**数 据 结 构 课 程 设 计**

**任 务 书**

东北大学软件学院

2018年3月

一 课程基本信息 3

二 教学目的和基本要求： 3

三．课程设计的内容和安排 3

四 课程设计的题目 4

五 考核的方法和成绩的评定 4

六 时间的安排 4

七 课程设计报告的格式 4

八 课程设计结果的提交 4

九 课程设计题目说明 5

十．注意事项 11

导游路线图

停车动画

文件读写

# 一 课程基本信息

* 1. 课程设计名称：数据结构课程设计
  2. 课程设计编号：C0801207040
  3. 课程设计学分：2学分
  4. 课程设计周数（学时）：3周（32学时）
  5. 课程设计授课单位：软件工程系
  6. 指导方式：集体辅导和个别辅导相结合
  7. 适用专业：软件工程，软件工程（国际（日语）），软件工程（国际（英语））
  8. 课程设计教材及主要参考资料：

[1] 自编. 数据结构课程设计指导书. 东北大学软件学院，2014.

[2] 殷人昆，陶永雷等. 数据结构(用面向对象方法与c++描述). 清华大学出版社，2014.

# 二 教学目的和基本要求：

1. 了解并掌握数据结构和算法设计的方法，具备初步的独立分析和设计能力
2. 初步掌握软件开发过程的问题分析分析、系统设计、程序编码和测试等基本方法和技能。
3. 提高综合运用所学的理论只是和方法，独立分析和解决问题的能力。
4. 训练用系统的观点和软件开发一般规范进行软件的开发，培养软件工作着所具备的科学工作方法和作风。

# 三．课程设计的内容和安排

1. 问题分析和任务定义：根据设计题目的要求，充分地分析和理解问题，明确问题要求做什么，限制的条件是什么。
2. 逻辑设计：对问题中涉及到的操作对象定义相应的数据类型，并按照以数据结构为中心的原则划分模块，定义主程序和各抽象数据类型，逻辑设计的结果应写出每个抽象数据类型的定义（包括数据结构的描述和每个基本操作的功能说明），各个主要模块的算法，并画出模块之间的调用关系图。
3. 物理设计：定义相应的存储结构并写出个函数的伪码算法。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试，抽象数据类型的实现尽可能做到数据封装，基本操作的规格说明尽可能明确具体。详细设计的结果是对数据结构和基本操作做出进一步的求精，写出数据结构存储结构的类型定义，写出函数形式的算法框架。
4. 程序编码：把详细设计的结果进一步求精为程序设计语言。同时加入一些注解和断言，使程序中逻辑概；念清晰。
5. 程序调试和测试：采用自底向上，分模块进行，即先调试低层函数。能够熟练掌握调试工具的各种功能，设计测试数据确定疑点，通过修改程序来证实它或绕过它。调试正确后，认真整理源程序和注释，形成格式和风格良好的源程序清单和结果。
6. 结果分析：程序运行结果包括正确的输入及其输出结果、含有错误的输入和输出结果、算法时间复杂度和空间复杂度分析。
7. 撰写课程设计报告：总结和提升上述过程和步骤，写出结构严谨、表述清楚、符合设计规范的报告。

# 四 课程设计的题目

本课程实践完成景区信息管理系统的实现，该系统有三个信息部分构成，分别为景点导游线路信息，制订景区道路铺设信息，停车场进出车辆信息。

# 五 考核的方法和成绩的评定

考核成绩分为三个部分：

1. 平时成绩（10%）

出勤

态度

对各部分知识掌握情况

(2) 验收程序部分(50%)

按照以下几个指标进行验收

程序的正确性

程序的运行和存储效率

程序代码编写规范程度，代码的可读性、

界面是否友好

程序设计的创新性

（3）实验报告（40%）

实验报告结构是否清晰，问题叙述是否具有逻辑性

关键代码设计是否正确无误

程序设计的时间、空间复杂性分析是否正确

# 六 时间的安排

共三周，每个同学可以根据自身实际情况掌握进度，最后一天验收程序。也可以按照阶段分步进行验收。

# 七 课程设计报告的格式

* 1. 问题的概述、分析和研究意义
  2. 数据结构的逻辑设计和物理存储设计
  3. 重要算法的设计、流程描述和代码描述
  4. 数据结构的时间、空间复杂性分析和重要算法的复杂度分析。
  5. 程序的最终实现结果（包括重点界面的抓取，能说明问题的重要实验结果数据的打印或其可视化结果）。
  6. 参考文献
  7. 附录：关键数据结构的定义及关键算法的源代码。

# 八 课程设计结果的提交

1. 课程设计报告（一份、A4纸打印，同时包括一份电子文档）
2. 完整的程序系统（电子方式提交），能够对输入产生相应的输出，同时尽量完成可视化演示。
3. 源程序文档（电子方式提交），源程序代码要求适当缩进编排、结构清晰、可读性好，对源程序的类说明（如采用面向对象方法设计）、函数说明、接口说明、关键变量说明等要进行注释。
4. 答辩报告（制作PPT报告，电子方式提交），要求重点突出，思路清晰，同时就此报告准备答辩。

所有电子方式提交的文件全部存放在一个目录中，并对其进行压缩，压缩后的文件按规定格式进行命名，命名格式为：班级号＋学号＋姓名。

# 九 课程设计题目说明

本次课程设计题目主要是设计一个大型景区的管理系统。系统用户包括管理员和游客两类，管理员负责管理景区的景点维护；游客可以根据自己的需求对景区进行各种信息查询，及路线规划等。

**1. 系统初始化**

首先将景区的初始结构保存于文本文件中，当系统打开时，系统从文件中读取景区信息， 初始化景区图结构**。**景区的信息包括景点的信息、近邻景点之间的通路和距离等。用邻接链表存储景区分布图的信息，带权无向图的邻接链表。首要完成创建景区景点分布图和输出景区景点分布图（邻接矩阵）。

* + - * 1. 文件中首先包含图中结点数目，然后是每个结点的信息，结点的信息包括结点的名字和相关介绍。之后是边的信息，每条边至少包括两个端点和边上的权值信息。（如后续设计有需要，可自行添加结点和边的其他信息）

下面给出一个文件的基本结构

景点——景点——距离权值

北门——狮子山——9

北门——仙云石——8

狮子山——一线天——7

狮子山——飞流瀑——6

仙云石——仙武湖——4

仙云石——九曲桥——5

仙武湖——九曲桥——7

一线天——观云台——11

飞流瀑——观云台——3

一线天——花卉园——10

观云台——红叶亭——15

花卉园——红叶亭——9

观云台——碧水亭——16

仙武湖——碧水亭——20

朝日峰——碧水潭——17

朝日峰——红叶亭——10

九曲桥——朝日峰——20



图1. 例图

* + - * 1. 建图子模块建立无向带权图，根据顶点信息和边的信息，构建邻接链表G。由于是无向边，输入一条边时构建两条边。顶点信息可以包括：景点名称，景点简介，景点欢迎度，有无休息区，有无公厕。边的信息可以包括两个景点的距离和需要的时间。

数据的结构体（仅供参考，可以自己定义合适的类或结构体）

typedef struct ArcNode

{

}ArcNode;//定义顶点信息

typedef struct VNode

{

}VNode,AdjList[MAX\_VERTEX\_NUM];//定义边信息

typedef struct

{

}ALGraph;//定义邻接表

功能函数：

void CreatGraph(ALGraph &G);//创建图的邻接表存储

void OutputGraph(ALGraph G);//输出图的邻接表

对于图的数据信息建立，可以进行多组数据的测试，以保证该函数的功能正确

* + - * 1. 输出图子模块：从邻接链表g转换成邻接矩阵a，并输出邻接矩阵a。图中边的权值∞用32767表示。

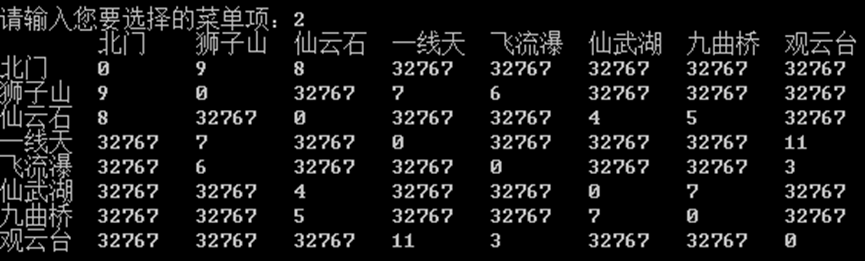


图2. 输出图示例

**2. 建立主程序应用菜单选项**

主程序应用菜单选项包含功能：

* + - 管理员登录
    - 输出景区景点分布图
    - 景点的查找与排序
    - 输出导游线路图
    - 两个景点间的最短路径和最短距离
    - 停车场车辆进出记录信息。

对选项采用数字标识进行选择，对其他错误输入可以进行判别，提示输入错误。

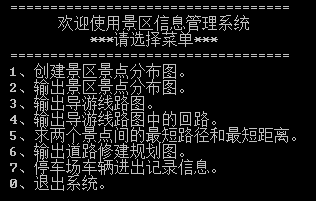


图3. 菜单例图(仅供参考，实际内容自定义)

**3. 管理员模块**

管理员模块主要负责维护整个景区景点以及发布各项通知。管理员登录后可以进行：

* + - 景点的插入和删除：景区一旦新开发了景点或者目前某个景点在维护中，需要将景点删除，而景点的删除同样会将相关的边删除。删除的方式尽量不要重新构建数组和图结构。
    - 路的插入和删除：维护各景点之间路的信息
    - 发布通知通告

**4. 景点的查找与排序**

* **查找功能(KMP,字典树)**

可以根据用户输入的关键字进行景点的查找，关键字可以在景点名称也可以在景点介绍中。查找成功则返回景点的相关简介，如果查找不成功请给予正确提示。

* **排序功能**

可以根据不同的需求进行排序，如按景点欢迎度或者景点的岔路数对景点进行排序，排序算法可以采用冒泡、快速、插入排序等。

排序后可以直接打印到控制台，检验排序或者查询后的数据是否完整。

**5. 输出两个景点之间最短路径和最短距离**

求两个景点间的最短路径和最短距离，由MiniDistanse(G1,path,D)函数实现；  
　 在参观途中，如果游客需要查找从当前结点到某一结点的最短路径或最短时间时调用该方法。在本线路图中将输出任意景点间的最短路径和最短距离。算法采用迪杰斯特拉算法（必做），弗洛伊德算法（选做）。

计算最短路径：void ShortestPath(ALGraph G,int path[][MAX\_VERTEX\_NUM],

double D[][MAX\_VERTEX\_NUM]);

求最短路径：void OutPutShortestPath(ALGraph G,int path[][MAX\_VERTEX\_NUM],

double D[][MAX\_VERTEX\_NUM],int i,int j);

例如：

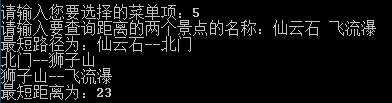


图4. 最短路径例图

**6. 输出导游线路图（保存导游线路图）**

* **功能函数**

输出导游线路图，由CreatTourSortGraph(G,G1)函数实现；  
** 建立路线图**

对游客来说，周游景点的方式有两种，第一种从入口A进入，最后从入口A出；另外一种从A进入，从另外的口B出。也就是遍历所有结点一遍的回路，而往往游客希望用最小耗费来遍历所有的景点，则该问题是一个最短哈密尔顿回路问题。该问题目前还没有一个最优解决方案，可以自己设计一个算法来完成这个问题的近似解法。

对于无向图g，这里提供一个备选近似解法：

void Traceablepath (Graph g)

{

(1)选择g的任一顶点r；

(2)用最小生成树算法找出带权图g的一棵以r为根的最小生成树T；

(3)前序遍历树T得到的顶点表L；

(4)将r加到表L的末尾，按表L中顶点次序组成回路H，作为计 算结果返回；

}

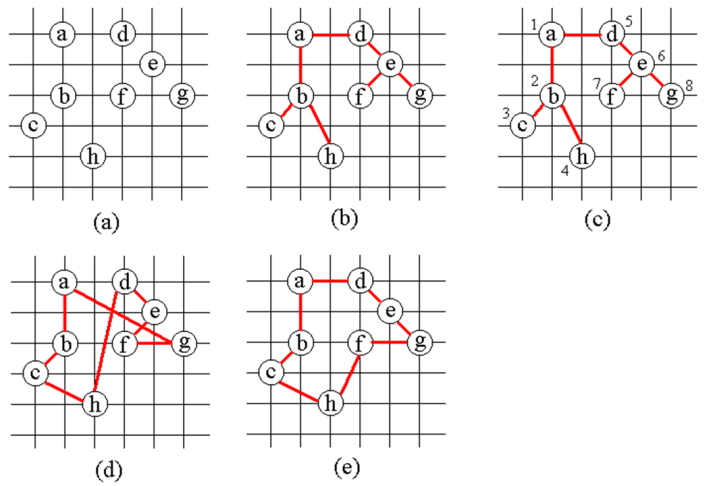


图5. 最小哈密尔顿回路近似算法例图

**(b)表示找到的最小生成树T；(c)表示对T作前序遍历的次序；(d)表示L产生的哈密顿回路H；**

**(e)是G的一个最小费用旅行售货员回路。**

**7．输出车辆的进出信息**

设停车场是一个可以停放n辆汽车的南北方向的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列（大门在最南端，最先到达的第一辆车停放在车场的最北端），若车场内已停满n辆车，那么后来的车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，待该辆车开出大门外，其它车辆再按原次序进入车场，每辆停放在车场的车在它离开停车场时必须按它停留的时间长短交纳费用。试为停车场编制按上述要求进行管理的模拟程序。要求程序输出每辆车到达后的停车位置（停车场或便道上），以及某辆车离开停车场时应缴纳的费用和它在停车场内停留的时间。

* **设计思路**

停车场的管理流程如下：

1. 当车辆要进入停车场时，检查停车场是否已满，如果未满则车辆进入停车场；如果停车场已满，则车辆进入便道等候。
2. 当车辆要求出栈时，先让在它之后进入停车场的车辆退出停车场为它让路，再让该车退出停车场，让路的所有车辆再按其原来进入停车场的次序进入停车场。之后，再检查在便道上是否有车等候，有车则让最先等待的那辆车进入停车场。

* **数据结构**

由于停车场只有一个大门，当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，先进停车场的后退出，后进车场的先退出，符合栈的“后进先出，先进后出”的操作特点，因此，可以用一个栈来模拟停车场。而当停车场满后，继续来到的其它车辆只能停在便道上，根据便道停车的特点，先排队的车辆先离开便道进入停车场，符合队列的“先进先出，后进后出”的操作特点，因此，可以用一个队列来模拟便道。排在停车场中间的车辆可以提出离开停车场，并且停车场内在要离开的车辆之后到达的车辆都必须先离开停车场为它让路，然后这些车辆依原来到达停车场的次序进入停车场，因此在前面已设的一个栈和一个队列的基础上，还需要有一个地方保存为了让路离开停车场的车辆，由于先退出停车场的后进入停车场，所以很显然保存让路车辆的场地也应该用一个栈来模拟。因此，本题求解过程中需用到两个栈和一个队列。栈以顺序结构实现，队列以链表结构实现。

* **车辆出入清单**

程序提示：以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入的输入数据序列进行模拟管理。每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码以及到达或离去的时刻。对每一组输入数据进行操作后的输出信息为：若是车辆到达，则输出汽车在停车场内或便道上的停车位置；若是车辆离去，则输出汽车在停车场内停留的时间和应交纳的费用（在便道上停留的时间不收费）。

数据结构：

模拟停车场的堆栈的性质；

typedef struct zanlind{       
    int number;   //汽车车号  
    int ar\_time;   //汽车到达时间  
}zanInode;

typedef struct{  
    zanInode   \*base;   //停车场的堆栈底  
   zanInode   \*top;   //停车场的堆栈顶  
   int stacksize\_curren;  
}stackhead;

例如：

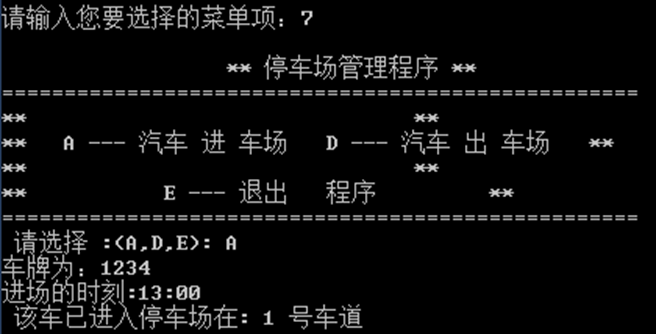


图6. 停车场管理例图

# 十．注意事项

1．在开始编码之前通读所有的任务，对项目进行整体分析和设计，不要仅仅考虑第一个任务，造成所设计的类不满足后续任务需求。

2．任务书中提供的结构体等仅供参考，同学们需要根据自己对需求的分析，确定自己的结构形式。

3．鼓励有创新性的功能改进，或者可视化的界面设计等。

数据结构代码（怎么搭建的）