**数据结构课程设计**

姓名： 朱亚非

学号： 161640230

班级： 1616403

指导老师： 孙涵

**目 录**

**[一、 购物网站信息管理 4](#_Toc16524)**

**[1 题目简介 4](#_Toc24720)**

**[2数据结构 4](#_Toc28370)**

**[3 算设法计思想 5](#_Toc9562)**

**[4测试数据和结果 6](#_Toc31152)**

**[二、 迷宫问题 7](#_Toc24745)**

**[1 题目简介 7](#_Toc24720)**

**[2数据结构 8](#_Toc28370)**

**[3 算设法计思想 8](#_Toc9562)**

**[4测试数据和结果 9](#_Toc31152)**

**[三、 二叉树的应用 1](#_Toc2816)1**

**[1 题目简介 1](#_Toc24720)1**

**[2数据结构 1](#_Toc28370)1**

**[3 算设法计思想 1](#_Toc9562)1**

**[4测试数据和结果 1](#_Toc31152)3**

**[四、 Huffman编码与解码 1](#_Toc24057)3**

**[1 题目简介 1](#_Toc24720)3**

**[2数据结构 1](#_Toc28370)3**

**[3 算设法计思想 1](#_Toc9562)4**

**[4测试数据和结果 1](#_Toc31152)5**

**[五、无线传感器网络 1](#_Toc8639)6**

**[1 题目简介 1](#_Toc24720)6**

**[2数据结构 1](#_Toc28370)7**

**[3 算设法计思想 1](#_Toc9562)7**

**[4测试数据和结果 1](#_Toc31152)8**

**[六、排序算法比较 1](#_Toc1727)9**

**[1 题目简介 1](#_Toc24720)9**

**[2数据结构 1](#_Toc28370)9**

**[3 算设法计思想 1](#_Toc9562)9**

**[4测试数据和结果 2](#_Toc31152)0**

**[5算法时间复杂度 2](#_Toc7219)0**

**[七、哈希表应用 2](#_Toc29741)1**

**[1 题目简介 2](#_Toc24720)1**

**[2数据结构 2](#_Toc28370)1**

**[3 算设法计思想 2](#_Toc9562)2**

**[4测试数据和结果 2](#_Toc31152)2**

**[八、南京公交路线 2](#_Toc26239)4**

**[1 题目简介 24](#_Toc24720)**

**[2数据结构 24](#_Toc28370)**

**[3 算设法计思想 26](#_Toc9562)**

**[4测试数据和结果 2](#_Toc31152)6**

**[九、 算术表达式求值 2](#_Toc26239)7**

**[1 题目简介 2](#_Toc24720)7**

**[2数据结构 2](#_Toc28370)8**

**[3 算设法计思想 2](#_Toc9562)8**

**[4测试数据和结果 2](#_Toc31152)8**

**十、 [课程设计总结](#_Toc24048) 30**

**[1 完成情况（代码行数）](#_Toc3603) 30**

**[2 心得体会 3](#_Toc6584)1**

**一．购物网站信息管理（必做）（链表）**

1.[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. 采用的数据结构

typedef struct Goods{

string name;

float price;

int sales;

Goods\* nextG; //下一个邻接的商品

void setvalue(string n, float p, int s)

{

name = n;

price = p;

sales = s;

}

bool operator > (Goods g)//为了插入排序

{

if(sales>g.sales) return true;

else return false;

}

}Goods,\*pGoods;

class Store{

public:

Store();

Store(int ,string,int,int);

void set(int ,string,int,int);

void addGoods(pGoods);

pGoods getfirstGoods();

int getNumber();

int getCredit();

string getname();

int getgoodsnum();

void setnumber(int i);

pGoods findGoods(string);

int deGoods(string );

void modifygoodsprice(pGoods,float);//修改商品价格

void addSales(pGoods);//增加销量

void showGoods();

private:

int number;

string name;

int credit;

int goodsnum;

pGoods first\_goods;

};

店铺信息链表

typedef struct SNode{

Store store;

SNode\* nextstore;

}SNode,\*Sptr;

当前查找商品的双链表：

typedef struct BGNode{//双向链表，存储当前查找的商品

Goods goods;

Sptr s;//存储商铺地址 ，方便购买时直接对店铺信息进行操作

BGNode\* pre;

BGNode\* next;

}BGNode,\*BGN;

1. 算法思想

文件读写：

每个店铺都有一个商品数量变量。读文件时获取，并将接下来指定数量的商品信息存入当前店铺中，后续店铺存储方式同。写入文件思想基本一致。

增加店铺：

新建一个店铺结点，并初始化。并写入文件

删除店铺：

通过pre指针查找到该店铺的前一个结点，并通过三步操作删除释放，并根据当前节点编号依次更新后续店铺编号。并写入文件

增加商品：

通过指针查找到对应店铺，新建商品结点，初始化并插入到该店铺的第一个商品结点位置。并写入文件

删除和修改商品：

找到对应店铺的该商品结点，对其进行修改和删除即可。并写入文件

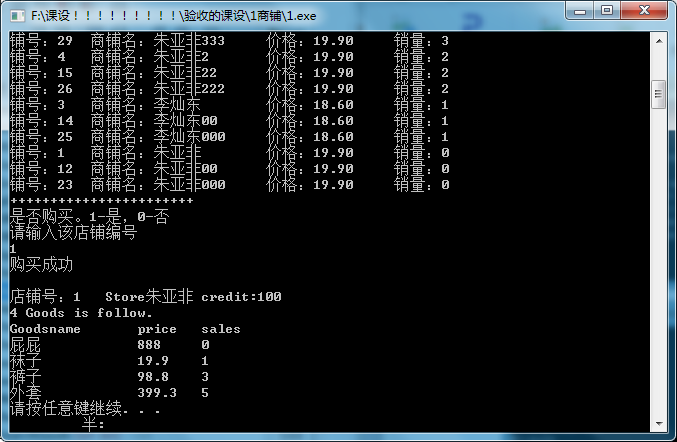
购买商品：

通过商品查找依次遍历所有店铺，若存在该商品则新建结点并初始化，同时记录该商品的店铺地址（方便购买时直接对店铺链表进行操作），将该节点插入到双链表的尾端。双链表建立结束后，通过插入排序对其每个节点按销量进行排序，排序完成后输出到屏幕。若需购买，通过对应节点中的店铺指针直接修改对应店铺中的该商品信息，购买完成。

1. 运行结果







**二、迷宫问题（必做）（栈与队列），(选做)（深度搜索，广度搜索）**

1.[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.采用的数据结构

int graph[ROW\_MAX][ROW\_MAX];//存储迷宫的每个单元的属性

class Cube{

public:

void move();//移动到方向所指定的方格

int get\_x(){return x; }

int get\_y(){return y; }

int get\_dir(){return nextDirection; }

bool next();//查找下一个可以移动的方向，若无（movetime已达三次），则返回false

void go()//转向下一个方向

{

nextDirection=(++nextDirection)%4;

movetime++;

}

void eraser();//擦出当前方格的属性，使其变为空白

void showGraph();//打印迷宫

void print();//设置当前方格的新属性，5678分别代表 ↑→↓←

void end();//判断是否结束

private:

int x,y;

int nextDirection; //存储下一个的方向

int movetime; //记录移动次数

};

stack<Cube> route;//栈中存储路线中每个位置的cube（位置，方向）。

3.算法思想：

迷宫文件内容读入到数组graph中。

建立一个cube初始化为起点坐标，通过next函数判断周围有无通路，若有通路则将该cube入栈route，更新graph，并让cube的x，y更新为下一个位置。如果无通路，则route栈pop一次，并让temp等于该cube，并让其转向一次，因为之前的方向已经退回，所以当前方向失效。继续寻找下一个位置，并不断移动。直到栈空或到达终点。

核心函数如下：

void f()//主函数，不断寻找下一个移动目标

{

route.push(Cube(start\_x,start\_y));//将初始位置存入

int kk=0;

while(!route.empty())//若四处碰壁不断pop最后空栈则游戏结束

{

Cube temp = route.top();//通路的火车头

route.pop();

if(kk>=1) //开始的位置因为还没判断所以无需转向 ，以后每次从队 //列里pop出的位置都要变向一次，

//因为之前的方向已经退回，所以当前方向失效。

temp.go();

kk++;

while(temp.next())//若可以移动到下一个位置

{

route.push(temp);

temp.move();//移动

temp.print();//修改该方格属性

temp.showGraph();//展示

// Sleep(200);

getch();

system("cls");

temp.end();//结束判断

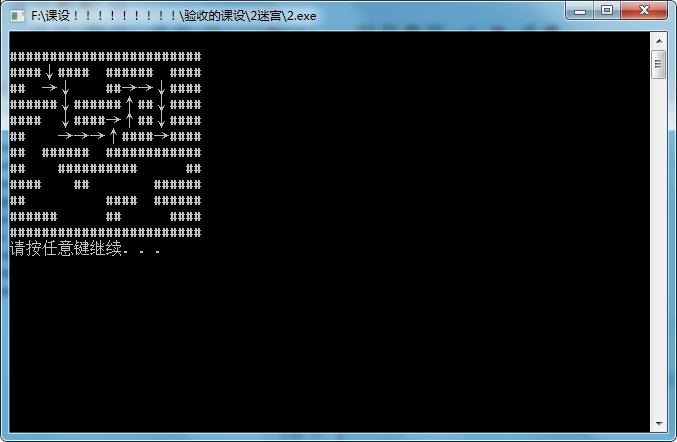
}

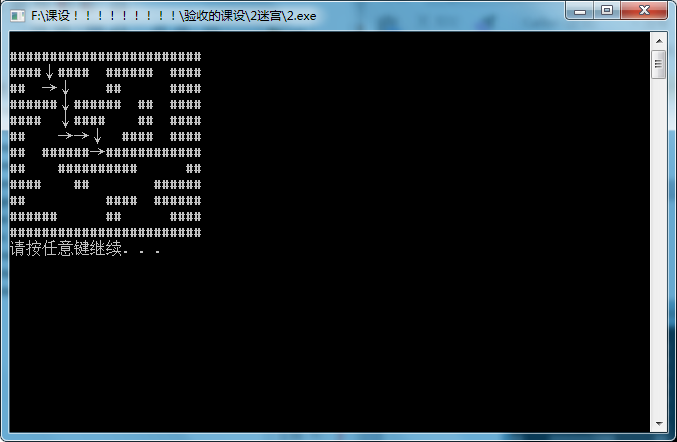
temp.eraser(); //擦除当前无下一个通路的方格

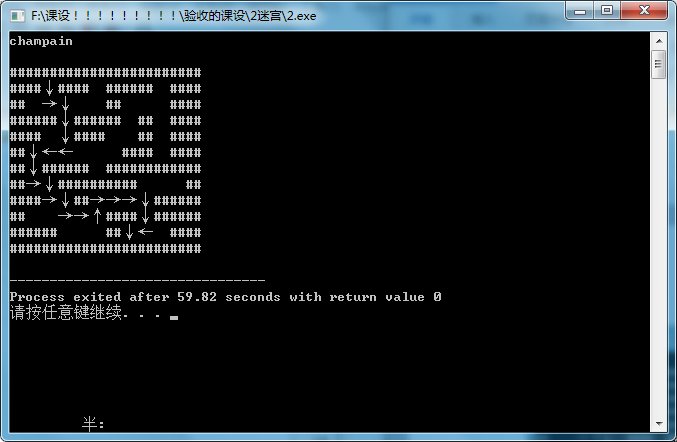
}

}

1. 运行结果







**三、二叉树的应用 (必做)（二叉树）**

1.[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，

1. 采用的数据结构

typedef struct BiTNode

{

TElemType data;

struct BiTNode \*lchild,\*rchild;

}BiTNode,\*BiTree;

1. 算法思想

深度：

若当前结点为空则返回0，不为空则递归求孩子结点的深度 ，返回较大的那个加一为当前结点的深度。

结点数：

访问函数运行了几次则结点数为几。

宽度以及层次遍历：

利用双队列层次遍历的思想实现。首先将根节点入a队，求其队列的长度，循环出队并将孩子入b队，求其长度并留最大值。B队列循环出队并将孩子结点存入a队，循环次过程，直到ab队均空。

排序二叉树判断：

通过中序遍历实现，将中序遍历的visit函数稍加修改，每次访问时比较与前一个访问的结点的大小关系，若大继续访问，若小，则跳出，并得出该二叉树不是排序二叉树。

非递归遍历思想：

1,先序遍历

每访问一个节点都将其右子树入栈，不管右子树是否为空。然后进入左子树，然后继续访问与入栈。当左子树不存在的时候，pop（），即返回到同一父母结点的右子树上，继续上述循环，从而完成先序遍历。

2，中序遍历

大致思想与先序类似。即每到达一个非空结点则将其入栈进入其左子树，不断循环，当左子树为空时出栈，即回到父母结点，然后访问，进入其右子树，继续循环，从而完成中序遍历。

3，后续遍历

后续遍历相对前两个多了个last指针，用来标记上一个被访问的树。

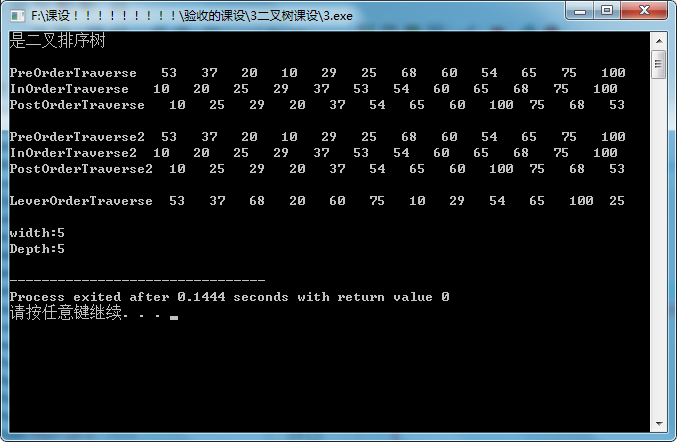
（一）．首先，若当前结点非空，则循环入栈进入左子树，即将其左下方 在一条直线上的左子树全部入栈，当当前树为空时出栈。

出栈就是退回到其父母结点：

（二）．（1）若其右子树非空，则用last判断是否刚刚被访问过，若未 访问则进入该右子树，继续循环存储左下方在一条直线上的左子 树，回到上一步。

（2）若右子树为空，或者last==其右子树，则访问当前结点。访问 之后要退回上一结点，即出栈，回到（二）。所以要让p=NULL， 无需将该结点继续作为左子树入栈。然后继续进行右子树是否被 访问过的判断。

1. 运行结果



**四、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。

2.采用的数据结构

typedef struct Letter//存储字符的值，出现的数量，以及编码

{

char value;

int num;

char \*code;

bool operator <(Letter l2)

{

if(num>l2.num) return true;

else return false;

}

}Letter;

typedef struct{

unsigned int weight;

unsigned int parent,lchild,rchild;

}HTNode,\*HuffmanTree;

typedef char \*\*HuffmanCode;

//int letter[256];//记录有效的，即存在的，个数非零的字符的位置

int letterNum;

int asc[256];//a-z,

int locate[256];//字符c的位置存在了letterlist【locate【c】】。该数组用来记录字符存在编码表的位置

Letter letterlist[256];//编码表

queue<char>compressCode;//存储压缩完成后的所有压缩字符

queue<char>rcode;//

3.算法思想

生成编码表：

首先通过asc[256]数组直接统计所有字符出现的次数，然后筛选掉那些次数为0的，其余的那些直接按次序存到编码表中，然后通过哈夫曼算法进行编码，将编码存到对应的编码表中。

压缩文章：

读取一篇文章，将文章中的字符替换成该字符的编码存到字符数组linecode中，当读取一定个数时进行压缩，压缩规则源代码如下：

unsigned char word = 0;

for(i=0;i<strlen(linecode);)

{

if(linecode[i]=='1') k=1;

else k=0;

word<<1;

word+=k;

i++;

if(i%8==0)

{

compressCode.push(word);

word=0;

}

}

通过定义一个unsigned 类型的字符变量，每来一个01符进行一次左移操作，并加上对应数值，当进行8次的时候将该压缩码入队。到文章结尾的时候可能凑不够八个01符，那么，将单独压缩，用两个变量，一个记录剩余几个01符，一个存储这些不足八个的01符，不足的用0代替。最终压缩完成整篇文章。

将压缩文件解码：

核心代码如下：

for(int i=0;i<8;i++,ch/=2)

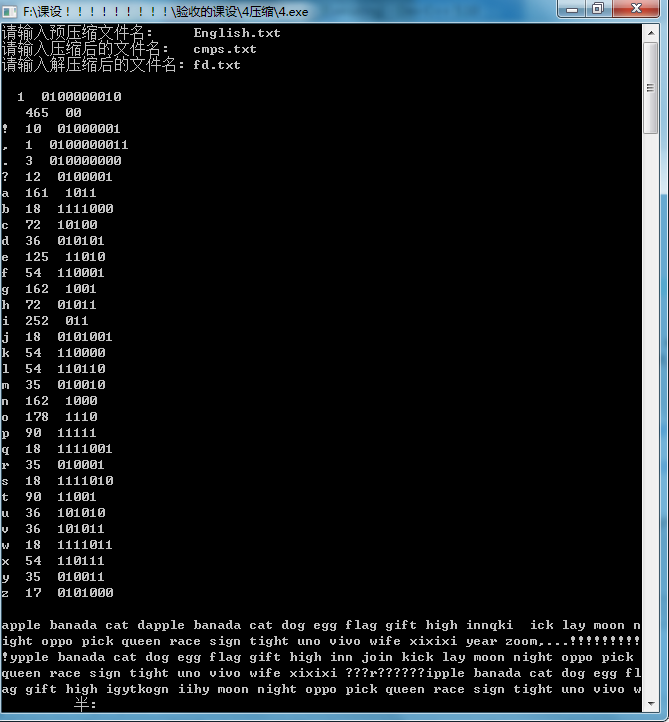
{

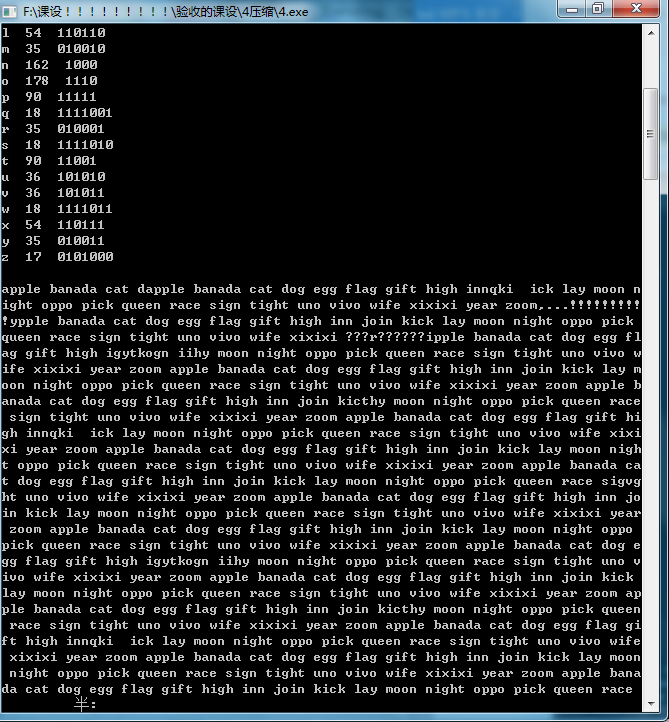
word[7-i] = (ch%2==0)?'0':'1';

}

对于每一个压缩符，都要通过8次取余，将每次取余的结果存储到字符串word中，这样所有的word拼装起来就是原文章的所有编码，然后从前往后遍历编码表，匹配成功则存入文件并更新下标，直到文件全部解码完毕。

4.运行结果





**五、无线传感器网络(必做) （图）**

1.[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，并给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。

1. 采用的数据结构

通过邻接表存储所有点的信息。

typedef struct Location{//坐标

int x;

int y;

}Location;

typedef struct Weight{//记录权值以及权值的平方

float weight;

int w\_weight;

}Weight;

typedef int Status;

typedef Location VertexType;

typedef Weight InfoType;

typedef struct ArcNode{

int adjvex;//存储当前邻接顶点在顶点向量中的位置

struct ArcNode\*nextarc;//存储下一个邻接点

InfoType \*inof;//存储权值信息

}ArcNode;

typedef struct VNode{

VertexType data;//储存顶点位置信息

ArcNode \*firstarc;//记录第一个邻接顶点的位置

}VNode,AdjList[MAX\_VERTEX\_NUM];

typedef struct{

AdjList vertices; //储存所有顶点连接情况

int vexnum;

}ALGraph;

1. 算法思想

邻接表的建立：

随机生成n个坐标，若该位置已经记录则重新生成。将这些坐标按x-》y排序，排序完成将其存到邻接表中，然后依次遍历所有顶点，扫描右下的10\*10的方格中有没有其他顶点，若存在判断距离是否＜10，若ok则这两个点互相邻接，遍历完成邻接表建立。

连通判断：

从汇点（0，0）开始深度遍历。遍历结束所有访问到的均为连通点，其余点删除。

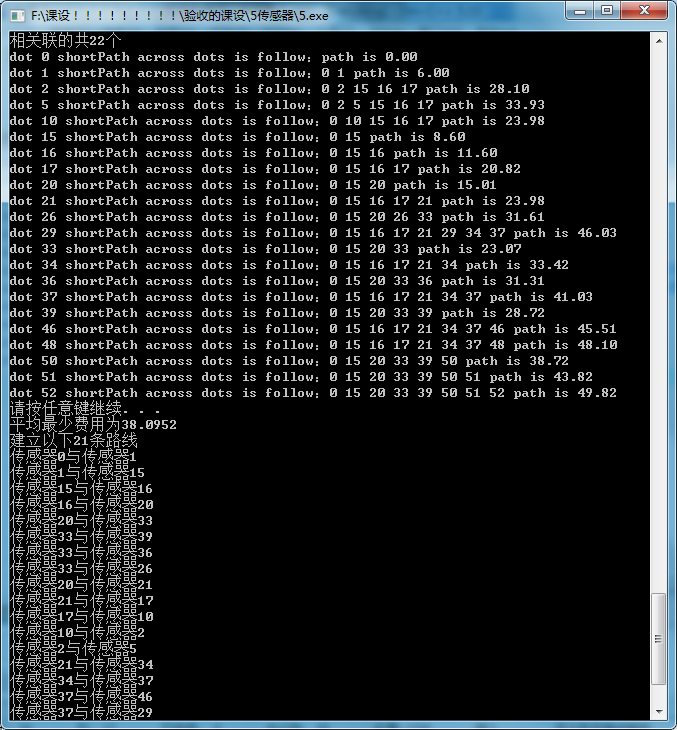
最少传输时间：

通过DIJ最短路径算法实现。不同的是最短距离只需求linkNum次，即将所有连通点的最短距离求出即可，结果即为最少传输时间。

最少能耗：

通过PRIM最小生成树算法实现。

1. 运行结果



**六、排序算法比较 （必做）（排序）**

1.[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；

3.算法思想

直接插入排序：从第二个数开始向后循环，若该数比前一个数大，则不做操作，否则，将其置入0号位置作为哨兵，向前遍历，若前一个元素大于该元素，就将前一个数向后移动一个位置，找到的第一个小于等于该数的位置之后，即为插入位置。

选择排序：

从第一个位置开始循环，通过Min（）函数找出当前元素往后所有元素中最小的一个，将其与第一个元素交换，继续进行第二个位置，直到循环结束。

交换排序：（冒泡排序）

通过两个for循环实现。i从1开始，j从i开始，依次比较a【i】和a【j】，若j小于该元素，则交换，反之，不操作，并让j++。直到j为Len结束第一次循环，结束时最小的元素已经移动到了最前面。使i++，继续循环此操作。

堆排序：构造一个大顶堆（或小顶堆），将堆顶数据输出后，最右下角的数据被移动到堆顶，再次从下到上调整成一个大顶堆，并输出堆顶元素，重复上述操作直至输出所有数据。

基数排序：建立F[0]到F[9]的数组，从个位开始，将每个数据连到对应的数组上，再讲数据按序收集，链到一起，对十位、百位重复此操作直至最高位。

希尔排序：先将整个待排记录序列分割成为若干子序列分别进行直接插入排，待整个序列中的记录基本有序时，对全体记录进行一次直接插入排序。本处即按增量序列依次为9,5,3,2,1进行处理。

归并排序：

通过多次递归的归并实现。

1. 归并思想：将两个有序序列合并到一个序列，比较首元素，小的赋给数组new，并让该数组指针加一，new数组的指针也加一。当一个数组元素已经合并完时直接将另一个数组剩余的元素接到new数组末尾即可。
2. 排序思想：递归归并排序该序列一分为二的子序列，重复此过程，直到当前序列长度为一，然后两两归并为一个序列，归并后序列变大，直到合并为一个序列。

快速排序：取第一个元素为数轴，所有小于他的数放在枢轴前，小于他的数放在枢轴后，再对划分后的每个部分重复此操作，直至某次划分后该部分的low==high，即只有一个数据。

4.时间复杂度：

直接插入排序：O(N\*N)

希尔排序：O（NlogN）

冒泡排序：O(N\*N)；

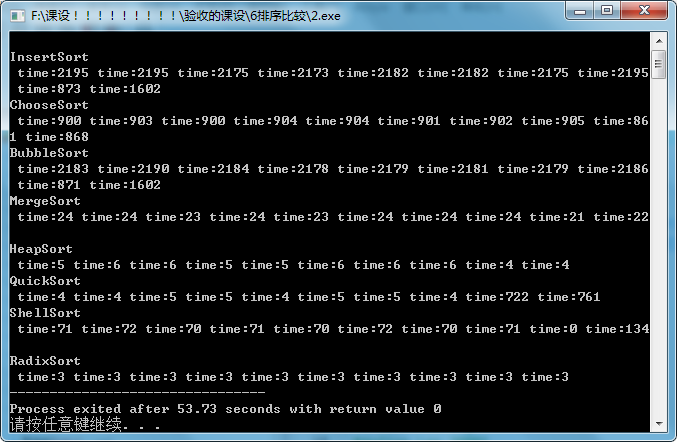
选择排序：O（N\*N）

归并排序：O（NlogN）

堆排序：O（NlogN）

基数排序：O（cN+d）

1. 运行结果



**七、Hash表应用 （选做） （查找）**

1.[问题描述]

设计散列表实现VIP客户发掘。对身份证号进行Hash, 通过对乘客某时间段内的乘机频率、里程数统计，发掘VIP客户。

[基本要求]

（1） 设每个记录有下列数据项：身份证号码（虚构，位数和编码规则与真实一致即可）、姓名、航班号、航班日期、里程。

（2） 从文件输入各记录，以身份证号码为关键字建立散列表。

（3） 分别采用开放定址（自行选择和设计定址方案）和链地址两种方案解决冲突；显示发生冲突的次数、每次中解决冲突进行重定位的次数。

（4）记录条数至少在100条以上。

（5） 从记录中实现乘客乘机频率、里程数统计，从而发掘VIP客户。

2．采用的数据结构

class record{//存数文件中单个记录

public:

string id;

string name;

string flightNum;//航班号

int mon;

int day;

int route;//航程

};

class people{//存储每个人的总统计信息

public:

string id;

string name;

int flytime;//飞行次数

int sum\_route;

people\* next;

};  
  
3．算法思想：

开放定址储存：

通过to\_ascii()函数将身份证的前六位与后四位的ascii码累加，取余 得出储存的地址，若该单元已经存在元素，判断是否为该乘客，若为该乘客，update该位置的飞行数据，否则通过二次线性再散列找出下一个位置，循环进行此判断，直到发现该乘客或空位。

开放定址查找：

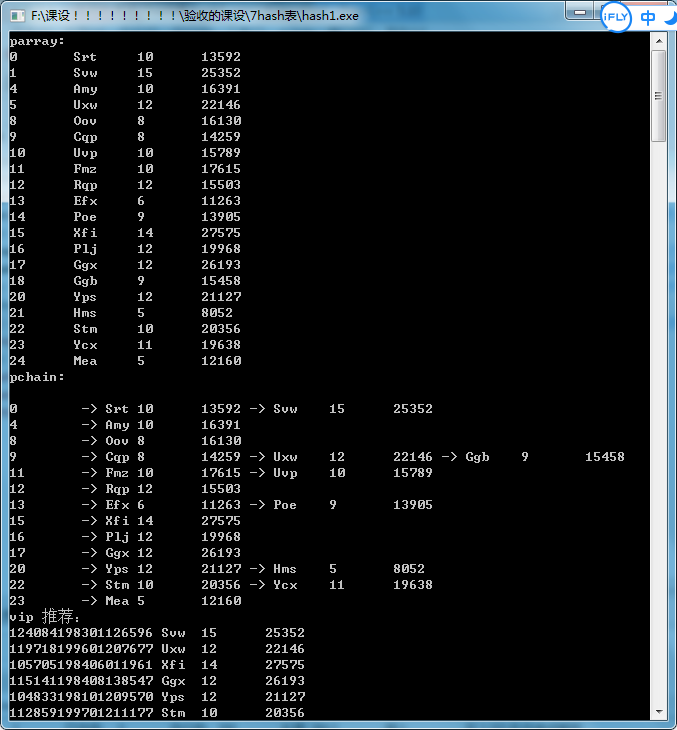
将要查找的元素通过to\_ascii()函数得出地址，判断当前位置上的元素是否有元素。若无，则查找失败，若有，判断id与其相等，若不等，循环二次线性再散列，直到找到该元素，否则查找失败。

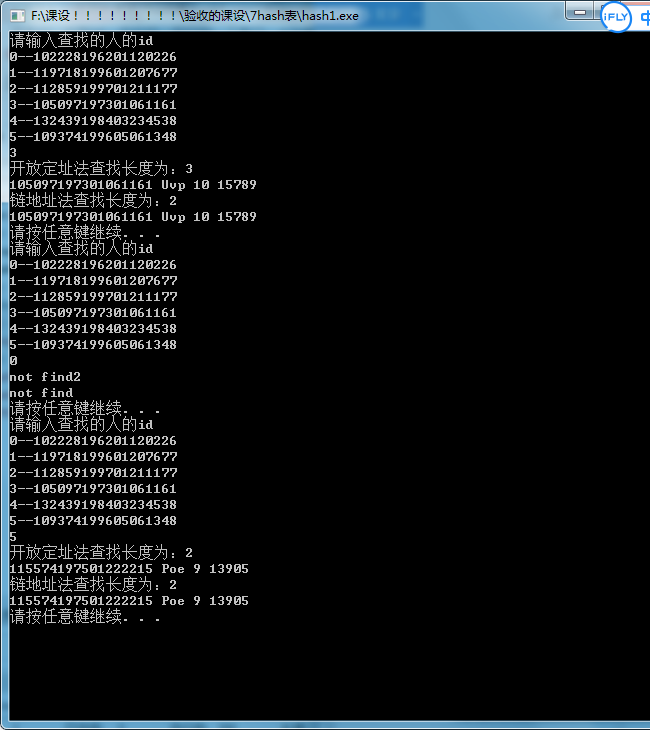
链地址法存储：

建立一个hash指针表，通过to\_ascii()函数得出hash地址，判断当前位置下有无乘客信息，若无，则添加，若有，循环判断是否为该乘客，直到找到该乘客或指针为NULL。

链地址法查找：

通过待查找的id通过to\_ascii()函数得出的地址循环比较，若找到该乘客的信息则输出，指针为NULL则查找失败。  
  
4．运行结果：





**八、公交线路提示 (选做) （图）**

1.[问题描述]

上网下载南京公交线路图，建立南京主要公交线路图的存储结构。

[基本要求]

（1）输入任意两站点，给出转车次数最少的乘车路线。

（2）输入任意两站点，给经过站点最少的乘车路线。

2.采用的数据结构

第一题所用的数据结构：

邻接矩阵存储公交信息，将公交车作为基本单位，建立所有公交车之间的邻接关系，若两者经过相同的公交站则二者相邻，距离设为一。将公交车经过的所有站点存在他的一个成员数组里。

class Bus{

public:

int busFlag;//公交车号

string station[MAX\_STATION];//经过的所有站

int staNum;//站的个数

Bus(){busFlag=0;staNum=0; }

bool FindStation(string s);//查询是否经过s站

void InsertStation(string s);//将s站存入

void busShow();

};

class crosspoint{

public:

int CPNumber;//两条公交路线共同经过的站的个数

string CPoint[20];//两条公交路线共同经过的公交站

};

typedef Bus VertexType;

typedef char\* InfoType;

class ArcCell{

public:

VRType adj;//存储权值，联通则为1，反之为 INITNUMBER

crosspoint CPt;//存储相交的公交站

};

typedef struct{

VertexType vexs[MAX\_BUS\_NUM];//存储所有公交车的信息

ArcCell arcs[MAX\_BUS\_NUM][MAX\_BUS\_NUM];//存储各个公交车与其他公交车的相邻情况

int busnum;//公交车个数

}MGraph;

第二题所用的数据结构：

邻接表存储所有站点信息,如果相邻则互相邻接，并存储该邻接关系是在哪条公交路线上。

typedef struct ArcNode{

int adjvex;//存储该邻接的站的位置

int samebus;//存储该邻接关系是在哪个公交路线上

struct ArcNode\*nextarc;//存储下一个邻接的站

int weight;//记录距离，该题相邻均为1

}ArcNode;

typedef struct VNode{

string data;//记录车站名

ArcNode \*firstarc;//出

}VNode,AdjList[MAX\_VERTEX\_NUM];

typedef struct{

AdjList vertices; //存储所有车站信息

int vexnum;

}ALGraph;

3.算法思想

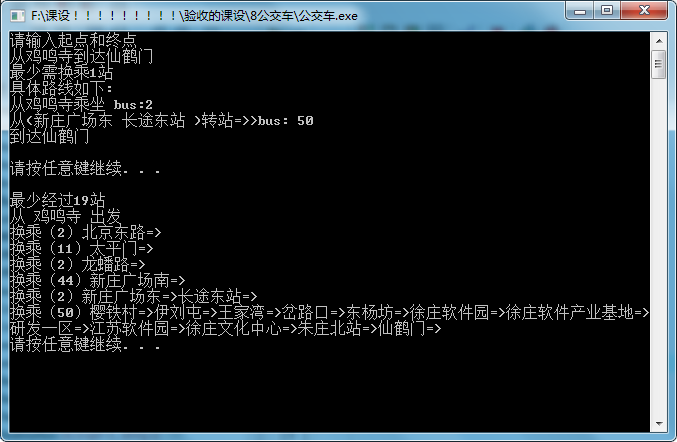
求最少换乘次数：

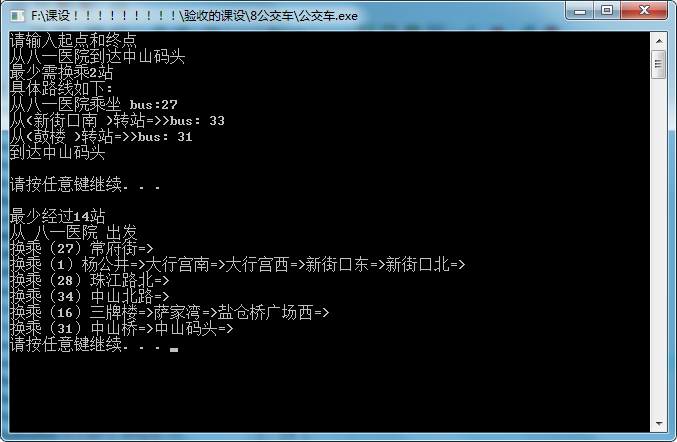
将公交车作为基本单位，经过的车站作为其成员变量。路线有交叉则二车相邻距离为一。

给定起点和终点，查找哪些公交车经过起点，哪些公交车经过终点，应用弗罗里达算法求这些公交车互相之间的最短距离，取最小的那一组输出。

求最少过站次数：

将公交站作为基本单位，从文件读取公交路线时，相邻的两站建立邻接关系，并记录该邻接关系在哪个公交路线上，邻接表建立完成后，从起点出发，运用DIJ算法求最短路径，当求的终点的最短路径时结束，并依次遍历这些经过的站点，从起点开始不断寻找下一个相邻的站点，并输出。

1. 运行结果
2. 



**九、算术表达式求值 (选做) （栈）**

1.[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。

2.采用的数据结构

stack<char>Ch;//存储操作符

stack<float>Num;//存储操作数

class P{

public:

int flag;//记录当前为操作数还是操作符，0--数，1--符

float num;//0

char c;//1

};

P p[MAX\_LENGTH];//存储表达式

1. 算法思想

表达式存储的思想：

首先将表达式读入到一个字符串中，然后从前往后依次判断并存储。若当前字符为操作符，设flag为1，存储。 若为操作数，则将字符串该位置往后映射，隔离出前面为float的部分。通过sscanf(&s[i],"%f%s",&t,k);实现。该函数在此处的功能就是将字符串前面的float格式的内容存入t，字符串剩余部分存入k。设flag为0，存入该数t。然后将k在拷贝到s中，再次从前往后遍历。

遍历结束后，操作数和操作符分离，通过flag区分。

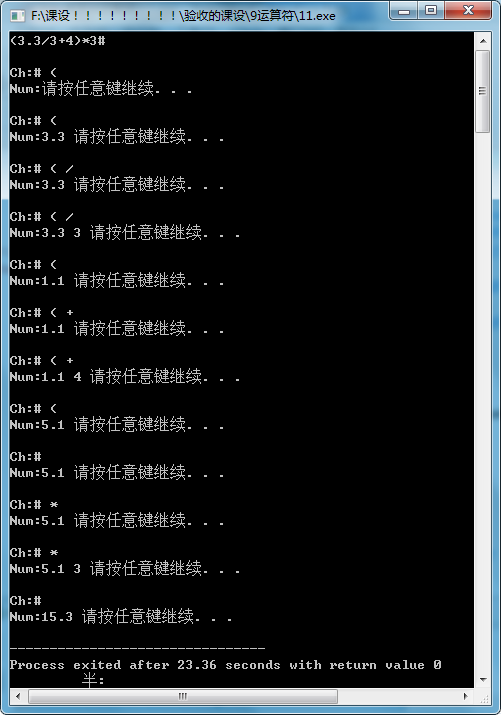
求值思想：

利用常规思想，依次读取数组中的元素，不断进行入栈出栈运算，最终得出表达式的值。

实现的出错检测：

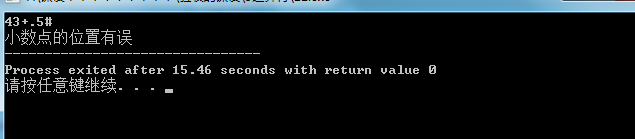
非法字符介入，表达式开头出现运算符，除数不能为零，括号不匹配，小数点位置非法，

1. 运行结果









**十、课程设计总结**

1. **代码行统计**

1.49+549=598

2.213

3.190+141=331

4.45+81+33+94+95+56=304

5.88+138+63+64+59=412

6.516

7.398

8.33+126+79+151+113+63+77=642

9.73+56+118=247

共3643行

**2.心得体会：**

这次数据结构课设真的刺激无比，也是这段时间让我感受到了大学的魅力。也是这段时间，让我感受到了时间的宝贵。更是在这段时间，让我感受到了生命的意义。 验收的前两晚，熬夜到四五点，甚至有种猝死的感觉，那两天平均每天睡觉不足四个小时，可以一整天不吃东西，晚上一个面包解决。这些感受都是大学的初体验。有骄傲也有烦躁，有喜悦也有疲劳，百感交集，为了debug甚至懒得穿鞋刷牙洗脸，一睁眼立马开始紧张的debug时光。真的挺刺激。

课设作业似乎从开学就布置下来了，老师也提醒过很多次，不要拖到最后，可自信的人呐，总是觉得时间还多的是，到时候大不了紧一紧，时间就有了，可到了最后几天，发现刚写了一两道题，依然充满自信，第二天消灭掉两题。然而到了第二天，一道题都没改好，这时候才发现了时间的紧迫性，白天几乎都泡在图书馆，虽说曾有一天连续灭掉运算符，hash，排序比较三道题，可毕竟这么多题难度不一。比较有趣的题之一就是文章压缩的那个，老师在刚讲过huffman编码后，曾提过这道课设题，当时由于正巧刚了解了左移右移运算符，联想到字符类型的二进制编码，顿时萌生了一些想法，而后便在课余时间进行检验并小有成就，后来稍加修饰变成品了。另一个比较好玩的就是公交车的那道选做题，某天晚上睡觉的时候看到这道题，晚上考虑了很久，第二天就成功实践，但后来有很多不足，比如显示了路线上的所有站点但不是按照顺序，有了顺序但没有标注是从哪转的车，显示了可以从哪站转车，但可供选择的等价的车又有很多，最终通过优化数据结构，在邻接顶点上添加了该邻接关系对应的公交车才完美解决掉了这道题。而后比较巧妙地就是运算符那道题，巧妙地运用了sscanf函数，当检测到有数字时，将算数表达式该位置往后划分两部分，一部分为float格式，另一部分仍为字符串格式，这样就可以巧妙地将数字部分与运算符部分巧妙分离，分别存储。然后很快就解决掉了这道题。

学了数据结构，平时刷csp考题的时候也有了更多的想法，对于之前做的题也有了更加高效的解法。感觉学到了很多特别有价值得东西，这是可以通过实践立马感受到的东西，是数据结构给我最直观地感觉。

这段时间的课设真的刻骨铭心，真正感受到了学习的趣味，不管是同学一起讨论问题的活跃还是自己成功做出来的成就感，都是刻骨铭心的，是很宝贵的记忆。