = 0.5 RAD：sin1 = 0.84\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "sin(cos45)，而不是sincos45";  
 **break**;  
 **case** 6:  
 tipmessage = tipmessage + "cos 函数用法示例：\n" +  
 "DEG：cos60 = 0.5 RAD：cos1 = 0.54\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "cos(sin45)，而不是cossin45";  
 **break**;  
 **case** 7:  
 tipmessage = tipmessage + "tan 函数用法示例：\n" +  
 "DEG：tan45 = 1 RAD：tan1 = 1.55\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "tan(cos45)，而不是tancos45";  
 **break**;  
 **case** 8:  
 tipmessage = tipmessage + "log 函数用法示例：\n" +  
 "log10 = log(5+5) = 1\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "log(tan45)，而不是logtan45";  
 **break**;  
 **case** 9:  
 tipmessage = tipmessage + "ln 函数用法示例：\n" +  
 "ln10 = le(5+5) = 2.3 lne = 1\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "ln(tan45)，而不是lntan45";  
 **break**;  
 **case** 10:  
 tipmessage = tipmessage + "n! 函数用法示例：\n" +  
 "n!3 = n!(1+2) = 3×2×1 = 6\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "n!(log1000)，而不是n!log1000";  
 **break**;  
 **case** 11:  
 tipmessage = tipmessage + "√ 用法示例：开任意次根号\n" +  
 "如：27开3次根为 27√3 = 3\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "(函数)√(函数) ， (n!3)√(log100) = 2.45";  
 **break**;  
 **case** 12:  
 tipmessage = tipmessage + "^ 用法示例：开任意次平方\n" +  
 "如：2的3次方为 2^3 = 8\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "(函数)√(函数) ， (n!3)^(log100) = 36";  
 **break**;  
 }  
 //将提示信息显示到tip  
 tip.setText(tipmessage);  
 }  
  
 */\*  
 \* 整个计算核心，只要将表达式的整个字符串传入calc().process()就可以实行计算了  
 \* 算法包括以下几部分：  
 \* 1、计算部分 process(String str) 当然，这是建立在查错无错误的情况下  
 \* 2、数据格式化 FP(double n) 使数据有相当的精确度  
 \* 3、阶乘算法 N(double n) 计算n!，将结果返回  
 \* 4、错误提示 showError(int code ,String str) 将错误返回  
 \*/* **public class calc** {  
 **public** calc() {  
  
 }  
  
 **final int** MAXLEN = 500;  
  
 */\*  
 \* 计算表达式  
 \* 从左向右扫描，数字入number栈，运算符入operator栈  
 \* +-基本优先级为1，×÷基本优先级为2，log ln sin cos tan n!基本优先级为3，√^基本优先级为4  
 \* 括号内层运算符比外层同级运算符优先级高4  
 \* 当前运算符优先级高于栈顶压栈，低于栈顶弹出一个运算符与两个数进行运算  
 \* 重复直到当前运算符大于栈顶  
 \* 扫描完后对剩下的运算符与数字依次计算  
 \*/* **public void** process(**String** str) {  
 **int** weightPlus = 0, topOP = 0,  
 topNum = 0, flag = 1, weightTemp = 0;  
 //weightPlus为同一（）下的基本优先级，weightTemp临时记录优先级的变化  
 //topOp为weight[]，operator[]的计数器；topNum为number[]的计数器  
 //flag为正负数的计数器，1为正数，-1为负数  
  
  
 **int** weight[];//保存operator栈中运算符的优先级,以topOp计数  
 **double** number[];//保存数字，以topNum计数  
  
  
 **char** ch, ch\_gai = 0, operator[];//operator[]保存运算符，以topOP计数  
 **String** num = **null**;//记录数字，str以+-×÷()sctgl!√^分段，+-×÷()sctgl!√^字符之间的字符串即为数字  
 weight = **new int**[MAXLEN];  
 number = **new double**[MAXLEN];  
 operator = **new char**[MAXLEN];  
 **String** expression = str;  
 **StringTokenizer** expToken = **new** StringTokenizer(expression, "+-×÷()sctgl!√^");  
 **int** i = 0;  
 **while** (i < expression.length()) {  
 ch = expression.charAt(i);  
 //判断正负数  
 **if** (i == 0) {  
 **if** (ch == '-') {  
 flag = -1;  
 }  
 } **else if** (expression.charAt(i - 1) == '(' && ch == '-') {  
 flag = -1;  
 }  
  
 //取得数字，并将正负号转移给数字  
 **if** (ch <= '9' && ch >= '0' || ch == '.' || ch == 'E') {  
 num = expToken.nextToken();//取出ch对应段的那个数字  
 ch\_gai = ch;  
 //将i向后移直至ch\_gai取得的字符不是数字  
 **while** (i < expression.length() && (ch\_gai <= '9' && ch\_gai >= '0' || ch\_gai == '.' || ch\_gai == 'E')) {  
 ch\_gai = expression.charAt(i++);  
 }  
 //将ch\_gai多移的位数移回来  
 **if** (i >= expression.length()) {  
 //说明已经取完了最后一个数字  
 i -= 1;  
 } **else** {  
 i -= 2;  
 }  
 //如果num只取到了一个 “.” 则将其当做0处理  
 **if** (num.compareTo(".") == 0) {  
 number[topNum++] = 0;  
 } **else** {  
 //将正负号转移到数值上  
 number[topNum++] = **Double**.parseDouble(num) \* flag;  
 flag = 1;  
 }  
 }  
 //计算运算符的优先级  
 //先检查有没有括号，如果有括号，就应该记录下来，然后在原来的基础上加4个优先级  
 **if** (ch == '(' || ch == ')') {  
 weightPlus += 4;  
 }  
 **if** (ch == '-' && flag == 1 || ch == '+' || ch == '×' || ch == '÷' ||  
 ch == 's' || ch == 'c' || ch == 't' || ch == 'g' || ch == 'l' ||  
 ch == '!' || ch == '√' || ch == '^') {  
 **switch** (ch) {  
 //+-的优先级最低，为1  
 **case** '+':  
 **case** '-':  
 weightTemp = 1 + weightPlus;  
 **break**;  
 //'×''÷'的优先级稍高，为2  
 **case** '÷':  
 **case** '×':  
 weightTemp = 2 + weightPlus;  
 **break**;  
 //sincos之类优先级为3：  
 **case** 's':  
 **case** 'c':  
 **case** 't':  
 **case** 'g':  
 **case** 'l':  
 **case** '!':  
 weightTemp = 3 + weightPlus;  
 **break**;  
 //其余优先级为4  
 //case '^':  
 //case '√':  
 **default**:  
 weightTemp = 4 + weightPlus;  
 **break**;  
 }  
 //如果当前优先级大于堆栈顶部元素，直接入栈  
 **if** (topOP == 0 || weight[topOP - 1] < weightTemp) {  
 weight[topOP] = weightTemp;  
 operator[topOP] = ch;  
 topOP++;  
 } **else** {  
 //否则将堆栈中运算符逐个取出，直到当前堆栈顶部运算符优先级小于当前优先级  
 **while** (topOP > 0 && weight[topOP - 1] >= weightTemp) {  
 **switch** (operator[topOP - 1]) {  
 //取出数字数组的相应元素进行计算  
 **case** '+':  
 number[topNum - 2] += number[topNum - 1];  
 **break**;  
 **case** '-':  
 number[topNum - 2] -= number[topNum - 1];  
 **break**;  
 **case** '×':  
 number[topNum - 2] \*= number[topNum - 1];  
 **break**;  
 **case** '÷':  
 //判断除数为0的情况  
  
 **if** (number[topNum - 1] == 0) {  
 showError(1, str\_old);  
 **return**;  
 }  
 number[topNum - 2] /= number[topNum - 1];  
 **break**;  
 **case** '√':  
 **if** (number[topNum - 1] == 0 || (number[topNum - 2] < 0 && number[topNum - 1] % 2 == 0)) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 2] = **Math**.pow(number[topNum - 2], 1 / number[topNum - 1]);  
 **break**;  
 **case** '^':  
 number[topNum - 2] = **Math**.pow(number[topNum - 2], number[topNum - 1]);  
 **break**;  
 //进行角度弧度判断的计算及转换  
 //sin  
 //sin  
 **case** 's':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.sin((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.sin(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 'c':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.cos((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.cos(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 't':  
 **if** (drg\_flag == **true**) {  
 **if** ((**Math**.abs(number[topNum - 1]) / 90) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 **if** ((**Math**.abs(number[topNum - 1]) / (pi / 2)) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 'g':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log10(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //ln  
 **case** 'l':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //阶乘  
 **case** '!':  
 **if** (number[topNum - 1] > 170) {  
 showError(3, str\_old);  
 **return**;  
 } **else if** (number[topNum - 1] < 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = N(number[topNum - 1]);  
 topNum++;  
 **break**;  
 }  
 //继续取堆栈中的下一个元素进行判断  
 topNum--;  
 topOP--;  
 }  
 //将运算符入堆栈  
 weight[topOP] = weightTemp;  
 operator[topOP] = ch;  
 topOP++;  
 }  
 }  
 i++;  
 }  
 //依次取出堆栈中的运算符进行运算  
 **while** (topOP > 0) {  
//+-x直接将数组的后两位数取出运算  
 **switch** (operator[topOP - 1]) {  
 **case** '+':  
 number[topNum - 2] += number[topNum - 1];  
 **break**;  
 **case** '-':  
 number[topNum - 2] -= number[topNum - 1];  
 **break**;  
 **case** '×':  
 number[topNum - 2] \*= number[topNum - 1];  
 **break**;  
 //涉及到除法时要考虑除数不能为零的情况  
 **case** '÷':  
 **if** (number[topNum - 1] == 0) {  
 showError(1, str\_old);  
 **return**;  
 }  
 number[topNum - 2] /= number[topNum - 1];  
 **break**;  
 **case** '√':  
 **if** (number[topNum - 1] == 0 || (number[topNum - 2] < 0 &&  
 number[topNum - 1] % 2 == 0)) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 2] =  
 **Math**.pow(number[topNum - 2], 1 / number[topNum - 1]);  
 **break**;  
 **case** '^':  
 number[topNum - 2] =  
 **Math**.pow(number[topNum - 2], number[topNum - 1]);  
 **break**;  
 //sin  
 **case** 's':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.sin((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.sin(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //cos  
 **case** 'c':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.cos((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.cos(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //tan  
 **case** 't':  
 **if** (drg\_flag == **true**) {  
 **if** ((**Math**.abs(number[topNum - 1]) / 90) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 **if** ((**Math**.abs(number[topNum - 1]) / (pi / 2)) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //对数log  
 **case** 'g':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log10(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //自然对数ln  
 **case** 'l':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //阶乘  
 **case** '!':  
 **if** (number[topNum - 1] > 170) {  
 showError(3, str\_old);  
 **return**;  
 } **else if** (number[topNum - 1] < 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = N(number[topNum - 1]);  
 topNum++;  
 **break**;  
 }  
 //取堆栈下一个元素计算  
 topNum--;  
 topOP--;  
 }  
 //如果数字太大，就提示错误信息  
 **if** (number[0] > 7.3E306) {  
 showError(3, str\_old);  
 **return**;  
 }  
 //输出最终结果  
 input.setText(**String**.valueOf(FP(number[0])));  
 tip.setText("计算完毕，要继续请按归零见C");  
 mem.setText(str\_old + "=" + **String**.valueOf(FP(number[0])));  
 }  
  
 */\*  
 \* FP = floating point 控制小数位数，达到精度  
 \* 否则会出现 0.6-0.2=0.39999999999999997的情况，用FP即可解决，使得数为0.4  
 \* 本格式精度为15位  
 \*/* **public double** FP(**double** n) {  
 //NumberFormat format=NumberFormat.getInstance(); //创建一个格式化类f  
 //format.setMaximumFractionDigits(18); //设置小数位的格式  
 **DecimalFormat** format = **new** DecimalFormat("0.#############");  
 **return Double**.parseDouble(format.format(n));  
 }  
  
 */\*  
 \* 阶乘算法  
 \*/* **public double** N(**double** n) {  
 **int** i = 0;  
 **double** sum = 1;  
 //依次将小于等于n的值相乘  
 **for** (i = 1; i <= n; i++) {  
 sum = sum \* i;  
 }  
 **return** sum;  
 }  
  
 */\*  
 \* 错误提示，按了"="之后，若计算式在process()过程中，出现错误，则进行提示  
 \*/* **public void** showError(**int** code, **String** str) {  
 **String** message = "";  
 **switch** (code) {  
 **case** 1:  
 message = "零不能作除数";  
 **break**;  
 **case** 2:  
 message = "函数格式错误";  
 **break**;  
 **case** 3:  
 message = "值太大了，超出范围";  
 }  
 input.setText("\"" + str + "\"" + ": " + message);  
 tip.setText(message + "\n" + "计算完毕，要继续请按归零键 C");  
 }  
 }  
}

**package com.example.xfg.calculator**;  
  
//import java.text.NumberFormat;  
  
**import java.text.DecimalFormat**;  
  
**import android.app.Activity**;  
**import android.os.Bundle**;  
**import android.widget.**\*;  
**import android.view.View**.\*;  
**import android.view.**\*;  
  
**import java.util.**\*;  
  
**public class MainActivity extends Activity** {  
 //0~9十个按键  
 **private Button**[] btn = **new** Button[10];  
 //显示器,用于显示输出结果  
 **private EditText** input;  
 //显示器下方的记忆器，用于记录上一次计算结果  
 **private TextView** mem;  
 //三角计算时标志显示：角度还是弧度  
 **private TextView** \_drg;  
 //小提示，用于加强人机交互的弱检测、提示  
 **private TextView** tip;  
 **private Button** div, mul, sub, add, equal, // ÷ × - + =  
 sin, cos, tan, log, ln, //函数  
 sqrt, square, factorial, bksp, //根号 平方 阶乘 退格  
 left, right, dot, exit, drg, //（ ） . 退出 角度弧度控制键  
 mc, c; // mem清屏键 input清屏键  
 //保存原来的算式样子，为了输出时好看，因计算时，算式样子被改变  
 **public String** str\_old;  
 //变换样子后的式子  
 **public String** str\_new;  
 //输入控制，true为重新输入，false为接着输入  
 **public boolean** vbegin = **true**;  
 //控制DRG按键，true为角度，false为弧度  
 **public boolean** drg\_flag = **true**;  
 //π值：3.14  
 **public double** pi = 4 \* **Math**.atan(1);  
 //true表示正确，可以继续输入；false表示有误，输入被锁定  
 **public boolean** tip\_lock = **true**;  
 //判断是否是按=之后的输入，true表示输入在=之前，false反之  
 **public boolean** equals\_flag = **true**;  
  
 **public void** onCreate(**Bundle** icicle) {  
 **super**.onCreate(icicle);  
 setContentView(**R**.**layout**.main\_activity);  
 //获取界面元素  
 input = (**EditText**) findViewById(**R**.**id**.input);  
 mem = (**TextView**) findViewById(**R**.**id**.mem);  
 tip = (**TextView**) findViewById(**R**.**id**.tip);  
 \_drg = (**TextView**) findViewById(**R**.**id**.\_drg);  
 btn[0] = (**Button**) findViewById(**R**.**id**.zero);  
 btn[1] = (**Button**) findViewById(**R**.**id**.one);  
 btn[2] = (**Button**) findViewById(**R**.**id**.two);  
 btn[3] = (**Button**) findViewById(**R**.**id**.three);  
 btn[4] = (**Button**) findViewById(**R**.**id**.four);  
 btn[5] = (**Button**) findViewById(**R**.**id**.five);  
 btn[6] = (**Button**) findViewById(**R**.**id**.six);  
 btn[7] = (**Button**) findViewById(**R**.**id**.seven);  
 btn[8] = (**Button**) findViewById(**R**.**id**.eight);  
 btn[9] = (**Button**) findViewById(**R**.**id**.nine);  
 div = (**Button**) findViewById(**R**.**id**.divide);  
 mul = (**Button**) findViewById(**R**.**id**.mul);  
 sub = (**Button**) findViewById(**R**.**id**.sub);  
 add = (**Button**) findViewById(**R**.**id**.add);  
 equal = (**Button**) findViewById(**R**.**id**.equal);  
 sin = (**Button**) findViewById(**R**.**id**.sin);  
 cos = (**Button**) findViewById(**R**.**id**.cos);  
 tan = (**Button**) findViewById(**R**.**id**.tan);  
 log = (**Button**) findViewById(**R**.**id**.log);  
 ln = (**Button**) findViewById(**R**.**id**.ln);  
 sqrt = (**Button**) findViewById(**R**.**id**.sqrt);  
 square = (**Button**) findViewById(**R**.**id**.square);  
 factorial = (**Button**) findViewById(**R**.**id**.factorial);  
 bksp = (**Button**) findViewById(**R**.**id**.bksp);  
 left = (**Button**) findViewById(**R**.**id**.left);  
 right = (**Button**) findViewById(**R**.**id**.right);  
 dot = (**Button**) findViewById(**R**.**id**.dot);  
 exit = (**Button**) findViewById(**R**.**id**.exit);  
 drg = (**Button**) findViewById(**R**.**id**.drg);  
 mc = (**Button**) findViewById(**R**.**id**.mc);  
 c = (**Button**) findViewById(**R**.**id**.c);  
 //为数字按键绑定监听器  
 **for** (**int** i = 0; i < 10; ++i) {  
 btn[i].setOnClickListener(actionPerformed);  
 }  
 //为+-x÷等按键绑定监听器  
 div.setOnClickListener(actionPerformed);  
 mul.setOnClickListener(actionPerformed);  
 sub.setOnClickListener(actionPerformed);  
 add.setOnClickListener(actionPerformed);  
 equal.setOnClickListener(actionPerformed);  
 sin.setOnClickListener(actionPerformed);  
 cos.setOnClickListener(actionPerformed);  
 tan.setOnClickListener(actionPerformed);  
 log.setOnClickListener(actionPerformed);  
 ln.setOnClickListener(actionPerformed);  
 sqrt.setOnClickListener(actionPerformed);  
 square.setOnClickListener(actionPerformed);  
 factorial.setOnClickListener(actionPerformed);  
 bksp.setOnClickListener(actionPerformed);  
 left.setOnClickListener(actionPerformed);  
 right.setOnClickListener(actionPerformed);  
 dot.setOnClickListener(actionPerformed);  
 exit.setOnClickListener(actionPerformed);  
 drg.setOnClickListener(actionPerformed);  
 mc.setOnClickListener(actionPerformed);  
 c.setOnClickListener(actionPerformed);  
 }  
  
 */\*  
 \* 键盘命令捕捉  
 \*/* //命令缓存，用于检测输入合法性  
 **String**[] Tipcommand = **new** String[500];  
 //Tipcommand的指针  
 **int** tip\_i = 0;  
 **private** OnClickListener actionPerformed = **new** OnClickListener() {  
 **public void** onClick(**View** v) {  
 //按键上的命令获取  
 **String** command = ((**Button**) v).getText().toString();  
 //显示器上的字符串  
 **String** str = input.getText().toString();  
 //检测输入是否合法  
 **if** (equals\_flag == **false** && "0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1) {  
 //检测显示器上的字符串是否合法  
 **if** (right(str)) {  
 **if** ("+-×÷√^)".indexOf(command) != -1) {  
 **for** (**int** i = 0; i < str.length(); i++) {  
 Tipcommand[tip\_i] = **String**.valueOf(str.charAt(i));  
 tip\_i++;  
 }  
 vbegin = **false**;  
 }  
 } **else** {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip\_lock = **true**;  
 tip.setText("欢迎使用！");  
 }  
  
 equals\_flag = **true**;  
 }  
 **if** (tip\_i > 0)  
 TipChecker(Tipcommand[tip\_i - 1], command);  
 **else if** (tip\_i == 0) {  
 TipChecker("#", command);  
 }  
 **if** ("0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1 && tip\_lock) {  
 Tipcommand[tip\_i] = command;  
 tip\_i++;  
 }  
 //若输入正确，则将输入信息显示到显示器上  
 **if** ("0123456789.()sincostanlnlogn!+-×÷√^".indexOf(command) != -1  
 && tip\_lock) { //共25个按键  
 print(command);  
 //若果点击了“DRG”，则切换当前弧度角度制，并将切换后的结果显示到按键上方。  
 } **else if** (command.compareTo("DRG") == 0 && tip\_lock) {  
 **if** (drg\_flag == **true**) {  
 drg\_flag = **false**;  
 \_drg.setText(" RAD");  
 } **else** {  
 drg\_flag = **true**;  
 \_drg.setText(" DEG");  
 }  
 //如果输入时退格键，并且是在按=之前  
 } **else if** (command.compareTo("Bksp") == 0 && equals\_flag) {  
 //一次删除3个字符  
 **if** (TTO(str) == 3) {  
 **if** (str.length() > 3)  
 input.setText(str.substring(0, str.length() - 3));  
 **else if** (str.length() == 3) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //依次删除2个字符  
 } **else if** (TTO(str) == 2) {  
 **if** (str.length() > 2)  
 input.setText(str.substring(0, str.length() - 2));  
 **else if** (str.length() == 2) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //依次删除一个字符  
 } **else if** (TTO(str) == 1) {  
 //若之前输入的字符串合法则删除一个字符  
 **if** (right(str)) {  
 **if** (str.length() > 1)  
 input.setText(str.substring(0, str.length() - 1));  
 **else if** (str.length() == 1) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 //若之前输入的字符串不合法则删除全部字符  
 } **else** {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 }  
 **if** (input.getText().toString().compareTo("-") == 0 || equals\_flag == **false**) {  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip.setText("欢迎使用！");  
 }  
 tip\_lock = **true**;  
 **if** (tip\_i > 0)  
 tip\_i--;  
 //如果是在按=之后输入退格键  
 } **else if** (command.compareTo("Bksp") == 0 && equals\_flag == **false**) {  
 //将显示器内容设置为0  
 input.setText("0");  
 vbegin = **true**;  
 tip\_i = 0;  
 tip\_lock = **true**;  
 tip.setText("欢迎使用！");  
 //如果输入的是清除键  
 } **else if** (command.compareTo("C") == 0) {  
 //将显示器内容设置为0  
 input.setText("0");  
 //重新输入标志置为true  
 vbegin = **true**;  
 //缓存命令位数清0  
 tip\_i = 0;  
 //表明可以继续输入  
 tip\_lock = **true**;  
 //表明输入=之前  
 equals\_flag = **true**;  
 tip.setText("欢迎使用！");  
 //如果输入的是”MC“，则将存储器内容清0  
 } **else if** (command.compareTo("MC") == 0) {  
 mem.setText("0");  
 //如果按”exit“则退出程序  
 } **else if** (command.compareTo("EXIT") == 0) {  
 **System**.exit(0);  
 //如果输入的是=号，并且输入合法  
 } **else if** (command.compareTo("=") == 0 && tip\_lock && right(str) && equals\_flag) {  
 tip\_i = 0;  
 //表明不可以继续输入  
 tip\_lock = **false**;  
 //表明输入=之后  
 equals\_flag = **false**;  
 //保存原来算式样子  
 str\_old = str;  
 //替换算式中的运算符，便于计算  
 str = str.replaceAll("sin", "s");  
 str = str.replaceAll("cos", "c");  
 str = str.replaceAll("tan", "t");  
 str = str.replaceAll("log", "g");  
 str = str.replaceAll("ln", "l");  
 str = str.replaceAll("n!", "!");  
 //重新输入标志设置true  
 vbegin = **true**;  
 //将-1x转换成-  
 str\_new = str.replaceAll("-", "-1×");  
 //计算算式结果  
 **new** calc().process(str\_new);  
 }  
 //表明可以继续输入  
 tip\_lock = **true**;  
 }  
 };  
  
 //向input输出字符  
 **private void** print(**String** str) {  
 //清屏后输出  
 **if** (vbegin)  
 input.setText(str);  
 //在屏幕原str后增添字符  
 **else** input.append(str);  
 vbegin = **false**;  
 }  
  
 */\*  
 \* 判断一个str是否是合法的，返回值为true、false  
 \* 只包含0123456789.()sincostanlnlogn!+-×÷√^的是合法的str，返回true  
 \* 包含了除0123456789.()sincostanlnlogn!+-×÷√^以外的字符的str为非法的，返回false  
 \*/* **private boolean** right(**String** str) {  
 **int** i = 0;  
 **for** (i = 0; i < str.length(); i++) {  
 **if** (str.charAt(i) != '0' && str.charAt(i) != '1' && str.charAt(i) != '2' &&  
 str.charAt(i) != '3' && str.charAt(i) != '4' && str.charAt(i) != '5' &&  
 str.charAt(i) != '6' && str.charAt(i) != '7' && str.charAt(i) != '8' &&  
 str.charAt(i) != '9' && str.charAt(i) != '.' && str.charAt(i) != '-' &&  
 str.charAt(i) != '+' && str.charAt(i) != '×' && str.charAt(i) != '÷' &&  
 str.charAt(i) != '√' && str.charAt(i) != '^' && str.charAt(i) != 's' &&  
 str.charAt(i) != 'i' && str.charAt(i) != 'n' && str.charAt(i) != 'c' &&  
 str.charAt(i) != 'o' && str.charAt(i) != 't' && str.charAt(i) != 'a' &&  
 str.charAt(i) != 'l' && str.charAt(i) != 'g' && str.charAt(i) != '(' &&  
 str.charAt(i) != ')' && str.charAt(i) != '!')  
 **break**;  
 }  
 **if** (i == str.length()) {  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 */\*  
 \* 检测函数，返回值为3、2、1 表示应当一次删除几个？ Three+Two+One = TTO  
 \* 为Bksp按钮的删除方式提供依据  
 \* 返回3，表示str尾部为sin、cos、tan、log中的一个，应当一次删除3个  
 \* 返回2，表示str尾部为ln、n!中的一个，应当一次删除2个  
 \* 返回1，表示为除返回3、2外的所有情况，只需删除一个（包含非法字符时要另外考虑：应清屏）  
 \*/* **private int** TTO(**String** str) {  
 **if** ((str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'i' &&  
 str.charAt(str.length() - 3) == 's') ||  
 (str.charAt(str.length() - 1) == 's' &&  
 str.charAt(str.length() - 2) == 'o' &&  
 str.charAt(str.length() - 3) == 'c') ||  
 (str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'a' &&  
 str.charAt(str.length() - 3) == 't') ||  
 (str.charAt(str.length() - 1) == 'g' &&  
 str.charAt(str.length() - 2) == 'o' &&  
 str.charAt(str.length() - 3) == 'l')) {  
 **return** 3;  
 } **else if** ((str.charAt(str.length() - 1) == 'n' &&  
 str.charAt(str.length() - 2) == 'l') ||  
 (str.charAt(str.length() - 1) == '!' &&  
 str.charAt(str.length() - 2) == 'n')) {  
 **return** 2;  
 } **else** {  
 **return** 1;  
 }  
 }  
  
 */\*  
 \* 检测函数，对str进行前后语法检测  
 \* 为Tip的提示方式提供依据，与TipShow()配合使用  
 \* 编号 字符 其后可以跟随的合法字符  
 \* 1 （ 数字|（|-|.|函数  
 \* 2 ） 算符|）|√ ^  
 \* 3 . 数字|算符|）|√ ^  
 \* 4 数字 .|数字|算符|）|√ ^  
 \* 5 算符 数字|（|.|函数  
 \* 6 √ ^ （ |. | 数字  
 \* 7 函数 数字|（|.  
 \*  
 \* 小数点前后均可省略，表示0  
 \* 数字第一位可以为0  
 \*/* **private void** TipChecker(**String** tipcommand1, **String** tipcommand2) {  
 //Tipcode1表示错误类型，Tipcode2表示名词解释类型  
 **int** Tipcode1 = 0, Tipcode2 = 0;  
 //表示命令类型  
 **int** tiptype1 = 0, tiptype2 = 0;  
 //括号数  
 **int** bracket = 0;  
 //“+-x÷√^”不能作为第一位  
 **if** (tipcommand1.compareTo("#") == 0 && (tipcommand2.compareTo("÷") == 0 ||  
 tipcommand2.compareTo("×") == 0 || tipcommand2.compareTo("+") == 0 ||  
 tipcommand2.compareTo(")") == 0 || tipcommand2.compareTo("√") == 0 ||  
 tipcommand2.compareTo("^") == 0)) {  
 Tipcode1 = -1;  
 }  
 //定义存储字符串中最后一位的类型  
 **else if** (tipcommand1.compareTo("#") != 0) {  
 **if** (tipcommand1.compareTo("(") == 0) {  
 tiptype1 = 1;  
 } **else if** (tipcommand1.compareTo(")") == 0) {  
 tiptype1 = 2;  
 } **else if** (tipcommand1.compareTo(".") == 0) {  
 tiptype1 = 3;  
 } **else if** ("0123456789".indexOf(tipcommand1) != -1) {  
 tiptype1 = 4;  
 } **else if** ("+-×÷".indexOf(tipcommand1) != -1) {  
 tiptype1 = 5;  
 } **else if** ("√^".indexOf(tipcommand1) != -1) {  
 tiptype1 = 6;  
 } **else if** ("sincostanloglnn!".indexOf(tipcommand1) != -1) {  
 tiptype1 = 7;  
 }  
 //定义欲输入的按键类型  
 **if** (tipcommand2.compareTo("(") == 0) {  
 tiptype2 = 1;  
 } **else if** (tipcommand2.compareTo(")") == 0) {  
 tiptype2 = 2;  
 } **else if** (tipcommand2.compareTo(".") == 0) {  
 tiptype2 = 3;  
 } **else if** ("0123456789".indexOf(tipcommand2) != -1) {  
 tiptype2 = 4;  
 } **else if** ("+-×÷".indexOf(tipcommand2) != -1) {  
 tiptype2 = 5;  
 } **else if** ("√^".indexOf(tipcommand2) != -1) {  
 tiptype2 = 6;  
 } **else if** ("sincostanloglnn!".indexOf(tipcommand2) != -1) {  
 tiptype2 = 7;  
 }  
  
 **switch** (tiptype1) {  
 **case** 1:  
 //左括号后面直接接右括号,“+x÷”（负号“-”不算）,或者"√^"  
 **if** (tiptype2 == 2 || (tiptype2 == 5 && tipcommand2.compareTo("-") != 0) ||  
 tiptype2 == 6)  
 Tipcode1 = 1;  
 **break**;  
 **case** 2:  
 //右括号后面接左括号，数字，“+-x÷sin^...”  
 **if** (tiptype2 == 1 || tiptype2 == 3 || tiptype2 == 4 || tiptype2 == 7)  
 Tipcode1 = 2;  
 **break**;  
 **case** 3:  
 //“.”后面接左括号或者“sincos...”  
 **if** (tiptype2 == 1 || tiptype2 == 7)  
 Tipcode1 = 3;  
 //连续输入两个“.”  
 **if** (tiptype2 == 3)  
 Tipcode1 = 8;  
 **break**;  
 **case** 4:  
 //数字后面直接接左括号或者“sincos...”  
 **if** (tiptype2 == 1 || tiptype2 == 7)  
 Tipcode1 = 4;  
 **break**;  
 **case** 5:  
 //“+-x÷”后面直接接右括号，“+-x÷√^”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6)  
 Tipcode1 = 5;  
 **break**;  
 **case** 6:  
 //“√^”后面直接接右括号，“+-x÷√^”以及“sincos...”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6 || tiptype2 == 7)  
 Tipcode1 = 6;  
 **break**;  
 **case** 7:  
 //“sincos...”后面直接接右括号“+-x÷√^”以及“sincos...”  
 **if** (tiptype2 == 2 || tiptype2 == 5 || tiptype2 == 6 || tiptype2 == 7)  
 Tipcode1 = 7;  
 **break**;  
 }  
 }  
 //检测小数点的重复性，Tipconde1=0,表明满足前面的规则  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo(".") == 0) {  
 **int** tip\_point = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 //若之前出现一个小数点点，则小数点计数加1  
 **if** (Tipcommand[i].compareTo(".") == 0) {  
 tip\_point++;  
 }  
 //若出现以下几个运算符之一，小数点计数清零  
 **if** (Tipcommand[i].compareTo("sin") == 0 || Tipcommand[i].compareTo("cos") == 0 ||  
 Tipcommand[i].compareTo("tan") == 0 || Tipcommand[i].compareTo("log") == 0 ||  
 Tipcommand[i].compareTo("ln") == 0 || Tipcommand[i].compareTo("n!") == 0 ||  
 Tipcommand[i].compareTo("√") == 0 || Tipcommand[i].compareTo("^") == 0 ||  
 Tipcommand[i].compareTo("÷") == 0 || Tipcommand[i].compareTo("×") == 0 ||  
 Tipcommand[i].compareTo("-") == 0 || Tipcommand[i].compareTo("+") == 0 ||  
 Tipcommand[i].compareTo("(") == 0 || Tipcommand[i].compareTo(")") == 0) {  
 tip\_point = 0;  
 }  
 }  
 tip\_point++;  
 //若小数点计数大于1，表明小数点重复了  
 **if** (tip\_point > 1) {  
 Tipcode1 = 8;  
 }  
 }  
 //检测右括号是否匹配  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo(")") == 0) {  
 **int** tip\_right\_bracket = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 //如果出现一个左括号，则计数加1  
 **if** (Tipcommand[i].compareTo("(") == 0) {  
 tip\_right\_bracket++;  
 }  
 //如果出现一个右括号，则计数减1  
 **if** (Tipcommand[i].compareTo(")") == 0) {  
 tip\_right\_bracket--;  
 }  
 }  
 //如果右括号计数=0,表明没有响应的左括号与当前右括号匹配  
 **if** (tip\_right\_bracket == 0) {  
 Tipcode1 = 10;  
 }  
 }  
 //检查输入=的合法性  
 **if** (Tipcode1 == 0 && tipcommand2.compareTo("=") == 0) {  
 //括号匹配数  
 **int** tip\_bracket = 0;  
 **for** (**int** i = 0; i < tip\_i; i++) {  
 **if** (Tipcommand[i].compareTo("(") == 0) {  
 tip\_bracket++;  
 }  
 **if** (Tipcommand[i].compareTo(")") == 0) {  
 tip\_bracket--;  
 }  
 }  
 //若大于0，表明左括号还有未匹配的  
 **if** (tip\_bracket > 0) {  
 Tipcode1 = 9;  
 bracket = tip\_bracket;  
 } **else if** (tip\_bracket == 0) {  
 //若前一个字符是以下之一，表明=号不合法  
 **if** ("√^sincostanloglnn!".indexOf(tipcommand1) != -1) {  
 Tipcode1 = 6;  
 }  
 //若前一个字符是以下之一，表明=号不合法  
 **if** ("+-×÷".indexOf(tipcommand1) != -1) {  
 Tipcode1 = 5;  
 }  
 }  
 }  
 //若命令式以下之一，则显示相应的帮助信息  
 **if** (tipcommand2.compareTo("MC") == 0) Tipcode2 = 1;  
 **if** (tipcommand2.compareTo("C") == 0) Tipcode2 = 2;  
 **if** (tipcommand2.compareTo("DRG") == 0) Tipcode2 = 3;  
 **if** (tipcommand2.compareTo("Bksp") == 0) Tipcode2 = 4;  
 **if** (tipcommand2.compareTo("sin") == 0) Tipcode2 = 5;  
 **if** (tipcommand2.compareTo("cos") == 0) Tipcode2 = 6;  
 **if** (tipcommand2.compareTo("tan") == 0) Tipcode2 = 7;  
 **if** (tipcommand2.compareTo("log") == 0) Tipcode2 = 8;  
 **if** (tipcommand2.compareTo("ln") == 0) Tipcode2 = 9;  
 **if** (tipcommand2.compareTo("n!") == 0) Tipcode2 = 10;  
 **if** (tipcommand2.compareTo("√") == 0) Tipcode2 = 11;  
 **if** (tipcommand2.compareTo("^") == 0) Tipcode2 = 12;  
 //显示帮助和错误信息  
 TipShow(bracket, Tipcode1, Tipcode2, tipcommand1, tipcommand2);  
 }  
  
 */\*  
 \* 反馈Tip信息，加强人机交互，与TipChecker()配合使用  
 \*/* **private void** TipShow(**int** bracket, **int** tipcode1, **int** tipcode2,  
 **String** tipcommand1, **String** tipcommand2) {  
 **String** tipmessage = "";  
 **if** (tipcode1 != 0)  
 tip\_lock = **false**;//表明输入有误  
 **switch** (tipcode1) {  
 **case** -1:  
 tipmessage = tipcommand2 + " 不能作为第一个算符\n";  
 **break**;  
 **case** 1:  
 tipmessage = tipcommand1 + " 后应输入：数字/(/./-/函数 \n";  
 **break**;  
 **case** 2:  
 tipmessage = tipcommand1 + " 后应输入：)/算符 \n";  
 **break**;  
 **case** 3:  
 tipmessage = tipcommand1 + " 后应输入：)/数字/算符 \n";  
 **break**;  
 **case** 4:  
 tipmessage = tipcommand1 + " 后应输入：)/./数字 /算符 \n";  
 **break**;  
 **case** 5:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字/函数 \n";  
 **break**;  
 **case** 6:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字 \n";  
 **break**;  
 **case** 7:  
 tipmessage = tipcommand1 + " 后应输入：(/./数字 \n";  
 **break**;  
 **case** 8:  
 tipmessage = "小数点重复\n";  
 **break**;  
 **case** 9:  
 tipmessage = "不能计算，缺少 " + bracket + " 个 )";  
 **break**;  
 **case** 10:  
 tipmessage = "不需要 )";  
 **break**;  
 }  
 **switch** (tipcode2) {  
 **case** 1:  
 tipmessage = tipmessage + "[MC 用法: 清除记忆 MEM]";  
 **break**;  
 **case** 2:  
 tipmessage = tipmessage + "[C 用法: 归零]";  
 **break**;  
 **case** 3:  
 tipmessage = tipmessage + "[DRG 用法: 选择 DEG 或 RAD]";  
 **break**;  
 **case** 4:  
 tipmessage = tipmessage + "[Bksp 用法: 退格]";  
 **break**;  
 **case** 5:  
 tipmessage = tipmessage + "sin 函数用法示例：\n" +  
 "DEG：sin30 = 0.5 RAD：sin1 = 0.84\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "sin(cos45)，而不是sincos45";  
 **break**;  
 **case** 6:  
 tipmessage = tipmessage + "cos 函数用法示例：\n" +  
 "DEG：cos60 = 0.5 RAD：cos1 = 0.54\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "cos(sin45)，而不是cossin45";  
 **break**;  
 **case** 7:  
 tipmessage = tipmessage + "tan 函数用法示例：\n" +  
 "DEG：tan45 = 1 RAD：tan1 = 1.55\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "tan(cos45)，而不是tancos45";  
 **break**;  
 **case** 8:  
 tipmessage = tipmessage + "log 函数用法示例：\n" +  
 "log10 = log(5+5) = 1\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "log(tan45)，而不是logtan45";  
 **break**;  
 **case** 9:  
 tipmessage = tipmessage + "ln 函数用法示例：\n" +  
 "ln10 = le(5+5) = 2.3 lne = 1\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "ln(tan45)，而不是lntan45";  
 **break**;  
 **case** 10:  
 tipmessage = tipmessage + "n! 函数用法示例：\n" +  
 "n!3 = n!(1+2) = 3×2×1 = 6\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "n!(log1000)，而不是n!log1000";  
 **break**;  
 **case** 11:  
 tipmessage = tipmessage + "√ 用法示例：开任意次根号\n" +  
 "如：27开3次根为 27√3 = 3\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "(函数)√(函数) ， (n!3)√(log100) = 2.45";  
 **break**;  
 **case** 12:  
 tipmessage = tipmessage + "^ 用法示例：开任意次平方\n" +  
 "如：2的3次方为 2^3 = 8\n" +  
 "注：与其他函数一起使用时要加括号，如：\n" +  
 "(函数)√(函数) ， (n!3)^(log100) = 36";  
 **break**;  
 }  
 //将提示信息显示到tip  
 tip.setText(tipmessage);  
 }  
  
 */\*  
 \* 整个计算核心，只要将表达式的整个字符串传入calc().process()就可以实行计算了  
 \* 算法包括以下几部分：  
 \* 1、计算部分 process(String str) 当然，这是建立在查错无错误的情况下  
 \* 2、数据格式化 FP(double n) 使数据有相当的精确度  
 \* 3、阶乘算法 N(double n) 计算n!，将结果返回  
 \* 4、错误提示 showError(int code ,String str) 将错误返回  
 \*/* **public class calc** {  
 **public** calc() {  
  
 }  
  
 **final int** MAXLEN = 500;  
  
 */\*  
 \* 计算表达式  
 \* 从左向右扫描，数字入number栈，运算符入operator栈  
 \* +-基本优先级为1，×÷基本优先级为2，log ln sin cos tan n!基本优先级为3，√^基本优先级为4  
 \* 括号内层运算符比外层同级运算符优先级高4  
 \* 当前运算符优先级高于栈顶压栈，低于栈顶弹出一个运算符与两个数进行运算  
 \* 重复直到当前运算符大于栈顶  
 \* 扫描完后对剩下的运算符与数字依次计算  
 \*/* **public void** process(**String** str) {  
 **int** weightPlus = 0, topOP = 0,  
 topNum = 0, flag = 1, weightTemp = 0;  
 //weightPlus为同一（）下的基本优先级，weightTemp临时记录优先级的变化  
 //topOp为weight[]，operator[]的计数器；topNum为number[]的计数器  
 //flag为正负数的计数器，1为正数，-1为负数  
  
  
 **int** weight[];//保存operator栈中运算符的优先级,以topOp计数  
 **double** number[];//保存数字，以topNum计数  
  
  
 **char** ch, ch\_gai = 0, operator[];//operator[]保存运算符，以topOP计数  
 **String** num = **null**;//记录数字，str以+-×÷()sctgl!√^分段，+-×÷()sctgl!√^字符之间的字符串即为数字  
 weight = **new int**[MAXLEN];  
 number = **new double**[MAXLEN];  
 operator = **new char**[MAXLEN];  
 **String** expression = str;  
 **StringTokenizer** expToken = **new** StringTokenizer(expression, "+-×÷()sctgl!√^");  
 **int** i = 0;  
 **while** (i < expression.length()) {  
 ch = expression.charAt(i);  
 //判断正负数  
 **if** (i == 0) {  
 **if** (ch == '-') {  
 flag = -1;  
 }  
 } **else if** (expression.charAt(i - 1) == '(' && ch == '-') {  
 flag = -1;  
 }  
  
 //取得数字，并将正负号转移给数字  
 **if** (ch <= '9' && ch >= '0' || ch == '.' || ch == 'E') {  
 num = expToken.nextToken();//取出ch对应段的那个数字  
 ch\_gai = ch;  
 //将i向后移直至ch\_gai取得的字符不是数字  
 **while** (i < expression.length() && (ch\_gai <= '9' && ch\_gai >= '0' || ch\_gai == '.' || ch\_gai == 'E')) {  
 ch\_gai = expression.charAt(i++);  
 }  
 //将ch\_gai多移的位数移回来  
 **if** (i >= expression.length()) {  
 //说明已经取完了最后一个数字  
 i -= 1;  
 } **else** {  
 i -= 2;  
 }  
 //如果num只取到了一个 “.” 则将其当做0处理  
 **if** (num.compareTo(".") == 0) {  
 number[topNum++] = 0;  
 } **else** {  
 //将正负号转移到数值上  
 number[topNum++] = **Double**.parseDouble(num) \* flag;  
 flag = 1;  
 }  
 }  
 //计算运算符的优先级  
 //先检查有没有括号，如果有括号，就应该记录下来，然后在原来的基础上加4个优先级  
 **if** (ch == '(' || ch == ')') {  
 weightPlus += 4;  
 }  
 **if** (ch == '-' && flag == 1 || ch == '+' || ch == '×' || ch == '÷' ||  
 ch == 's' || ch == 'c' || ch == 't' || ch == 'g' || ch == 'l' ||  
 ch == '!' || ch == '√' || ch == '^') {  
 **switch** (ch) {  
 //+-的优先级最低，为1  
 **case** '+':  
 **case** '-':  
 weightTemp = 1 + weightPlus;  
 **break**;  
 //'×''÷'的优先级稍高，为2  
 **case** '÷':  
 **case** '×':  
 weightTemp = 2 + weightPlus;  
 **break**;  
 //sincos之类优先级为3：  
 **case** 's':  
 **case** 'c':  
 **case** 't':  
 **case** 'g':  
 **case** 'l':  
 **case** '!':  
 weightTemp = 3 + weightPlus;  
 **break**;  
 //其余优先级为4  
 //case '^':  
 //case '√':  
 **default**:  
 weightTemp = 4 + weightPlus;  
 **break**;  
 }  
 //如果当前优先级大于堆栈顶部元素，直接入栈  
 **if** (topOP == 0 || weight[topOP - 1] < weightTemp) {  
 weight[topOP] = weightTemp;  
 operator[topOP] = ch;  
 topOP++;  
 } **else** {  
 //否则将堆栈中运算符逐个取出，直到当前堆栈顶部运算符优先级小于当前优先级  
 **while** (topOP > 0 && weight[topOP - 1] >= weightTemp) {  
 **switch** (operator[topOP - 1]) {  
 //取出数字数组的相应元素进行计算  
 **case** '+':  
 number[topNum - 2] += number[topNum - 1];  
 **break**;  
 **case** '-':  
 number[topNum - 2] -= number[topNum - 1];  
 **break**;  
 **case** '×':  
 number[topNum - 2] \*= number[topNum - 1];  
 **break**;  
 **case** '÷':  
 //判断除数为0的情况  
  
 **if** (number[topNum - 1] == 0) {  
 showError(1, str\_old);  
 **return**;  
 }  
 number[topNum - 2] /= number[topNum - 1];  
 **break**;  
 **case** '√':  
 **if** (number[topNum - 1] == 0 || (number[topNum - 2] < 0 && number[topNum - 1] % 2 == 0)) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 2] = **Math**.pow(number[topNum - 2], 1 / number[topNum - 1]);  
 **break**;  
 **case** '^':  
 number[topNum - 2] = **Math**.pow(number[topNum - 2], number[topNum - 1]);  
 **break**;  
 //进行角度弧度判断的计算及转换  
 //sin  
 //sin  
 **case** 's':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.sin((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.sin(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 'c':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.cos((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.cos(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 't':  
 **if** (drg\_flag == **true**) {  
 **if** ((**Math**.abs(number[topNum - 1]) / 90) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 **if** ((**Math**.abs(number[topNum - 1]) / (pi / 2)) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 **case** 'g':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log10(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //ln  
 **case** 'l':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //阶乘  
 **case** '!':  
 **if** (number[topNum - 1] > 170) {  
 showError(3, str\_old);  
 **return**;  
 } **else if** (number[topNum - 1] < 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = N(number[topNum - 1]);  
 topNum++;  
 **break**;  
 }  
 //继续取堆栈中的下一个元素进行判断  
 topNum--;  
 topOP--;  
 }  
 //将运算符入堆栈  
 weight[topOP] = weightTemp;  
 operator[topOP] = ch;  
 topOP++;  
 }  
 }  
 i++;  
 }  
 //依次取出堆栈中的运算符进行运算  
 **while** (topOP > 0) {  
//+-x直接将数组的后两位数取出运算  
 **switch** (operator[topOP - 1]) {  
 **case** '+':  
 number[topNum - 2] += number[topNum - 1];  
 **break**;  
 **case** '-':  
 number[topNum - 2] -= number[topNum - 1];  
 **break**;  
 **case** '×':  
 number[topNum - 2] \*= number[topNum - 1];  
 **break**;  
 //涉及到除法时要考虑除数不能为零的情况  
 **case** '÷':  
 **if** (number[topNum - 1] == 0) {  
 showError(1, str\_old);  
 **return**;  
 }  
 number[topNum - 2] /= number[topNum - 1];  
 **break**;  
 **case** '√':  
 **if** (number[topNum - 1] == 0 || (number[topNum - 2] < 0 &&  
 number[topNum - 1] % 2 == 0)) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 2] =  
 **Math**.pow(number[topNum - 2], 1 / number[topNum - 1]);  
 **break**;  
 **case** '^':  
 number[topNum - 2] =  
 **Math**.pow(number[topNum - 2], number[topNum - 1]);  
 **break**;  
 //sin  
 **case** 's':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.sin((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.sin(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //cos  
 **case** 'c':  
 **if** (drg\_flag == **true**) {  
 number[topNum - 1] = **Math**.cos((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 number[topNum - 1] = **Math**.cos(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //tan  
 **case** 't':  
 **if** (drg\_flag == **true**) {  
 **if** ((**Math**.abs(number[topNum - 1]) / 90) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan((number[topNum - 1] / 180) \* pi);  
 } **else** {  
 **if** ((**Math**.abs(number[topNum - 1]) / (pi / 2)) % 2 == 1) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.tan(number[topNum - 1]);  
 }  
 topNum++;  
 **break**;  
 //对数log  
 **case** 'g':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log10(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //自然对数ln  
 **case** 'l':  
 **if** (number[topNum - 1] <= 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = **Math**.log(number[topNum - 1]);  
 topNum++;  
 **break**;  
 //阶乘  
 **case** '!':  
 **if** (number[topNum - 1] > 170) {  
 showError(3, str\_old);  
 **return**;  
 } **else if** (number[topNum - 1] < 0) {  
 showError(2, str\_old);  
 **return**;  
 }  
 number[topNum - 1] = N(number[topNum - 1]);  
 topNum++;  
 **break**;  
 }  
 //取堆栈下一个元素计算  
 topNum--;  
 topOP--;  
 }  
 //如果数字太大，就提示错误信息  
 **if** (number[0] > 7.3E306) {  
 showError(3, str\_old);  
 **return**;  
 }  
 //输出最终结果  
 input.setText(**String**.valueOf(FP(number[0])));  
 tip.setText("计算完毕，要继续请按归零见C");  
 mem.setText(str\_old + "=" + **String**.valueOf(FP(number[0])));  
 }  
  
 */\*  
 \* FP = floating point 控制小数位数，达到精度  
 \* 否则会出现 0.6-0.2=0.39999999999999997的情况，用FP即可解决，使得数为0.4  
 \* 本格式精度为15位  
 \*/* **public double** FP(**double** n) {  
 //NumberFormat format=NumberFormat.getInstance(); //创建一个格式化类f  
 //format.setMaximumFractionDigits(18); //设置小数位的格式  
 **DecimalFormat** format = **new** DecimalFormat("0.#############");  
 **return Double**.parseDouble(format.format(n));  
 }  
  
 */\*  
 \* 阶乘算法  
 \*/* **public double** N(**double** n) {  
 **int** i = 0;  
 **double** sum = 1;  
 //依次将小于等于n的值相乘  
 **for** (i = 1; i <= n; i++) {  
 sum = sum \* i;  
 }  
 **return** sum;  
 }  
  
 */\*  
 \* 错误提示，按了"="之后，若计算式在process()过程中，出现错误，则进行提示  
 \*/* **public void** showError(**int** code, **String** str) {  
 **String** message = "";  
 **switch** (code) {  
 **case** 1:  
 message = "零不能作除数";  
 **break**;  
 **case** 2:  
 message = "函数格式错误";  
 **break**;  
 **case** 3:  
 message = "值太大了，超出范围";  
 }  
 input.setText("\"" + str + "\"" + ": " + message);  
 tip.setText(message + "\n" + "计算完毕，要继续请按归零键 C");  
 }  
 }  
}

*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <!--结果显示框-->  
 <**EditText** android:id="@+id/input"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:cursorVisible="false"  
 android:focusable="false"  
 android:gravity="right"  
 android:text="0"  
 android:textColor="#FF1493"  
 android:textSize="20sp" />  
  
 <!-- 采用TableRow的格式进行布局设计-->  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <!--用于显示存储结果-->  
 <**TextView** android:id="@+id/M"  
 android:layout\_width="53sp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/mem"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--默认显示字符串“0”-->  
 <**TextView** android:id="@+id/mem"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="0"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
  
 </**TableRow**>  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
 <!--设置当前角度是角度还是弧度，默认是角度-->  
 <**TextView** android:id="@+id/\_drg"  
 android:layout\_width="53sp"  
 android:layout\_height="wrap\_content"  
 android:text=" DEG"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--清除存储结果-->  
 <**Button** android:id="@+id/mc"  
 android:layout\_width="106sp"  
 android:layout\_height="wrap\_content"  
 android:width="106sp"  
 android:background="#00000000"  
 android:text="MC"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
  
 <!--清除窗口的所有内容-->  
 <**Button** android:id="@+id/c"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#00000000"  
 android:text="C"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 </**TableRow**>  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <!--在角度和弧度制之间切换-->  
 <**Button** android:id="@+id/drg"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="DRG"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
 <!--计算正弦-->  
 <**Button** android:id="@+id/sin"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="sin"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--计算余弦-->  
 <**Button** android:id="@+id/cos"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="cos"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--计算正切-->  
 <**Button** android:id="@+id/tan"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="tan"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--计算阶乘-->  
 <**Button** android:id="@+id/factorial"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="n!"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <!--退格键-->  
 <**Button** android:id="@+id/bksp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="Bksp"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
 </**TableRow**>  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**Button** android:id="@+id/seven"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="7"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/eight"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="8"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/nine"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="9"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/divide"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="÷"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/left"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="("  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/right"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text=")"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
 </**TableRow**>  
  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**Button** android:id="@+id/four"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="4"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/five"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="5"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/six"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="6"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/mul"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="×"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/sqrt"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="√"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/square"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="^"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
 </**TableRow**>  
  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**Button** android:id="@+id/one"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="1"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/two"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="2"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/three"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="3"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/sub"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="-"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/log"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="log"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/ln"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="ln"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
 </**TableRow**>  
  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**Button** android:id="@+id/zero"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="0"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/dot"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="."  
 android:textAllCaps="false" />  
  
 <**Button** android:id="@+id/equal"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="="  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 <**Button** android:id="@+id/add"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:background="#00000000"  
 android:text="+"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="40dp" />  
  
 <**Button** android:id="@+id/exit"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.8"  
 android:background="#00000000"  
 android:text="EXIT"  
 android:textAllCaps="false"  
 android:textColor="#4EEE94"  
 android:textSize="20dp" />  
  
 </**TableRow**>  
  
 <**TableRow** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**TextView** android:id="@+id/T"  
 android:layout\_width="45sp"  
 android:layout\_height="wrap\_content"  
 android:background="#00000000"  
 android:text="@string/tips"  
 android:textColor="#FF1493"  
 android:textSize="20sp" />  
  
 <**TextView** android:id="@+id/tip"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#00000000"  
 android:text="@string/welcomeuse"  
 android:textColor="#FF1493"  
 android:textSize="20sp" />  
 </**TableRow**>  
</**LinearLayout**>

# 八、课程设计总结

8.1 代码行数

本课程设计总代码大约6000行

8.2实验感想

数据结构作为计算机院最重要的一门课程，其算法思想对编程能力的提高具有很大帮助，由于能力和时间有限，我只做了六个必做题和一个选做题，七个课设六千余行代码，感受最深的是代码能力有了很大的提高。

这次课程设计，加深了我对课本理论知识的理解，使我了解了软件开发的一些流程及注意事项，提高了我对C语言和C++的运用能力，同时也培养了我的思维。

很悲惨的是在本次课程设计验收的前两天我患上了很严重的感冒，严重到卧床不起，当时我睡在床上抱着我的笔记本电脑告诉自己不能倒下，不然课程设计完不成就完蛋了，正是靠着这种压力我坚持着高烧在床上写代码，最终代码写完了感冒也更严重了，导致最近一直在打针，虽然后果比较严重，但是没有什么比按时完成课程设计更加高兴了、更加有成就感的。

通过本次课程设计，我的C语言能力得到了很大的提高，对C语言的特性更加了解，相信这会对以后的学习工作带来巨大的好处。此外，为了实现图形界面，我在课程设计中使用了自学的java，这让我可以系统地使用这门自学的编程语言，使我可以将其和C语言对比起来，更加了解它的特性和使用方法，有利于我以后更好地使用它。

这次课程设计的时间虽然不长，但依然意义重大。每一个题目的一个功能，在于运用学习成果，检验学习成果，看一看课堂学习与实际工作的距离， 并通过综合分析找出学习中的不足，从而完善我们的学习与实践。

在此次课程设计过程中，我在收获知识的同时，也收获了一定的阅历。因为我在设计过程中查阅书本、查找资料，每当达到了题目中的一个要求，实现了一个功能，心中都会有一丝隐隐的自豪感，后面的学习中，还会继续努力，将所学知识更好地运用到实际的代码中去、让我的代码更加高效、简洁。