**南京理工大学紫金学院数据结构**

**课程设计报告**

|  |  |
| --- | --- |
| 课 程： | 数据结构 |
| 学 院： | 计算机学院 |
| 班 级： | 软件工程2班 |
| 学 号： | 180310230 |
| 姓 名： | 刘俊 |
| 选题名称： | 学生信息管理系统  学校导航图 |
| 起止时间： | 2019.11~2019.12 |
| 指导教师： | 朱保平 |

**2019年12月**

**1 问题描述**

设计一个学生管理系统，用于管理学生成绩，输入参数时，若输入的数字超出给出的范围，则提示出错，要求重新输入。

**2 基本要求**

能正常运行，进行简单的学生成绩管理。

**3 需求分析**

　　设计一个可以正常执行的学生成绩管理系统

　　输入的形式为：表单大小，学生学号，学生姓名，学生各项成绩，操作选项

输入值得范围为：0<=学生各项成绩<=100，0<=操作选项<=10

　　系统输出形式：请输入学生的数量：

录入数据：

学生学号：

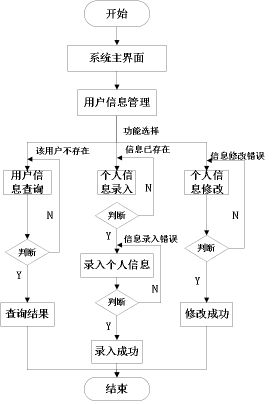
学生姓名：

学生的数据结构成绩：

学生的英语成绩：

学生的高数成绩：

**4 流程图**



**5 概要设计**

　　先调用linklist()全局函数，初始化学生管理系统，再调用initlink(int m)，明确学生数量,依次调用orderid(); rderEngscore(); orderMathscore();

orderDatascore();showid(elemtypee);add();changeid(elemtypee);deleteid(elemtype;showlink();等函数来输入学生的信息，最后通过main函数来对学生信息进行增删查改。

**6 详细设计**

initlink()作用是初始化学生信息管理系统，

flag=m;

node \*p=head,\*s;

for(int i=0;i<m;i++){

s=new node;

cout<<"学生学号：";

cin>>s->id;

cout<<endl;

cout<<"学生姓名：";

cin>>s->name;

cout<<endl;

cout<<"学生的数据结构成绩：";

cin>>s->datascore;

cout<<endl;

cout<<"学生的英语成绩：";

cin>>s->engscore;

cout<<endl;

cout<<"学生数学成绩：";

cin>>s->mathscore;

cout<<endl;

s->next=NULL;

p->next=s;

p=s;

}

orderformid()作用是将学生的信息按照学号排序，如果成功则输出success，

orderformEngscore()作用是讲学生的信息按照英语成绩排序，如果成功则输出success，

orderformMathscore()作用是讲学生的信息按照数学成绩排序，如果成功则输出success，

orderformDatascore()作用是讲学生的信息按照数据结构成绩排序，如果成功则输出success，

for(int i=0;i<flag;i++){

node \*q=head,\*p=head->next,\*s=p->next,\*a;

for(int j=i+1;j<flag;j++){

if(p->id>=s->id){

p->next=s->next;

s->next=p;

q->next=s;

q=q->next;

s=p->next;

}

else{

q=q->next;

p=p->next;

s=s->next;

}

}

}

showfromid(elemtype e)作用是根据学号查询学生信息

node \*q=head;

while(q->next&&q->id!=e){

q=q->next;

};

if(q->id==e){

cout<<"姓名："<<q->name<<endl<<"学号:"<<q->id<<endl<<"数据结构成绩："<<q->datascore<<endl<<"英语成绩："<<q->engscore<<endl<<"数学成绩："<<q->mathscore<<endl;

}else{

cout<<"id:"<<e<<"不存在"<<endl;

}

add()作用是增加其他学生信息

node \*s,\*q=head;

s=new node;

cout<<"学生学号：";

cin>>s->id;

cout<<endl;

cout<<"学生姓名：";

cin>>s->name;

cout<<endl;

cout<<"学生的数据结构成绩：";

cin>>s->datascore;

cout<<endl;

cout<<"学生的英语成绩：";

cin>>s->engscore;

cout<<endl;

cout<<"学生的数学成绩：";

cin>>s->mathscore;

cout<<endl;

s->next=q->next;

q->next=s;

flag++;

orderfromid();

changefromid(elemtype e)作用是根据学号修改学生信息

node \*q=head;

while(q->next&&q->id!=e){

q=q->next;

}

if(q->id==e){

cout<<"学生学号：";

cin>>q->id;

cout<<endl;

cout<<"学生姓名：";

cin>>q->name;

cout<<endl;

cout<<"学生的数据结构成绩：";

cin>>q->datascore;

cout<<endl;

cout<<"学生的英语成绩：";

cin>>q->engscore;

cout<<endl;

cout<<"学生的数学成绩：";

cin>>q->mathscore;

cout<<"success"<<endl;

}

else{

cout<<"学生学号:"<<e<<"不存在"<<endl;

};

deleteromid(elemtype e)作用是根据学号删除学生信息

node \*q=head,\*p=head->next;

while(p->next&&p->id!=e){

q=p;

p=p->next;

}

if(p->id==e){

q->next=p->next;

delete p;

flag--;

cout<<"success"<<endl;

}

else{

cout<<"学生学号:"<<e<<"不存在"<<endl;

}

showlink()作用是展示全部学生信息

node \*q=head->next;

while(q){

cout<<"学生姓名："<<q->name<<endl<<"学生学号:"<<q->id<<endl<<"学生的数据结构成绩："<<q->datascore<<endl<<"学生的英语成绩："<<q->engscore<<endl<<"学生的数学成绩："<<q->mathscore<<endl;

q=q->next;

}

**7 用户使用说明**

先设定表单大小，再输入学生信息，通过操作（0-10）来实现对学生信息的增删查改。

**8 测试结果**

请输入学生的数量：3

信息录入

学生学号：1

学生姓名：蒋大

学生的数据结构成绩：78

学生的英语成绩：75

学生数学成绩：82

学生学号：2

学生姓名：李二

学生的数据结构成绩：82

学生的英语成绩：85

学生数学成绩：81

学生学号：3

学生姓名：张三

学生的数据结构成绩：88

学生的英语成绩：86

学生数学成绩：87

finish

0:初始化

1：根据学号进行排序

2：根据英语成绩排序

3：根据数学成绩排序

4：根据数据结构成绩进行排序

5：根据学号查询

6：添加学生信息

7：根据学号修改学生信息

8：根据学号删除学生信息

9：展示列表

10：退出

5

输入学号：2

姓名：李二

学号:2

数据结构成绩：82

英语成绩：85

数学成绩：81

6

学生学号：4

学生姓名：阿达

学生的数据结构成绩：85

学生的英语成绩：59

学生的数学成绩：76

success

7

输入学号：4

学生学号：4

学生姓名：阿达

学生的数据结构成绩：77

学生的英语成绩：59

学生的数学成绩：74

success

9

学生姓名：蒋大

学生学号:1

学生的数据结构成绩：78

学生的英语成绩：75

学生的数学成绩：82

学生姓名：李二

学生学号:2

学生的数据结构成绩：82

学生的英语成绩：85

学生的数学成绩：81

学生姓名：张三

学生学号:3

学生的数据结构成绩：88

学生的英语成绩：86

学生的数学成绩：87

学生姓名：阿达

学生学号:4

学生的数据结构成绩：77

学生的英语成绩：59

学生的数学成绩：74

finish

10

--------------------------------

Process exited after 183.6 seconds with return value 0

请按任意键继续. . .

**10 课程设计总结**

**此次课程设计主要是运用了链表储存结构。让我对链表使用有了更深的认识。在课程设计的过程中主要遇到的问题是对链表的排序。所以结合以往学过的数组的冒泡排序法。在链表上进行运用。经过尝试，发现链表与数组如出一辙，冒泡排序的大体过程基本没有变化。课程设计采用的是去头链表，经过对带头链表，双链表，环链表的学习，发现这次的课程设计用去头链表更加方便。通过此次课程设计加深了我对链表的认识。**

# 学校导航图

**1 问题描述**

本次设计为数据结构中图类的问题。将校园景点作为图的结点，将景点间的路径作为图的边，路径距离作为边的权值。本次设计主要的目的是方便查询不同地点之间的最短路径，方便学生更快了解校园。

**2 基本要求**

程序能够正常运行，并能通过程序查询到校园两个不同地点之间的最短距离和路线。

**3 需求分析**

本次设计要求：尽量包含校园的所有场所，程序可以找到任一地点到另一地点的最短距离及其路线。

取值范围：1-25

输出形式：链状输出

**4 流程图**

查看地图--》输入坐标--》路径显示--》结束

**5 基本概要**

通过floyd算法对图结构进行最短路径的判断

**6 详细设计**

**根据地图进行节点表和边表的设计（地图由软件一班李一凡设计）**

**建立顶点表和边表**

int G[vertexNum][vertexNum]={

/\*0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 \*/

/\*0\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*1\*/ {INF,0, 80, INF,INF,400,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*2\*/ {INF,80, 0, 150,INF,INF,INF,INF,400,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,300,300,INF},

/\*3\*/ {INF,INF,150,0, 150,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*4\*/ {INF,INF,INF,150,0, INF,INF,INF,300,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,80, 300},

/\*5\*/ {INF,400,INF,INF,INF,0, 100,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*6\*/ {INF,INF,INF,INF,INF,100,0, 100,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*7\*/ {INF,INF,INF,INF,INF,INF,100, 0,110,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*8\*/ {INF,INF,400,INF,300,INF,INF,110,0, 80, INF,INF,INF,INF,300,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,300},

/\*9\*/ {INF,INF,INF,INF,INF,INF,INF,INF,80, 0, 110,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*10\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,110,0, 50, INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*11\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,50, 0, 150,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*12\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,0, 100,110,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*13\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,150,100,0, 150,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*14\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,110,150,0, 100,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*15\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,100,0, 110,INF,INF,INF,INF,INF,INF,INF,INF,INF},

/\*16\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,110,0, INF,INF,INF,INF,INF,INF,INF,400,110},

/\*17\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,0, INF, 80,INF,INF,INF,INF,400,INF},

/\*18\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,0, 100,50,INF, INF,INF,INF,INF},

/\*19\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,80, 100,0, INF,INF,INF,INF,INF,INF},

/\*20\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,50, INF,0, 110,INF,INF,250,INF},

/\*21\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,110,0, 110,INF,INF,INF},

/\*22\*/ {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,110,0, 110,250,INF},

/\*23\*/ {INF,INF,300,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,110,0, INF,INF},

/\*24\*/ {INF,INF,300,INF, 80,INF,INF,INF,INF,INF,INF,INF,INF,INF,500,INF,400,400,INF,INF,250,INF,250,INF,0, INF},

/\*25\*/ {INF,INF,INF,INF,300,INF,INF,INF,300,INF,INF,INF,INF,INF,INF,INF,110,INF,INF,INF,INF,INF,INF,INF,INF,0 }

};//邻接矩阵

string vertex[]={"0","行政楼","教a","教b","教c","a4","a6","综合楼","一食堂","瑞幸食堂","b5","b6","b7","b10","三食堂","c6","c8","d7","d2","d4","d1","行政办公12","实b","实d","图书馆","中国联通"}; //顶点表组

通过floyd算法进行最短路径的检索所以两点间的最短路径

int dist[vertexNum][vertexNum],i,k,j; //dist为权值存储数组

string path[vertexNum][vertexNum];

for(i=0;i<vertexNum;i++)

for(j=0;j<vertexNum;j++)

{

dist[i][j]=G[i][j]; //用G初始化dist

path[i][j]=vertex[i]+"-->"+vertex[j];

}

for(k=1;k<vertexNum;k++)

for(i=1;i<vertexNum;i++)

for(j=1;j<vertexNum;j++)

{

if((dist[i][k]+dist[k][j]<dist[i][j])&&(dist[i][k]!=INF)&&(dist[k][j]!=INF)&&(i!=j))

{

dist[i][j]=dist[i][k]+dist[k][j]; //存储i到j的最短路径长度

path[i][j]=path[i][k]+"-->"+vertex[j]; //

}

}

在主函数中给出开始和结束start和end参数 在floyd方法中输出路径和最短路径长度

do{

cout<<"请输入起点和终点的序号(终止请输入0，0)：";

cin>>start>>end;

Floyd(start,end);

} while(start!=0&&end!=0);

效果展示：

请输入起点和终点的序号(终止请输入0，0)：1 5

顶点:1到顶点5的最短路径及权值和为：

行政楼-->a4 400

请输入起点和终点的序号(终止请输入0，0)：0 0

--------------------------------

Process exited after 9.163 seconds with return value 0

请按任意键继续. . .

**10 课程设计总结**

校园导航图主要运用的最短路径的算法。书本上的两种算法大体的意思能够了解，但是在代码的编写上总是遇到问题。不过在多次的错误下。对floyd算法逐渐熟悉，并能清楚明白其中原理。